ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΡΓΑΛΕΙΟ ΔΙΑΧΕΙΡΙΣΗΣ ΜΟΥΣΙΚΩΝ ΕΡΓΩΝ

Των φοιτητών
ΑΛΕΞΑΝΔΡΟΥ ΘΕΩΔΩΡΟΥ ΤΣΑΤΣΟΥ
Αρ. Μητρώου: 123956

ΑΛΕΞΑΝΔΡΟΥ ΜΠΑΡΚΟΛΙΑ
Αρ. Μητρώου: 123970

Επιβλέπων
ΔΕΛΗΓΙΑΝΝΗΣ ΙΓΝΑΤΙΟΣ
Καθηγητής

ΣΕΠΤΕΜΒΡΙΟΣ 2020

Τίτλος Δ.Ε **Εργαλείο διαχείρισης μουσικών έργων**
Κωδικός Π.Ε. **19040**
Ονοματεπώνυμο φοιτητών **Τσάτσος Θεόδωρος – Αλέξανδρος, Μπαρκολίας Αλέξανδρος**
Ονοματεπώνυμο εισηγητή **Δεληγιάννης Ιγνάτιος**
Ημερομηνία ανάληψης Δ.Ε. **Οκτώβριος 2019**
Ημερομηνία περάτωσης Δ.Ε. **Σεπτέμβριος 2020**

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Στις μέρες μας οι κινητές συσκευές όχι μόνο έχουν μπει στην καθημερινότητα μας αλλά αποτελούν και ένα σημαντικό κομμάτι στην καθημερινότητα μας, για ψυχαγωγία και επικοινωνία αλλά ακόμη και ως ένα επικουρικό μέσο υποβοήθησης της εργασίας πολλών ανθρώπων. Το βασικό πλεονέκτημα τους είναι η φορητότητα και το μικρό μέγεθος. Τα χαρακτηριστικά αυτά μας προκάλεσαν το ενδιαφέρον να επιλέξουμε να υλοποιήσουμε αυτήν την εφαρμογή στην πλατφόρμα Android. Στόχος μας ήταν να υλοποιήσουμε μια καθημερινή εφαρμογή για την ψυχαγωγία των χρηστών, καθώς επίσης και την χρήση της από μουσικούς ως μέσο διευκόλυνσης για την εκμάθηση μουσικής.

Με την υλοποίηση της παρούσας εργασίας είχαμε την ευκαιρία να γνωρίσουμε ένα νέο περιβάλλον ανάπτυξης εφαρμογών το Android Studio καθώς και να εμβαθύνουμε σε νέες τεχνολογίες. Τέλος μας δόθηκε η ευκαιρία να εφαρμόσουμε γνώσεις που αποκτήσαμε από την σχολή μας.

Περίληψη

Η παρούσα πτυχιακή έχει ως στόχο την ανάπτυξή ενός έργου λογισμικού διαχείρισης μουσικών έργων. Ουσιαστικά η εφαρμογή έχει ως στόχο την αναπαραγωγή βίντεο από τον κατάλογο της συσκευής όσο και από την πλατφόρμα του YouTube, την εμφάνιση η την προσθήκη των στοίχων και την δυνατότητα τμηματοποίησης σε μέρη του βίντεο.

Για την ανάπτυξη του έργου χρησιμοποιήθηκε η σύγχρονη και ευέλικτη (modern agile) μέθοδος Scrum όσον αφορά την οργάνωση και την λειτουργία της ομάδας καθώς και στην διαχείριση του έργου. Επίσης χρησιμοποιήθηκε η ενοποιημένη προσέγγιση (Unified Process) για τον χειρισμό των απαιτήσεων και την γραφική ανάπτυξη με UML.

Το παρόν έργο αναπτύχθηκε σε Android χρησιμοποιώντας τεχνολογίες και βιβλιοθήκες που αναλύονται στο παρόν έγγραφό.

Η εφαρμογή χωρίζεται σε δύο μέρη, το ένα μέρος της αφορά τα αρχεία video της συσκευής . Το δεύτερο μέρος αφορά τα βίντεο που ο χρήστης αναζητά στην πλατφόρμα του YouTube.

.

«Music project management tool»

«Tsatsos Theodoros- Alexandros, Barkolias Alexandros»

Abstract

This thesis aims to develop a software of management music project. To be more specific, this application aims to play videos directly from the device, as well as from the YouTube platform, to display or add the elements and the ability to create parts of the video.

For the development of the project, the organization and operation of the team as well as the project management, it was used the modern and flexible (modern agile) Scrum method.

The Unified Process approach was also used for requirements handling and graphical development with UML.

This project was developed on Android using the technologies and the libraries that are being discussed in this document.

The application is divided into two parts, one part concerns the video files of the device. The second part is about the videos that the user is looking for on the YouTube platform.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε στα πλαίσια του προπτυχιακού προγράμματος του τμήματος Πληροφορικής και Ηλεκτρονικών συστημάτων της σχολής Μηχανικών του ΔΙ.ΠΑ.Ε. κατά το ακαδημαϊκό έτος 2019-2020.

Αρχικά θα θέλαμε να ευχαριστήσουμε τον καθηγητή και επιβλέποντα Ιγνάτιο Δεληγιάννη για την εμπιστοσύνη που μας έδειξε αναθέτοντας μας αυτήν την εργασία, όσο και για την στήριξη και την υπομονή μου μας έδειξε καθ' όλη την διάρκεια μέχρι την ολοκλήρωση του έργου.

Εν συνεχεία θα θέλαμε να ευχαριστήσουμε συνάδελφούς και φίλους που μας βοήθησαν ο καθένας με την δική του οπτική στο τελικό αποτέλεσμα.

Τέλος είμαστε ευγνώμονες στους γονείς μας για την αμέριστη συμπαράστασή σε όλη την διάρκεια τον σπουδών μας τόσο σε ψυχολογικό όσο και σε οικονομικό επίπεδο.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1ο: Εισαγωγή στο Android

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο επιχειρείται μια ανασκόπηση του τι είναι το λειτουργικό σύστημα Android πάνω στο οποίο αναπτύχθηκε το έργο μας. Γίνεται αναφορά στο ιστορικό των εκδόσεων του καθώς και μια συνοπτική ανάλυση της αρχιτεκτονικής του και των χαρακτηριστικών του.

1.2 Τι είναι το Android

Είναι ένα λειτουργικό σύστημα ανοικτού κώδικα για κινητές συσκευές όπως τα έξυπνα τηλέφωνα και οι ταμπλέτες βασισμένο στο λειτουργικό Linux. Στην πορεία προσαρμόστηκε για χρήση και σε άλλες συσκευές όπως οι τηλεοράσεις, τα αυτοκίνητα καθώς και σε συσκευές διαδικτύου. Πρόκειται για το πιο διαδεδομένο λογισμικό. Η ανάπτυξη του γίνεται από την Open Handset Alliance με βασικό συντελεστή την Google. Η τελευταία έκδοση η οποία είναι διαθέσιμη τη παρούσα στιγμή είναι η android 11.

1.3 Εκδόσεις του Android

Πίνακας 1-1 Εκδόσεις του Android

Κωδικό όνομα	**Νούμερο έκδοσης**	**Ημερομηνία αρχικής κυκλοφορίας**	**Επίπεδο API**
Alpha	1.0	23 Σεπτεμβρίου 2008	1
Beta	1.1	9 Φεβρουάριου 2009	2
Cupcake	1.5	27 Απριλίου 2009	3
Donut	1.6	15 Σεπτεμβρίου 2009	4
Eclair	2.0 – 2.1	26 Οκτωβρίου 2009	5 – 7
Froyo	2.2 – 2.2.3	20 Μαΐου 2010	8
Gingerbread	2.3 – 2.3.7	6 Δεκεμβρίου 2010	9 – 10
Honeycomb	3.0 – 3.2.6	22 Φεβρουάριου 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	18 Οκτωβρίου 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	9 Ιουλίου 2012	16 – 18
KitKat	4.4 – 4.4.4	31 Οκτωβρίου 2013	19 – 20

Lollipop	5.0 – 5.1.1	12 Νοεμβρίου 2014	21 – 22
Marshmallow	6.0 – 6.0.1	5 Οκτωβρίου 2015	23
Nougat	7.0 – 7.1.2	22 Αυγούστου 2016	24 – 25
Oreo	8.0 – 8.1	21 Αυγούστου 2017	26 – 27
Pie	9.0	6 Αυγούστου 2018	28
Q	10.0	3 Σεπτεμβρίου 2019	29
R	11.0	19 Φεβρουαρίου 2020	30

1.3.1 ANDROID 1.1

Είναι η δεύτερη γενική έκδοση του λειτουργικού Android και έκανε την εμφάνιση της τον Φεβρουάριο του 2009 και αποτέλεσε τον διάδοχο του Android 1. Διόρθωσε αρκετά σφάλματα αλλά δεν παρείχε κάτι το επαναστατικό.

Εικόνα 1-1 Android 1.1

1.3.2 Cupcake 1.5

Αποτέλεσε μια έκδοση ορόσημο προσφέροντας πολλές νέες λειτουργείες με σημαντικότερη εξ' αυτών το πληκτρολόγιο αφής ενώ μέχρι αυτήν την έκδοση δεν υπήρχε υποστήριξη πληκτρολογίου. Κυκλοφόρησε ως αναβάθμιση στο μοντέλο G1.

Μια ακόμη σημαντική προσθήκη είναι η δυνατότητα ανάπτυξης widgets (μικρές εφαρμογές εργαλεία αρχικής οθόνης) από προγραμματιστές.

Στις μετέπειτα βελτιώσεις αυτής της έκδοσης συναντάμε για πρώτη φορά την λειτουργεία της αντιγραφής και επικόλλησης προς και από το πρόχειρο όπως στους προσωπικούς υπολογιστές και την επιλογή αυτόματης περιστροφής οθόνης.

Τέλος, γίνεται διαθέσιμη η αναπαραγωγή και καταγραφή video κάτι το οποίο είχε παραλείψει η πρώτη έκδοση του λειτουργικού ενώ υποστηρίζεται και το ανέβασμα αρχείων βίντεο στο YouTube καθώς και φωτογραφιών στο Picasa.

Εικόνα 1-2 Cupcake 1.5

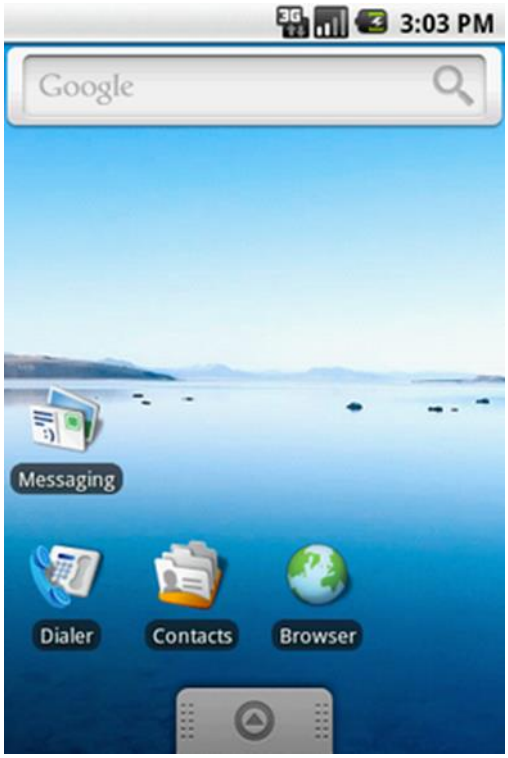
1.3.3 Donut 1.6

Σε σχέση με την 1.5 δεν παρατηρούνται τόσο μεγάλες αλλαγές. Η σημαντική αλλαγή αφορά την υποστήριξη διαφορετικών μεγεθών οθόνης. Το πρώτο τηλέφωνο κυκλοφόρησε με ανάλυση οθόνης 320x480 η οποία θεωρήθηκε η βασική ανάλυση του λειτουργικού τα dp-pixels θα λύσουν το πρόβλημα των τηλεφώνων που θα κυκλοφορήσουν σε άλλες διαστάσεις οθόνης ώστε να μην υπάρχει πρόβλημα στην εμφάνιση των στοιχείων που θα είχαν πολύ μικρό μέγεθος.

Η έκδοση αυτή εισάγει την ολοκληρωμένη αναζήτηση οπού ο χρήστης μπορεί να κάνει αναζήτηση τοπικά στην συσκευή του όπως σε εφαρμογές, αρχεία και επαφές πέραν της αναζήτησης στον ιστό μέσω του google.com.

Σημαντικές αλλαγές παρατηρούνται και στον ανασχεδιασμό του καταστήματος εφαρμογών Android Market όπως λίστες κορυφαίων εφαρμογών δωρεάν ή επί πληρωμή σε μια χρονική στιγμή που αυξάνονται οι εφαρμογές τρίτων κατακόρυφα.

Το Gallery και η κάμερα είναι πλήρως ανανεωμένα και ο χρήστης έχει την δυνατότητα να επιλέξει πολλά αρχεία ταυτόχρονα και να τα επεξεργαστεί όπως για παράδειγμα να διαγράψει πολλαπλά αρχεία φωτογραφιών. [1]



Εικόνα 1-3 Donut 1.6

1.3.4 Éclair (2.0 / 2.1)

Σε αυτήν την έκδοση ο χρήστης μπορεί να συνδέσει περισσότερους από έναν λογαριασμούς Google πράγμα που συμβαίνει για πρώτη φορά δίνοντας την ευκολία στον χρήστη να ελέγχει για παράδειγμα τα email του ξεχωριστά.

Οι χάρτες έρχονται προ εγκατεστημένοι στην εργοστασιακή έκδοση με σημαντικές βελτιώσεις όπως η εισαγωγή της λειτουργίας πλοήγησης από την συσκευή, ενώ παράλληλα κάνει την εμφάνιση του το Bluetooth 2.1.

Γίνονται σαφώς βελτιώσεις στην κάμερα με νέες δυνατότητες όπως το ψηφιακό zoom, εφέ χρώματος υποστήριξη φλάς κ.α.. Η εφαρμογή φωτογραφιών περιέχει βασικά εργαλεία επεξεργασίας φωτογραφιών.

Το πρόγραμμα περιήγησης πλέον υποστηρίζει την γλώσσα HTML5 ακολουθώντας το παράδειγμα του google chrome.

Μια ακόμα σημαντική αλλαγή είναι η λειτουργία μετατροπής ομιλίας σε κείμενο, καθώς αυτή η λειτουργία προστίθεται στο πληκτρολόγιο μέσω ενός κουμπιού.

Εικόνα 1-4 Éclair (2.0 / 2.1)

1.3.5 Froyo 2.2

Σε αυτήν την έκδοση παρέχεται βελτιστοποίηση στην ταχύτητα την μνήμη και την απόδοση. Υποστήριξη της άμεσης μεταγλώττισης (Just In Time compilation - JIT) η οποία αυτόματα μετατρέπει τα java bytoce αρχεία σε μητρικό κώδικα android κατά τη διάρκεια της εκτέλεσής τους κάτι που οδήγησε στην αύξηση της ταχύτητας του λειτουργικού συστήματος,

Ο Χρήσης μπορεί να κάνει εγκατάσταση των εφαρμογών του Android Market σε έναν εξωτερικό χώρο αποθήκευσης (κάρτα SD) απελευθερώνοντας έτσι χώρο στην μνήμη τηλεφώνου ενώ παρέχεται πλέον και η υποστήριξη μετατροπής του κινητού σε Wi-Fi hotspot. Το πληκτρολόγιο υποστηρίζει την πολυγλωσσικότητα ενώ η οθόνη κλειδώματος εφαρμόζει μια πολιτική με αριθμητικούς κωδικούς πρόσβασης.

Τέλος, γίνεται η χρήση του Adobe Flash κατά την περιήγηση στον ιστό και οι ενημερώσεις στο Android Market μπορούν να γίνουν αυτόματα. [2] [3] [4]

Εικόνα 1-5 Froyo 2.2

1.3.6 GingerBread (2.3)

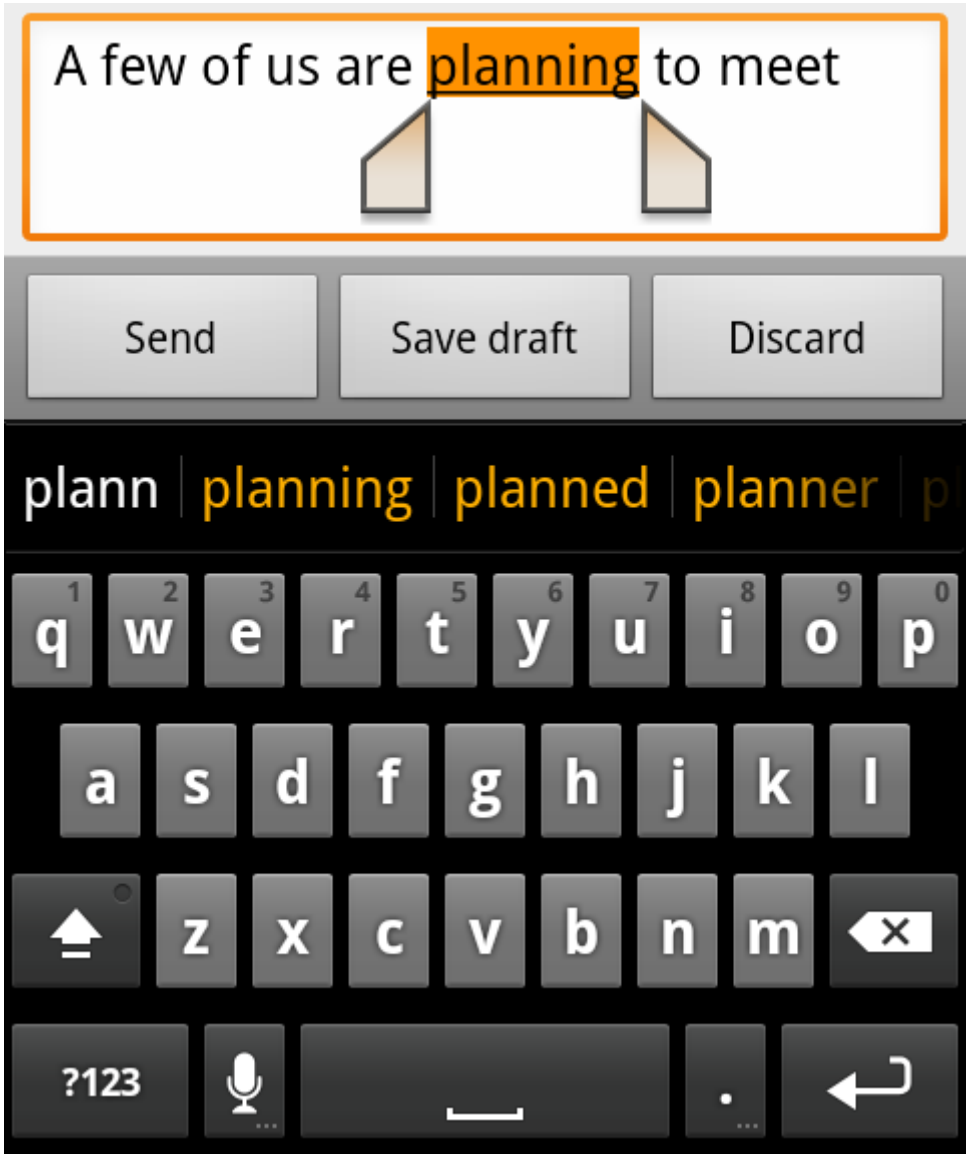
Στην έκδοση αυτή υπάρχουν βελτιώσεις σε υπάρχουσες λειτουργιές κάνοντας τες πιο εύχρηστες. Μια από αυτές είναι η αντιγραφή και η επικόλληση κειμένου που παλιότερα γινόταν με κέρσορες ενώ τώρα ο χρήστης μπορεί με το δάκτυλο του να επιλέξει ολόκληρες λέξεις.

Στο πληκτρολόγιο συναντάμε και την δυνατότητα αυτόματης διόρθωσης μέσω τον προτάσεων που εμφανίζει το λεξικό.

Σημαντική προσθήκη αποτελεί και η εισαγωγή εργαλείων διαχείρισης εφαρμογών όπου ο χρήστης έχει την δυνατότητα να αναγνωρίσει ποίες εφαρμογές εκτελούνται στο παρασκήνιο με σκοπό ο χρήστης να έχει την δυνατότητα να τις τερματίσει και να βελτιώσει την κατανάλωση ενέργειας της συσκευής.

Δίνεται επίσης η δυνατότητα κλήσεων μέσω διαδικτύου σε άλλους χρήστες που διαθέτουν λογαριασμό SIP.

Τέλος, γίνεται χρήση της τεχνολογίας συνδεσιμότητας NFC που έχει ως στόχο την ανάπτυξη τεχνολογιών ανέπαφων πληρωμών την ανταλλαγή πληροφοριών κ.α. [5]



Εικόνα 1-6 GingerBread (2.3)

1.3.7 HoneyComb (3.0 - 3.2)

Είναι μια έκδοση η οποία αφορά τις ταμπλέτες. Παρατηρούμε σημαντικές αλλαγές τόσο στην εμφάνιση όσο και στην δομή του λειτουργικού.

Το κουμπί αναζήτησης και το κουμπί του μενού δίνει την θέση του στο κουμπί των ενεργών εφαρμογών ενώ η αρχική οθόνη είναι πλέον πιο προσαρμόσιμη από τον χρήστη.

Επιπλέον έχουμε τα κουμπιά της αρχικής οθόνης και της επιστροφής που βρίσκονται όλα μαζί στο κάτω μέρος της οθόνης συνθέτοντας μια μπάρα, η μπάρα αυτή (action bar) γίνεται επίσημο πρότυπο σχεδίασης εφαρμογών. [6]

Εικόνα 1-7 HoneyComb (3.0 - 3.2)

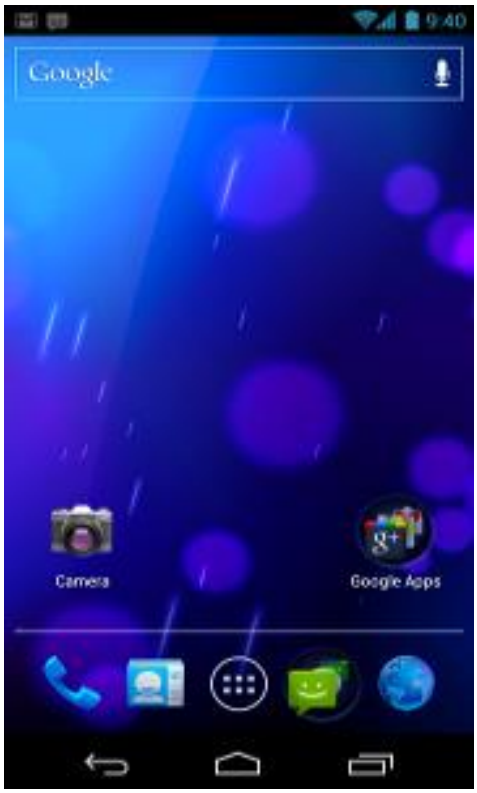
1.3.8 Ice Cream Sandwich (4.0)

Οι αλλαγές που έγιναν στην έκδοση HoneyComb γίνονται διαθέσιμες πλέον και για τα τηλέφωνα.

Στο κάτω μέρος της οθόνης που μέχρι πρότινως υπήρχε μόνο η εφαρμογή κλήσεων και η εφαρμογή του περιηγητή ο χρήστης μπορεί να προσθέσει τις εφαρμογές που επιθυμεί μετατρέποντας την έτσι σε μπάρα αγαπημένων εφαρμογών.

Για πρώτη φορά βλέπουμε και ένα νέο εργαλείο καταμέτρησης των δεδομένων που καταναλώνει η κάθε εφαρμογή ενώ ο χρήστης έχει την δυνατότητα να ορίσει όριο κατανάλωσης δεδομένων πράγμα που είναι ιδιαίτερά χρήσιμο όταν γίνεται παροχή δεδομένων από εταιρία τηλεφωνίας.

Ιδιαιτέρα σημαντική είναι και η χρήση του Wi-Fi Direct οπού οι συσκευές μπορούν να συνδεθούν μεταξύ τους ασύρματα χωρίς την μεσολάβηση κάποιου access point κάνοντας την επικοινωνία πιο αξιόπιστη και ταχύτερη.



Εικόνα 1-8 Ice Cream Sandwich (4.0)

1.3.9 Jelly Bean(4.1-4.2-4.3)

Η αρχική έκδοση Jelly Bean 4.1 παρέχει βελτιωμένη προσβασιμότητα επιτρέποντας στον χρήστη να κάνει περισσότερες ενέργειες με την χρήση αγγίγματός στην οθόνη.

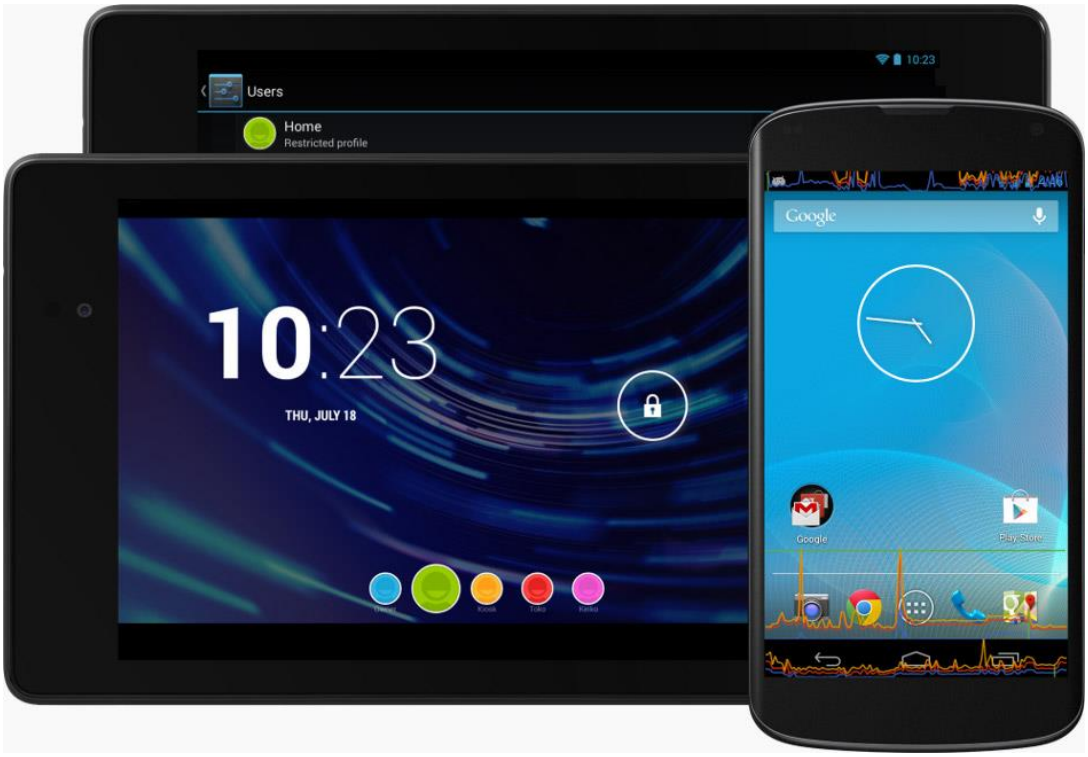
Το πληκτρολόγιο αποκτά την δυνατότητα πρόβλεψης της επόμενης λέξης που θα πληκτρολογήσει ο χρήστης βάση του ιστορικού συγγραφής του.

Δίνεται επίσης έμφαση στα γραφικά στοιχεία όπως την αυτόματη προσαρμογή widget σε κάθε οθόνη, την δυνατότητα να αλλάζει το μέγεθος του widget καθώς και να το αφαιρεί σύροντας το εκτός οθόνης.

Στην επόμενη έκδοση 4.2 εισάγετε το Miracast που κάνει δυνατή την προβολή της οθόνης της συσκευής του κινητού σε εξωτερικές συσκευές, ενώ για πρώτη φορά μπορούν να δημιουργηθούν στην συσκευή περισσότεροι από έναν λογαριασμοί.

Τέλος, έχουμε την τελευταία έκδοση του Jelly Bean 4.3 η οποία υποστηρίζει την βιβλιοθήκη γραφικών OpenGL που θα είναι κομβική για την δημιουργία πιο εξελιγμένων παιχνιδιών. Παράλληλα θα υπάρξει βελτιωμένη συνδεσιμότητα σε συσκευές και αισθητήρες με χαμηλή κατανάλωση ενέργειας χρησιμοποιώντας την τεχνολογία Bluetooth smart.

Βελτιώσεις παρατηρούνται και στις υπηρεσίες τοποθεσίας για τοποθεσίες που παρέχονται από ασύρματα δίκτυα. [7]



Εικόνα 1-9 Jelly Bean(4.1-4.2-4.3)

1.3.10 KitKat (4.4)

Η βασική αλλαγή στην KitKat είναι στις απαιτήσεις του λειτουργικού ως προς το υλικό που χρειάζεται με στόχο το λειτουργικό να είναι αποδοτικότερο πιο γρήγορο και να χρησιμοποιεί λιγότερους πόρους ούτος ώστε οι κατασκευαστές να μπορούν να ενημερώνουν και παλαιότερες συσκευές με την έκδοση αυτή.

Σημαντικές αλλαγές πραγματοποιούνται και στην εμφάνιση, σημαντικότερης εξ' αυτών στην εφαρμογή κλήσεων που πλέον μπορείς να κάνεις με ευκολία και ταχύτητα αναζήτηση στις επαφές αλλά ακόμα και σε Google Apps accounts.

