Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

Πρόγραμμα Μεταπτυχιακών Σπουδών

“Ψηφιακά Συστήματα & Υπηρεσίες”

Κατεύθυνση: Ψηφιακές Επικοινωνίες και Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΔΙΑΜΟΙΡΑΣΗΣ ΑΡΧΕΙΩΝ ΜΕΤΑΞΥ ΔΙΑΦΟΡΕΤΙΚΩΝ ΣΥΣΚΕΥΩΝ ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ ANDROID

ΕΠΙΜΕΛΕΙΑ: ΔΗΜΗΤΡΑ – ΝΙΚΗ ΜΑΖΑΡΑΚΗ

Α.Μ. : ΜΕ/10065

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΓΕΩΡΓΙΟΣ ΕΥΘΥΜΟΓΛΟΥ,
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ ΤΜΗΜΑΤΟΣ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πειραιάς, Δεκέμβριος 2014

Εγκρίθηκε από την ..... εξεταστική επιτροπή την .....

.....	.....	.....
Καθηγητής	Καθηγητής	Καθηγητής

Στους γονείς μου Ανδρέα και Παρθένα και στον αδερφό μου Δημήτρη.

Στους φίλους μου Νεκταρία, Φώτη, Ευτυχία και Παναγιώτη.

Για την αγάπη, τη συμπαράσταση και την υποστήριξή τους.

Ευχαριστίες

Η παρούσα Μεταπτυχιακή Διπλωματική Εργασία (ΜΔΕ), εκπονήθηκε στο πλαίσιο ολοκλήρωσης των σπουδών μου στο Μεταπτυχιακό Πρόγραμμα Σπουδών "Ψηφιακά Συστήματα και Υπηρεσίες", με κατεύθυνση τα του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.

Στο πλαίσιο αυτής, θέλω να ευχαριστήσω θερμά τον Επιβλέποντα Καθηγητή μου κο **Γεώργιο Ευθύμογλου**, Αναπληρωτή Καθηγητή του Τμήματος Ψηφιακών Συστημάτων, για την ευκαιρία εκπόνησης της παρούσας εργασίας και της υποστήριξης που μου παρείχε στη διάρκεια αυτής.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους ανθρώπους οι οποίοι κατά τη διάρκεια της διπλωματικής εργασίας μου παρείχαν χώρους για την ολοκλήρωσή της, αλλά και τη βοήθειά τους και στάθηκαν αρωγοί στο έργο που μου είχαν ανατεθεί. Ευχαριστώ εκ βάθους καρδιάς όλους όσους μου έδωσαν την ευκαιρία και τη στήριξη να ασχοληθώ με το συγκεκριμένο αντικείμενο.

Πειραιάς, Δεκέμβριος 2014

Δήμητρα – Νίκη Μαζαράκη (Dimitra – Niki Mazaraki)

Συντομογραφίες

ΒΔ= Βάση Δεδομένων

AIDL = Android Interface Definition Language

ADT = Android Developer Tools

IPC = Inter-Process Communication

RPC = Remote Procedure Call

SDK = Software Developer Tools

Λέξεις-κλειδιά

Android, Application software – development, Android (Electronic Resource), FileShare, Διαμοίραση Αρχείων, Έξυπνες συσκευές, Εφαρμογή Android, Εφαρμογή για κινητές συσκευές, Mobile Computing, Programming, Smartphones – Programming.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια παρατηρείται η τάση της αγοράς ως προς την τεχνολογία να στρέφεται προς μικρότερες “έξυπνες” συσκευές και λογισμικό που μπορεί να τις υποστηρίξει λόγω των περιορισμένων πόρων τους. Πλέον, τα κινητά τηλέφωνα αποκτούν και άλλες λειτουργίες και παύουν να χρησιμοποιούνται μόνο για ομιλία και αποστολή μηνυμάτων. Επίσης, όλο και περισσότερο τόσο το ευρύ καταναλωτικό κοινό όσο και φορείς Δημόσιας Διοίκησης χρησιμοποιούν λογισμικό ανοιχτού κώδικα.