Βελτίωση παρατηρείται και στην υπηρεσία φωνητικών αναζητήσεων ως προς την ακρίβεια, ενώ το Google Now ενεργοποιείται με ένα άγγιγμα από τα αριστερά στα δεξιά στην αρχική σελίδα. [8]

Εικόνα 1-10 KitKat (4.4)

1.3.11 Lollipop (5.0)

Η πρώτη σημαντική αλλαγή είναι στις διεργασίες εκτέλεσης των εφαρμογών που το Dalvik δίνει την θέση του στο Android runtime. Πλέον ο κώδικας εκτέλεσης της εφαρμογής δεν εκτελείται κάθε φορά που ο χρήστης προσπαθεί να ανοίξει την εφαρμογή αλλά μία φόρα κατά την εγκατάσταση. Έτσι η εκτέλεση των εφαρμογών γίνεται πολύ πιο γρήγορα.

Το περιβάλλον του χρήστη είναι επανασχεδιασμένο βασισμένο γύρω από μία σχεδιαστική γλώσσα γνωστή ως Material Desing η οποία αλλάζει την εμφάνιση των εφαρμογών του Android, όπως του GMAIL, του YOUTUBE και του GOOGLE MAPS.

Ακόμα, το LOLLIPOP παρέχει συμβατότητα σε πολλές νέες συσκευές όπως τηλεοράσεις, αυτοκίνητα και smart-watch.

Επιπλέον, επαναδημιουργούνται τα εικονίδια, τα animation και το multitasking menu. Η οθόνη κλειδώματος γίνεται πολύ πιο χρήσιμη για τον χρήστη μιας και μπορούν να διαχειριστούν και να επιλέξουν τις ειδοποιήσεις που θέλουν να λαμβάνουν και ποιες θέλουν να αποκρύψουν. [9]

Εικόνα 1-11 Lollipop (5.0)

1.3.12 Marshmallow 6.0

Σε αυτή την έκδοση υπάρχει η βελτίωση ότι ο χρήστης έχει καλύτερο έλεγχο και καλύτερη διαχείριση δικαιωμάτων των εφαρμογών κατά τον χρόνο εκτέλεσης τους. Με αυτόν τον τρόπο ο χρήστης μπορεί να επιλέγει ποια δεδομένα και ποιους αισθητήρες μπορούν να χρησιμοποιούν οι εκάστοτε εφαρμογές του.

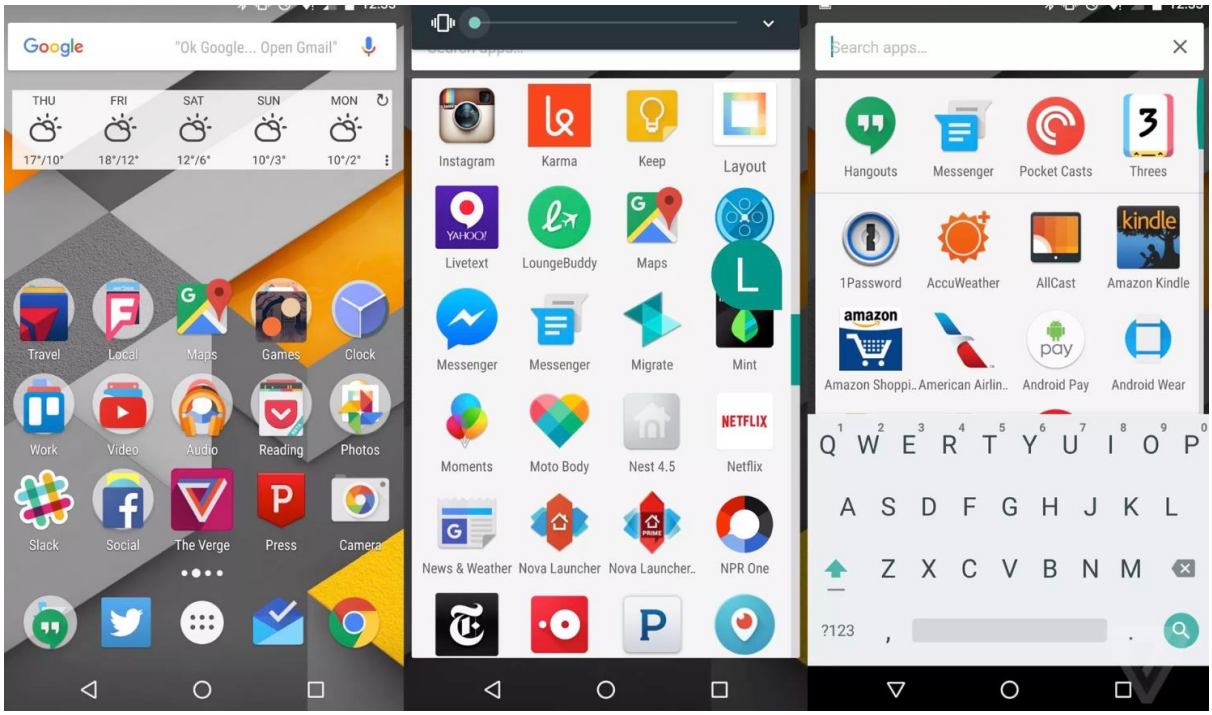
Σημαντική είναι ακόμα η παροχή εξοικονόμησης ενέργειας όσον αφορά την συσκευή και τις εφαρμογές όταν βρίσκεται σε αδράνεια. Όταν ο χρήστης αφήσει την συσκευή του για ένα χρονικό διάστημα τότε η συσκευή μπαίνει σε αδράνεια (doze). Σε αυτή την λειτουργεία θα μείνουν ενεργές μόνο οι βασικές λειτουργείες και θα ενεργοποιηθούν και πάλι όταν έρθει κάποια ειδοποίηση. [10]

Μία ακόμη χρήσιμη λειτουργεία για την εξοικονόμηση ενέργειας είναι η εφαρμογή αναμονής. Αυτή επιτρέπει στο σύστημα να καταλάβει ότι μία εφαρμογή είναι σε αδράνεια όταν ο χρήστης δεν την χρησιμοποιεί. Με αυτόν το τρόπο αναστέλλονται ο συγχρονισμός και οι εργασίες της εφαρμογής.

Σε αυτή την έκδοση υπάρχει προ εγκατεστημένο το android pay που επιτρέπει για πρώτη φορά πραγματοποίηση πληρωμών μέσω αναγνώρισης δακτυλικών αποτυπωμάτων.

Η Google αναβαθμίζει την λειτουργεία του google now δίνοντας την δυνατότητα στους χρήστες να την χρησιμοποιούν την στιγμή που το χρειάζονται. Ο χρήστης μπορεί να ζητήσει βοήθεια από το Google Now, κρατώντας πατημένο το home button, χωρίς να φύγει από την εφαρμογή και να του εμφανιστούν όλες οι πληροφορίες εκεί.

Τέλος, παρέχεται η υποστήριξη usb type C, η δυνατότητα ρύθμισης του ήχου σε διαφορετικά επίπεδα για multitasking (ήχος κλήσης, ήχος εφαρμογής, ήχος ειδοποιήσεων), νέα λίστα εμφάνισης της κατανάλωσης μνήμης RAM για κάθε εφαρμογή. [11]

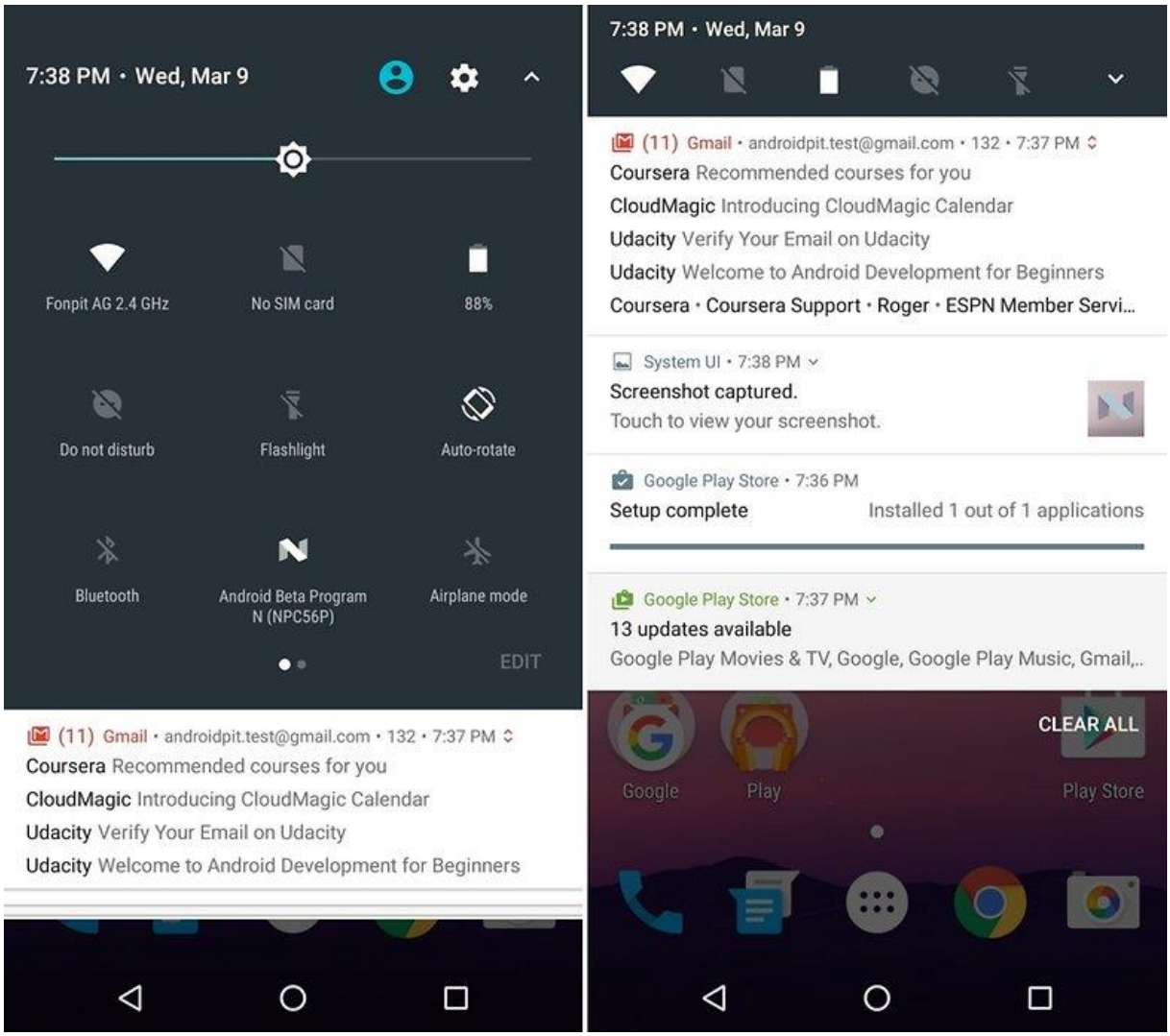


Εικόνα 1-12 Marshmallow 6.0

1.3.13 Nougat 7.0, 7.1

Ένα από τα βασικότερα χαρακτηριστικά αυτής της έκδοσης είναι η δυνατότητα να εκτελούνται πολλές εφαρμογές στην ίδια οθόνη (multitasking). Επιπλέον ο χρήστης μπορεί να αλλάζει το μέγεθος της εφαρμογής όταν δυο εφαρμογές βρίσκονται σε λειτουργεία split screen. [12]

Όσον αφορά την περιοχή των ειδοποιήσεων με τον επανασχεδιασμό που έγινε υπάρχουν μεγάλες βελτιώσεις. Σέρνοντας από την κορυφή της οθόνης προς τα κάτω εμφανίζονται οι ειδοποιήσεις, ενώ επαναλαμβάνοντας την κίνηση για δεύτερη φορά εμφανίζονται οι βασικές γρήγορες ρυθμίσεις, όπως Wi-Fi, Bluetooth, ηχητικά προφίλ, φωτεινότητα κλπ. Μία ακόμα δυνατότητα είναι η αναδιάταξη των εικονιδίων στις γρήγορες ρυθμίσεις [13]



Εικόνα 1-13 Nougat 7.0, 7.1

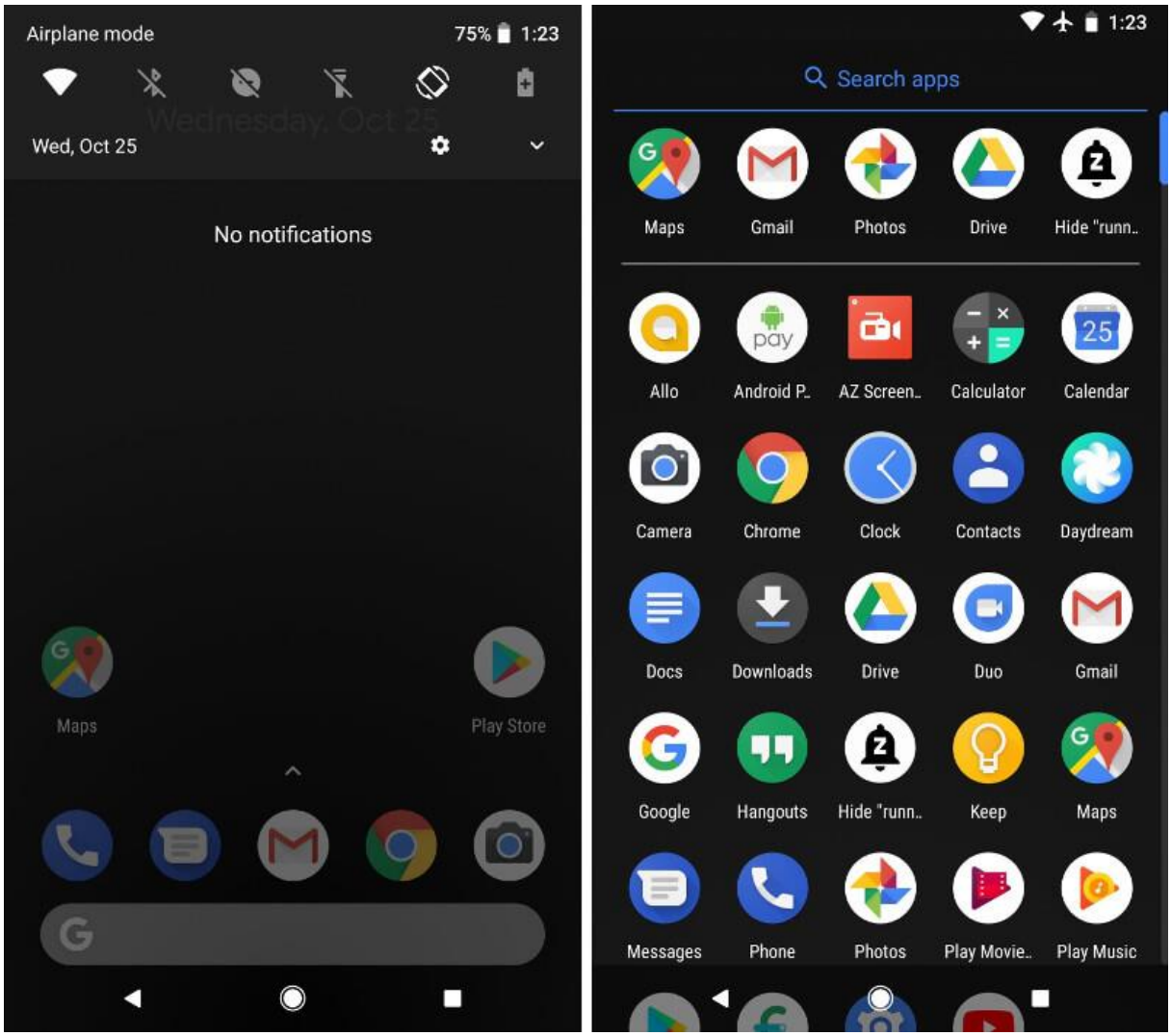
1.3.14 Oreo 8.0, 8.1

Στην αρχική έκδοση παρατηρείτε σημαντική βελτίωση μέσω της επανασχεδίασης των ειδοποιήσεων. Εισάγοντας ομαδοποιημένα κατηγορίες ειδοποιήσεων, ακόμη εισάγονται οι κουκίδες ειδοποιήσεων στα εικονίδια εκκίνησης των εφαρμογών που περιλαμβάνουν ενημερώσεις που ο χρήστης δεν έχει κάνει ακόμα κάποια ενέργεια. Επίσης, ο χρήστης μπορεί να καθορίζει χρονικά όρια για τις ειδοποιήσεις αλλά και να τις απορρίπτει.

Ακόμα γίνεται διαθέσιμο το πλαίσιο αυτόματης συμπλήρωσης. Ο χρήστης διευκολύνεται στην αυτόματη συμπλήρωση σε φόρμες που χρησιμοποιεί συχνά. Παρέχει νέα emoji και νέα εικονίδια εφαρμογών βελτιώνοντας σημαντικά την αισθητική. Υπάρχει δυνατότητα ένα video να παίζει σε ένα μέρος της οθόνης ενώ παράλληλα εκτελείται κάποια άλλη λειτουργεία. [14]

Τέλος, υποστηρίζει το πρωτόκολλο Wi-Fi Aware για την επικοινωνία των συσκευών χωρίς να χρειάζεται σύνδεση στο διαδίκτυο.

Στην έκδοση 8.1 υποστηρίζεται ένα API νευρωνικού δικτύου το οποίο είναι σχεδιασμένο ώστε να επιτρέπει την μηχανική μάθηση σε κινητές συσκευές. [15]

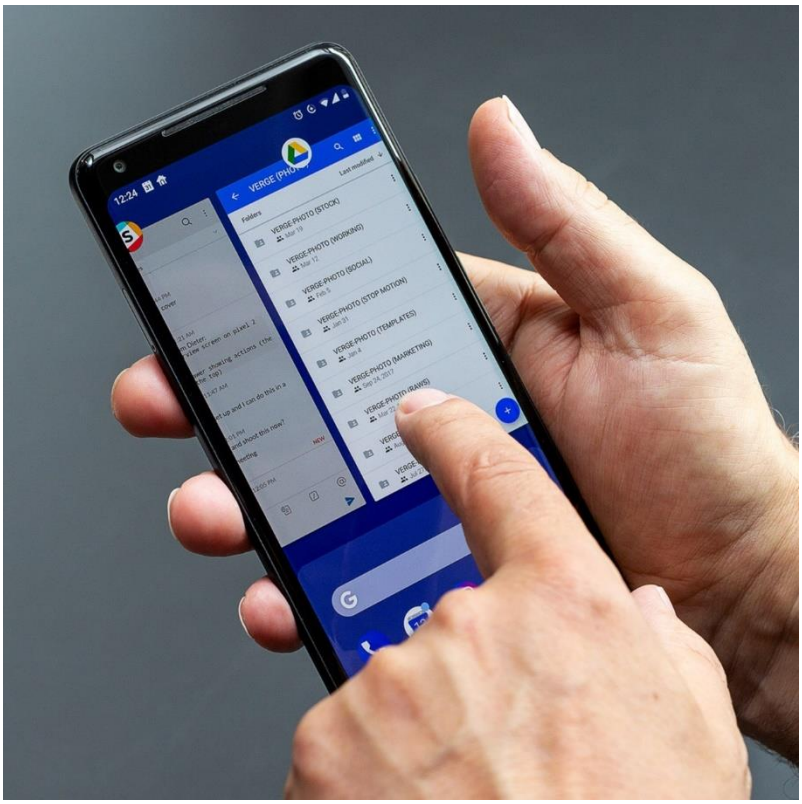


Εικόνα 1-14 Oreo 8.0, 8.1

1.3.15 Pie 9.0

Στην έκδοση Pie ο χρήστης συναντάει την περιήγηση με swipes. Ο χρήστης μπορεί με swipe από κάτω προς το μέσο της οθόνης να μπει άμεσα στο multitasking μενού και να περιηγηθεί στις ανοιχτές εφαρμογές της συσκευής του για να επιλέξει αυτή που θέλει. Στην ίδια οθόνη, εμφανίζονται συντομεύσεις για εφαρμογές που χρησιμοποιεί συχνά ώστε να μπορεί να έχει άμεση πρόσβαση. Με swipe μέχρι το άνω μέρος της οθόνης, πηγαίνει στο μενού εφαρμογών που είναι εγκατεστημένες στη συσκευή του. [16]

Επιπλέον χαρακτηριστικά είναι το κουμπί λήψης στιγμιότυπου όπως και η περιστροφή οθόνης που πλέον γίνεται και χειροκίνητα με ένα απλό πάτημα στον αντίστοιχό δείκτη.



Εικόνα 1-15 Pie 9.0

1.3.16 Q 10

Η έκδοσή αυτή έγινε διαθέσιμη τον Σεπτέμβριο του 2019. Για πρώτη φορά γίνεται διαθέσιμη η σκοτεινή λειτουργεία (Dark Mode) μέσω της οποίας ο χρήστης θα μπορεί να θέτει τις εφαρμογές του σε dark mode ενεργοποιώντας την δυνατότητα αυτή μεσώ των ρυθμίσεων προβολής ή μέσω της εξοικονόμησης ενέργειας.

Ακόμη προστίθενται νέες χειρονομίες όπως το σύρσιμο του δαχτύλου προς τα πάνω για επιστροφή στηνα ρχική οθόνη, το παρατεταμένο σύρσιμο προς τα πάνω για το άνοιγμα της λειτουργείας πολλαπλών εφαρμογών και το σύρσιμο αριστερά δεξιά για τις επιλογές πίσω ή μπροστά.

Σημαντική βελτίωση παρατηρείται ακόμη στην ενίσχυση της προστασίας απορρήτου και στην παροχή νέων αδειών. Δίνεται η επιλογή στον χρήστη οι εφαρμογές να έχουν πρόσβαση στην τοποθεσία ενώ εκτελούνται στο παρασκήνιο δηλαδή η τοποθεσία σας θα ενεργοποιείται μόνο ενώ χρησιμοποιείτε την εφαρμογή. Υπάρχει επίσης μία νέα σελίδα απορρήτου στις ρυθμίσεις μέσω της οποίας θα μπορεί να δει ο χρήστης ποιες εφαρμογές έχουν πρόσβαση σε πληροφορίες καθώς επίσης θα μπορεί να κάνει αλλαγή όπου επιθυμεί. [17]

Ένα νέο χαρακτηριστικό είναι αυτό των (φυσαλλίδων Bubbles) είναι μία νέα μέθοδος πλοήγησης μεταξύ πολλαπλών εφαρμογών, η οποία επιτρέπει την ελαχιστοποίηση οποιασδήποτε εφαρμογής σε μία μικρή φυσαλίδα.

Τέλος, γίνεται δυνατή η βιντεοσκόπηση της οθόνης ενώ οι λήψη ενημερώσεις στο λογισμικό γίνονται χωρίς να γίνεται επανεκκίνηση της συσκευής και σε πιο γρήγορο χρόνο.

Εικόνα 1-16 Q 10

1.3.17 R 11.0 Beta

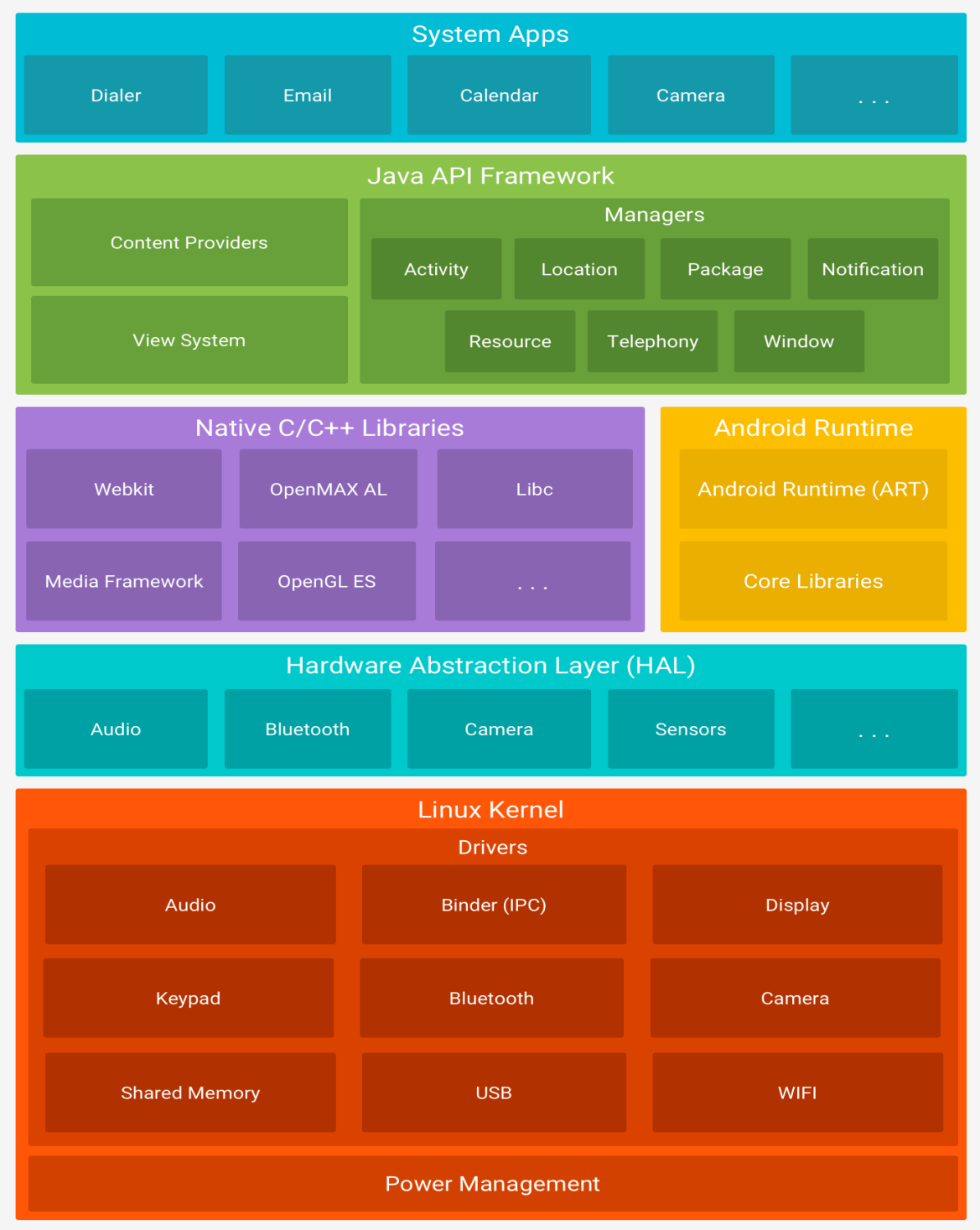
Η έκδοση αυτή αποτελεί την νεότερη έκδοση android και κυκλοφόρησε στις 19 Φεβρουαρίου του τρέχοντος έτους. Ορισμένα νέα χαρακτηριστικά είναι για παράδειγμα ένα νέο εικονίδιο εγγραφής της οθόνης αλλά και η σίγαση των ειδοποιήσεων κατά την διαρκεία της εγγραφής βίντεο.

Σημαντική θα είναι η ανάκληση της άδειας σε εφαρμογές, η ενεργοποίηση αυτής της επιλογής να ανακαλεί αυτόματα όλα τα δικαιώματα που έχει κάποια εφαρμογή εάν δεν την έχετε χρησιμοποιήσει για μεγάλο χρονικό διάστημα. Με το Android 11 θα μπορεί να εγκρίνει τα δικαιώματα μόνο μία φορά και το λειτουργικό σύστημα θα ανακαλέσει την άδεια αργότερα. [18]

Εικόνα 1-17 R 11.0 Beta

1.4 Αρχιτεκτονική του Android

Το Android είναι μια στοίβα λογισμικού ανοιχτού κώδικα, βασισμένη στο Linux, που δημιουργήθηκε για ένα ευρύ φάσμα συσκευών. Η αρχιτεκτονική της περιλαμβάνει τέσσερα βασικά επίπεδα τον πυρήνα linux (Linux Kernel), τις βιβλιοθήκες (Libraries), το πλαίσιο εφαρμογής (Application Framework) και το επίπεδο εφαρμογών (Application).



Εικόνα 1-18 Η στοίβα λογισμικού Android

1.4.1 Πυρήνας Linux (Linux Kernel)

Ο πυρήνας Linux βρίσκεται στην βάση της αρχιτεκτονικής του Android. Ο πυρήνας παρέχει προγράμματα οδήγησης χαμηλού επιπέδου για τα στοιχεία του υλικού, διαχειρίζεται την πρόσβαση στην μνήμη, τις διαδικασίες της CPU, τη δικτύωση και την ενέργεια. [19]

1.4.2 Επίπεδο αφαίρεσης υλικού (HAL)

Παρέχει διεπαφές που υποστηρίζουν τις δυνατότητες των υλικών που περιέχει μια συσκευή όπως κάμερα, ηχεία, Bluetooth, αισθητήρες κ.α.. Αποτελείται από διάφορες βιβλιοθήκες που κάθε μια από αυτές υλοποιεί ένα interface για έναν συγκεκριμένο τύπο υλικού. Όταν μία εφαρμογή κάνει μια κλήση για πρόσβαση στο υλικό της συσκευής, το σύστημα Android φορτώνει τη λειτουργική μονάδα της κατάλληλης βιβλιοθήκης για το εν λόγω στοιχείο υλικού. [19]

1.4.3 Χρόνος εκτέλεσης Android (Android Runtime)

Για συσκευές με έκδοση Android 5.0 (επίπεδο 21 API) ή υψηλότερη, κάθε εφαρμογή εκτελείται με τη δική της διαδικασία και με το δικό της στιγμιότυπο στο Android Runtime (ART). Περιλαμβάνει βιβλιοθήκες γραμμένες σε γλώσσα προγραμματισμού Java. Το ART είναι γραμμένο για να εκτελεί πολλές εικονικές μηχανές σε συσκευές χαμηλής μνήμης εκτελώντας αρχεία DEX, μια μορφή bytecode που έχει σχεδιαστεί ειδικά για Android και βελτιστοποιείται για ελάχιστο αποτύπωμα μνήμης. Μέχρι και την έκδοση 5.0 την θέση του Android Runtime (ART) κατείχε το Dalvik Virtual Machine. Το Android περιλαμβάνει επίσης ένα σύνολο βασικών βιβλιοθηκών χρόνου εκτέλεσης (runtime libraries) που παρέχουν το μεγαλύτερο μέρος της λειτουργικότητας της γλώσσας προγραμματισμού Java. [19]

1.4.4 Εγγενείς βιβλιοθήκες C / C ++ (Native C/C++ Libraries)

Υπάρχει ένα σύνολο βιβλιοθηκών γραμμένων σε C/C++ οι οποίες χρησιμοποιούνται από διάφορα στοιχεία του συστήματος του Android από τα υψηλότερα επίπεδα όπως HAL, ART, Java OpenGL API για χρήση της βιβλιοθήκης OpenGL ES για χρήση 2D και 3D γραφικών, Android NDK για δημιουργία νέων βιβλιοθηκών. [19]

1.4.5 Java API Framework

Όλο το set των δυνατοτήτων και των χαρακτηριστικών του Android είναι διαθέσιμο στον προγραμματιστή μέσω Apis γραμμένα σε Java. Τα API σχηματίζουν τα δομικά στοιχεία που χρειάζονται για να δημιουργηθεί μια εφαρμογή Android επαναχρησιμοποιώντας τα βασικά και δομικά στοιχεία όπως και τις υπηρεσίες. Βασικότερα συστατικά και υπηρεσίες είναι τα παρακάτω:

- **View System:** προσφέρει τα απαραίτητα στοιχεία γραφικών στις διέπαφες (π.χ. lists, grids, text boxes, buttons).
- **Content Provider:** υποστηρίζει, με τις κατάλληλες διαδικασίες, τη δυνατότητα των εφαρμογών να έχουν πρόσβαση σε δεδομένα από άλλες εφαρμογές ή να μοιράζουν τα δικά τους δεδομένα.

- **Resource Manager:** διαχειρίζεται την πρόσβαση σε πόρους της συσκευής εκτός των αρχείων κώδικά.
- **Notification Manager:** ελέγχει και διαχειρίζεται τα μηνύματα και τις ειδοποιήσεις που διακινούνται κατά τη λειτουργία μιας εφαρμογής στην γραμμή κατάστασης.
- **Activity Manager:** ελέγχει τις διαδικασίες της εφαρμογής κατά τη λειτουργία της και υποστηρίζει την ομαλή μετάβαση από τη μία οθόνη της εφαρμογής στην άλλη.
- **Location Manager:** ανιχνεύει και καταγράφει την ακριβή γεωγραφική τοποθεσία της συσκευής. [19]

1.4.6 Εφαρμογές συστήματος (System Apps)

Στην κορυφή της ιεραρχίας βρίσκεται ένα σύνολο βασικών εφαρμογών όπως το email, τα sms, τα calendars, οι browser και τα contacts. Οι εφαρμογές αυτές δεν έχουν διάκριση σε σχέση με αυτές που εγκαθιστά ο χρήστης, με αυτόν τον τρόπο εφαρμογές τρίτων μπορούν να γίνουν προκαθορισμένες με επιλογή του χρήστη όπως για παράδειγμα το πληκτρολόγιο. Εξαίρεση σε αυτό αποτελούν οι ρυθμίσεις που έχουν οι εφαρμογές. [20]

1.5 Βασικά συστατικά μιας εφαρμογής

Τα app component (συστατικά εφαρμογής) είναι τα βασικά δομικά στοιχεία μιας εφαρμογής Android. Κάθε συστατικό είναι ένα σημείο εισόδου που μέσω αυτού το σύστημα η κάποιος χρήστης μπορεί να εισέλθει στην εφαρμογή. Τα βασικά συστατικά είδη είναι τέσσερα:

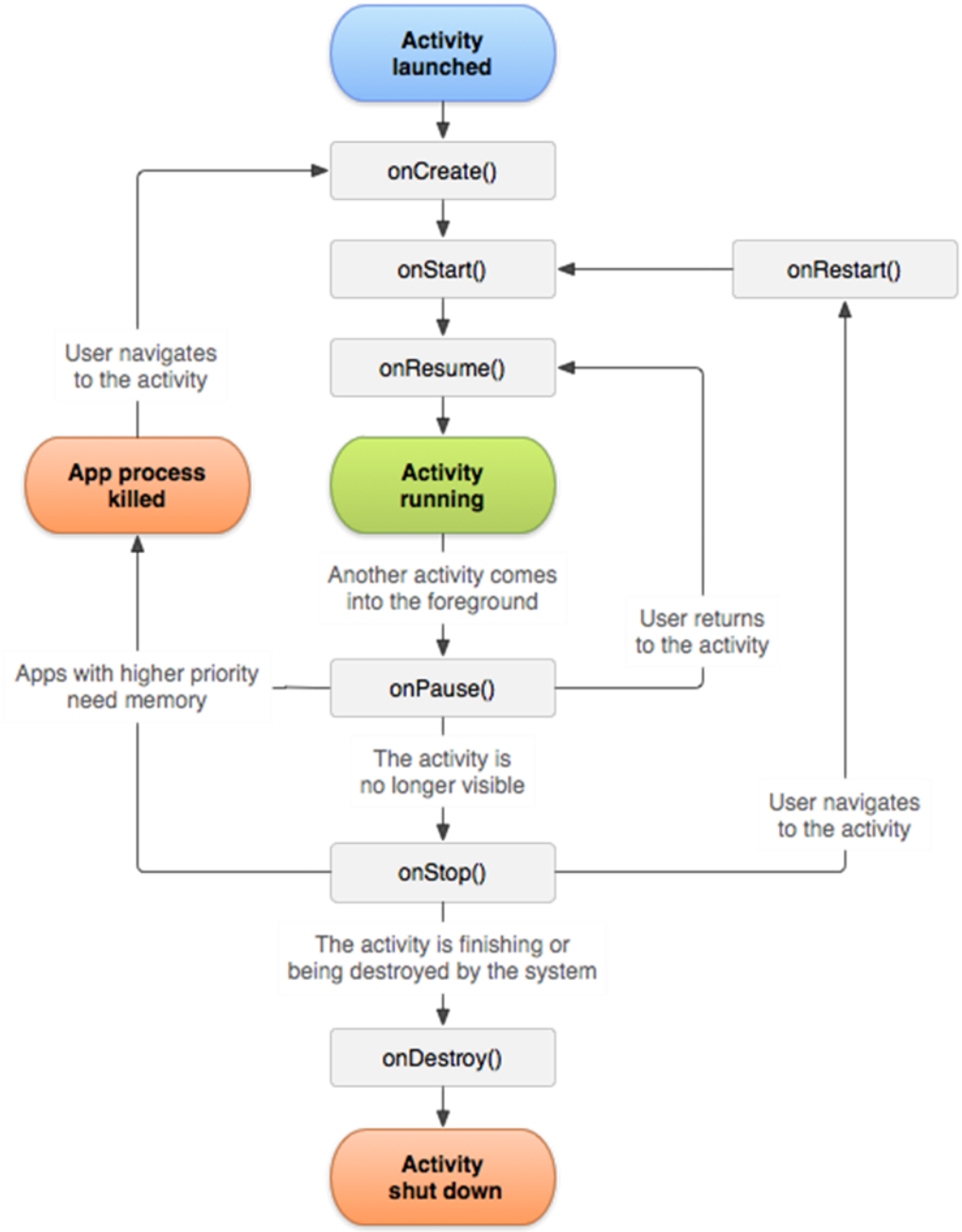
- Activities
- Services
- Broadcast receivers
- Content providers

Κάθε τύπος εξυπηρετεί έναν ξεχωριστό σκοπό και έχει έναν ξεχωριστό κύκλο ζωής που καθορίζει τον τρόπο με τον οποίο δημιουργείται και καταστρέφεται ένα συστατικό. [19] [21]

1.5.1 Activity

Μία δραστηριότητα (Activity) είναι το σημείο εισόδου για την αλληλεπίδραση με τον χρήστη και αντιπροσωπεύει μια και μόνο μία οθόνη. Διαχειρίζεται την διά δράση που έχει η εφαρμογή με τον χρήστη, την επεξεργασία των δεδομένων και τις υπηρεσίες που παρέχει. Μία εφαρμογή μπορεί να έχει ένα ή πολλά περισσότερα activities για να διαχειρίζεται διάφορες φάσεις κατά την εκτέλεση της. Κάθε δραστηριότητα είναι ανεξάρτητη από τις άλλες, ενώ μπορεί να χρησιμοποιηθεί και από κάποια άλλη εφαρμογή αν είναι επιτρεπτό από την αρχική εφαρμογή της. Κάθε activity έχει τον δικό της κύκλο ζωής.

Για να δημιουργηθεί ένα Activity αρκεί μια κλάση να επεκτείνει την Activity και να ορίσει οπωσδήποτε την onCreate() μέθοδο η οποία αναπαριστά το στάδιο onCreate στο κύκλο ζωής. Σημαντική κλήση που θα πρέπει να οριστεί στην onCreate() είναι η μέθοδος setContentView() με την οποία ορίζεται η διάταξη (layout) της διεπιφάνειας χρήστη που θα χρησιμοποιηθεί για το συγκεκριμένο Activity. [22]



Εικόνα 1-19 Κύκλος ζωής activity

1.5.2 Υπηρεσίες (Services)

Είναι ένα συστατικό το όποιο εκτελείται στο παρασκήνιο για να πραγματοποιήσει ενέργειες μεγάλης διάρκειας καθώς και απομακρυσμένες διεργασίες. Μία υπηρεσία δεν παρέχει γραφική αναπαράσταση σε οθόνη. Για παράδειγμα μια υπηρεσία μπορεί να αναπαράγει μουσική στο παρασκήνιο ενώ ο χρήστης βρίσκεται σε αλληλεπίδραση με μια διαφορετική εφαρμογή ή να κάνει λήψη δεδομένων από το διαδίκτυο χωρίς να επηρεάζει τον χρήστη από μια άλλη δραστηριότητα. Υπάρχουν δύο περιπτώσεις διαχείρισης των υπηρεσιών η μία περίπτωση είναι να τρέχει η υπηρεσία στο παρασκήνιο και να το γνωρίζει ο χρήστης σε αυτήν την περίπτωση ο χρήστης έχει τη δυνατότητα να την σταματήσει. Στην άλλη περίπτωση μια υπηρεσία τρέχει στο παρασκήνιο αλλά ο χρήστης δεν το γνωρίζει και το σύστημα

έχει την ευχέρεια να την διαχειριστεί. Μια υπηρεσία υλοποιείται ως υπο κλάση της κλάσης Service. [19]

1.5.3 Broadcast receivers

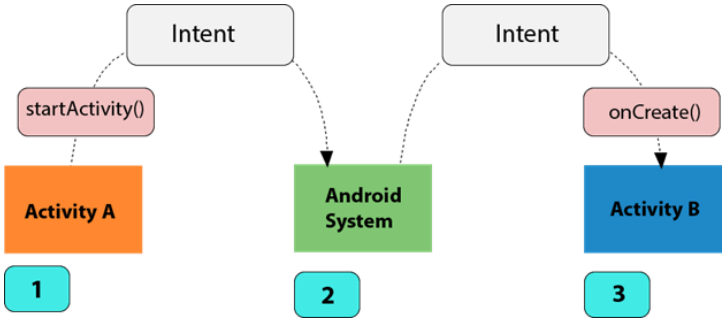
Ένα broadcast receiver είναι ένα συστατικό που επιτρέπει στο σύστημα να παραδίδει συμβάντα στην εφαρμογή εκτός από μια κανονικής ροής χρηστών, επιτρέποντας στην εφαρμογή να ανταποκρίνεται σε ανακοινώσεις μετάδοσης σε όλο το σύστημα. Οι ανακοινώσεις αυτές μπορεί να είναι είτε από το σύστημα όπως για παράδειγμα όταν το σύστημα ενημερώνει ότι η στάθμη της μπαταρίας είναι χαμηλή ή να είναι από τον χρήστη όπως το ξεκίνημα μίας διαδικασίας. Το σύστημα τέλος μπορεί να στείλει ανακοινώσεις και σε εφαρμογές που δεν είναι απαραίτητο να εκτελούνται. Ένα broadcast receiver υλοποιείται ως υπο κλάση της κλάσης Broadcast Receiver. [19]

1.5.4 Content providers

Ένας content provider διαχειρίζεται ένα σύνολο από δεδομένα εφαρμογών που μπορούν να αποθηκευτούν στο σύστημα αρχείων της συσκευής σε μία SQLite βάση δεδομένων, στη Firebase στο web ή σε οποιοδήποτε είδος μόνιμης αποθήκευσης δεδομένων που μπορεί η εφαρμογή να έχει πρόσβαση. Άλλες εφαρμογές έχουν την δυνατότητα μέσω του content provider αν το επιτρέπει να διαχειριστούν δεδομένα. Αντιστοιχίζεται σε ένα URI scheme και μία εφαρμογή αντιστοιχεί τα δεδομένα μέσω ενός URI namespace. [19]

1.6 Ο μηχανισμός Intent

Τρείς από τους τέσσερις τύπους συστατικών στο Android (Activities, Services, Broadcast receivers) ενεργοποιούνται από ένα ασύγχρονο μήνυμα που ονομάζεται intent. Ένα intent ζητάει μέσω ενός μηνύματος να γίνει μια ενέργεια που επιθυμεί ο χρήστης. Για τις δραστηριότητες και τις υπηρεσίες ένα intent καθορίζει την δράση που πρέπει να γίνει και το URI των δεδομένων στα οποία θα γίνει μια ενέργεια. Για παράδειγμα, μια πρόθεση μπορεί να μεταφέρει ένα αίτημα για μια δραστηριότητα να εμφανίσει μια εικόνα ή να ανοίξει μια ιστοσελίδα. Για τους Broadcast Receiver καθορίζει το περιεχόμενο του μηνύματος που θα εμφανιστεί. Για παράδειγμα, μια εκπομπή που υποδεικνύει ότι η μπαταρία της συσκευής είναι χαμηλή περιλαμβάνει μόνο μια γνωστή συμβολοσειρά δράσης που δείχνει ότι η μπαταρία είναι χαμηλή. Ένα activity κάνει κλήση ενός άλου activity με την εντολή startActivity().



Εικόνα 1-20 Ο μηχανισμός Intent

1.7 Content Resolver

Σέ αντίθεση με τα άλλα βασικά συστατικά οι content providers ενεργοποιούνται από ένα ContentResolver που χειρίζεται όλες τις άμεσες συναλλαγές με τον content provider έτσι ώστε το στοιχείο που εκτελεί συναλλαγές με τον πάροχο να μην χρειάζεται και να καλεί μεθόδους στο ContentResolver αντικείμενο.

1.8 Android Manifest

Πριν το σύστημα Android μπορεί να ξεκινήσει ένα συστατικό μίας εφαρμογής, το σύστημα ελέγχει ότι έχει δηλωθεί στο αρχείο δήλωσης της εφαρμογής το AndroidManifest.xml. Όλα τα συστατικά θα πρέπει να δηλωθούν σε αυτό το αρχείο το όποιο θα πρέπει να βρίσκεται στη ρίζα του καταλόγου του έργου εφαρμογής. Το Manifest είναι σημαντικό για πολλές ενέργειες πέραν της δήλωσης των στοιχείων της εφαρμογής όπως:

- Προσδιορισμός δικαιωμάτων χρήστη που απαιτεί η εφαρμογή, όπως για παράδειγμα πρόσβαση στο διαδίκτυο.
- Καθορισμός του ελάχιστου επίπεδου API που απαιτεί η εφαρμογή.
- Δήλωση δυνατοτήτων υλικού και λογισμικού που απαιτούνται από την εφαρμογή (Κάμερα, Bluetooth).
- Λίστα Βιβλιοθηκών στις οποίες πρέπει να συνδεθεί η εφαρμογή.

Ο κύριος σκοπός του manifest είναι να ενημερώσει το σύστημα για τα στοιχεία της εφαρμογής. Πρέπει να δηλωθούν όλα τα στοιχεία της εφαρμογής χρησιμοποιώντας τους παρακάτω κόμβους:

- <activity>, στοιχεία για δραστηριότητες
- <service>, στοιχεία για υπηρεσίες.
- <receiver>, στοιχεία για δέκτες εκπομπής.
- <provider>, στοιχεία για παρόχους περιεχομένου.

[19]

1.9 Δήλωση Συστατικών

Αν τα βασικά συστατικά activities, services και content providers που περιλαμβάνονται στον πηγαίο κώδικα δεν έχουν δηλωθεί στο manifest τότε δεν θα είναι ορατά από το σύστημα με αποτέλεσμα να μην μπορούν να εκτελεστούν πράγμα το οποίο δεν ισχύει για τους broadcast receivers που μπορούν να δηλωθούν στο manifest, ενώ εναλλακτικά μπορούν να δημιουργηθούν δυναμικά στον κώδικα ως αντικείμενα BroadcastReceiver και να εγγράφουν στο σύστημα με κλήση της μεθόδου registerReceiver().

1.10 Δήλωση πόρων

Για κάθε πόρο που συμπεριλαμβάνετε στο android project, αντιστοιχίζεται ένα μοναδικό αναγνωριστικό (ID) ακέραιου αριθμού, το οποίο μπορεί να χρησιμοποιηθεί για να γίνει αναφορά στον πόρο από τον κώδικα της εφαρμογής ή από άλλους πόρους που ορίζονται στη XML. Για παράδειγμα, εάν η εφαρμογή περιέχει ένα αρχείο εικόνας με όνομα logo.png (αποθηκευμένο στον res/drawable/κατάλογο), τα

εργαλεία SDK δημιουργούν ένα αναγνωριστικό πόρου με όνομα R.drawable.logo. Αυτό το αναγνωριστικό αντιστοιχεί σε έναν ακέραιο αριθμό συγκεκριμένης εφαρμογής, τον οποίο είναι δυνατόν να χρησιμοποιηθεί για να γίνει αναφορά στην εικόνα και να τοποθετηθεί στη διεπαφή χρήστη.

Μία σημαντική δυνατότητα της παροχής πόρων είναι η παροχή εναλλακτικών πόρων για συσκευές διαφορετικής διαμόρφωσης. Για παράδειγμα, ορίζοντας τις συμβολοσειρές UI σε XML, είναι δυνατόν να οριστούν οι συμβολοσειρές σε πολλές γλώσσες και να αποθηκευτούν αυτές οι συμβολοσειρές σε ξεχωριστά αρχεία. Στη συνέχεια, το android εφαρμόζει τις κατάλληλες συμβολοσειρές γλώσσας στο UI με βάση ένα qualifier γλώσσας που προστίθεται στο όνομα του καταλόγου πόρων και τη ρύθμιση γλώσσας του χρήστη. Ο qualifier είναι μια σύντομη συμβολοσειρά που συμπεριλαμβάνετε στο όνομα των καταλόγων πόρων για να οριστεί η διαμόρφωση της συσκευής για την οποία πρέπει να χρησιμοποιηθούν οι πόροι αυτοί. Το android υποστηρίζει πολλά διαφορετικά qualifier για τους εναλλακτικούς πόρους μιας εφαρμογής.

1.11 Επίλογος

Στο κεφάλαιο αυτό έγινε μια εισαγωγή στον κόσμο του λειτουργικού συστήματος Android με μια σύντομη ανάδρομή στην εξέλιξη του καθώς και στην αρχιτεκτονική τα συστατικά και τα χαρακτηριστικά που το συνοδεύουν.

Κεφάλαιο 2ο: Android Studio

2.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μια εισαγωγή στο περιβάλλον του Android Studio, παρουσιάζοντας την δομή ενός έργου καθώς επίσης και τα κύρια χαρακτηριστικά της διεπαφής χρήστη.

2.2 Εισαγωγή στο Android Studio

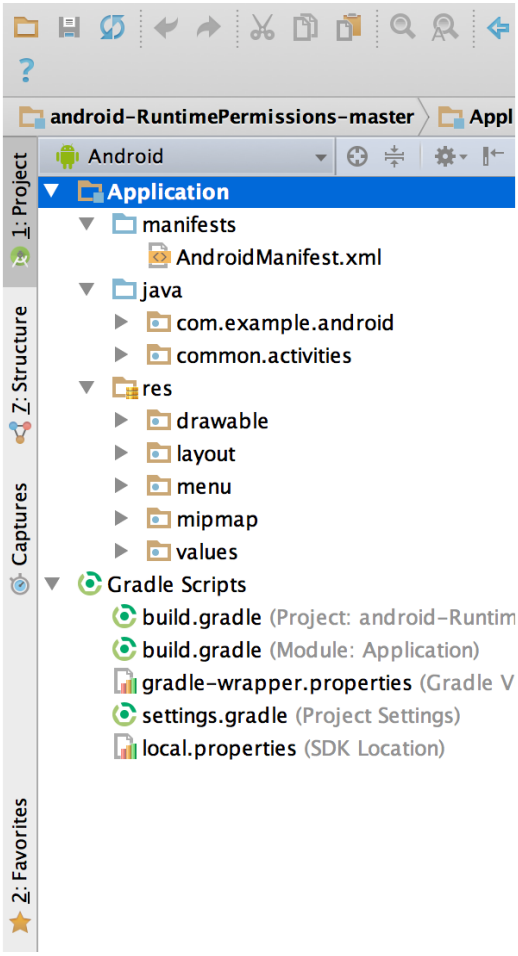
Το Android Studio είναι το επίσημο Περιβάλλον Ολοκληρωμένης Ανάπτυξης (IDE) για ανάπτυξη εφαρμογών android, με βάση το IntelliJ IDEA. Πέρα από τον ισχυρό επεξεργαστή κώδικα της IntelliJ και τα εργαλεία προγραμματιστών, το Android Studio προσφέρει ακόμη περισσότερες δυνατότητες που βελτιώνουν την παραγωγικότητά σας κατά τη δημιουργία εφαρμογών android, όπως: [23]

- Ένα ευέλικτο σύστημα κατασκευής με βάση το Gradle.
- Ένας γρήγορος και πλούσιος σε χαρακτηριστικά εξομοιωτής.
- Ένα ενιαίο περιβάλλον όπου μπορείτε να αναπτύξετε για όλες τις συσκευές Android.
- Πρότυπα κώδικα και χρήση του GitHub για τη δημιουργία βιβλιοθηκών και την εισαγωγή sample code.
- Ειδικά εργαλεία για debug και test.
- Υποστήριξη C ++ και NDK.
- Ενσωματωμένη υποστήριξη για την πλατφόρμα Google Cloud.

2.3 Δομή έργου

Κάθε έργο στο Android Studio περιέχει μία ή περισσότερες ενότητες με αρχεία πηγαίου κώδικα και αρχεία πόρων. Οι τύποι των ενοτήτων περιλαμβάνουν:

- Ενότητες εφαρμογών Android.
- Ενότητες βιβλιοθήκης.
- Ενότητες Google App Engine.

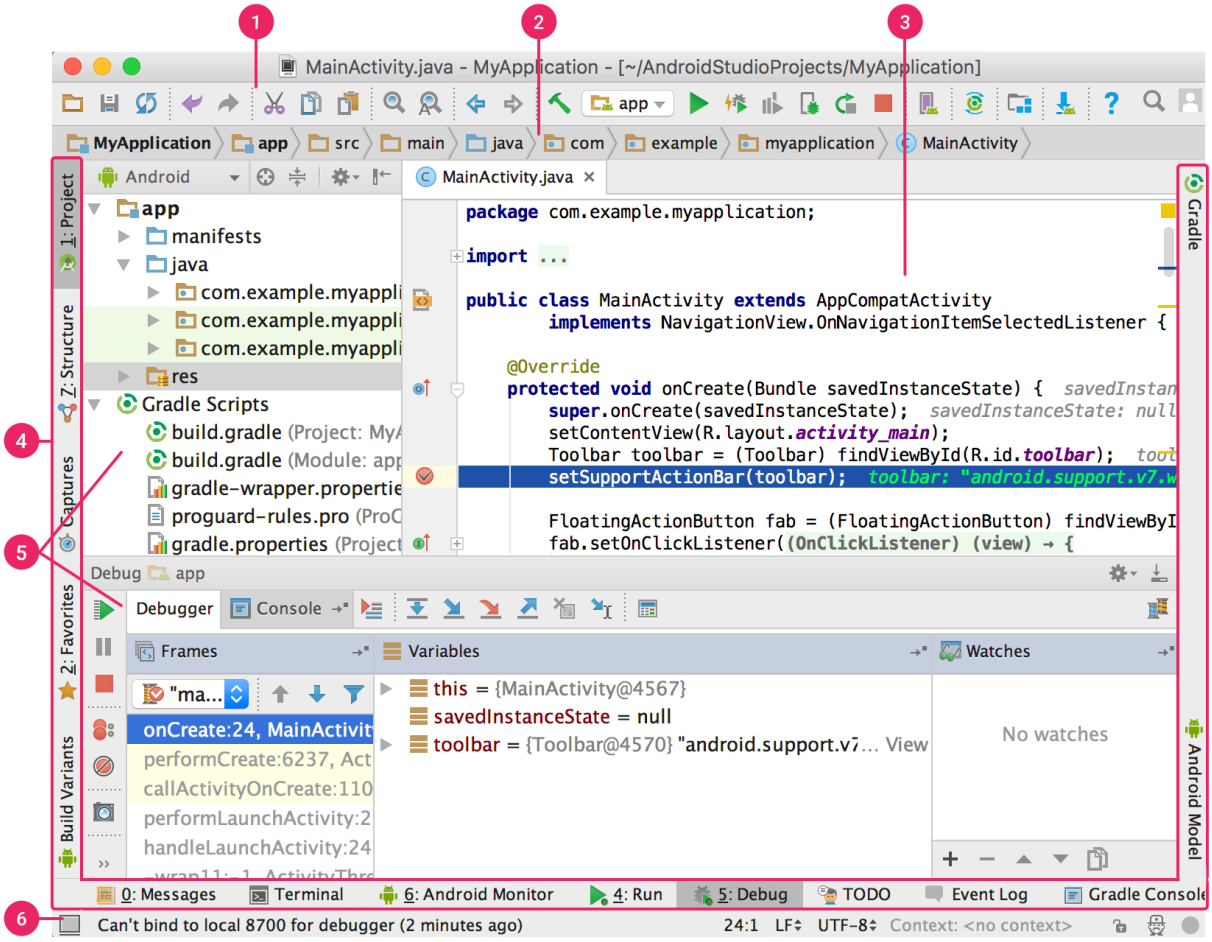


Εικόνα 2-1 Τα αρχεία έργου σε προβολή Android.

Όλα τα αρχεία build είναι ορατά στο ανώτερο επίπεδο στο Gradle Scripts και κάθε ενότητα εφαρμογής περιέχει τους ακόλουθους φακέλους:

- manifests: Περιέχει το AndroidManifest.xml αρχείο.
- **java**: Περιέχει τα αρχεία πηγαίου κώδικα Java, συμπεριλαμβανομένου του δοκιμαστικού κώδικα JUnit.
- **res**: Περιέχει όλους τους πόρους χωρίς κώδικα, όπως διατάξεις XML, συμβολοσειρές UI και εικόνες bitmap.

2.4 Περιβάλλον Android Studio



Εικόνα 2-2 Το Περιβάλλον του Android Studio

1. Γραμμή εργαλείων δίνει την δυνατότητα σε ένα μεγάλο φάσμα ενεργειών όπως η εκτέλεση της εφαρμογής
2. Η γραμμή Navigation Bar όπου γίνεται πλοήγηση σε αρχεία που έχει ανοίξει ο χρήστης.
3. Εδώ εμφανίζεται ο editor σε αυτό ο χρήστης γράφει τον κώδικα της εφαρμογής.
4. Windows tool που βρίσκεται στο αριστερό μέρος της οθόνης και παρέχει δυνατότητες διαχείρισης των φακέλων και των αρχείων του project.
5. Παροχή σε συγκεκριμένες ενέργειες όπως ο έλεγχος.
6. Status bar η οποία βρίσκεται στην τελευταία περιοχή κάτω και δίνει status σε σχέση με το project όπως ειδοποιήσεις, ελέγχους σφαλμάτων κ.α.

2.5 Grandle

Το σύστημα δημιουργίας του android μεταγλωτίζει τον κώδικα και τους πόρους που παράγει το apk του project. Το Android Studio το χρησιμοποιεί ως ενσωματωμένο εργαλείο για την διαχείριση και αυτοματοποίηση της διαδικασίας δημιουργίας κάνοντας επιτρεπτό παράλληλα τον ορισμό προσαρμοσμένων διαμορφώσεων της μεταγλώτισης.