Για να ανταποκριθεί σε αυτές τις επιταγές, η βιομηχανία της κινητής τηλεφωνίας εφαρμόζει πλέον τεχνολογίες οι οποίες κατευθύνονταν μέχρι πρότινος σε προσωπικούς υπολογιστές. Με χρήση ενός "έξυπνου" κινητού τηλεφώνου είναι δυνατόν να υπάρχει πρόσβαση στο Διαδίκτυο, να ανακτώνται πληροφορίες που σχετίζονται με γεωγραφικά δεδομένα, να γίνονται αναπαραγωγή μουσικής, λήψεις από ιστοσελίδες, πραγματοποίηση τραπεζικών συναλλαγών, και φυσικά τηλεφωνικές κλήσεις και αποστολή μηνυμάτων.

Μεγάλο μερίδιο της αγοράς καταλαμβάνει το Android, ως ένα “οικοσύστημα” που υποστηρίζεται από υλικό, λειτουργικό σύστημα και λογισμικό.

Άρα η επιλογή αυτής της τεχνολογίας ως προς μελέτη θέμα κρίθηκε απαραίτητη. Η παρούσα διπλωματική εργασία έχει ως στόχο τη μελέτη, κατανόηση και ανάλυση μίας εφαρμογής διαμοίρασης αρχείων για την πλατφόρμα Android που προορίζεται για εγκατάσταση σε κινητές “έξυπνες” συσκευές και πρόσβαση από όλες τις πλατφόρμες, έτσι ώστε τα αρχεία να είναι διαθέσιμα από όλες τις συσκευές (προσωπικοί υπολογιστές, κινητά τερματικά) που έχουν σύνδεση στο Διαδίκτυο.

Ένας σημαντικός σκοπός είναι να αναδειχθεί το Android ως μέσο τόσο για την ανάπτυξη εφαρμογών απαραίτητων για καθημερινή χρήση και ως μία πρώτη επαφή με τον αντικειμενοστραφή προγραμματισμό μέσω εργαλείων εύκολων στη χρήση τους. Είναι εξίσου σημαντικό να αναδειχθεί και ως ένα χρηστικό εργαλείο για το ευρύ κοινό κατανοώντας το ευρύ φάσμα των δυνατοτήτων του. Μία λεπτομερής ανάλυση της χρήσης των τεχνολογιών που σχετίζονται με το Android οδηγεί στο συμπέρασμα ότι οι τεχνολογίες αυτές βοηθούν στη συνολική εμπειρία χρήστη.

ABSTRACT

Over the last years the trend of the technology market is directed towards smaller “smart" devices and the software that can support them due to their limited resources. Nowadays, mobile phones provide wider functionality and they are not only used for phone calls and SMS. Also, the general public and public administrations use open source software increasingly.

To meet these requirements, the mobile industry is now applying technologies which were directed previously on PCs. Using a "smart" phone it is possible to have access to the Internet, to retrieve geographical data, listen to music, download from websites, carry out banking transactions, and of course do phone calls and send messages.

A large share of the market belongs to Android as an "ecosystem", which is supported by hardware, operating system and software.

Therefore, the choice of said technology for the subject to be studied was considered of great importance. This thesis aims to study, understand and analyze a file-sharing application for the Android platform that aims to be installed in "smart" mobile devices and be accessed by all platforms via the Internet, so that files are available on all devices (PCs, mobile terminals) that have established an Internet connection.

An important goal of this diploma thesis is for the Android to be emerged as a tool for developing applications required for daily use and as an introduction to Object Oriented Programming via tools that are easy to use. It is essential to be also considered useful for the public, should they have understood the vast range of its possibilities. A detailed analysis of the use of the Android-related technologies leads to the conclusion that these technologies help in the overall user experience.