2.6 Επίλογος

Στόχος αυτού του κεφαλαίου ήταν ο αναγνώστης να πάρει μια πρώτη εικόνα ως αφορά την δομή του έργου και τα βασικά εργαλεία της κύριας διεπαφής χρήστη.

Κεφάλαιο 3ο: Επιλογή μεθοδολογίας ανάπτυξης

3.1 Εισαγωγή

Στο παρόν κεφάλαιο γίνεται μια ανάλυση των μεθόδων ανάπτυξης λογισμικού που επιλέχθηκαν για την σχεδίαση του έργου με μια παρουσίαση και εισαγωγή των χαρακτηριστικών της μεθοδολογίας Scrum και της ενοποιημένης προσέγγισης (Unified Process).

3.2 SCRUM

Για την διαχείριση του έργου χρησιμοποιήθηκε η σύγχρονη και ευέλικτη (modern agile) μέθοδος Scrum, μια επαναληπτική και αυξητική μεθοδολογία ανάπτυξης έργων. Η Scrum χρησιμοποιείται για την ανάπτυξη λογισμικού, όσον αφορά στην οργάνωση και τη λειτουργία των ομάδων ανάπτυξης αλλά και στο σχεδιασμό και τη διαχείριση των έργων. Είναι μια ευέλικτη διαδικασία που μας επιτρέπει να επικεντρωθούμε στην παροχή της υψηλότερης επιχειρηματικής αξίας στο συντομότερο χρονικό διάστημα.

Το Scrum είναι:

• Χαμηλής πολυπλοκότητας

• Απλό να το κατανοήσεις

• Δύσκολο να το κατακτήσεις

Το πλαίσιο Scrum αποτελείται από τις Ομάδες Scrum και τους σχετικούς με αυτές ρόλους και κανόνες, δραστηριότητες και αντικείμενα. Καθένα από τα συστατικά στοιχεία του πλαισίου εξυπηρετεί συγκεκριμένο σκοπό και είναι ουσιώδες τόσο για την χρήση όσο και για την επιτυχία του Scrum. [24]

3.3 Δραστηριότητες του Scrum

Το Scrum στηρίζετε στις παρακάτω δραστηριότητες:

- Σχεδιασμός του Sprint
- Καθημερινό Scrum
- Επισκόπηση του Sprint
- Αναδρομή στο Sprint

Όλες οι δραστηριότητες έχουν έναν μέγιστο χρόνο διάρκειας. Το Sprint έχει συγκεκριμένο χρόνο ο οποίος δεν μπορεί να μεταβληθεί ενώ οι υπόλοιπες δραστηριότητες μπορούν να ολοκληρωθούν όταν έχει έρθει εις πέρας ο σκοπός τους. Η κάθε δραστηριότητα αποτελεί και μια ευκαιρία για επιθεώρηση και προσαρμογή.

Sprint

Το sprint είναι ουσιαστικά ένα συγκεκριμένο χρονικό διάστημα με τυπική διάρκεια από δύο έως τέσσερις εβδομάδες το μέγιστο. Σημαντικό είναι να έχουν σταθερή διάρκεια για όλο το χρονικό διάστημα της ανάπτυξης του έργου. Με το που ολοκληρώνεται το ένα Sprint αμέσως αρχίζει ένα νέο.

Κάθε Sprint περιλαμβάνει τα καθημερινά Scrums, τις εργασίες ανάπτυξης την επισκόπηση καθώς και την αναδρομή του Sprint.

Κατά την διάρκεια του Sprint δεν μπορούν να γίνουν αλλαγές που θα επηρεάσουν τον αρχικό στόχο του Sprint ενώ το αντικείμενο των εργασιών μπορεί να συζητηθεί και να δοθούν λύσεις καθώς και να τεθεί σε επαναδιαπραγμάτευση μεταξύ του Product Owner και της υπόλοιπης ομάδας.

3.4 Ρόλοι

Product owner

Κάνει τον ορισμό του προϊόντος ορίζει την τελική ημερομηνία ολοκλήρωσης του έργου. Βελτιστοποιεί την αξία του έργου που παράγει η ομάδα, ενώ διασφαλίζει το να είναι ξεκάθαρες σε όλους οι απαιτήσεις του έργου.

Scrum Master

Έχει τον ρόλο να βοηθάει στην αλληλεπίδραση της ομάδας με όσους βρίσκονται εκτός αυτής στο να κατανοούν ποιες από αυτές είναι χρήσιμες και ποιες όχι . Βοηθά την ομάδα να είναι πλήρως λειτουργική και τους μεταδίδει τις πρακτικές του scrum καθώς και καθοδηγεί την Ομάδα Ανάπτυξης στην αυτό-οργάνωση και ανάπτυξη δεξιοτήτων από διαφορετικές λειτουργικές περιοχές.

Ομάδα

Οι ομάδες συνήθως αποτελούνται από μικρό αριθμό ατόμων, λειτουργούν με αυτό οργάνωση έχουν δεξιότητες από διαφορετικές περιοχές δεν έχουν τίτλους και δεν επιτρέπεται να λειτουργούν σε υπό ομάδες.

3.5 Αντικείμενα του Scrum

Τα αντικείμενα που ορίζονται στο Scrum είναι ειδικά σχεδιασμένα για να επιτυγχάνουν μέγιστη διαφάνεια των σημαντικότερων πληροφοριών και τελικά κοινή κατανόηση του κάθε αντικειμένου από όλους.

Product Backlog

Περιλαμβάνει τα βασικά χαρακτηριστικά του έργου βάση του ορισμού τους από τον πελάτη καθώς και βελτιώσεις ή ελλείψεις που θα χρειαστούν προσοχή.

Sprint Backlog

Είναι το σύνολο από τα χαρακτηριστικά του Product Backlog που έχουν επιλεγεί για το Sprint συνοδευόμενο από ένα πλάνο για τον στόχο του Sprint καθώς και για το πώς θα παραδοθεί η επαύξηση του προϊόντος. Το Sprint Backlog δείχνει όλες τις εργασίες που θα κάνει η ομάδα ανάπτυξης ώστε να υπάρξει επίτευξη του στόχου. Κάθε Sprint Backlog περιλαμβάνει τουλάχιστον μια ενέργεια με υψηλή προτεραιότητα. [24]

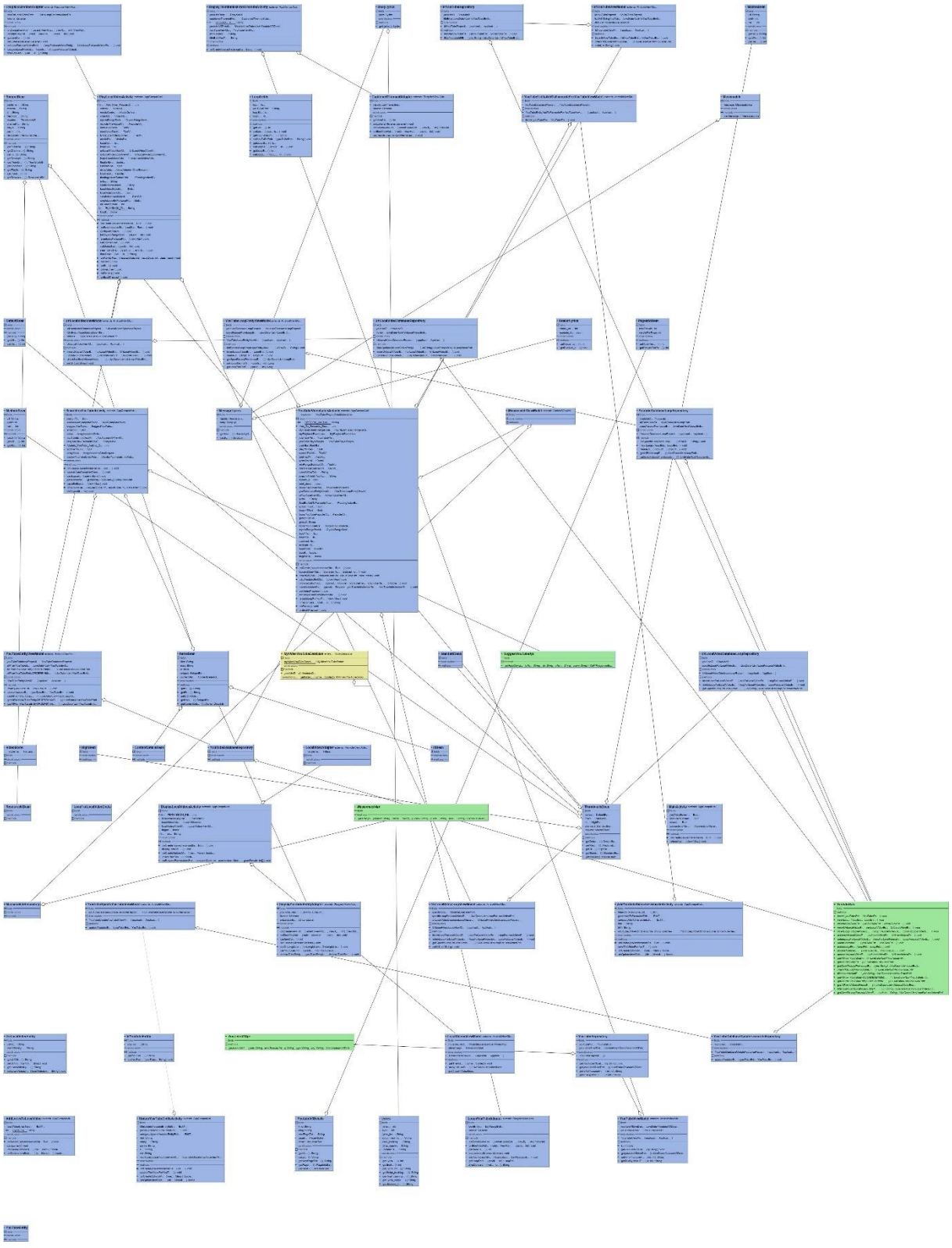
3.6 Unified Process

Η ενοποιημένη προσέγγιση (Unified Process ή, εν συντομία, UP) αποτελεί μια από τις πλέον διαδεδομένες διαδικασίες ανάπτυξης λογισμικού. Η UP βασίζεται στον καλό χειρισμό των απαιτήσεων με το μοντέλο περιπτώσεων χρήσης καθώς και στην γραφική ανάπτυξη με χρήση UML.

Είναι μια επαναληπτική διαδικασία που αποτελείται από τέσσερις φάσεις:

- Σύλληψη (inception)
- Σε αυτήν την φάση παρουσιάζεται η αρχική ιδέα του συστήματος χαράσσεται η στρατηγική οι απαιτήσεις και οι περιπτώσεις χρήσης του συστήματος καθώς και τους πιθανούς κινδύνους που μπορεί να υπάρχουν.
- Επεξεργασία (elaboration)
- Εδώ γίνεται καταγραφή των απαιτήσεων με λεπτομερή ανάλυση, προσδιορίζεται το αντικείμενο των εργασιών ενώ αναπτύσσεται και η βασική αρχιτεκτονική του έργου.
- Κατασκευή (construction)
- Σε αυτήν την φάση πραγματοποιείται η υλοποίηση και ο έλεγχος τμημάτων πού είναι ορισμένα.
- Μετάβαση (transition)
- Τέλος, στην φάση της μετάβασης γίνεται παράδοση του τελικού έργου που περιλαμβάνει την εκπαίδευση των χρηστών και την πιλοτική λειτουργία του συστήματος.

3.6.1 Διάγραμμα κλάσεων συστήματος



Εικόνα 3-1 Διάγραμμα κλάσεων

3.7 Επίλογος

Σ αυτό το κεφάλαιο ο αναγνώστης έχει την δυνατότητα να μάθει τις βασικές αρχές που διέπουν τις μεθοδολογίες ανάπτυξης που χρησιμοποιήθηκαν για το παρόν έργο και να κατανοήσει την χρησιμότητα τους ως βασική αρχή για την ανάπτυξη ενός «καλού» λογισμικού.

Κεφάλαιο 4ο: Εξαγωγή απαιτήσεων

4.1 Εισαγωγή

Για να αναπτυχθεί ένα σύστημα λογισμικού απαραίτητη προϋπόθεση είναι να γίνει σαφές το τι εργασίες θα κάνει καθώς και αλλά χαρακτηριστικά τα οποία θα πρέπει να έχει. Για αυτό είναι απαραίτητο να γίνει ο ορισμός των ιστοριών χρηστών (user stories), ή αλλιώς των λειτουργιών που θα καλείται να εκτελεί το σύστημα από τους χρήστες του.

4.2 Ιστορίες χρηστών (User stories)

Αρμόδιος για τον ορισμό των ιστοριών χρηστών είναι, σύμφωνα με την SCRUM, ο Product Owner.

Στο παρόν έργο οι ρολόι έχουν ως εξής:

Όνομα Scrum master: Δεληγιάννης Ιγνάτιος

Όνομα 1ου μέλους: Τσάτσος Θεόδωρος - Αλέξανδρος

Όνομα 2ου μέλους: Μπαρκολίας Αλέξανδρος

Πίνακας 4-1 user stories

	ΕΦΑΡΜΟΓΗ ΜΟΥΣΙΚΟΥ ΕΡΓΑΛΕΙΟΥ MEDIAYOUTUBELYRICS			
ID	**STORY**	**Περιγραφή**	**Estimation**	**Priority**
	(Product owner)		(Scrum Team)	(PO)
US01	Ως χρήστης ανοίγω την εφαρμογή.	Στο UI εμφανίζονται δύο κουμπιά το ένα αφορά το YouTube και το δεύτερο τα αρχεία video του καταλόγου του κινητού.	xxs	1
US02	Ως χρήστης επιλέγω το κουμπί «Youtube video»	Στο UI εμφανιζεται ένα textview στο οποίο πληκτρολογώ τον τίτλο ή κάτι σχετικό με αυτό που θέλω να βρω.	xxs	2

US03	Ως χρήστης θέλω να κάνω αναζήτηση κάποιου βίντεο στην πλατφόρμα του YouTube.	Στο UI εμφανίζεται ένα textview στο οποίο πληκτρολογώ τον τίτλο ή κάτι σχετικό με αυτό που θέλω να βρω. Πατώντας το κουμπί search στο UI εμφανίζετε ένα νέο UI που περιέχει έως 50 αποτελέσματά σε σχέση με την αναζήτηση που έχω κάνει.	l	3
US04	Ως χρήστης επιλέγω να ανοίξω ένα βίντεο από την πλατφόρμα του Youtube.	Αφού έχω επιλέξει ένα βίντεο από τα αποτελέσματα της αναζήτησης του Youtube εμφανίζεται ένα UI το όποιο περιέχει ένα κουμπί για την αναπαραγωγή του video και δύο switch button, το ένα για την εμφάνιση των στοίχων και το άλλο για τα loops καθώς και ένα κουμπί αποθήκευσης του συγκεκριμένου βίντεο στα αγαπημένα.	m	4
US05	Ως χρήστης θέλω να επιλέξω ένα βίντεο από την λίστα των αγαπημένων.	Αφού έχω επιλέξει το κουμπί «Youtube video» στο UI εμφανίζεται μια λίστα η οποία περιέχει εγγραφές αγαπημένων βίντεο που έχω κάνει. Επιλέγω την εγγραφή που επιθυμώ και εμφανίζετε το UI αναπαραγωγής .	m	5
US06	Ως χρήστης θέλω να επεξεργαστώ κάποια εγγραφή από τα αγαπημένα μου.	Αφού έχω εισέλθει στο UI για τα «YouTube Videos» πατώντας παρατεταμένα πάνω σε μια εγγραφή της λίστας των αγαπημένων δίνεται η δυνατότητα να επεξεργαστώ τον τίτλο την κατηγορία και το είδος.	m	6
US07	Ως χρήστης θέλω να δημιουργήσω loops ή να διαγράψω υπάρχοντα loops για video από το YouTube.	Αφού έχω επιλέξει το κουμπί «Youtube video» και έχω επιλέξει να ανοίξω ένα βίντεο, επιλεγώ το switch button Loops και εμφανίζεται στο UI η μπάρα δημιουργίας loop. Επιλέγω μέσω αυτής τα σημεία που θέλω	xl	7

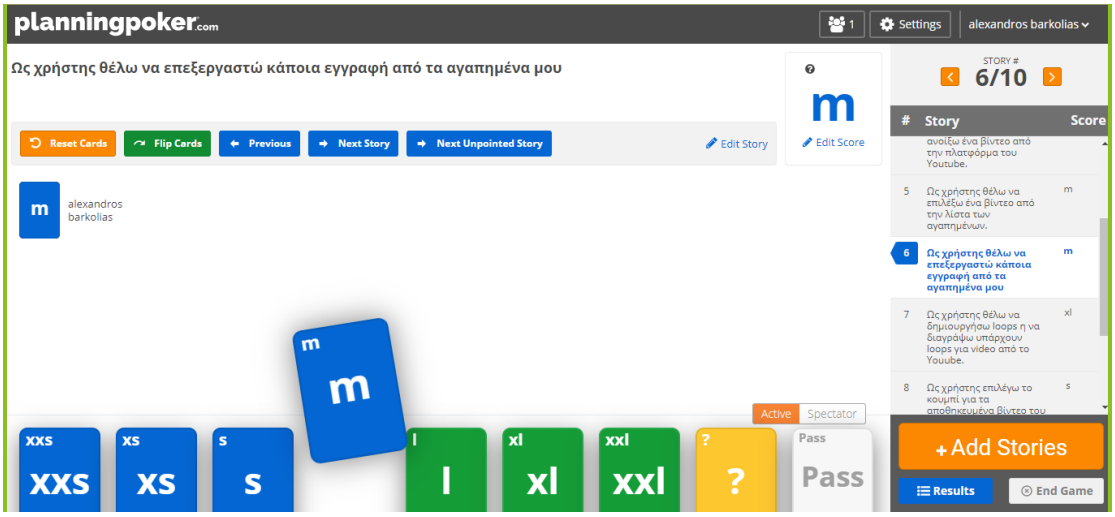
		να είναι η αρχή και το τέλος της κάθε λούπας και πατάω το κουμπί create loop. Αν θέλω να διαγράψω κάποια λούπα την σύρω δεξιά η αριστερά για την διαγραφή της.		
US08	Ως χρήστης επιλέγω το κουμπί για τα αποθηκευμένα βίντεο του καταλόγου του κινητού (Phone videos)	Εμφανίζεται ένα UI που περιέχει στο πάνω μέρος μια μπάρα αναζήτησης που φιλτράρει τα αποτελέσματά με βάση τον τίτλο του κάθε βίντεο και κάτω εμφανίζεται μια λίστα με τα βίντεο του κινητού.	s	8
US09	Ως χρήστης επιλέγω να ανοίξω ένα βίντεο από τον κατάλογο το κινητού.	Αφού έχω επιλέξει ένα βίντεο από τα βίντεο του κινητού, εμφανίζεται ένα UI στο οποίο γίνεται αυτόματα η αναπαραγωγή και δύο switch button. Το ένα switch button για την εμφάνιση των στοίχων και το άλλο για την δημιουργία (μέσω του κουμπιού create loop) και διαγραφή loop. Επιπλέον εμφανίζετε ένα κουμπί εισαγωγής και επεξεργασίας στοίχων και αποθήκευσης τους για το συγκεκριμένο βίντεο. Τέλος υπάρχει και ένα spinner για την επιλογή της ταχύτητας αναπαραγωγής του επιλεγμένου βίντεο.	l	9
US10	Ως χρήστης θέλω να δημιουργήσω loops η να διαγράψω υπάρχοντα loops για video από το τον κατάλογο του κινητού.	Αφού έχω επιλέξει το κουμπί «Phone videos» και έχω επιλέξει να ανοίξω ένα βίντεο, επιλεγώ το switch button Loops και εμφανίζεται στο UI η μπάρα δημιουργίας loop. Επιλέγω μέσω αυτής τα σημεία που θέλω να είναι η αρχή και το τέλος της κάθε λούπας και πατάω το κουμπί create loop. Αν θέλω να διαγράψω κάποια λούπα την σύρω δεξιά η αριστερά για την διαγραφή της.	xl	10

4.3 Αξιολόγηση user Stories

Πρόκειται ουσιαστικά για την αξιολόγηση της δυσκολίας και των απαιτήσεων των user Stories.

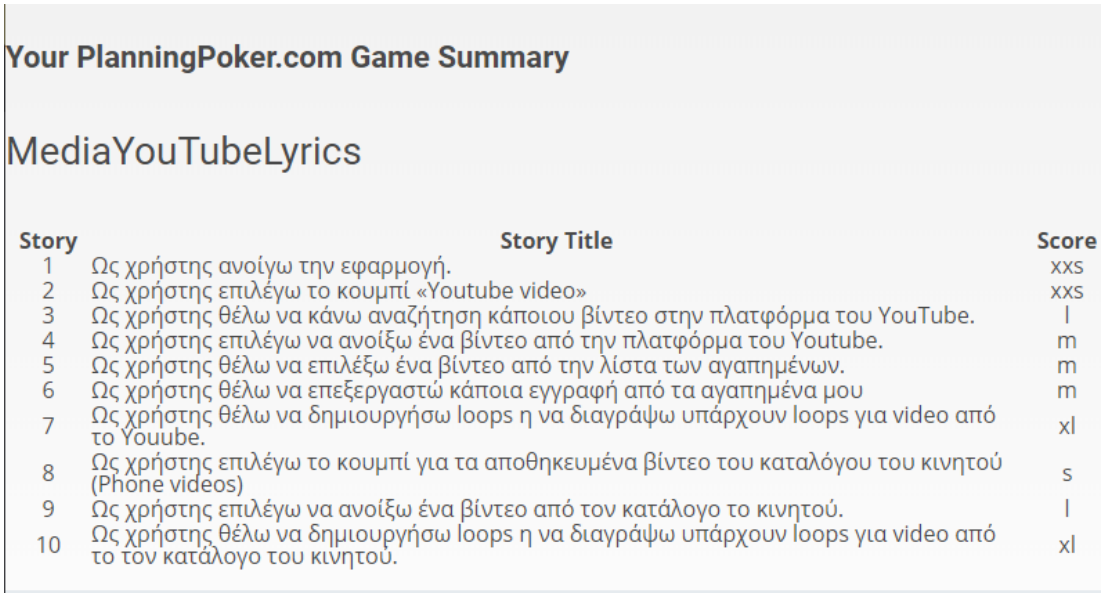
Οι τιμές της στήλης ‘estimation’ του προηγούμενου πίνακα προέκυψαν από την αξιολόγηση αυτήν.

PlanningPoker



Εικόνα 4-1 Αξιολόγιση U.S. με το Planning Poker

Συγκεντρωτικά αποτελέσματα



Your PlanningPoker.com Game Summary

MediaYouTubeLyrics

Story	Story Title	Score
1	Ως χρήστης ανοίγω την εφαρμογή.	xxs
2	Ως χρήστης επιλέγω το κουμπί «Youtube video»	xxs
3	Ως χρήστης θέλω να κάνω αναζήτηση κάποιου βίντεο στην πλατφόρμα του YouTube.	l
4	Ως χρήστης επιλέγω να ανοίξω ένα βίντεο από την πλατφόρμα του Youtube.	m
5	Ως χρήστης θέλω να επιλέξω ένα βίντεο από την λίστα των αγαπημένων.	m
6	Ως χρήστης θέλω να επεξεργαστώ κάποια εγγραφή από τα αγαπημένα μου	m
7	Ως χρήστης θέλω να δημιουργήσω loops η να διαγράψω υπάρχουν loops για video από το Youube.	xl
8	Ως χρήστης επιλέγω το κουμπί για τα αποθηκευμένα βίντεο του καταλόγου του κινητού (Phone videos)	s
9	Ως χρήστης επιλέγω να ανοίξω ένα βίντεο από τον κατάλογο το κινητού.	l
10	Ως χρήστης θέλω να δημιουργήσω loops η να διαγράψω υπάρχουν loops για video από το τον κατάλογο του κινητού.	xl

Εικόνα 4-2 estimation U.S.

4.4 Επίλογος

Στο τέταρτο κεφάλαιο δίνεται ο πίνακας των ιστοριών χρηστών (User Stories) του συγκεκριμένου έργου καθώς και ο τρόπος που έγινε η αξιολόγηση της δυσκολίας και των απαιτήσεων τους. Ο αναγνώστης αποκτά μια εικόνα για το πώς θα είναι η τελική εφαρμογή ως προς τις λειτουργίες που θα παρέχονται στον χρήστη.

Κεφάλαιο 5ο: Τεχνολογίες που χρησιμοποιήθηκαν

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα περιγράφουν οι τεχνολογίες (γλώσσες προγραμματισμού και βιβλιοθήκες) που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας εφαρμογής.

5.2 Java

Η γλώσσα Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού δημιουργήθηκε από τον James Gosling και πλέον βρίσκεται στην έκδοση 14. 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 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 συμβολικός κώδικας (assembly) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της Εικονικής Μηχανής (Virtual Machine).Αφού γραφεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι 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 τα 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 δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ . [25]

5.3 Xml

Η XML(eXtensible Markup Language) είναι μία γλώσσα για τη δόμηση δεδομένων. Με την έννοια δομημένα δεδομένα εννοούμε μία συλλογή στοιχείων δεδομένων όπως είναι για παράδειγμα τα λογιστικά φύλλα, οι κατάλογοι διευθύνσεων, οι παράμετροι διαμόρφωσης, οι οικονομικές συναλλαγές και τα τεχνικά σχέδια. Η XML είναι δηλαδή, ένα σύνολο κανόνων (ή διαφορετικά ένα πακέτο κατευθυντήριων γραμμών ή συμβάσεων) για το σχεδιασμό μορφών κειμένου οι οποίες διευκολύνουν τη δόμηση των δεδομένων σας. Ορίζεται κυρίως, στην προδιαγραφή XML 1.0 (XML 1.0 Specification), που δημιούργησε ο διεθνής οργανισμός προτύπων W3C (World Wide Web Consortium), αλλά και σε διάφορες άλλες σχετικές προδιαγραφές ανοιχτών προτύπων. Η XML διευκολύνει τον υπολογιστή να παράγει δεδομένα, να διαβάζει δεδομένα και να εξασφαλίζει τη σαφήνεια της δομής των δεδομένων. Η XML αποφεύγει τις συνήθεις παγίδες του σχεδιασμού γλωσσών: είναι επεκτάσιμη, ανεξάρτητη συστήματος υλικού και μπορεί να υποστηρίξει διεθνείς και τοπικές προσαρμογές. Η XML είναι πλήρως

συμβατή με Unicode. Η XML, όπως η HTML, χρησιμοποιεί ετικέτες (tags) (λέξεις μέσα σε γωνιακές αγκύλες '<' και '>') και γνωρίσματα (τύπου όνομα = "τιμή"). Σε αντίθεση με την HTML η οποία διευκρινίζει τη σημασία κάθε ετικέτας και γνωρίσματος και συχνά προσδιορίζει πως θα εμφανίζεται σε φυλλομετρητή το κείμενο το οποίο περιλαμβάνεται σε αυτά, η XML χρησιμοποιεί ετικέτες μόνο για να οριοθετήσει κομμάτια δεδομένων και αφήνει την ερμηνεία των δεδομένων στη εφαρμογή που τα διαβάζει. [26]

5.4 YouTube Android Player API

Το youtube android player Api παρέχει την δυνατότητα να ενσωματώσουμε την λειτουργία αναπαραγωγής video μέσα σε μια android εφαρμογή. Το Api ορίζει μεθόδους για την φόρτωση, προσαρμογή, έλεγχο και αναπαραγωγή Youtube video. Χρησιμοποιώντας το Api, μπορούμε να φορτώσουμε ή να προετοιμάσουμε κάποια videos μέσα στο player view που είναι ενσωματωμένος στην εφαρμογής το UI. Έπειτα μπορούμε να ελέγξουμε την αναπαραγωγή video προγραμματιστικά. Για παράδειγμα, μπορούμε να κάνουμε παύση, ή να προχωρήσουμε το τρέχων video σε ένα συγκεκριμένο σημείο.

5.4.1 Πως λειτουργεί

Η Api βιβλιοθήκη πελάτη αλληλεπιδρά με μια υπηρεσία που είναι κατανεμημένη ως ένα κομμάτι της YouTube εφαρμογής για την Android πλατφόρμα. Η βιβλιοθήκη πελάτη έχει ελαφρύ αποτύπωμα, σημαίνοντας οτι δεν έχει αρνητική επίδραση στο μέγεθος του αρχείου της εφαρμογής. [27]

5.4.2 Πακετο com.google.android.youtube.player

Το πακέτο αυτό περιέχει όλα τα interface και τις κλάσεις του YouTube Android Player Api. Με το Api μας δίνεται η δυνατότητα να κάνουμε εύκολη την αναπαραγωγή YouTube Video.

Υπάρχουν δύο τρόποι για την αναπαραγωγή video. Η πρώτη επιλογή είναι η τοποθέτηση του YouTubePlayerFragment ή YouTubePlayerActivity μέσα στην View ιεραρχία και ύστερα η χρησιμοποίηση του YouTubePlayer για τον έλεγχο της αναπαραγωγής του video .

Η δεύτερη επιλογή είναι η τοποθέτηση YouTube Standalone Player στο οποίο θα γίνει η αναπαραγωγή του video σε ξεχωριστή δραστηριότητα (Activity). Αυτός ο τρόπος είναι ο πιο εύκολος ωστόσο μας δίνει μικρότερη δυνατότητα για τον έλεγχο του video.