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1

1.1. Εισαγωγή – Ορισμός προβλήματος

Σήμερα οι σύγχρονες επιταγές επιβάλλουν όλο και περισσότερο τη χρήση της τεχνολογίας. Καθημερινές λειτουργίες τείνουν να απλοποιούνται με τη χρήση τεχνολογιών ήδη γνωστών, αλλά που δεν χρησιμοποιούνταν μέχρι πρότινος ευρέως από το κοινό. Οι τεχνολογίες αυτές πλέον αποτελούν κομμάτι της καθημερινότητάς μας, καθώς εργασίες που απαιτούσαν φυσική παρουσία και ήταν χρονοβόρες, ήδη παρέχουν τη δυνατότητα να διεκπεραιώνονται απομακρυσμένα. Καθώς οι σύγχρονες τάσεις θέτουν στο κέντρο τους την εξυπηρέτηση του καταναλωτή, και η βιομηχανία που σχετίζεται με τις κινητές συσκευές μπορεί πια να παραδώσει προς εμπορική χρήση συσκευές οι οποίες έχουν τη δυνατότητα να παίρνουν τη θέση ενός υπολογιστή για διάφορες εργασίες, κρίθηκε απαραίτητη η μελέτη μίας εφαρμογής που σχετίζεται με τις νέες αυτές τεχνολογίες και των τεχνολογιών που περικλείει. Η εφαρμογή αυτή αφορά τη διαμοίραση αρχείων μέσω Δικτύου και τη διαχείριση τους από οποιαδήποτε συσκευή έχει τη δυνατότητα πρόσβασης στο Διαδίκτυο. Το συγκεκριμένο ζήτημα πλέον αποτελεί μέρος της καθημερινότητάς μας, καθώς παρατηρείται η ανάγκη πρόσβασης στα αρχεία μας από οπουδήποτε και οποιαδήποτε στιγμή.

Αναλυτικά σε κάθε παράγραφο μελετάται:

- Αρχικά, γίνεται ιστορική αναδρομή σχετικά με τη δημιουργία του Android και παρουσίασή του, γίνεται σύγκριση έναντι άλλων λειτουργικών που προορίζονται για κινητές συσκευές. Κατόπιν, επεξηγείται τι ακριβώς είναι το Android και σε ποιες τεχνολογίες έχει στηριχθεί, τα εργαλεία, η ασφάλεια και οι άδειες υπό τις οποίες υφίσταται. Μελετάται εκτενώς η αρχιτεκτονική του Android και η σύζευξη με άλλες τεχνολογίες.
- Στη συνέχεια, ακολουθεί παρουσίαση του Android, η οποία αφορά την εγκατάσταση των εργαλείων του και του περιβάλλοντος ανάπτυξης, τα δομικά συστατικά του και άλλα στοιχεία απαραίτητα για την ανάπτυξη εφαρμογών.
- Στο επόμενο κεφάλαιο αναλύεται η εφαρμογή που έχει αναπτυχθεί στα πλαίσια της παρούσας εργασίας, και ειδικότερα πώς εξαρχής εγκαθίστανται τα απαραίτητα αρχεία στο περιβάλλον ανάπτυξης και πώς έχουν αναπτυχθεί κάποια εξ αυτών. Σε κάθε παράγραφο επεξηγείται ποια είναι η λειτουργία τους, αλλά και πώς εξελίσσουν την εφαρμογή.
- Στο τελευταίο κεφάλαιο δίδονται τα συμπεράσματα από τη διπλωματική εργασία και προτάσεις περαιτέρω ανάπτυξης.