5.4.2.1 YouTubePlayer public interface YouTubePlayer

Ο YouTubePlayer παρέχει μεθόδους για την φορτώση, αναπαραγωγή και έλεγχο του YouTube Video. Μπορούμε να πάρουμε ένα στιγμιότυπο από την κλάση αυτή καλώντας την μέθοδο initialize σε ένα YouTubePlayer.Provider όπως το YouTubePlayerFragment ή το YouTubePlayerView.

Τα id των video ή τις playlist είναι απαραίτητα για να μπορούμε να χρησιμοποιήσουμε τις μεθόδους όπως την loadVideo(String) ή την cue PlayList(String).

Ο Youtubeplayer υποστηρίζει την αποθήκευση της κατάστασης, η οποία διαχειρίζεται αυτόματα από τον YouTubePlayer Provide του player. Η αποθηκευμένη κατάσταση περιέχει το φορτωμένο video, την τρέχων θέση και τις ρυθμίσεις της συσκευής αναπαραγωγής. Η κατάσταση δεν εμπεριέχει τους ακροατές που πιθανόν έχουν εφαρμοστεί στον player και άρα θα πρέπει να προσθέτουν εκ νέου στην περίπτωση που ο player ξαναδημιουργηθεί.

5.4.2.2 public static interface YouTubePlayer.OnInitializedListener

Ορισμός διεπαφής για ανάκληση κλήσεων που καλούνται όταν η αρχικοποίηση του player πετυχαίνει ή αποτυγχανει. Η διεπαφή OnInitializedListener περιέχει δύο public μεθόδους:

- abstract void onInitializationFailure(YouTubePlayer.Provider provider,YouTubeInitializationResult error) - που καλείται όταν η αρχικοποίηση του player αποτυγχάνει.
- abstract void onInitializationSuccess (YouTubePlayer.Provider provider,YouTubePlayer player,boolean wasRestored) - που καλείται όταν η αρχικοποίηση του player πετυχαίνει.

5.4.2.3 public static interface YouTubePlayer.PlaybackEventListener.

Ορισμός διεπαφής για επιστροφές κλήσεων που καλούνται όταν γεγονότα αναπαραγωγής προκύπτουν.

Η διεπαφή PlaybackEventListener περιέχει τις παρακάτω public μεθόδους:

- **public abstract void onBuffering (boolean isBuffering) -** Καλείται όταν η προσωρινή αποθήκευση αρχίζει ή τελειώνει.
- **public abstract void onPaused () -** Καλείται όταν η αναπαραγωγή του video έχει προσωρινή παύση, είτε λόγο της μέθοδου **pause(),** είτε σε κάποια ενέργεια του χρήστη.
- **public abstract void onPlaying () -** Καλείται όταν η αναπαραγωγή αρχίζει είτε λόγο της μέθοδου play() είτε σε κάποια ενέργεια του χρήστη.
- **public abstract void onSeekTo (int newPositionMillis)** - Καλείται όταν επεμβαίνουμε στην αλλαγή θέσης των millisecond του player που μπορεί να γίνει είτε από ενέργεια του χρήστη είτε να οφείλεται στην μέθοδο (seekToMillis(int)).
- **public abstract void onStopped () -** Καλείται όταν η αναπαραγωγή σταματάει, όπως όταν το video φτάσει στο τέλος του ή κάποιο σφάλμα προκύπτει.

5.4.2.4 public static interface YouTubePlayer.PlayerStateChangeListener.

Ορισμός διεπαφής για επιστροφές κλήσεων που καλούνται όταν υψηλού επιπέδου καταστάσεις για τον player αλλάζουν. Η διεπαφή YouTubePlayer.PlayerStateChangeListener περιέχει τις παρακάτω μεθόδους

- **public abstract void onAdStarted () -** Καλείται όταν γίνεται αναπαραγωγή μιας διαφήμισης.
- **public abstract void onError (YouTubePlayer.ErrorReason reason) -** Καλείται όταν ένα σφάλμα προκύπτει, αύτη η μέθοδος μπορεί να κληθεί σε οποιαδήποτε κατάσταση.
- **public abstract void onLoaded (String videoId) -** Καλείται όταν η φόρτωση του video έχει φτάσει στο τέλος της. Οι μέθοδοι όπως η play(),pause() ή seekTomills(int) μπορούν να κληθούν μετά την επιστροφή της onLoaded μεθόδου.
- **public abstract void onLoading () -** Καλείται όταν αρχίζει η φόρτωση του video στον player και δεν είναι ακόμα έτοιμος να δειχθεί εντολές που επηρεάζουν την αναπαραγωγή του video(όπως η play() ή pause() μέθοδο).
- **public abstract void onVideoEnded () -** Καλείται όταν το video φτάσει στο τέλος του.
- **public abstract void onVideoStarted () -** Καλείται όταν η αναπαραγωγή του video αρχίζει.

5.4.2.5 public final class YouTubePlayerView extends ViewGroup implements YouTubePlayer.Provider.

Ένα view για την απεικόνιση YouTube video. Χρησιμοποιώντας αυτό το View είναι μια εναλλακτική απο το να χρησιμοποιούμε την Κλάση YouTubePlayerFragment. Αυτό το view θα αποθηκεύσει και θα

επαναφέρει την κατάσταση του YouTubePlayer που σχετίζεται με το view ως ένα κομμάτι της onSaveInstanceState/onRestoreInstanceState ροής.

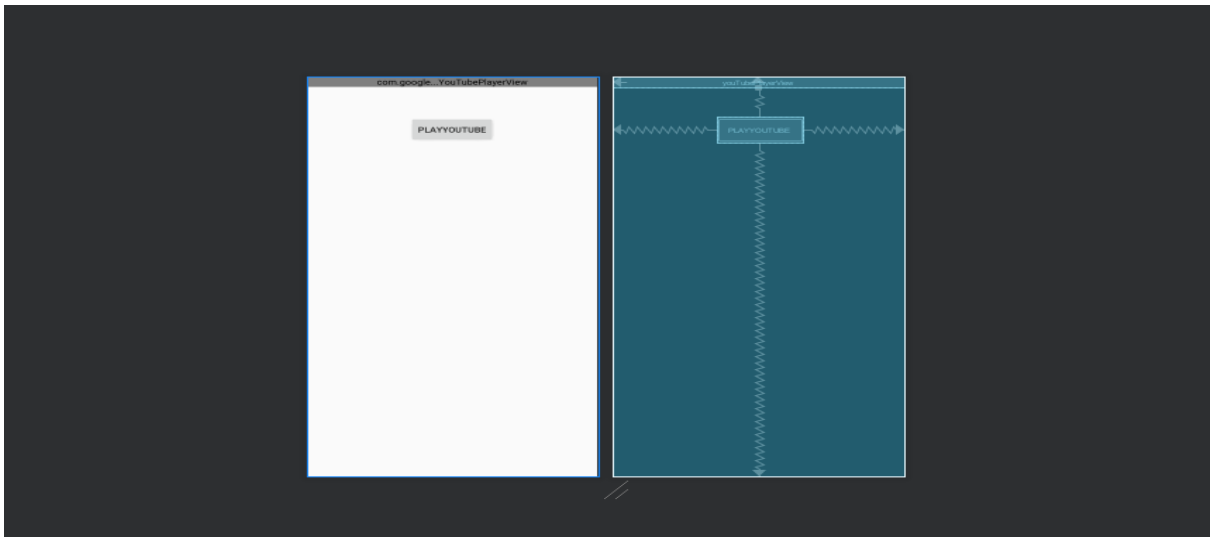
5.4.2.6 public class YouTubeBaseActivity extends Activity

Οποιαδήποτε δραστηριότητα(Activity) χρειάζεται να ενσωματώσει κατευθείαν το YouTubePlayerView view στο UI της πρέπει να κάνει επέκταση αυτής της δραστηριότητας(Activity).

Παράδειγμα χρήσης των κλάσεων YouTubePlayerView, YouTubeBaseActivity για την αναπαραγωγή ενός youtube video:

```xml
<?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">

    <com.google.android.youtube.player.YouTubePlayerView
        android:id="@+id/youTubePlayerView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/youTubePlayBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Playyoutube"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/youTubePlayerView"
        app:layout_constraintVertical_bias="0.081" />

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

Εικόνα 5-1 παράδειγμα 1 _ activity.xml



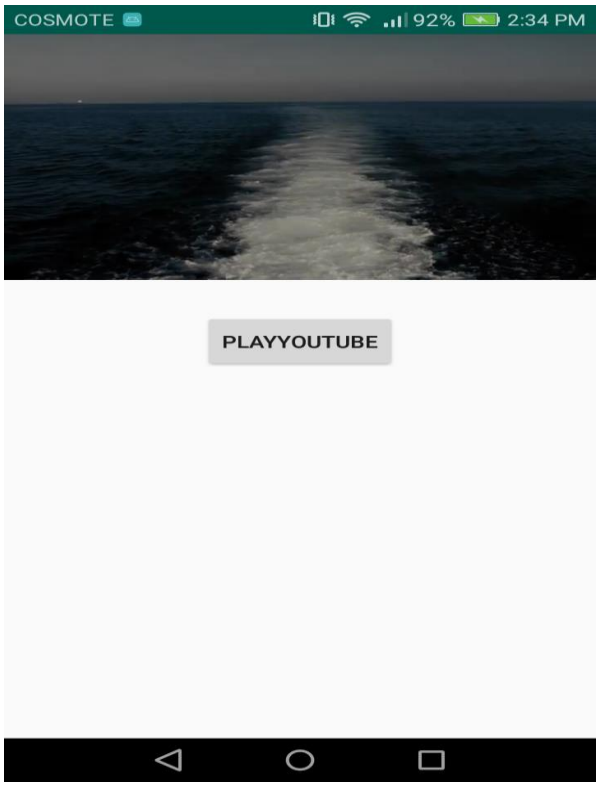
Εικόνα 5-2 Παράδειγμα 1 προεπισκόπηση activity.xml

Το αρχείο MainActivity.java όπου κάνουμε επέκταση της κλάσης YouTubeBaseActivity και υλοποιούμε τις μεθόδους onInitializationSuccess, onInitializationFailure του OnInitializedListener ακροατή, με την μέθοδο initialize (String developerKey, YouTubePlayer.OnInitializedListener listener) γίνεται η αρχικοποίηση του YouTubePlayer και με την μέθοδο loadVideo(String videoId) κάνουμε την φόρτωση και την αναπαραγωγή του συγκεκριμένου video:

```
public class MainActivity extends YouTubeBaseActivity {
   private YouTubePlayerView youtubePlayerView;
    private Button youTubePlayeBtn;
    private YouTubePlayer.OnInitializedListener mOnInitializedListener;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        youTubePlayeBtn = findViewById(R.id.youTubePlayBtn);
        youtubePlayerView = findViewById(R.id.youTubePlayerView);
        mOnInitializedListener = new YouTubePlayer.OnInitializedListener() {
            @Override
            public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer youTubePlayer, boolean b) {
                youTubePlayer.loadVideo( s: "54fea7wuV6s");

            }

            @Override
            public void onInitializationFailure(YouTubePlayer.Provider provider, YouTubeInitializationResult youTubeInitializationResult) {
                Toast.makeText( context: MainActivity.this,youTubeInitializationResult.toString(),Toast.LENGTH_SHORT);
            }
        };
        youTubePlayeBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                youtubePlayerView.initialize( s: "AIzaSyBWp_ZfKx8DA7z9v34AcqmkzqM3YqbAm7U", mOnInitializedListener);
            }
        });
    }
}
```

Εικόνα 5-3 Παράδειγμα 1 MainActivity.java

Το αποτέλεσμα που προκύπτει είναι το παρακάτω:



Εικόνα 5-4 Παράδειγμα 1 Οθόνη αποτελέσματος

5.4.2.7 public class YouTubePlayerFragment extends Fragment implements YouTubePlayer.Provider.

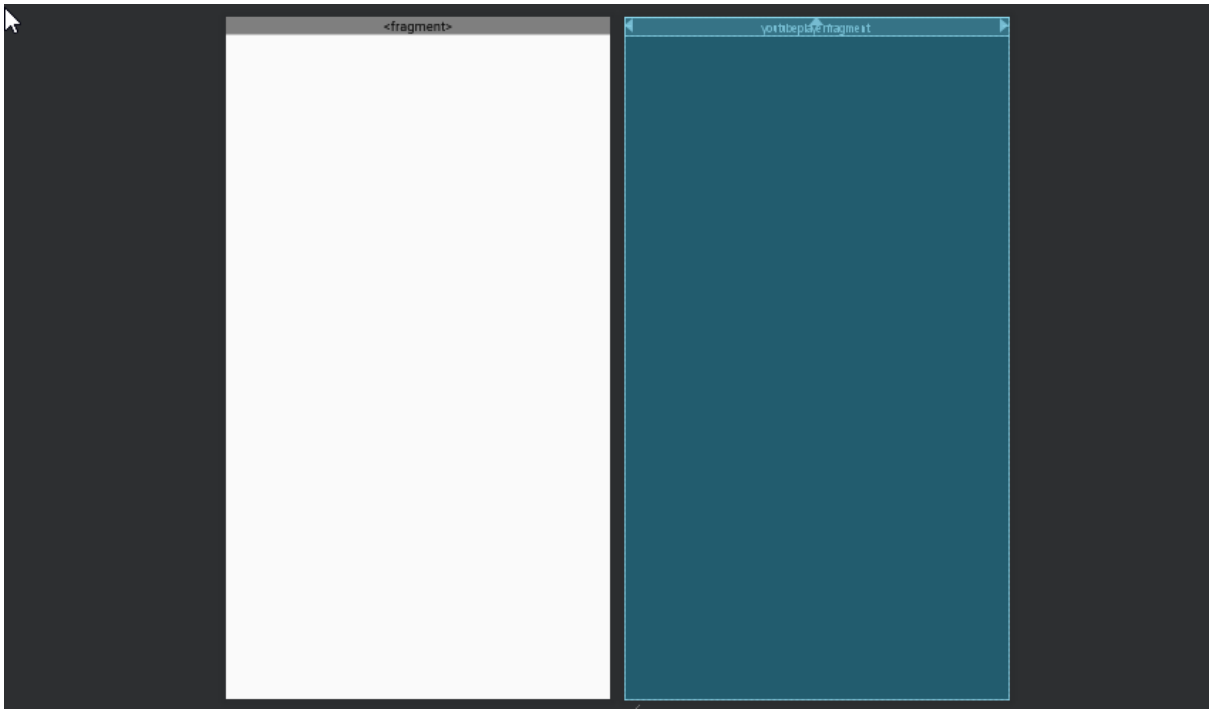
Ένα fragment που περιλαμβάνει το YouTubePlayerView. Χρησιμοποιώντας αυτό το fragment και σύμφωνα με το έγγραφο της google είναι ο προτιμητέος τρόπος για την αναπαραγωγή Youtube video, καθώς δεν χρειάζεται να κάνουμε επέκταση οποιαδήποτε δραστηριότητας (Activity) που παρέχεται από την youtube βιβλιοθήκη. Το fragment θα αποθηκεύσει και θα επαναφέρει την κατάσταση του YouTubePlayer που σχετίζεται με το μέρος της ροής onSaveInstanceState/onRestoreInstanceState.

Παράδειγμα χρήσης της κλάσης YouTubePlayerFragment :

Στο αρχείο activity_main.xml εισάγουμε ένα fragment.

```xml
<?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">

    <fragment
        android:id="@+id/youtubeplayerfragment"
        android:name="com.google.android.youtube.player.YouTubePlayerFragment"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Εικόνα 5-5 παράδειγμα 2 _ activity.xml



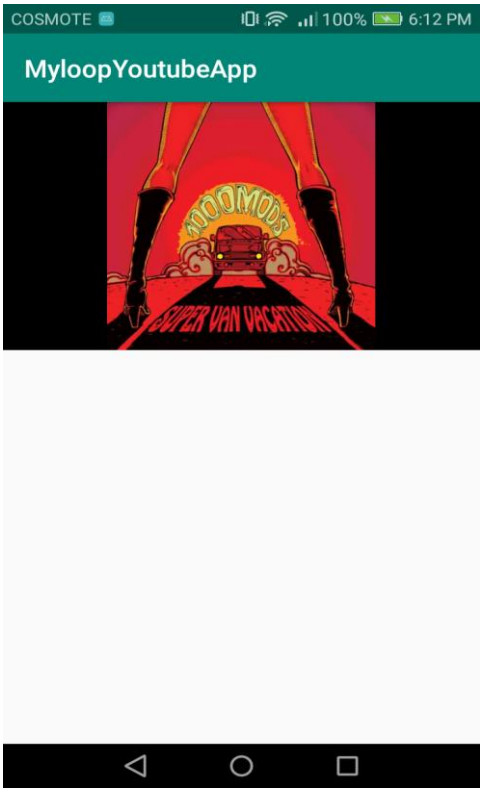
Εικόνα 5-6 Παράδειγμα 2 προεπισκόπηση activity.xml

Στο Αρχείο MainActivity.java υλοποιούμε το YouTubePlayer.OnInitaizedListener και κάνουμε υπερφόρτωση των μεθόδων onInitializationSuccess, onInitializationFailure. Η μέθοδος setFullscreenControlFlags(int controlFlags) αυτοματοποιεί την συμπεριφορά της οθόνης όταν αυτή εισέρχεται στην κατάσταση της πλήρους οθόνης. Η μέθοδος αυτή στην δικής μας περίπτωση δέχεται η δύο σταθερές σαν παραμέτρους την int FULLSCREEN_FLAG_CONTROL_ORIENTATION για τον αυτόματο έλεγχο του προσανατολισμού την οθόνης και την int FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE η οποία αναγκάζει τον player να μπει αυτόματα στην κατάσταση της πλήρους οθόνης οποτεδήποτε η android συσκευή εισέλθει σε οριζόντιο προσανατολισμό. Με την μέθοδο initialize (String developerKey, YouTubePlayer.OnInitializedListener listener) γίνεται η αρχικοποίηση του YouTubePlayer και με την

μέθοδο cueVideo(String videoId) γίνεται η φόρτωση του thumbnail του συγκεκριμένου video και γίνεται η προετοιμασία του player για την αναπαραγωγή του video αλλά δεν γίνεται η λήψη της ροής του video έως ότου κληθεί η μέθοδος play().

```
public class MainActivity extends AppCompatActivity implements YouTubePlayer.OnInitializedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        YouTubePlayerFragment youTubePlayerFragment = (YouTubePlayerFragment) getFragmentManager()
                .findFragmentById(R.id.youtubeplayerfragment);
        youTubePlayerFragment.initialize( s: "AIzaSyBWp_ZfKx8DA7z9v34AcqmkzqM3YqbAm7U",  onInitializedListener: this);

    }

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer youTubePlayer, boolean b) {
        youTubePlayer.setFullscreenControlFlags(YouTubePlayer.FULLSCREEN_FLAG_CONTROL_ORIENTATION |
                YouTubePlayer.FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE);
                youTubePlayer.cueVideo( s: "zsSaMCMUbaw");

    }
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider, YouTubeInitializationResult youTubeInitializationResult) {
    }
}
```

Εικόνα 5-7 Παράδειγμα 2 MainActivity.java

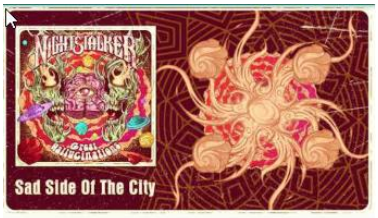


Εικόνα 5-8 Παράδειγμα 2 Οθόνη αποτελέσματος

Επίσης μπορούμε να δώσουμε διάφορα στυλ εμφάνισης στον Player με την μέθοδο setPlayerStyle(PlayerStyle) μπορούμε να τα ορίσουμε. Τα διαθέσιμα στυλ είναι τα παρακάτω: [28]

- YouTubePlayer.PlayerStyle.CHROMELESS - Στυλ που δεν δείχνει κανένα διαδραστικό στοιχείο ελέγχου της αναπαραγωγής.
- YouTubePlayer.PlayerStyle.DEFAULT - Το προεπιλεγμένο στυλ που δείχνει όλα τα διαδραστικά στοιχεία ελέγχου της αναπαραγωγής.
- YouTubePlayer.PlayerStyle.MINIMAL - Στυλ που δείχνει την χρονική γραμμή, και τα διαδραστικα στοιχεια (αναπαραγωγής,παύσης).

Πίνακας 5-1 Στυλ εμφάνισης Player

CHROMELESS STYLE	DEFAULT STYLE	MINIMAL STYLE
		

5.5 Αναπαραγωγή βίντεο στο Android με χρήση VideoView και MediaController κλάσεις.

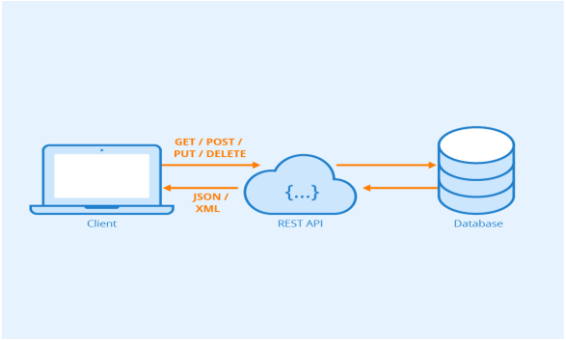
Το android SDK περιλαμβάνει δύο κλάσεις που κάνει την υλοποίηση της αναπαραγωγή βίντεο σε android συσκευές αρκετά εύκολη όταν γίνεται ανάπτυξη μιας εφαρμογής. Ένας τρόπος για αναπαράσταση βίντεο μέσα σε μια Android εφαρμογή είναι η χρησιμοποίηση της VideoView Κλάσης . Η videoview Κλάση συνδυάζει ένα media player (Την MediaPlayer κλάση)για την αποκωδικοποίηση και την αναπαραγωγή του βίντεο με το SurfaceView για την πραγματική απεικόνιση του βίντεο στην οθόνη .Το videoview συστατικό το οποίο, όταν προστίθεται στο layout μιας δραστηριότητας(Activity), παρέχει μια επιφάνεια όπου το βίντεο μπορεί να παιχτεί. Ωστόσο δεν παρέχει πολλά χαρακτηριστικά ή κάποιου είδους προσαρμογής, η videoview κλάση υλοποιεί την στοιχειώδη συμπεριφορά για την αναπαραγωγή ενός βίντεο σε μια εφαρμογή ,μπορούμε να κάνουμε αναπαραγωγή τα πολυμεσικά αρχεία που βρίσκονται στην εσωτερική ή εξωτερική μνήμη(SD κάρτα) ή ακόμα και αρχεία που είναι στο διαδίκτυο. Το android αυτή τη στιγμή υποστηρίζει το ακόλουθα video format:

- H.263
- H.264 AVC
- H.265 HEVC
- MPEG-4 SP
- VP8
- VP9

Αν σε ένα βίντεο έχει γίνει αναπαραγωγή χρησιμοποιώντας την VideoView Κλάση, ο χρήστης δεν θα έχει κανένα έλεγχο για την αναπαραγωγή του βίντεο ως που το βίντεο φτάσει στο τέλος τους. Αυτό το

ζήτημα μπορεί να διευθετηθεί τοποθετώντας ένα στιγμιότυπο της MediaController Class στο videoView στιγμιότυπο. Αφού γίνει αυτό ο Media Controller μας παρέχει μια σειρά από ελέγχους επιτρέποντας στον χρήστη να διαχειρίζεται την αναπαραγωγή(όπως την προσωρινή παύση του βίντεο ,πηγαίνοντας προς τα εμπρός και προς τα πίσω το βίντεο). [29] [30]

5.6 REST API



Εικόνα 5-9 REST API

5.6.1 RESTful Api

Ένα RESTful API αποτελεί μια διεπαφή προγραμματισμού εφαρμογών (application programming interface), και καθορίζει ένα σύνολο κανόνων και μηχανισμών με τους οποίους μια εφαρμογή ή ένα στοιχείο (component-microservice) αλληλεπιδρά με άλλα προγράμματα λογισμικού (User Interfaces, external services, third-party APIs κλπ).

Όταν καλείται ένα RESTful API, ο server θα μεταφέρει στον client μια αναπαράσταση της κατάστασης του ζητούμενου πόρου. Τέτοια είδους συμπεριφορά βασίζεται στο μοντέλο REST (Representational State Transfer).

5.6.2 Χαρακτηριστικά αρχιτεκτονικής REST

Τα κύρια χαρακτηριστικά της αρχιτεκτονικής REST είναι τα ακόλουθα:

- Πρόκειται για μια κατανεμημένη αρχιτεκτονική client-server.
- Η εφαρμογή στην πλευρά του πελάτη (client) βρίσκεται σε διάφορες καταστάσεις (application state), όπως τις αντιλαμβάνεται ο χρήστης.
- Στη μεριά του εξυπηρετητή (server) αποθηκεύονται οι πόροι (resources) που διαχειρίζεται η εφαρμογή. Οι πόροι βρίσκονται και αυτοί σε διάφορες καταστάσεις (resource state), ως προς το περιεχόμενό τους.
- Ο χρήστης μέσω υπερμέσων (hypermedia), όπως π.χ. οι σύνδεσμοι HTML, προκαλεί αλλαγές στην κατάσταση των πόρων του εξυπηρετητή.
- Με τη σειρά του ο εξυπηρετητής στέλνει στον πελάτη αναπαραστάσεις της κατάστασης των πόρων, προκαλώντας την αλλαγή της κατάστασης της εφαρμογής του πελάτη.

Στον παγκόσμιο ιστό η αρχιτεκτονική REST υλοποιείται μέσω των χαρακτηριστικών του πρωτοκόλλου HTTP και των μεθόδων που αυτό υποστηρίζει: [31]

Πίνακας 5-2 HTTP Μέθοδοι

Μέθοδος:	Χρήση:
GET	Προσπέλαση ενός απλού πόρου ή μιας συλλογής. Η μέθοδος GET δεν προκαλεί αλλαγές στην κατάσταση των πόρων (safe) και πολλαπλές συνεχόμενες αιτήσεις GET για τον ίδιο πόρο θα φέρουν το ίδιο αποτέλεσμα (idempotent)
POST	Δημιουργία νέου πόρου μέσα σε μια συλλογή. Η μέθοδος POST προκαλεί αλλαγές στην κατάσταση των πόρων.
PUT	Αντικατάσταση της πληροφορίας ενός πόρου. Όμοιες συνεχόμενες αιτήσεις PUT για τον ίδιο πόρο θα έχουν το ίδιο αποτέλεσμα (idempotent). Χρησιμοποιείται και για τη δημιουργία νέων πόρων.

DELETE	Διαγραφή ενός πόρου. Πολλαπλές συνεχόμενες αιτήσεις DELETE για τον ίδιο πόρο θα έχουν το ίδιο αποτέλεσμα (idempotent).

Έτσι λοιπόν για κάθε resource (πόρο) είναι δυνατόν να υπάρχουν πολλές διαφορετικές αναπαραστάσεις, όπου οι πιο συχνά χρησιμοποιούμενες είναι οι απλές HTML σελίδες και οι σελίδες που ακολουθούν τις μορφοποιήσεις XML και JSON. Αυτό επιτρέπει σε κάθε πελάτη να μπορεί κατά την αποστολή ενός αιτήματος για την τρέχουσα κατάσταση ενός πόρου, να ζητήσει και συγκεκριμένη μορφοποίηση για την επιστρεφόμενη αναπαράσταση. Βεβαίως υπάρχει περίπτωση ο εξυπηρετητής να μην μπορεί να επιστρέψει την αναπαράσταση στην αιτούμενη μορφοποίηση, ενημερώνοντας τον πελάτη με κατάλληλο μήνυμα του HTTP πρωτοκόλλου. [32]

5.7 JSON

Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και το γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 Έκδοση 3η - Δεκέμβριος 1999. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

Το JSON είναι χτισμένο σε δύο δομές:

- Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object, καταχώριση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.
- Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία. [33]

5.7.1 Μόρφες JSON

- Αντικείμενο(Object)είναι ένα άτακτο σύνολο από ζευγάρια ονόματων/τιμών. Ένα αντικείμενο ξεκινάει με {αριστερο *άγκιστρο* και τελειώνει με δεξί *άγκιστρο*}. Κάθε όνομα ακολουθείται από **:** *άνω-κάτω τελεία* και τα ζευγάρια ονόματος/τιμής χωρίζονται από **,** *κόμμα*, παράδειγμα:
  {

  "firstName":"Tsatsos",

  "lastName":"Alex"

  }

- Ένας πίνακας (array) είναι μια συλλογή από τιμές σε σειρά. Ένας πίνακας (array) ξεκινάει με **[**αριστερή αγκύλη και τελειώνει με **]**δεξιά αγκύλη. Οι τιμές χωρίζονται με **,** κόμμα, παράδειγμα:
  "employees":[

  {

  "firstName":"Tsatsos", "lastName":"Alex"

  },

  {

  "firstName":"Mparkolias", "lastName":"Alexandros"

  }

  ]

- Μία τιμή μπορεί να είναι string μέσα σε διπλά quotes, ή αριθμός (number), ή **true** ή **false** ή **null**, ή αντικείμενο (object) ή πίνακας.
  - String,παράδειγμα:
    {

    "name":"Alex"

    }
  - Λογικές τιμές(true-false),παράδειγμα:
    {

    “distinction”:false

    }
  - Κενό,παράδειγμα:
    {

    “name”:null

    }
  - Αριθμός,παράδειγμα:
    {

    “number”:11

    }

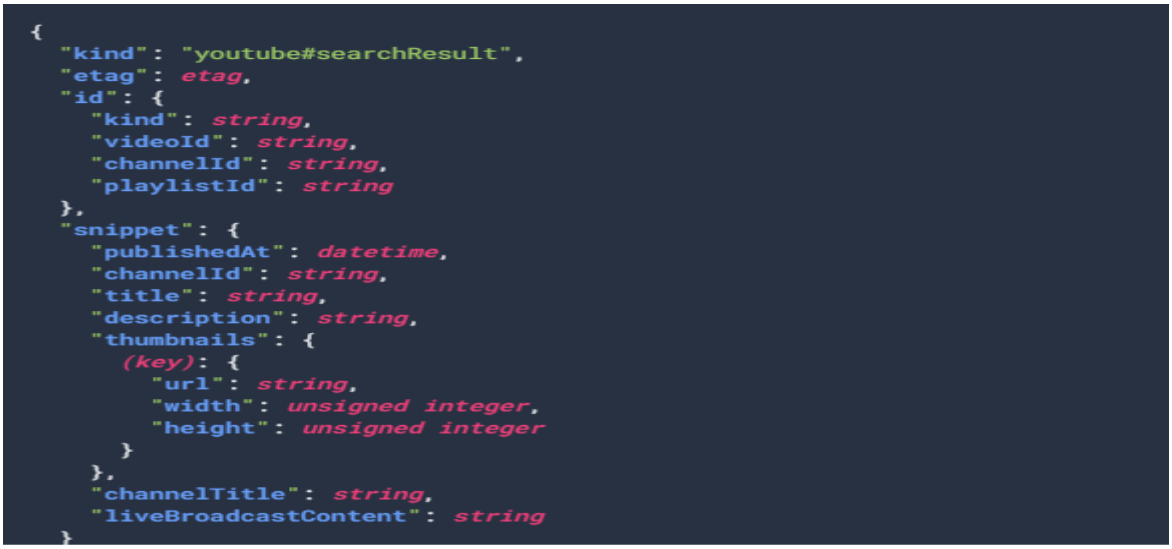
5.8 YouTube Data Api

Το YouTube Data api μας επιτρέπει να ενσωματώσουμε λειτουργίες που εκτελούνται στην YouTube ιστοσελίδα στην εφαρμογή μας. Ένας πόρος (resource) αντιπροσωπεύει έναν τύπου αντικείμενου που είναι κομμάτι της Youtube όπως ένα video ή μια λίστα. Κάθε πόρος αναπαριστάται ως ένα json αντικείμενο. [34]

5.8.1 Τύπος πόρου Search

Ένα search αποτέλεσμα περιέχει πληροφορίες για ένα YouTube video, κανάλι, λίστα που ταιριάζει με τις παραμέτρους αναζήτησης που προσδιορίζονται σε ένα Api αίτημα(request). [35]

Η παρακάτω εικόνα δείχνει την απεικόνιση των επιστρεφόμενων αποτελεσμάτων αναζήτησης σε JSON μορφοποίηση:



Εικόνα 5-10Απεικόνιση του πόρου σε json

Πίνακας 5-3 Επεξήγηση ιδιοτήτων του πόρου search

kind	string Προσδιορίζει τον τύπο του API πόρου.
etag	etag Το etag αυτού του πόρου.

id	object Το id αντικείμενο περιέχει πληροφορία η οποία μπορεί να χρησιμοποιηθεί για την μοναδική αναγνώριση του πόρου.
id.kind	string Ο τύπους του API πόρου.
id.videoId	string Αναπαριστά το id που η YouTube χρησιμοποιεί για την μοναδικη αναγνωριση του video.
id.channelId	string Αναπαριστά το id που η YouTube χρησιμοποιεί για την μοναδική αναγνώριση του καναλιού.
id.playlistId	Αναπαριστά το id που η YouTube χρησιμοποιεί για την μοναδικη αναγνωριση της λίστας αναπαραγωγής.
snippet	object Το snippet αντικείμενο περιέχει λεπτομέρειες για τα αποτελέσματα για τα οποία έγινε ή αναζήτηση, όπως ο τίτλος και η περιγραφή του

	video.
snippet.publishedAt	datetime Η ημερομηνία και η ώρα δημιουργίας του πόρου.
snippet.channelId	string Τιμή την οποία την χρησιμοποιεί η YouTube για την μοναδική αναγνώριση του καναλιού που δημοσίευσε το πόρο.
snippet.title	string Ο τίτλος από το αποτέλεσμα της αναζήτησης.
snippet.description	string Η περιγραφή από το αποτέλεσμα της αναζήτησης.
snippet.thumbnails	object Μια αντιστοιχίζει απο thumbnails εικόνες που σχετίζονται με το αποτέλεσμα της αναζήτησης. Για κάθε αντικείμενο σε αυτή της αντιστοιχίσει το κλειδι (key) είναι το όνομα απο την thumbnail εικόνα, και η τιμή είναι ένα αντικείμενο που περιέχει άλλες πληροφορίες σχετικά με την

	thumbnail εικόνα.
snippet.thumbnails.(key)	object Οι έγκυρες τιμές είναι:
	● default - Η προεπιλεγμένη thumbnail εικόνα για ένα video. Οι διαστασεις της είναι 120px x 90px ● medium - Μια thumbnail εικόνα υψηλότερης εύκρινας για ένα video. Οι διαστάσεις της οποίας είναι 320px x 180px. ● high - Μια thumbnail εικόνα υψηλότερης εύκρινας για ένα video. Οι διαστάσεις της οποίας είναι 480px x 360px.
snippet.thumbnails.(key).url	string Το URL της εικόνας.
snippet.thumbnails.(key).width	unsigned integer Το πλάτος της εικόνας.

snippet.thumbnails.(key).height	unsigned integer Το ύψος της εικόνας.
snippet.channelTitle	string Ο τίτλος του καναλιού.
snippet.liveBroadcastContent	string Μία ένδειξη για το εάν ένα video ή ένα κανάλι έχει ζωντανή μετάδοση περιεχομένου.

Το API υποστηρίζει την μεθόδο Search by Keyword function για αναζήτηση:

- list-Μας επιστρέφει μια συλλογή αποτελεσμάτων αναζήτησης που ταιριάζουν στους παραμέτρους ερωτήσεων (query parameters) στο συγκεκριμένο αίτημα του API.

5.8.2 HTTP Request:

Το URL στο οποίο θα κάνουμε αιτήματα για να πάρουμε αποτελέσματα μέσο του REST API είναι: https://www.googleapis.com/youtube/v3/search, στο οποίο μπορούμε να προσθέσουμε διάφορες παραμέτρους (παρακάτω επισυνάπτεται μια λίστα με τις διαθέσιμες παραμέτρους).

Παράμετροι:

Πίνακας 5-4 Παράμετροι Αιτήματος

part	string Η παράμετρος part καθορίζει μια λίστα η οποία είναι χωρισμένη με κόμμα για να μπορούμε να χρησιμοποιήσουμε επιπλέον ιδιότητες πόρων αναζήτησης όπου η απόκριση του API θα εμπεριέχει. Ορίζουμε την τιμή της παραμέτρους

	part να ισούται με το snippet(part=snippet).
maxResult	unsigned integer Η maxResult παράμετρος καθόριζει τον μέγιστο αριθμό απο αντικείμενα που πρέπει να επιστραφούν στο αποτέλεσμα της αναζήτησης. Αποδέκτες τιμές είναι το εύρος από 0 μέχρι 50 η προεπιλεγμένη τιμή είναι το 5. Παράδειγμα χρήσης (maxResult=25).
q	string Η q παράμετρος καθορίζει τον όρο ερωτήματος για το οποιο κάνουμε αναζήτηση. Επίσης στο αίτημας μας, μπορούμε να προσθέσουμε Boolean τελεστές [NOT(-),OR(|)] για να αποκλείσουμε κάποια video ή για την αναζήτηση video που σχετίζεται με τον ένα ή τον άλλον όρο. Για παράδειγμα για την αναζήτηση video που ταιριάζουν στους όρους metal ή rock ορίζουμε την τιμή της πάμετρου q με τιμή metal|rock(q=metal|rock). Με τον ίδιο τρόπο αν θέλουμε να κάνουμε αναζήτηση για video που ταιριάζουν με τους metal ή rock όρους αλλα δεν θέλουμε να υπάρχει αντιστοίχιση για τον pop όρο, ορίζουμε για τιμη της παραμετρο q την metal|rock-pop(q=metal|rock-pop).
type	string Η type παράμετρος περιορίζει το εύρος αναζήτησης σε ένα συγκεκριμένο τύπο. Αποδεκτές τιμές για την παράμετρο τύπου είναι το channel ή playlist και το video. Παράδειγμα χρήσης (type=video).

pageToken	string Η pageToken παράμετρος προσδιορίζει μια συγκεκριμένη σελίδα ως μέρος των επιστρεφόμενων αποτελεσμάτων. Στην απόκριση του API , η nextPageToken και prevPageToken ιδιότητες μπορούν να χρησιμοποιηθούν για την ανακτήσει επιπλέον σελίδων αναζήτησης.

5.8.3 HTTP Response

Εκτελώντας λοιπόν το URL https://www.googleapis.com/youtube/v3/search αν η αναζήτηση που κάναμε ήταν επιτυχής, η απόκριση που θα λάβουμε είναι σαν τη παρακάτω εικόνα **5-11** όπου η δομής της απόκρισης του σώματος(Response body) είναι σε μορφοποίηση json. [36]

```
{
  "kind": "youtube#searchListResponse",
  "etag": etag,
  "nextPageToken": string,
  "prevPageToken": string,
  "regionCode": string,
  "pageInfo": {
    "totalResults": integer,
    "resultsPerPage": integer
  },
  "items": [
    search Resource
  ]
}
```

Εικόνα 5-11 Απόκριση

Πίνακας 5-5 Επεξήγηση των ιδιοτήτων για το σώμα απόκρισης (Response body).

kind	string Προσδιορίζει τον τύπου του API πόρου.

etag	etag Το etag από αυτόν τον πόρο.
nextPageToken	string Το προσδιοριστικό που μπορεί να χρησιμοποιηθεί ως τιμή της παραμέτρου pageToken για την ανάκτηση της επόμενης σελίδας που περιέχει επιπλέον αποτελέσματα.
prevPageToken	string Το προσδιοριστικό που μπορεί να χρησιμοποιηθεί ως τιμή της παραμέτρου pageToken για την ανάκτηση της προηγούμενης σελίδας με τα αποτελέσματα.
regionCode	string Ο κωδικός της περιοχής από όπου έγινε το ερώτημα αναζήτησης, η τιμή του οποίου είναι υπό την μορφή κωδικού ISO χώρας των δύο γραμμάτων που προσδιορίζουν μια περιοχή. Παράδειγμα(“GR”).
pageInfo	object Το pageInfo αντικείμενο ενσωματώνει πληροφορίες για την σελιδοποιήση σχετικά με το

	σύνολο των αποτελεσμάτων.
pageInfo.totalResults	integer Ο συνολικός αριθμός αποτελεσμάτων.
pageInfo.resultPerPage	integer Ο αριθμός των αποτελεσμάτων .
items[]	list Μια λίστα αποτελεσμάτων που ταιριάζει με τα κριτήρια αναζήτησης.

5.9 Musixmatch lyrics API

Το Musixmatch lyrics API είναι μια υπηρεσία που μας επιτρέπει να αναζητήσουμε και να ανακτήσουμε στίχους τραγουδιών. Η βάση δεδομένων που διαθέτει περιέχει πάνω από 14 εκατομμύρια στίχους σε 50 διαφορετικές γλώσσες. Η τρέχουσα έκδοση του API είναι η 1.1 και το ριζικό URL είναι το https://api.musixmatch.com/ws/1.1/ .

5.9.1 API μέθοδος matcher.lyrics.get

Με την matcher.lyrics.get δημιουργούμε ένα GET HTTP ερώτημα έτσι ώστε να λάβουμε τους στίχους βασισμένη στο όνομα του τραγουδιού και όνομα του καλλιτέχνη που τα παίρνει ως παραμέτρους.

Δημιουργώντας λοιπόν το GET ερώτημα στο URL:

https://api.musixmatch.com/ws/1.1/matcher.lyrics.get?format=json&callback=callback&q_track=Children of the sun &q_artist=Nightstalker&apikey=73d2a5f27d171a616f3b83ef5c60bf6e η μορφή απάντησης που θα πάρουμε θα είναι σε JSON και θα περιέχει του στίχους για το όνομα το τραγουδιού children of the sun (παράμετρος q_track) από το συγκρότημα Nightstalker (παράμετρος q_artist).

Εικόνας της εκτέλεσης του του παραπάνω URL σε JSON μορφή εικόνα 5-12 [37]

Εικόνα 5-12 Εκτέλεση του url σε Json μορφή

5.10 Retrofit

Η Retrofit είναι μια REST Client βιβλιοθήκη, που χρησιμοποιείται από την Java και το Android για την δημιουργία HTTP αιτημάτων (request) και για την επεξεργασία HTTP αποκρίσεων (response). Δημιουργήθηκε απο την Square,Inc. Για να μπορούμε να την χρησιμοποιήσουμε πρέπει την προσθέσουμε σαν dependency στο bulid.gradle αρχείο του επιπέδου εφαρμογής.
Παράδειγμα εισαγωγής βιβλιοθήκης Retrofit στο bulid.gradle:

Implementation 'com.squareup.retrofit2:retrofit:2.4.0'

5.10.1 Retrofit Αιτήματα(Request)

Η Retrofit μας παρέχει μια αρκετές δυνατότητες για το χειρισμό των αιτημάτων και των παραμέτρων τους. Με αυτές τις δυνατότητες εισάγουμε τα αιτήματα που θέλουμε να κάνουμε με τα annotations και τα σύμβολα αντικατάστασης και η Retrofit αναλαμβάνει τα υπόλοιπα για εμάς. Επίσης η Retrofit παρέχει την δυνατότητα για σύγχρονα και ασύγχρονα αιτήματα. [38]

5.10.1.1 Δήλωση API με την Retrofit

Η Retrofit χρησιμοποιεί σχολιασμούς (annotations) στις μεθόδους Java διεπαφών(Interface) για να περιγράψει το χειρισμό των αιτημάτων. Ένα απο το πρώτα πράγματα που πρέπει να ορίσουμε είναι οι μέθοδοι των Http αιτημάτων όπως η GET, POST, DELETE, PUT και τα λοιπά. Η Retrofit παρέχει annotations για κάθε μια request μέθοδο. Επίσης πρέπει να προσθέσουμε το σχετικό Url.

Παράδειγμα:

```
public interface TaskService {
 @GET("/relative/url/path")
 void method();
}
```

Ο παραπάνω ορισμός μεταφράζεται απο την Retrofit σε GET αίτημα (request) με το σχετικό Url /relative/url/path στο τελικό σημείου το πόρου. Η Retrofit κλάση ενσωματώνει το απόλυτο Url και το

σχετικό url συνδυάζεται με το απόλυτο Url. Η Retrofit συνθέτει το URL τελικού αιτήματος και από τις δύο τιμές της συμβολοσειράς.

5.10.1.2 Παράμετρος ερωτήματος(Query parameter)

Όταν εργαζόμαστε με APIs πολλές φορές θέλουμε να φιλτράρουμε τους πόρους για ένα συγκεκριμένο σύνολο. Μια κοινή περίπτωση χρήσης είναι η επιλογή ενός στοιχείου που προσδιορίζεται από μια μοναδική τιμή id.

Οι παράμετροι ερωτήματος είναι ένας κοινός τρόπος για μετάδοση απλών δεδομένων ανάμεσα σε πελάτες και διακομιστές. Κοινά παραδείγματα των παραμέτρων ερωτήματος είναι ένα αιτήματα για επιλογή συγκεκριμένων στοιχείων, ταξινόμηση των στοιχείων. Μπορούμε να τοποθετήσουμε τις παραμέτρους ερωτήματος κατευθείαν μέσα στο πόρου το σχετικού url. Στο παρακάτω παράδειγμα κάνουμε ένα αίτημα στο API για την εργασία με id=123:

```
public interface TaskService {
 @GET("/tasks?id=123")
 Call<Task> getTask123();
 }
```

Έτσι λοιπόν η απόκριση που θα λάβουμε θα είναι τα δεδομένα για την εργασία με id το 123.Το παραπάνω παράδειγμα όμως στο πραγματικό κόσμο των Apis δεν είναι τόσο συνηθισμένο διότι χρειαζόμαστε μια πιο δυναμική συμπεριφορά για τα αιτήματα που θα περιλαμβάνουν μεταβαλλόμενες τιμές για τις παραμέτρους ερωτήματος, για αυτόν τον λόγο μπορούμε να προσθέσουμε το @Query(“key”) annotation(σχολιασμό). Το κλειδί (key) τίμη μέσα στο @Query σχολιασμό(annotation) ορίζει το όνομα της παραμέτρους μέσα στο url. Η Retrofit αυτόματα προσθέτη αυτούς τους παραμέτρους στο url που θα πραγματοποιήθει το αίτημα.

```
public interface TaskService {
 @GET("/tasks")
 Call<Task> getTask(@Query("id") long taskId);
 }
```

Η μέθοδος getTask() απαιτεί την παράμετρο taskId. Αυτή η παράμετρος θα αντιστοιχιστεί απο την Retrofit σε ορισμένο όνομα παραμέτρου που καθορίζεται στο σχολιασμό (annotation) @Query. Σε αυτή την περίπτωση, το όνομα της παραμέτρου είναι το id το οποίο θα έχει ως αποτέλεσμα στο κομμάτι του url αιτήματος το: /tasks?id=<taskId>.Φυσικά μπορούμε να προσθέσουμε περισσότερες παραμέτρους σε ένα αίτημα. Παράδειγμα :

```
public interface TaskService {
 @GET("/tasks")
 Call<Task> getTask(
 @Query("id") long taskId,
 @Query("order") String order,
 @Query("page") int page);
 }
```

Η προκύπτουσα διεύθυνση URL του αιτήματος μοιάζει με την : /tasks?id=<taskId>&order=<order>&page=<page>

Μέσα στις διεπαφές απαιτείται για κάθε μέθοδο να είναι τύπου Call, η επιστρεφόμενη τιμή περιτυλίγει την απόκριση σε ένα Call αντικείμενο που μέσα στα Generics περιλαμβάνεται ο τύπους του αποτελέσματος. Στο παραπάνω κώδικα περιγράφεται μόνο ο ορισμός του API από την μεριά του πελάτη (client side). Μέσο της Retrofit κλάσης θα γίνει και η δημιουργία της υλοποίησης της διεπαφη. Ετσι λοιπόν αρχικοποιούμε ένα αντικείμενο της κλάσης Retrofit μαζί με τη κύρια διεύθυνση url του API.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.doyourtasks.com")
    .build();
TaskService service = retrofit.create(TaskService.class);
```

Η δήλωση retrofit.create επιστρέφει μια υλοποίηση για την TaskService διεπαφή παρέχοντας τις καθορισμένες μεθόδους.

5.10.2 Σύγχρονη και ασύγχρονη εκτέλεση.

Η retrofit υποστηρίζει ασύγχρονη και σύγχρονη εκτέλεσης των ερωτημάτων. Για την ασύγχρονη εκτέλεση θα καλέσουμε την μέθοδο enqueue().

```
Call< Task > call = service.getTask(123);
call.enqueue(new Callback<Task>() {
  @Override
  public void onResponse(Call<Task> call, Response<TaskS> response) {
   //do something
  }
  @Override
  public void onFailure(Call<Task> call, Throwable t) {
    //do something
  }
});
```

Η εκτέλεση με σύγχρονο γίνεται με την μέθοδο execute() της retrofit. Η εκτέλεση αυτής της μεθόδου θα γίνει στο τρέχων νήμα της εφαρμογής και εάν συμβεί κάποιο λάθος θα έχουμε ένα IOException.

```
Response<Task> response =service.getTask(123).execute();
```

5.10.3 Χειρισμός απόκρισης

Κάθε φορά που κάνουμε αιτήματα κατά κανόνα παίρνουμε και μια απόκριση. Μια συνηθισμένη αναπαράσταση των δεδομένων που λαμβάνονται από ένα διακομιστή είναι σε JSON. Έτσι λοιπόν αν ένας διακομιστής στέλνει δεδομένα σε μια java εφαρμογή σε μορφή JSON θα πρέπει να γίνει κάποια μετατροπή των JSON αντικειμένων σε Java αντικείμενα για να είναι γίνει με επιτυχία η επικοινωνία τους. Η Retrofit υποστηρίζει αρκετούς μετατροπείς όπως τους Gson, Moshi, Jackson οι οποίοι κάνουν

αυτή την μετατροπή - αντιστοίχιση των αντικειμένων. Έτσι λοιπόν για να μπορούμε να χρησιμοποιήσουμε το Gson μετατροπέα στην Retrofit πρέπει να προσθέσουμε στο αντικείμενο της Retrofit την μέθοδο addConverterFactory(GsonConverterFactory.create()),

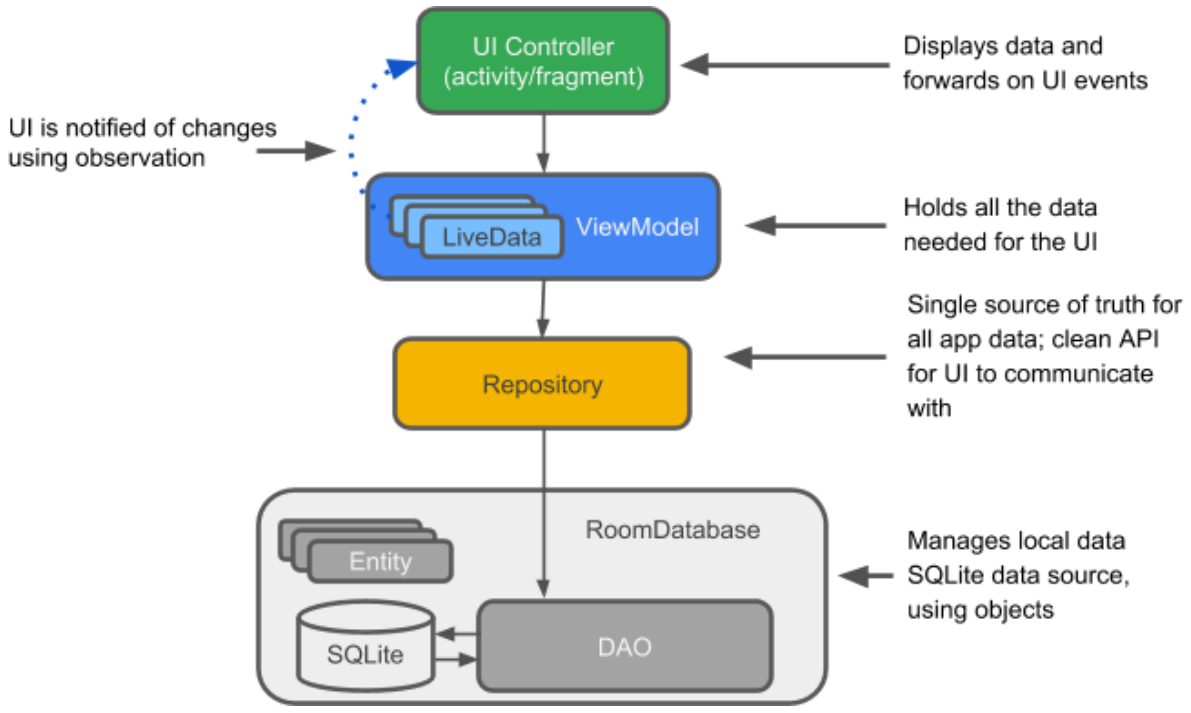
```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.doyourtasks.com").
addConverterFactory(GsonConverterFactory.create())
    .build();
```

Για παράδειγμα αν έχουμε μια μεθόδου με επιστρεφόμενο τύπο τον List<Task> ,η Gson θα αντιστοίχιση τα Json δεδομένα απόκρισης σε αντικείμενα της κλάσης Task.

5.11 Στοιχεία Αρχιτεκτονικής Android (Android Architecture Components)

Τα στοιχεία αρχιτεκτονικής στο android είναι μια συλλογή από βιβλιοθήκες οι οποίες μας βοηθούν να δημιουργήσουμε ισχυρές, διατηρήσιμες και ελέγξιμες εφαρμογές.

5.11.1 Εισαγωγή στα στοιχεία Αρχιτεκτονικής.



Εικόνα 5-13 Εισαγωγή στα στοιχεία αρχιτεκτονικής

- Οντότητα(Entity) - Η entity είναι μια κλάση που περιέχει την σήμανση @Entity και περιγράφει ένα πίνακα βάσης δεδομένων.
- SQLite database - Σε μια συσκευή, τα δεδομένα αποθηκεύονται στην SQLite βάση δεδομένων. Η room βιβλιοθήκη δημιουργεί και διατηρεί αυτή την βάση δεδομένων.

- DAO (Αντικείμενο πρόσβασης δεδομένων) Data access object - Μια αντιστοίχιση SQL ερωτημάτων σε συναρτήσεις. Όταν χρησιμοποιούμε το DAO απλά καλούμε τις μεθόδους και η Room αναλαμβάνει την υπόλοιπη διαδικασία.
- Room database - Επίπεδο βάσης δεδομένων που βρίσκεται πάνω από την SQLite. Απλοποιεί τις εργασίες με την βάση δεδομένων και εξυπηρετεί ως σημείο πρόσβασης για την SQLite. Η Room χρησιμοποιεί το Dao για την εκτέλεση ερωτημάτων στην SQLite.
- Repository - Μια κλάση που δημιουργείται για την διαχείριση πολλών πηγών δεδομένων.
- ViewModel - Παρέχει δεδομένα στο User Interface(UI) και λειτουργεί ως κέντρο επικοινωνίας με το Repository και το UI. Τα αντικείμενα ViewModel επιβιώνουν όταν τα Activities/Fragments δημιουργούνται εκ νέου.
- liveData - Μια κλάση που διατηρεί δεδομένα και ακολουθεί το πρότυπο παρατηρητή (observer pattern), που σημαίνει ότι τα δεδομένα μπορούν να παρακολουθούνται. Επίσης διατηρεί /αποθηκεύει την τελευταία έκδοση των δεδομένων. Ακόμα ενημερώνει τον παρατηρητή όταν τα δεδομένα υποστούν μια αλλαγή. Τα liveData γνωρίζουν τον κύκλο ζωής, έτσι διαχειρίζονται αυτόματα τη διακοπή και την επανάληψη της παρακολούθησης βασισμένα στην κατάσταση της Activity/fragment που έχουν τεθεί σε παρακολούθηση. Τα στοιχεία του UI παρακολουθούν τα σχετικά δεδομένα.

5.11.1.1 Entity (οντότητα).

Μια οντότητα (Entitiy) αποτελεί τα δεδομένα μιας εφαρμογής που έχει βάση δεδομένων. Η room μοντελοποιεί την βάση δεδομένων Sqlite, και υλοποιείται με την SQlite στο backend. Η SQlite βάση δεδομένων αποθηκεύει τα δεδομένα σε γραμμές και στήλες. Κάθε μια γραμμή αναπαριστά μια οντότητα (Entity) ή (μια εγγραφή) και μια στήλη αναπαριστά μια τιμή για αυτήν την οντότητα.

Παράδειγμα για μια οντότητα(Entitiy) ,είναι η οντότητα για ένα άτομο, το οποίο έχει ένα μοναδικό id, με ένα όνομα, επίθετο και ένα email. Παράδειγμα:

ID	**FirstName**	**LastName**	**Email**
123956	Alex	Tsatsos	theo.alex.tsatsos@gmail.com

123970	Alex	Mparkolias	alexandros.mparkolias@gmail.com

Όταν θέλουμε να ορίζουμε μια οντότητα (Enitity) ή πολλές οντότητες δημιουργούμε μια κλάση/κλάσεις. Κάθε αντικείμενο μιάς κλάσης απεικονίζει μια γραμμή, και κάθε στήλη αναπαρίσταται από μεταβλητές μέλη της κλάσης.

Παράδειγμα κώδικα που αναπαριστά την οντότητα (Entity) άτομο:

```
public class Person {
   private int id;
   private String firstName;
   private String lastName;
   private String email;
}
```

7.1.1.1 Entity Annotations.

Για την Room προκειμένου να χρησιμοποιήσουμε μια οντότητα, πρέπει να δώσουμε στην Room κάποιες πληροφορίες οι οποίες σχετίζονται με το περιεχόμενο της Java κλάσης της οντότητας(για παράδειγμα της κλάσης Person), για αυτό που θέλουμε να αναπαρίσταται στον πίνακα της βάσης δεδομένων, το κάνουμε αυτό με τα annotations.

Λίστα με κάποια annotations:

- @Entity(tableName=”Person_table”) - Κάθε @Entity απεικονίζει μια οντότητα στον πίνακα. Επίσης μπορούμε να ορίζουμε το όνομα του πίνακα στην περίπτωση που θέλουμε να είναι διαφορετικό από το όνομα της κλάσης (στο παραπάνω ονομάσαμε τον πίνακα “Person_table”).
- @PrimaryKey-Κάθε μια οντότητα χρειάζεται ένα κύριο κλειδί. Για την αυτόματη-δημιουργία μοναδικού κλειδιού για κάθε οντότητα, προσθέτουμε το autoGenerate=true.
- @NonNull - Δηλώνει ότι μια παράμετρος, ένα πεδίο ή η τιμή της επιστροφής μίας μεθόδου δεν μπορεί να είναι null. Το κύριο κλειδί σε έναν πίνακα πρέπει να δηλώνεται και με αυτόν το annotation, όπως και επίσης τα υποχρεωτικά πεδία σε μια γραμμή.
- @ColumnInfo(name=”last_Name”) - Ορίζουμε το όνομα της στήλης του πίνακα, στη περίπτωση που θέλουμε να είναι διαφορετικό από την μεταβλητή μέλος της κλάσης.

Κάθε πεδίο που αποθηκεύεται στη βάση πρέπει είτε να είναι public, είτε να έχει μεθόδους get() έτσι ώστε η Room να έχει πρόσβαση σε αυτό.

Παράδειγμα κώδικα που περιέχει τα πιο πάνω annotations και getter, setters μεθόδους.