ΚΕΦΑΛΑΙΟ 2

2.1. Η τεχνολογία Android

2.1.1. Γενικά

Το Android είναι ένα λειτουργικό σύστημα, βασισμένο στον πυρήνα (kernel) του Linux έκδοση 2.6 και πιο συγκεκριμένα σε έναν τροποποιημένο πυρήνα που βασίζεται στον πυρήνα του Linux (βλέπε σε πιο κάτω παράγραφο) και το οποίο έχει κυρίως, αλλά όχι αποκλειστικά, σχεδιαστεί για κινητές συσκευές με οθόνη αφής, όπως τα smart phones και τα tablets computers. Έχει, επίσης, χρησιμοποιηθεί σε τηλεοράσεις, κονσόλες παιχνιδιών, ψηφιακές κάμερες και σε άλλες συσκευές. Η διεπιφάνεια χρήστη (User Interface, UI) του Android, εφόσον πρόκειται για συσκευές με οθόνη αφής, ανταποκρίνεται σε "πραγματικές" κινήσεις του χρήστη.

Ο πηγαίος κώδικας του Android κυκλοφορεί από την Google με άδειες ανοιχτού προτύπου, όμως οι συσκευές καταλήγουν στους τελικούς χρήστες με ένα συνδυασμό λογισμικού ανοιχτού και κλειστού κώδικα.

Αρχικά αναπτύχθηκε από την εταιρεία Android Inc. η οποία στην πορεία χρηματοδοτήθηκε και τελικά εξαγοράστηκε το 2005 από την Google. Εμφανίζεται στο ευρύ κοινό το 2007, παράλληλα με την ίδρυση της κοινοπραξίας εταιρειών υλικού, λογισμικού και τηλεπικοινωνιών *Open Handset Alliance (OHA)*, της οποίας σκοπός είναι η ανάπτυξη και προώθηση ανοιχτών προτύπων για κινητά τερματικά.

Το Android SDK παρέχει APIs για τη χρήση web browser, εμφάνιση δισδιάστατων και τρισδιάστατων γραφικών, δομημένη αποθήκευση δεδομένων σε βάση δεδομένων, εμφάνιση πολυμεσικού υλικού (ήχος, βίντεο, εικόνες), χρήση των τεχνολογιών GSM, Bluetooth, EDGE, 3/4G και Wi-Fi, χρήση συσκευών όπως η φωτογραφική μηχανή, GPS, και άλλα. [1]

2.1.2. Evolution Android

Το 2003 ιδρύεται η Android Inc. στο Palo Alto της Καλιφόρνια. Σύμφωνα με τους ιδρυτές, προσπάθησαν να παράγουν ένα λειτουργικό σύστημα ως αντίπαλο δέος του Symbian και του Windows Mobile, το οποίο είναι σε θέση να αναγνωρίζει τις προτιμήσεις του κατόχου του.

Στις 17 Αυγούστου 2005 η Google αγοράζει την Android Inc. και αρχίζει την ανάπτυξη ενός λειτουργικού συστήματος βασισμένου στον πυρήνα του Linux. Πολλοί υποστηρίζουν ότι η

Google προσπαθεί να εισέλθει στην αγορά της κινητής τηλεφωνίας. Μάλιστα, η Google προωθεί το Android σε εταιρείες κινητών συσκευών και σε παρόχους τηλεπικοινωνιών.

Έπονται εικασίες για την πρόθεση της Google να εισέλθει στην αγορά των κινητών επικοινωνιών, οι οποίες διακόπτονται από την εμφάνιση του iPhone της Apple στις αρχές του 2007, ενώ μέσα στη χρονιά η Google έχει καταθέσει πατέντες για εφαρμογές κινητών συσκευών, αλλάζοντας κατά τη διάρκεια τα πρότυπα τα οποία επεξεργαζόταν αρχικά.

Εικόνα 1. Το λογότυπο της OHA.