```
@Entity(tableName="Person_table")
```

```
public class Person {
    @PrimaryKey(autoGenerate=true)
    private int id;
    @ColumnInfo(name = "first_name")
    private String firstName;
    @ColumnInfo(name = "last_name")
    private String lastName;
    @ColumnInfo(name="email")
    private String email;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;}}
```

5.11.1.1.1 Σχέσεις μεταξύ οντοτήτων.

Χρησιμοποιώντας τα annotations μπορούμε να ορίσουμε συσχετίσεις μεταξύ των οντοτήτων(Entities).

Υπάρχουν τρεις τύποι σχέσεων:

- Ένα προς ένα.
- Ένα προς πολλά.
- Πολλά προς πολλά.
  [39]

5.11.1.1.2 Σχέση ένα προς ένα (One to One)

Για παράδειγμα έστω ότι θέλουμε να κατασκευάσουμε ένα πίνακα που να περιέχει διευθύνσεις και κάθε διεύθυνση θα έχει ως πεδία της μια πόλη, μια οδό, μια πολιτεία και ένα μοναδικό id.

Επίσης θέλουμε να δημιουργήσουμε και ένα πίνακα με πελάτες και κάθε πελάτης έχει για πεδία του ένα μοναδικο id, ένα όνομα και την διεύθυνση της κατοικίας του που αποτελεί ξένο κλειδί και αναφέρεται στο id του πίνακα με τις διευθυνσης. Αν λοιπόν θέλουμε να απεικονίσουμε ότι κάθε διεύθυνση θα ανήκει σε ένα και μόνο πελάτη τότε θα έχουμε μια σχέση ένα προς ένα.

Εικόνα 5-14 σχέση ένα προς πολλά.

Παράδειγμα κώδικα για την Οντότητα (Entity) Address:

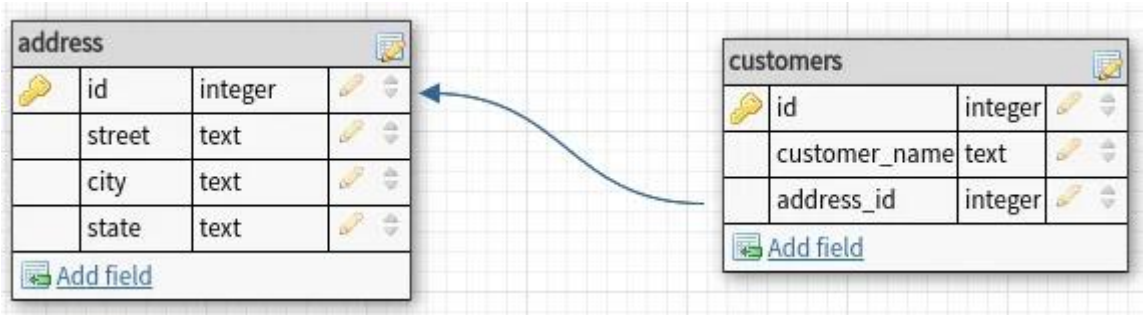
```
@Entity(tableName = "address")
public class Address {
   @PrimaryKey(autoGenerate = true)
   @ColumnInfo(name = "id")
   public long addressId;
   public String street;
   public String city;
   public String state;
}
```

Ακόμα πρέπει να ορίσουμε τις σχέσεις μεταξύ των customer και address αντικειμένων.

Η Room μας επιτρέπει να ενσωματώσουμε περιορισμό ξένου κλειδιού μεταξύ οντοτήτων.

Παράδειγμα κώδικα δημιουργίας οντότητας Customer με το ορισμό της συσχέτισης που έχει προς την Address οντότητα:

```
@Entity(tableName = "customers" ,foreignKeys =
@ForeignKey(entity = Address.class,parentColumns = "id",childColumns = "address_id"))
public class Customer {
   @PrimaryKey(autoGenerate = true)
   @ColumnInfo(name = "id")
   public long customerId;
   @ColumnInfo(name = "customer_name")
   public String customerName;
   @ColumnInfo(name = "address_id")
   public long addressId;}
```



Εικόνα 5-15 Παραδείγματα πινάκων Address και Customer

5.11.1.1.3 Σχέση ένα προς πολλά (One to many)

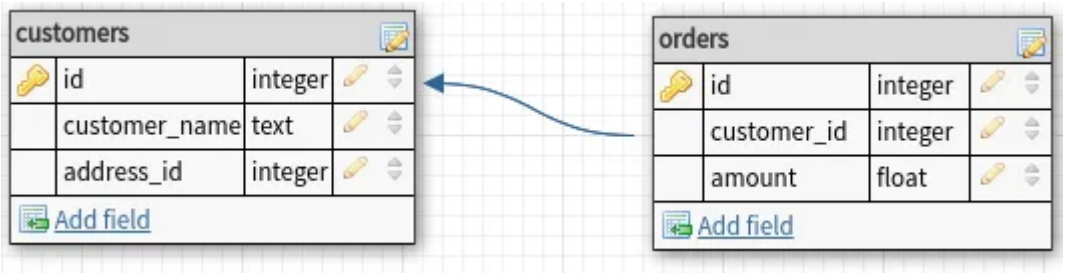
Συνεχίζοντας το προηγούμενο παράδειγμα, έστω ότι έχουν προστεθεί οι εξείς απαιτήσεις:

- Ένα πελάτης να μπορεί κάνει καμία, μία η περισσότερες παραγγελίες.
- Οι παραγγελιες να μπορούν περιέχουν πολλά αντικείμενα.

Σε αυτή τη περίπτωση πρέπει να δημιουργήσουμε μια σχέση ένα προς πολλά.

Εικόνα 5-16 σχέση ένα προς πολλά

Σύμφωνα με την παραπάνω σχέση, ένας Πελάτης (Customer) μπορεί να έχει μία η περισσότερες παραγγελίες (Orders). Για να δημιουργήσουμε αυτή τη σχέση πρέπει να έχουμε ένα πίνακα Πελάτη (Customer) και ένα πίνακα Παραγγελία (order) σαν αύτους που είναι στην ακόλουθη εικόνα.



Εικόνα 5-17 σχέση πελάτης παραγγελίες

Κάθε πελάτης μπορεί να έχει καμία, μια ή περισσότερες παραγγελίες, αλλά κάθε παραγγελία μπορεί να ανήκει σε έναν μόνο πελάτη. Για να κατασκευάσουμε τον πίνακα Orders πρέπει δημιουργήσουμε τον ακόλουθο Java κώδικα.

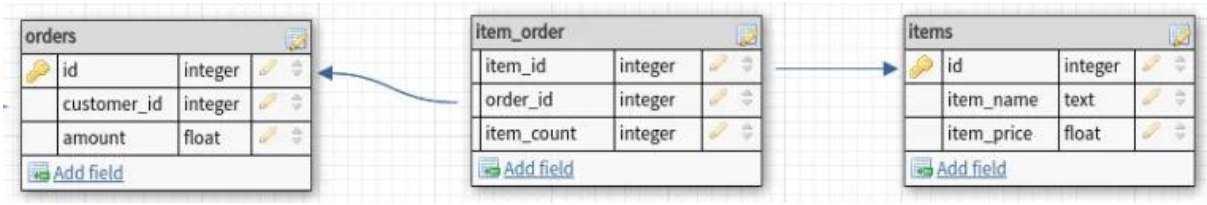
```
@Entity(tableName = "orders",foreignKeys =
@ForeignKey(entity=Customer.class,parentColumns = "id",childColumns = "customer_id"))
public class Order {
   @PrimaryKey(autoGenerate = true)
   public long id;
   @ColumnInfo(name = "customer_id")
   public long customerId;
   public float amount;
}
```

5.11.1.1.4 Σχέση πολλά προς πολλά (Many to many)

Κάποιες φορές μπορεί να χρειαζόμαστε πολλαπλά αντικείμενα και από τις δύο μεριές της συσχέτισης. Για παράδειγμα κάθε παραγγελία μπορεί να έχει πολλά αντικείμενα και κάθε αντικείμενο μπορεί να βρίσκεται σε πολλές παραγγελιές.

Εικόνα 5-18 Σχέση πολλά προς πολλά

Για τις παραπάνω σχέσεις χρειαζόμαστε να δημιουργήσουμε ένα ακόμα πίνακα.



Εικόνα 5-19 εισαγωγή πίνακα item order

Έτσι λοιπόν ο item_order πίνακας δημιουργεί μια πολλά προς πολλά ανάμεσα στον item και oder πίνακα. Για να κατασκευάσουμε τον πίνακα items πρέπει δημιουργήσουμε τον ακόλουθο Java κώδικα.

```
@Entity(tableName = "items")
public class Item {
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id")
    public long id;
    @ColumnInfo(name = "item_name")
    public String itemName;
    @ColumnInfo(name = "item_price")
    public float itemPrice;
}
```

Επιπρόσθετα κάποιες φορές, μερικά πεδία ή ένα γκρουπ πεδίων στη βάση δεδομένων πρέπει να είναι μοναδικά. Μπορούμε να επιβάλουμε αύτη την μοναδικότητα θέτοντας την unique ιδιότητα της @Index annotation σε true. Ο παρακάτω Java κώδικας αποτρέπει ένα πίνακα από το να έχει δύο γραμμές οι οποίες περιέχουν το ίδιο σετ από τιμές για τις order_id και item_id στήλες.

```
@Entity(tableName = "items_orders", indices = {@Index("name"),
        @Index(value = {"order_id", "item_id"})},
foreignKeys = {@ForeignKey(entity = Order.class, parentColumns = "id", childColumns = "order_id"),
            @ForeignKey(entity = Item.class, parentColumns = "id", childColumns = "item_id")})
public class ItemOrder {
    @ColumnInfo(name = "order_id")
    public long orderId;
    @ColumnInfo(name = "item_id")
    public long itemId;
    @ColumnInfo(name = "item_count")
```

```
    public int itemCount;
}
```

5.11.2 Dao (Data access object-Αντικείμενο πρόσβασης δεδομένων)

Για να έχουμε πρόσβαση στα δεδομένα μιας εφαρμογής χρησιμοποιώντας την Room βιβλιοθήκη εργαζόμαστε με τα αντικείμενα πρόσβασης δεδομένων (DAOs). Ένα σύνολο αντικειμένων Dao (DAO) αποτελεί το κύριο συστατικό της Room βάσης δεδομένων. Κάθε DAO περιλαμβάνει μεθόδους που προσφέρουν αφηρημένη πρόσβαση στη βάση δεδομένων της εφαρμογής.

Εισάγουμε το Dao για να καθορίσουμε SQL ερωτήματα (Queries) και να τα συνδέσουμε με κλήσεις μεθόδων. Ο μεταγλωττιστής (compiler) ελέγχει για SQL σφάλματα και έπειτα παράγει ερωτήματα (Queries) απο τα annotations. Για κάποια συχνά ερωτήματα, η βιβλιοθήκη μας παράγει τα κατάλληλα annotations, όπως @Insert(εισαγωγή),@Delete(διαγραφή),@Updata(Ενημέρωση).

Επίσης, πρέπει να δώσουμε ιδιαίτερη προσοχή ότι το Dao πρέπει να είναι διεπαφή (Interface) η αφημερεμήνη κλάση (abstract Class).Από προεπιλογή τα ερωτήματα(@Queries) πρέπει να εκτελούνται σε διαφορετικό νήμα από το κύριο νήμα. Για λειτουργίες όπως η εισαγωγή ή διαγραφή, η Room κάνει τη διαχείριση του νήματος.

Παράδειγμα κώδικα χρήσης των annations για το Dao μαζί με τους διαφορετικούς τύπους των ερωτημάτων.

```
 @Dao
public interface ExampleDao {
  @Insert
  void insertCustomer(Customer customer);  //Με την μέθοδο insertCustomer γίνεται η εισαγώγη ενός Πελάτη(Customer)
  @Update
  void updateCustomer(Customer customer); //Με την μέθοδο updateCustomer γίνεται η ενημέρωση του πίνακα Πελάτη(customers)

  @Delete
  void deleteCustomer(Customer customer); //Με την μέθοδο deleteCustomer γίνεται η διαγραφή ενός Πελάτη(Customer)
  @Query("SELECT * from customers") //Η μέθοδος getAllCustomerς μας επιστρέφει μια λίστα  με όλους του πελάτες
  List<Customer> getAllCustomers();
}
```

5.11.3 LiveData (Ζωντανά δεδομένα)

Όταν μια εφαρμογή προβάλει δεδομένα, συνήθως θέλουμε να κάνουμε κάποια ενέργεια όταν τα δεδομένα αλλάζουν. Αυτό λοιπόν σημαίνει ότι η εφαρμογή πρέπει να παρακολουθεί (observe) τα δεδομένα έτσι ώστε όταν αλλάξουν, η εφαρμογή να μπορεί να αντιδράσει με κάποιο τρόπο. Η παρατήρηση για αλλαγές στα δεδομένα σε όλα τα στοιχεία μια εφαρμογής μπορεί μετατρέψει την αποσφαλμάτωση (debugging) και τον έλεγχο (testing) σε αρκετά δύσκολες λειτουργίες. Έτσι λοιπόν για να αποφύγουμε αυτά τα προβλήματα γίνεται η χρήση των LiveData, τα οποία είναι μια βιβλιοθήκη κύκλου ζωής (life cycle library) κλάση για την παρακολούθηση των δεδομένων. Αν χρησιμοποιήσουμε τον επιστρεφόμενο τύπο LiveData στην υπογραφή μιας μεθόδου, η Room παράγει όλο τον απαιτούμενο κώδικα για την ενημέρωση των LiveData όταν ενημερώνεται η βάση δεδομένων.

5.11.3.1 Πλεονεκτήματα χρήσης των LiveData.

- Διασφαλίζουν ότι το UI αντιστοιχεί με την κατάσταση των δεδομένων. Τα LiveData ακολουθούν το πρότυπο παρατηρητή (observer pattern) με αποτέλεσμα να ειδοποιούν τα αντικείμενα του παρατηρητή όταν η κατάσταση του κύκλου ζωής αλλάζει.
- Δεν υπάρχει διαρροή μνήμης.
- Δεν υπάρχουν σφάλματα λόγω της διακοπής μια δραστηριότητας (Activity). Αν ο κύκλος ζωής του παρατηρητής είναι αδρανής, όπως στην περίπτωση μιας δραστηριότητας (Activity) που βρίσκεται στην πίσω στοίβα (Back,Stack), σε αυτή την περίπτωση λοιπόν δεν λαμβάνει γεγονότα LiveData.
- Όχι πια χειροκίνητος χειρισμός του κύκλου ζωής. Τα στοιχεία από τα οποία αποτελείται το UI παρακολουθούν τα δεδομένα και δεν σταματούν ή ξαναρχίζουν την παρατήρηση. Τα LiveData αντιλαμβάνονται τις αλλαγές της κατάστασης του κύκλου ζωής όσο παρακολουθούν, έτσι τα LiveData διαχειρίζονται αυτόματα την παύση ή την συνέχεια της παρακολούθησης.
- Τα δεδομένα είναι πάντα ενημερωμένα. Εάν ο κύκλος ζωής περάσει στο στάδιο αδράνειας, λαμβάνει τα πιο πρόσφατα δεδομένα όταν ενεργοποιηθεί ξανά. Για παράδειγμα, εάν μια δραστηριότητα που ήταν στο παρασκήνιο (background) θα λάβει τα πιο πρόσφατα δεδομένα όταν επιστρέψει στο προσκήνιο (foreground).
- Οι αλλαγές διαμόρφωσης χειρίζονται με τον σωστό τρόπο. Αν μια δραστηριότητα (Activity) ή ένα Fragment δημιουργείται εκ νέου λόγω αλλαγής της διαμόρφωσης, όπως για παράδειγμα την αλλαγή του προσανατολισμού της οθόνης, το Fragment ή η δραστηριότητα θα λάβουν τα πιο πρόσφατα ενημερωμένα δεδομένα.

Κάνοντας τα δεδομένα παρατηρήσιμα χρησιμοποιώντας τα LiveData

Για κάνουμε τα δεδομένα παρατηρήσιμα άπλα τα "τυλίγουμε" με LiveData. Για το παράδειγμα μας με το Πελάτη (Customer) στο ExampleDao όπου επιστρέφουμε μια λίστα με όλους τους καταχωρημένους πελάτες " τυλίγουμε" τα επιστρεφόμενα αποτελέσματα μέσα στα LiveData.

Παράδειγμα:

```
@Query("SELECT * from customers")
LiveData<List<Customer>> getAllCustomers();
```

5.11.3.2 MutableLiveData(Μεταβλητά ζωντανά δεδομένα).

Μπορούμε να χρησιμοποιήσουμε LiveData ανεξάρτητα απο την Room, αλλά για να το κάνουμε αυτό πρέπει να διαχειριστούμε τις ενημερώσεις των δεδομένων. Ωστόσο, τα LiveData δεν διαθέτουν δημόσιες (public) μεθόδους για την ενημέρωση των αποθηκευμένων μεθόδων. Εάν λοιπόν θέλουμε να ενημερώσουμε τα αποθηκευμένα δεδομένα πρέπει να χρησιμοποιήσουμε τα MutableLiveData αντί των LiveData. Η MutableLiveData κλάση έχει δύο δημόσιες (public) μεθόδους που μας επιτρέπουν να ορίσουμε την τιμή ενός αντικειμένου LiveData:

- postValue(T) - Δημοσιεύει μια εργασία στο κύριο νήμα για να ορίσει την δεδομένη τιμή.
- setValue(T) - Ορίζει την τιμή για ένα αντικείμενο. Αν υπάρχουν ενεργοί παρατηρητές (observers) η τιμή θα αποσταλεί σε αυτούς.

5.11.4 Room βάση δεδομένων.

Η Room είναι ένα επίπεδο βάσης δεδομένων πάνω από την SQLite. Η Room αναλαμβάνει να κάνει τις εργασίες της SQLiteOpenHelper. Για να την χρησιμοποιήσουμε δημιουργούμε μια δημόσια αφηρημένη κλάση (public abstract class) που κάνει επέκταση της Room Database.

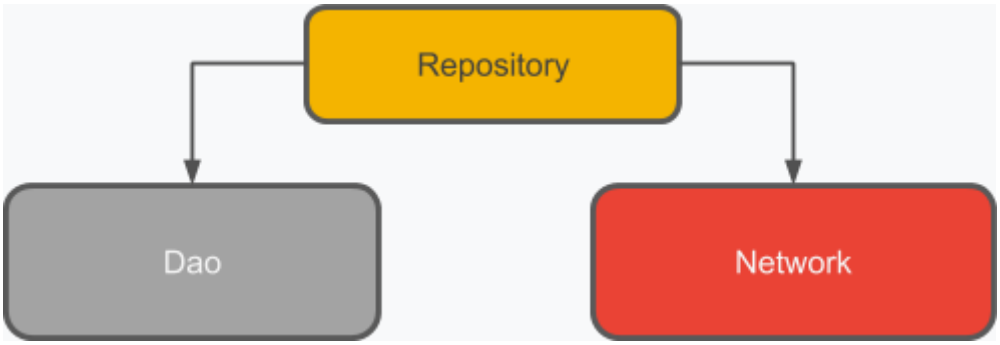
Παράδειγμα κώδικα χρήσης της RoomDatabase όπου μετράπουμε της κλάση μας σε singleton για να αποτρέψουμε να έχουμε πολλαπλά αντικείμενα της βάσης δεδομένων τα οποία να δημιουργούνται την ίδια ώρα.

```
@Database(entities = {Customer.class}, version = 1, exportSchema = false)
public abstract class ExampleDatabase extends RoomDatabase {

    private static ExampleDatabase exampleDatabase;

    public abstract ExampleDao exampleDao();

    public static synchronized ExampleDatabase getInstance(Context context) {
        if (exampleDatabase == null){
            exampleDatabase = Room.databaseBuilder(context.getApplicationContext(), ExampleDatabase.class,  name: "example_database")
                    .fallbackToDestructiveMigration()
                    .build();
        }
        return exampleDatabase;
    }

}
```

Εικόνα 5-20 Χρήση Room DataBase

5.11.5 Repository (Αποθετήριο)

Ένα Repository είναι μια κλάση που αποτελεί αφαίρεση προσβάσης σε πολλαπλές πηγές δεδομένων. Το Repository δεν είναι μέρος των βιβλιοθηκών Architecture Components αλλά είναι μια προτεινόμενη βέλτιστη πρακτική για διαχωρισμό κώδικα και αρχιτεκτονικής. Μία κλάση Repository διαχειρίζεται τις λειτουργίες των δεδομένων και παρέχει ένα καθαρό API στην υπόλοιπη εφαρμογή για τα δεδομένα εφαρμογής. Συμπεριφέρεται ως την γέφυρα μεταξύ views και της βάσης δεδομένων. Το αποθετήριο εφαρμόζει τη λογική για να αποφασίσει εάν θα πάρει δεδομένα από το διαδίκτυο ή θα χρησιμοποιήσει αποτελέσματα που έχουν αποθηκευτεί στην κρυφή μνήμη σε μια τοπική βάση δεδομένων.



Εικόνα 5-21 Repository

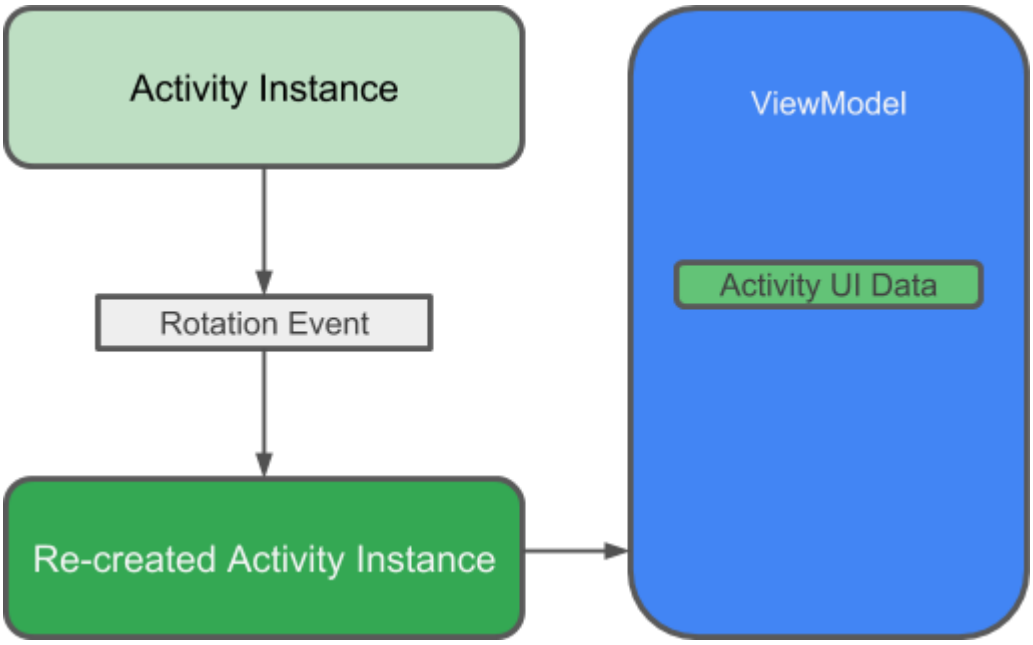
Παράδειγμα κώδικα χρήσης του Repository για την εισαγωγή πελατών και την επιστροφή όλων των πελατών που είναι αποθηκευμένοι στην βάση δεδομένων.

```
public class ExampleRepository {
    private ExampleDao exampleDao;
    private LiveData<List<Customer>> allCustomers;

    ExampleRepository(Application application) {
        ExampleDatabase db = ExampleDatabase.getInstance(application);
        exampleDao = db.exampleDao();
        allCustomers = exampleDao.getAllCustomers();
    }

    LiveData<List<Customer>> getAllCustomers() {
        return allCustomers;
    }

    public void insert (Customer customer) {
        new insertAsyncTask(exampleDao).execute(customer);
    }

    private static class insertAsyncTask extends AsyncTask<Customer, Void, Void> {

        private ExampleDao mAsyncTaskDao;

        insertAsyncTask(ExampleDao dao) {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(final Customer... params) {
            mAsyncTaskDao.insertCustomer(params[0]);
            return null;
        }
    }
}
```

Εικόνα 5-22 Χρήση του Repository

5.11.6 ViewModel

Η ViewModel είναι κλάση της οποία ο ρόλος είναι να παρέχει δεδομένα στο UI (διεπαφή χρήστη) και να αντέχει στις αλλαγές διαμόρφωσης. Το viewModel συμπεριφέρεται σαν το κέντρο επικοινωνίας ανάμεσα στο Repository και το UI. Επίσης μπορούμε να χρησιμοποιήσουμε το ViewModel και κοινή χρήση δεδομένων μεταξύ Fragments. Το ViewModel είναι κομμάτι της lifeCircle (κύκλου ζωής) βιβλιοθήκης.



Εικόνα 5-23ViewModel.

Το ViewModel διατηρεί τα δεδομένα UI(Διεπαφής χρήστη) με ένα τρόπο συνειδητοποιήσεις του κύκλου ζωής που αντέχει στις αλλαγές διαμόρφωσης. Διαχωρίζοντας τα δεδομένα της UI (διεπαφής χρήστη) από την δραστηριότητα (Activity) ή από το fragment είναι μια καλή τακτική καθώς μας επιτρέπεται να ακολουθήσουμε την μοναδική αρχή ευθύνης: ότι οι δραστηριότητες (Activities) και τα fragments είναι υπεύθυνα για την απεικόνιση δεδομένων στην οθόνη, ενώ το ViewModel είναι υπεύθυνο για να κρατάει και επεξεργάζεται όλα τα δεδομένα που χρειάζονται για την UI (διεπαφη χρήστη).

Παράδειγμα κώδικα για το ViewModel, χρησιμοποιούμε LiveData για τα μεταβλητά δεδομένα όπου η UI(διεπαφή χρήστη) θα τα χρησιμοποιήσει ή θα τα απεικονίσει, έτσι ώστε να μπορούμε να προσθέσουμε έναν παρατηρητή (observer) και να ανταποκρίνεται στις πιθανές αλλαγές που μπορεί να υπάρξουν.

.

```
public class ExampleViewModel extends AndroidViewModel {
    private ExampleRepository exampleRepository;
    private LiveData<List<Customer>> allCustomers;

    public ExampleViewModel(@NonNull Application application) {
        super(application);
        exampleRepository = new ExampleRepository(application);
        allCustomers = exampleRepository.getAllCustomers();
    }
    LiveData<List<Customer>> getAllCustomers(){
        return allCustomers;
    }
    public  void insert(Customer customer){
        exampleRepository.insert(customer);
    }

}
```

Εικόνα 5-24 χρήση ViewModel

Αφού κατασκευάσαμε το ViewModel, έχουμε πλέον τη δυνατότητα να εμφανίσουμε τα δεδομένα στο View. Εάν για παράδειγμα εμφανίσουμε τα δεδομένα σε ένα RecyclerView, με την onChange() μέθοδο ενημερώνονται τα δεδομένα που βρίσκονται στην κρυφή μνήμη του προσαρμογέα.

Παράδειγμα:

```
exampleViewModel.getAllCustomers.observe(this, new Observer<List<Customer>>() {
  @Override
  public void onChanged(@Nullable final List<Customer> customer) {
     adapter.setResult(customer);
  }
});
```

Στο παραπάνω παράδειγμα επίσης συνδέεται ένας παρατηρητής με τα LiveData. [40] [41]

5.12 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια προσπάθεια να περιγράφουν λεπτομερώς οι τεχνολογίες που χρησιμοποιήθηκαν προκειμένου να ολοκληρωθεί η εφαρμογή.

Κεφάλαιο 6ο: Αξιολόγηση

6.1 Εισαγωγή

Προκειμένου να αξιολογήσουμε το λογισμικό θα πάρουμε κάποιες μετρήσεις από την εσωτερική αξιολόγηση πού έχουν να κάνουν με τον κώδικα της εφαρμογής και κάποιες εξωτερικές από χρήστες που δοκίμασαν την εφαρμογή.

6.2 Εσωτερική αξιολόγηση βάσει μετρικών

Κάνοντας χρήση του plugin **Metrics Reloaded** στο android studio συλλέξαμε τις παραπάνω τιμές για τις ακόλουθες μετρικές:

- **CC (Cyclomatic Complexity**): κυκλωματική πολυπλοκότητα (πολυπλοκότητα μεθόδου). Είναι μια μετρική που μετρά τον αριθμό τον πιθανών μονοπατιών μέσα στον αλγόριθμο μιας μονάδας λογισμικού όπως για παράδειγμα σε μια μέθοδο ελέγχει τον αριθμό των if, switch, for, while. Η τιμή αυτής της μέτρησης δεν θα πρέπει να ξεπερνά τον αριθμό δέκα.
- **WMC**: υπολογίζει το άθροισμα των μετρικών της πολυπλοκότητας κάθε μεθόδου για το σύνολο τους.
- **LOC (Lines of Code)**: Είναι μια μετρική μεγέθους που μετρά τον αριθμό των γραμμών κώδικα χωρίς να υπολογίζει τα σχόλια και τις κενές γραμμές.

Πίνακας 6-1 Μετρικές CC - WMC

ΜΕΤΡΙΚΕΣ	CC	WMC
Κλάση:		
AddLyricsToLocalVideo	1.5	6.0
AddYouTubeVideoToFavouriteActivity	1.75	7.0
DisplayLocalVideosActivity	1.55	14.0
DisplayThumbnailsFromYoutubeActivity	1.0	4.0
Entities.IdYouTubeEntity	1.0	2.0
Entities.LoopEntity	1.0	8.0
Entities.LoopForLocalVideoEntity	1.0	8.0
UriLocalVideoEntity	1.0	4.0
.Entities.YouTubeEntity	1.0	10.0
.ExampleInstrumentedTest	1.0	1.0
.ExampleUnitTest	1.0	1.0
.LocalVideoModel.VideoModel	1.0	12.0
.MainActivity	1.25	5.0
.MusixmatchModel.BodyLyrics	1.0	1.0
.MusixmatchModel.HeaderLyrics	1.0	2.0
.MusixmatchModel.Lyrics	1.0	7.0
.MusixmatchModel.MessageLyrics	1.0	2.0
.MusixmatchModel.Musixmatch	1.0	1.0
.MyAdapters.CaptionedThumnailAdapter	1.16	7.0
.MyAdapters.CaptionedThumnailAdapter.ViewHolder	1.0	1.0
.MyAdapters.DisplayYouTubeEntityAdapter	1.22	11.0
.MyAdapters.DisplayYouTubeEntityAdapter.ViewHolder	1.0	1.0
.MyAdapters.LocalVideoAdapter	1.4	14.0
.MyAdapters.LocalVideoAdapter.ViewHolder	1.0	1.0

.MyAdapters.LoopsLocalVideoAdapter	1.25	10.0
.MyAdapters.LoopsLocalVideoAdapter.ViewHolder	1.0	1.0
.MyAdapters.LoopsYouTubeAdapter	1.22	11.0
.MyAdapters.LoopsYouTubeAdapter.ViewHolder	1.0	1.0
.MyVideoYouTubeDatabase	2.0	2.0
.MyViewModel.IdYouTubeViewModel	1.0	4.0
.MyViewModel.LocalVideosViewModel	1.5	6.0
.MyViewModel.MusixmatchViewModel	1.0	4.0
.MyViewModel.UriLocalVideoLoopViewModel	1.0	5.0
.MyViewModel.UriLocalVideoViewModel	1.0	5.0
.MyViewModel.YouTubeEntityAddToFavouriteForYouTubeViewModel	1.0	2.0
.MyViewModel.YouTubeEntityViewModel	1.0	6.0
.MyViewModel.YouTubeLoopEntityViewModel	1.0	7.0
.MyViewModel.YouTubeUpdateYouTubeViewModel	1.0	2.0
.MyViewModel.YouTubeViewModel	1.0	6.0
.PlayLocalVideoActivity	1.57	44.0
.Repositories.IdYouTubeRepository	1.0	3.0
.Repositories.IdYouTubeRepository.InsertIdYouTubeEntityAsyncTask	1.0	2.0
.Repositories.MusixmatchResository	1.0	5.0
.Repositories.UriLocalVideoDatabaseLoopRepository	1.0	4.0
.Repositories.UriLocalVideoDatabaseLoopRepository.DeleteLoopForLocalVideoEntityAsynTask	1.0	2.0
.Repositories.UriLocalVideoDatabaseLoopRepository.InsertLoopForLocalVideoEntityAsynTask	1.0	2.0
.Repositories.UriLocalVideoDatabaseRepository	1.0	4.0
.Repositories.UriLocalVideoDatabaseRepository.InsertUriLocalVideoEntityAsynTask	1.0	2.0
.Repositories.UriLocalVideoDatabaseRepository.UpdateUriLocalVideoEntityAsynTask	1.0	2.0
.Repositories.YoutubeDatabaseLoopRepository	1.0	6.0
.Repositories.YoutubeDatabaseLoopRepository.deleteLoopEntityAsyncTask	1.0	2.0
.Repositories.YoutubeDatabaseLoopRepository.InsertLoopEntityAsyncTask	1.0	2.0
.Repositories.YouTubeDatabaseRepository	1.0	6.0
.Repositories.YouTubeDatabaseRepository.DeleteYoutubeEntityAsyncTask	1.0	2.0
.Repositories.YouTubeDatabaseRepository.InsertYoutubeEntityAsyncTask	1.0	2.0
.Repositories.YouTubeDatabaseUpdateFavouriteRepository	1.0	2.0
.Repositories.YouTubeDatabaseUpdateFavouriteRepository.updateYoutubeEntityAsyncTask	1.0	2.0
.Repositories.YouTubeRepository	1.0	7.0
.SearchForYouTubeActivity	1.31	29.0
.UpdateYouTubeEntityActivity	1.5	6.0
.YouTubeV3Bean.ContentDetailsBean	1.0	2.0
.YouTubeV3Bean.DefaultBean	1.0	3.0
.YouTubeV3Bean.HighBean	1.0	3.0
.YouTubeV3Bean.IdBean	1.0	2.0
.YouTubeV3Bean.ItemsBean	1.0	5.0

.YouTubeV3Bean.MaxresBean	1.0	3.0
.YouTubeV3Bean.MediumBean	1.0	3.0
.YouTubeV3Bean.PageInfoBean	1.0	2.0
.YouTubeV3Bean.ResourceIdBean	1.0	2.0
.YouTubeV3Bean.SnippetBean	1.0	9.0
.YouTubeV3Bean.StandardBean	1.0	3.0
.YouTubeV3Bean.ThumbnailsBean	1.0	5.0
.YouTubeV3Bean.YoutubeV3Details	1.0	5.0
.YouTubeVideoLyricsActivity	1.31	38.0
.YouTubeVideoLyricsActivity.MyPlaybackEventListener	1.0	5.0
.YouTubeVideoLyricsActivity.MyPlayerStateChangeListener	1.0	7.0
Total		**435.0**
Average	**1.15**	**5.72**

Πίνακας 6-2 Μετρικές LOC

package	LOC
com	
com.alextsatsos	
.mediayoutubelyrics	1386.0
.mediayoutubelyrics.Entities	197.0
.mediayoutubelyrics.InterfacesApi	54.0
.mediayoutubelyrics.InterfacesDao	68.0
mediayoutubelyrics.LocalVideoModel	57.0
mediayoutubelyrics.MusixmatchModel	85.0
mediayoutubelyrics.MyAdapters	411.0
mediayoutubelyrics.MyViewModel	310.0
mediayoutubelyrics.Repositories	357.0
mediayoutubelyrics.YouTubeV3Bean	288.0
drawable	199.0
drawable-v24	34.0
layout	709.0
layout-land	118.0
menu	27.0
mipmap-anydpi-v26	10.0

values	32.0
Total	4342.0
Average	255.41

6.3 Εξωτερική αξιολόγηση

Για την εξωτερική αξιολόγηση του έργου χρησιμοποιήθηκε μια φόρμα ερωτηματολογίου (Google form) με ερωτήσεις που έχουν να κάνουν με την αξιοπιστία την λειτουργικότητα την πολυπλοκότητα και την ευχρηστία. Οι απαντήσεις έχουν ως κλίμακα το 1 (καθόλου) έως το 5 (παρά πολύ).

Στην ερώτηση:

1. Βαθμολογήστε την εφαρμογή ως προς την αξιοπιστία της σε κλίμακα από το 1(καθόλου) έως το 5 (πάρα πολύ). Το 61.5% απάντησε 5, το 30.8% απάντησε 4 και μόλις το 7.7% απάντησε 3.
2. Βαθμολογήστε την εφαρμογή ως προς την λειτουργικότητα της σε κλίμακα από το 1 (καθόλου) έως το 5 (πάρα πολύ). Το 38.5% απάντησε 5, το 50% απάντησε 4 και μόλις το 11.5% απάντησε 3.
3. Βαθμολογήστε την εφαρμογή ως προς την πολυπλοκότητα της σε κλίμακα από το 1 (καθόλου) έως το 5 (πάρα πολύ). Το 92.5% απάντησε 1, το 7.7% απάντησε 2.
4. Βαθμολογήστε την εφαρμογή ως προς την ευχρηστία της σε κλίμακα από το 1 (καθόλου) έως το 5 (πάρα πολύ). Το 30,8% απάντησε 5, το 46,2% απάντησε 4, το 19.2% απάντησε 3 ενώ το 3.8 απάντησε 2.

Βάση αυτών τον απαντήσεων η βαθμολογία που παίρνει από την εξωτερική αξιολόγηση το έργο είναι:

Ως προς την αξιοπιστία: 4.5

Ως προς λειτουργικότητα: 4.2

Ως προς πολυπλοκότητα: 1.1

Ως προς ευχρηστία: 4

6.4 Επίλογος

Βάση τον παραπάνω αποτελεσμάτων που συλλέχθηκαν εξάγουμε τα εξής συμπεράσματα:

CC (κυκλωματική πολυπλοκότητα) έχουμε μέσο όρο 1.1 που είναι μια τιμή αρκετά χαμηλή και συνεπώς η πολυπλοκότητα του κώδικα είναι μικρή με αποτέλεσμα η κατανόηση του κώδικα να είναι σχετικά εύκολη.

WMC: έχουμε μέσο όρο 5.7 πού είναι μια καλή μέτρηση για την αποφυγή σφαλμάτων και συντήρησής.

LOC (Linew of Code): εδώ έχουμε μέσο όρο 255.4 ανά πακέτο.

Τέλος στην εξωτερική αξιολόγηση η εφαρμογή συγκεντρώνει βαθμούς πάνω από το 4 στα 5 και στην πολυπλοκότητα 1.1 που μας οδηγεί στο συμπέρασμα ότι η εφαρμογή βρίσκεται στον σωστό δρόμο με περιθώρια βελτίωσης.

Κεφάλαιο 7ο: Οδηγός Χρήσης

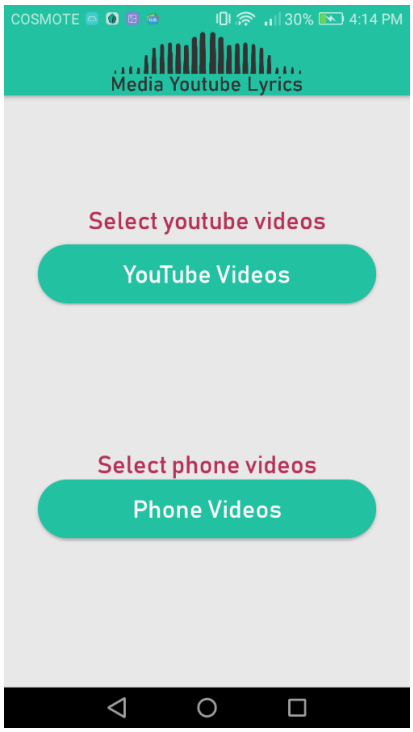
7.1 Εισαγωγή

Στο κεφάλαιο αυτό θα δοθεί στον αναγνώστη μια πιο οπτικοποιημένη άποψη της εφαρμογής με περιγραφή των λειτουργειών της.

7.2 Οδηγός

Στην πρώτη οθόνη με το που ανοίγει ο χρήστης την εφαρμογή θα συναντήσει δύο επιλογές(κουμπιά)

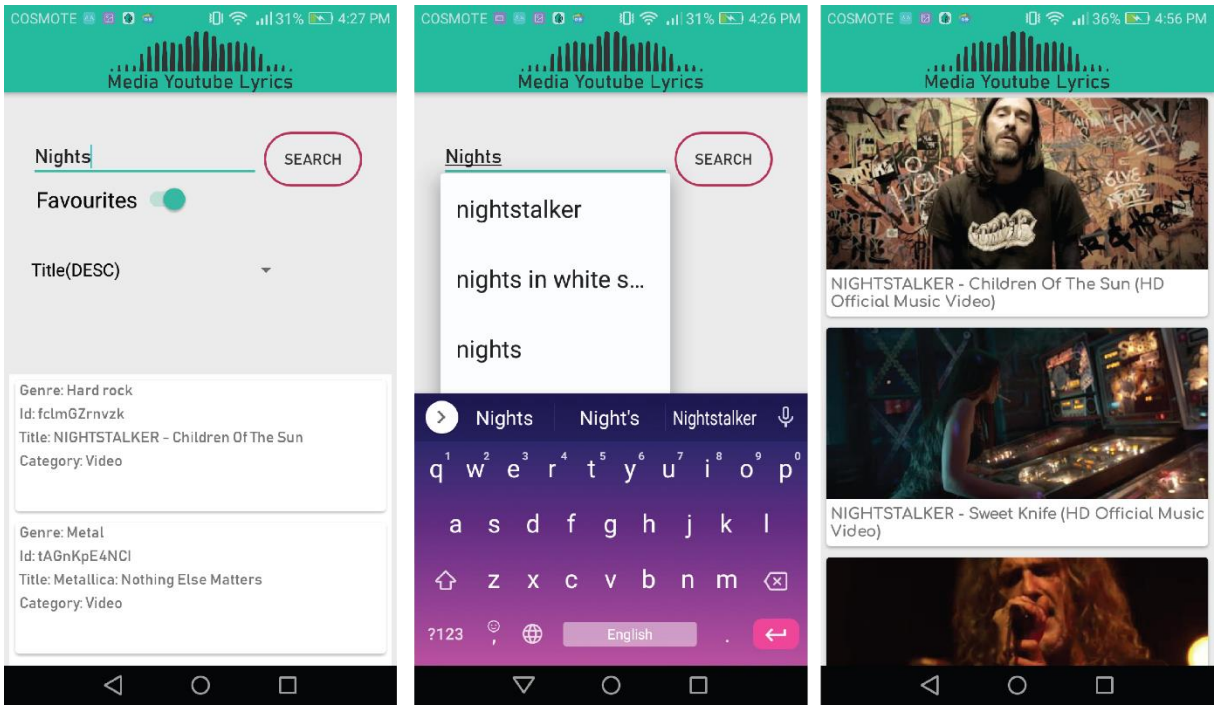
Το πρώτο είναι για βίντεο από την πλατφόρμα του YouTube και το δεύτερο για τα βίντεο του κινητού.



Εικόνα 7-1 Η αρχική οθόνη της εφαρμογής

Στην περίπτωση που ο χρήστης θέλει να αναζητήσει κάποιο βίντεο από το YouTube επιλέγει το κουμπί YouTube Videos , αυτόματος μεταφέρετε σε μια νέα οθόνη . Στην οθόνη αυτή υπάρχει η δυνατότητα ο χρήστης να πληκτρολογήσει τον τίτλο η κάτι σχετικό με αυτό που θέλει να αναζητήσει ενώ παράλληλα του δίνεται η δυνατότητα να επιλέξει και κάποιο από τα αγαπημένα βίντεο που έχει προσθέσει. Τέλος μπορεί να κάνει ταξινόμηση βάση τίτλου στην λίστα αυτή καθώς και να διαγράψει κάποια εγγραφή της.

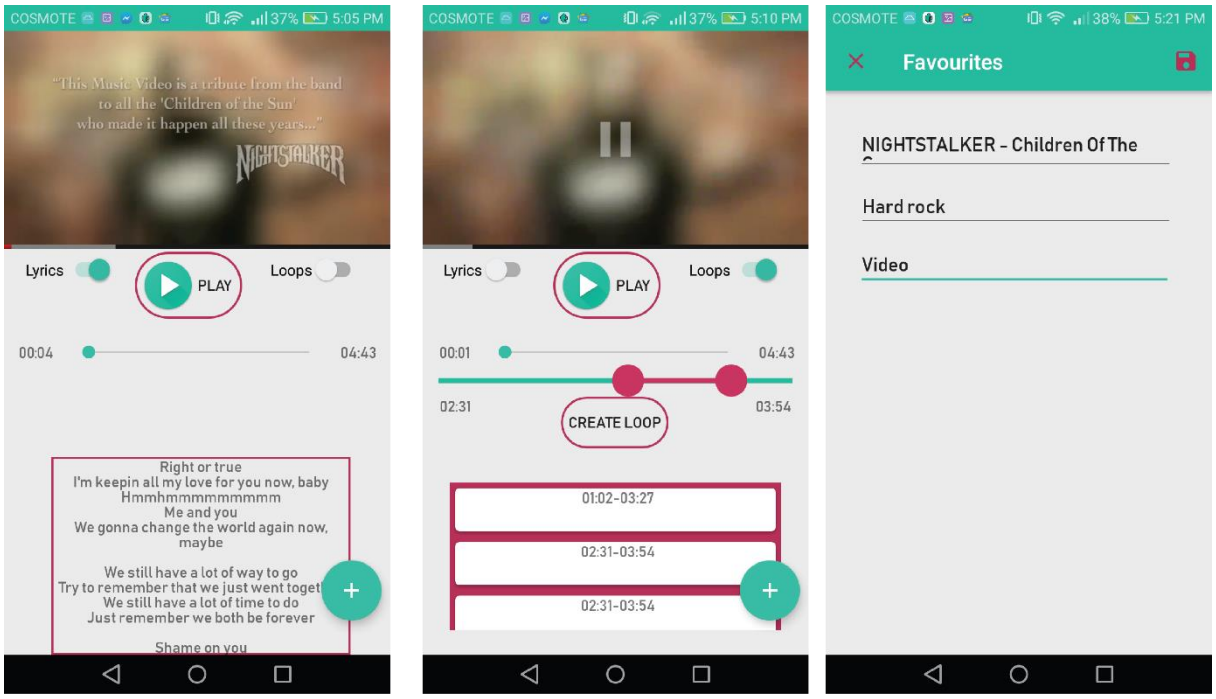
Όταν ο χρήστης κάνει αναζήτηση βάση αυτού που έχει γράψει πατώντας το κουμπί search θα μεταφερθεί σε μία οθόνη που θα του εμφανίζει μια λίστα αποτελεσμάτων βίντεο.



Εικόνα 7-2 Αρχική οθόνη για βίντεο YouTube

Επιλέγοντάς το βίντεο που επιθυμεί ο χρήστης θα μεταφερθεί σε μια νέα οθόνη που περιλαμβάνει τις παρακάτω δυνατότητες :

- Εμφάνιση στοίχων ενεργοποιώντας το κουμπί επιλογής Lyrics.
- Δημιουργία τμημάτων(loops) του βίντεο επιλέγοντας τα σημεία έναρξης και λήξης στην μπάρα που ανοίγει και πατώντας το κουμπί create loop.
- Δυνατότητα διαγραφής αυτόν τον τμημάτων σύροντας με το δάχτυλό δεξιά η αριστερά.
- Και τέλος πατώντας το κουμπί με το σύμβολο συν δίνεται η δυνατότητα προσθήκης του βίντεο στην λίστα των αγαπημένων προσθέτοντας του κάποια χαρακτηριστικά όπως η κατηγόρια και το είδος της μουσικής που ανήκει.



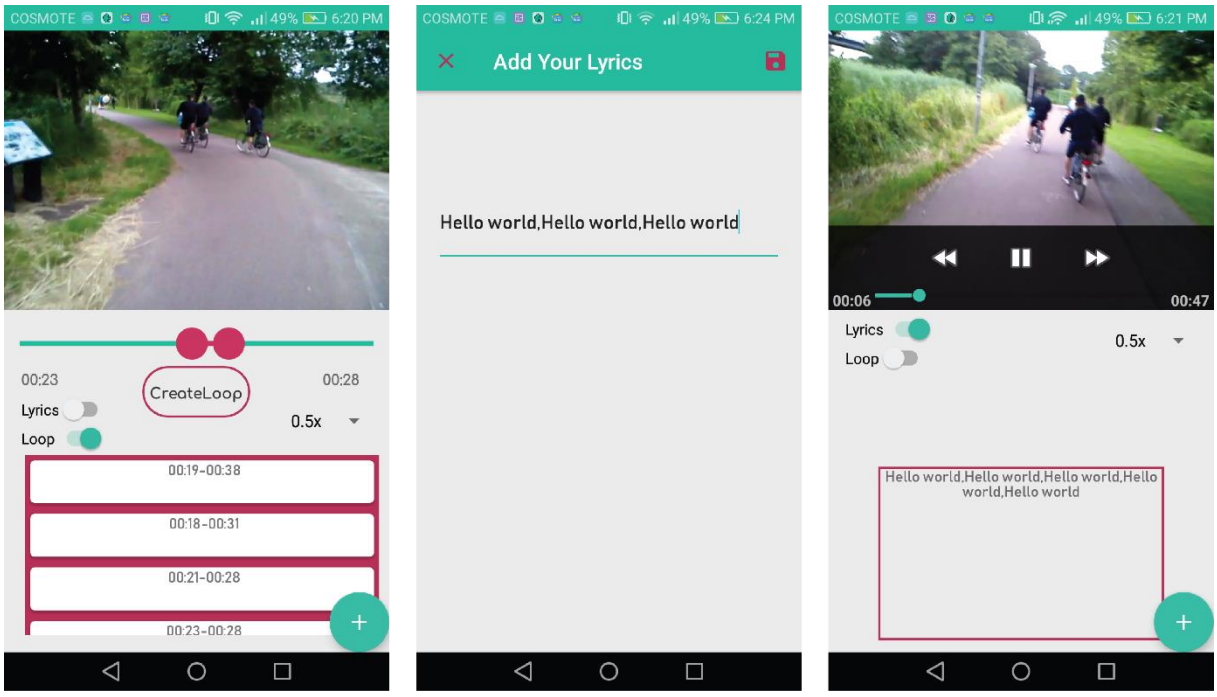
Εικόνα 7-3 Δυνατότητες για βίντεο YouTube

Στην περίπτωση που ο χρήστης θέλει να αναζητήσει κάποιο βίντεο από τον κατάλογο της συσκευής του επιλέγει το κουμπί Phone Videos . Ανοίγει μια νέα οθόνη στην οποία ο χρήστης μπορεί να επιλέξει στην λίστα με τα video ή και να κάνει αναζήτηση βάση του τίτλου .

Εικόνα 7-4 Αρχική οθόνη για τα βίντεο του κινητού

Επιλέγοντάς το βίντεο που επιθυμεί ο χρήστης θα μεταφερθεί σε μια νέα οθόνη που περιλαμβάνει τις παρακάτω δυνατότητες :

- Εμφάνιση στοίχων ενεργοποιώντας το κουμπί επιλογής Lyrics.
- Δημιουργία τμημάτων(loops) του βίντεο επιλέγοντας τα σημεία έναρξης και λήξης στην μπάρα που ανοίγει και πατώντας το κουμπί create loop.
- Δυνατότητα διαγραφής αυτόν τον τμημάτων σύροντας με το δάχτυλό δεξιά η αριστερά.
- Επίσης δίνεται η δυνατότητα επιλογής ταχύτητας αναπαραγωγής σε τρις ταχύτητες.
- Και τέλος πατώντας το κουμπί με το σύμβολο συν δίνεται η δυνατότητα προσθήκης στοίχων για το συγκεκριμένο βίντεο.



Εικόνα 7-5 Δυνατότητες για βίντεο απο την συσκευή

7.3 Επίλογος

Σκοπός αυτού του κεφαλαίου ήταν ο αναγνώστης να δει οπτικά πώς είναι η εφαρμογή και ποιες δυνατότητες του παρέχει

Κεφάλαιο 8ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Στην παρούσα πτυχιακή εργασία αναλύθηκε η ανάπτυξη και υλοποίηση μιας εφαρμογής για κινητές συσκευές που χρησιμοποιούν λειτουργικό σύστημα Android. Μας δόθηκε η δυνατότητα να εφαρμόσουμε γνώσεις από τα προπτυχιακά μας μαθήματα και να γνωρίσουμε τον κόσμο του Android αποκομίζοντας πολλές καινούργιες γνώσεις σε νέες τεχνολογίες.

Ο τρόπος με τον οποίο έχει αναπτυχθεί η εφαρμογή δίνει την δυνατότητα να υπάρξουν μελλοντικά νέα χαρακτηριστικά. Θα μπορούσε η εφαρμογή να συνδεθεί και με άλλες μεγάλες πλατφόρμες εκτός του YouTube όπως και να δίνει την δυνατότητα της παροχής συγχορδιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Κεφάλαιο 9ο: Βιβλιογραφία

[1] «Android 1.6 Donut,» Android Headlines, [Ηλεκτρονικό]. Available: https://www.androidheadlines.com/android-1-6-donut.

[2] C. Sorrel, «Android 2.2 'Froyo' Features USB, Wi-Fi Tethering,» [Ηλεκτρονικό]. Available: https://www.wired.com/2010/05/android-22-froyo-features-usb-wi-fi-tethering/.

[3] «2.2, Android 2.1 to Android» «TalkAndroid.com,» [Ηλεκτρονικό]. Available: https://www.talkandroid.com/guides/update/new-features-from-android-2-1-to-android-2-2-froyo/.

[4] «Developers, Android 2.2 Platform Highlights »|. A. Developers, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/android-2.2-highlights.

[5] android-2.3-highlights, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/android-2.3-highlights.

[6] android-3.0-highlights, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/android-3.0-highlights.

[7] jelly-bean, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/jelly-bean#android-4.3.

[8] Android 4.4 KitKat, «Digital Life!,» [Ηλεκτρονικό]. Available: https://www.digitallife.gr/android-4-4-kit-kat-ayta-einai-ta-xaraktiristika-tou-95605.

[9] android-5.0-changes, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/android-5.0-changes.

[10] marshmallow-6-0, «android.com,» [Ηλεκτρονικό]. Available: https://www.android.com/versions/marshmallow-6-0/ .

[11] android-6.0-changes, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/marshmallow/android-6.0-changes#behavior-apache-http-client .

[12] android-7.0, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/nougat/android-7.0.

[13] nougat-7-0, «Android Developers,» [Ηλεκτρονικό]. Available: https://www.android.com/versions/nougat-7-0/ .

[14] android-8.0, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/oreo/.

[15] android-8.1, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/versions/oreo/android-8.1.

[16] pie-9-0, «Android Developers,» [Ηλεκτρονικό]. Available: https://www.android.com/versions/pie-9-0/.

[17] Android Q, «Techgear.gr,» [Ηλεκτρονικό]. Available: https://www.techgear.gr/android-q-new-features-google-io-2019-712.

[18] Prakhar, Android 11 vs Android 10: What's new?, «Pocketnow,» [Ηλεκτρονικό]. Available: https://pocketnow.com/android-11-vs-android-10.

[19] platform, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/platform/index.html.

[20] fundamentals, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/components/fundamentals.

[21] Architecture, An Overview of the Android, «Techotopia.com,» [Ηλεκτρονικό]. Available: ttps://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture.

[22] activity-lifecycle, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle.

[23] Meet Android Studio, «Android Developers,» [Ηλεκτρονικό]. Available: https://developer.android.com/studio/intro.

[24] Ο Οδηγός του Scrum, «Scrumguides.org,» [Ηλεκτρονικό]. Available: https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-GR.pdf.

[25] Java, «El.wikipedia.org,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/Java.

[26] XML, «W3c.gr,» [Ηλεκτρονικό]. Available: http://www.w3c.gr/docs/XML10points.el.htm.

[27] YouTube Android Player API, «Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/youtube/android/player.

[28] YouTube Android Player API, «Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/package-summary.

[29] N. Smyth, Android studio 3.0 Development Essentials, 2017.

[30] P. v. w. VideoView, «Google-developer-training.github.io,» [Ηλεκτρονικό]. Available: https://google-developer-training.github.io/android-developer-advanced-course-practicals/unit-5-advanced-graphics-and-views/lesson-13-media/13-1-p-playing-video-with-videoview/13-1-p-playing-video-with-videoview.html.

[31] «Repository.kallipos.gr,» [Ηλεκτρονικό]. Available: https://repository.kallipos.gr/pdfviewer/web/viewer.html?file=/bitstream/11419/1343/1/07_chapter_6.pdf.

[32] «Repfiles.kallipos.gr,» [Ηλεκτρονικό]. Available: http://repfiles.kallipos.gr/html_books/9536/Chapter%209/Chapter09.html#SOAP.

[33] JSON, «Json.org,» [Ηλεκτρονικό]. Available: https://www.json.org/json-en.htm.

[34] Developers, PI Reference | YouTube Data API | Google, «Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/youtube/v3/docs.

[35] Google, Search | YouTube Data API , «Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/youtube/v3/docs/search.

[36] Developers, Search: list | YouTube Data API «Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/youtube/v3/docs/search/list.

[37] Music Meta Data, «Developer.musixmatch.com,» [Ηλεκτρονικό]. Available: https://developer.musixmatch.com/documentation/music-meta-data.

[38] M. Pöhls, Retrofit: Love Working with APIs on Android, 2016.

[39] Transfer, Room: Database Relationships - knowledge, «knowledge Transfer,» [Ηλεκτρονικό]. Available: https://androidkt.com/database-relationships/.

[40] Architecture Components ·, «Google-developer-training.github.io,» [Ηλεκτρονικό]. Available: https://google-developer-training.github.io/android-developer-advanced-course-concepts/unit-6-working-with-architecture-components/lesson-14-architecture-components/14-1-c-architecture-components/14-1-c-architecture-components.html#chapterstart.

[41] Android fundamentals 10.1 Part A: Room, LiveData, and ViewModel, «Codelabs.developers.google.com,» [Ηλεκτρονικό]. Available: https://codelabs.developers.google.com/codelabs/android-training-livedata-viewmodel/.