Στις 5 Νοεμβρίου 2007 υποστηρίζεται από την Open Handset Alliance (OHA), μία κοινοπραξία (consortium) 84 εταιρειών τεχνολογίας, όπως των Google, HTC, Sony, Dell, Samsung Electronics, Nvidia, Motorola και άλλων, δηλαδή εταιρειών κατασκευαστών κινητών τερματικών, συσκευών, εταιρείες ανάπτυξης εφαρμογών, αλλά και παρόχων υπηρεσιών ασύρματων δικτύων (carriers). Τα μέλη της OHA δεν μπορούν να κατασκευάσουν οτιδήποτε μη συμβατό με το λειτουργικό Android, προσβλέποντας σε ένα μελλοντικά ενιαίο "οικοσύστημα". Τα μέλη της κοινοπραξίας έχουν ως σκοπό να αναπτύξουν ανοιχτά πρότυπα για κινητά τερματικά. Έτσι, κυκλοφορεί το HTC Dream, το πρώτο κινητό τερματικό που χρησιμοποιεί Android στις 22 Οκτωβρίου 2008, βασισμένο στην έκδοση του Linux kernel 2.6. Από εδώ και μπρος η Google αναπτύσσει και διανείμει το Android σε κατασκευαστές, οι οποίοι μπορούν να παραμετροποιήσουν και να εμπλουτίσουν την εκάστοτε έκδοση, έτσι ώστε να ταιριάζει στις συσκευές που προωθούν.

Το 2010 η Google λανσάρει τη σειρά κινητών συσκευών Nexus, κατασκευασμένη από συνεργάτες της OHA. Η συγκεκριμένη σειρά θεωρείται από τη Google η ναυαρχίδα της, καθώς παρουσιάζει στο καταναλωτικό κοινό τα τελευταία χαρακτηριστικά του Android, τόσο σε επίπεδο λογισμικού όσο και υλικού. Από το 2008 και έπειτα, έχουν υπάρξει πολλές ενημερώσεις για το λειτουργικό σύστημα Android, όπου η κάθε μία εισάγει καινούρια χαρακτηριστικά, αλλά και βελτιώνει το υπάρχον λειτουργικό σύστημα διορθώνοντας προβλήματα που είχαν παρουσιαστεί σε προηγούμενες εκδόσεις. Κάθε μία από αυτές τις ενημερώσεις παίρνει την ονομασία της ως εξής: ένας αριθμός, οποίος αντιπροσωπεύει ένα επίπεδο Android API, ακολουθούμενο από ένα όνομα γλυκού σε αλφαβητική σειρά, για παράδειγμα η πρώτη εμπορική έκδοση ήταν Android 1.5 Cupcake (Android API level 3). Η τελευταία εμπορική έκδοση είναι Android 4.4.2 KitKat (Android API level 19), που κυκλοφόρησε στις 9 Δεκεμβρίου 2013, ενώ η επόμενη έκδοση, η Android 5.0 Lollipop,

αναμένεται να κυκλοφορήσει στα τέλη του 2014 και έχει κυκλοφορήσει δοκιμαστικά για επιλεγμένες συσκευές Google Nexus στις 26 Ιουνίου 2014. Εκτός από κινητά τερματικά, στοχεύει και σε άλλες πλατφόρμες που χρησιμοποιούν τεχνολογία Android, για παράδειγμα το Android TV και το Android Wear για smartwatches. Οι ανακοινώσεις αναφέρουν ότι υπάρχουν αλλαγές στη διεπιφάνεια χρήστη, στη διαχείριση ενημερώσεων, στην ασφάλεια της συσκευής με τη δημιουργία διαφορετικών λογαριασμών χρηστών, αλλά και την αντικατάσταση της εικονικής μηχανής Dalvik από το Android Runtime (ART) για τη βελτίωση της απόδοσης των εφαρμογών, καθώς το ART θα επιτρέπει τη μεταγλώττιση του πηγαίου κώδικα πριν από τη χρήση από το χρήστη με μεγαλύτερα εκτελέσιμα αλλά και καλύτερη τελική απόδοση των εφαρμογών και εν τέλει βελτιστοποίηση της χρήσης της μπαταρίας, εφόσον η μεταγλώττιση γίνεται μία φορά κατά την εκκίνηση της εφαρμογής, και άλλες αλλαγές που στοχεύουν στη βελτίωση της χρήσης των υπολογιστικών πόρων.

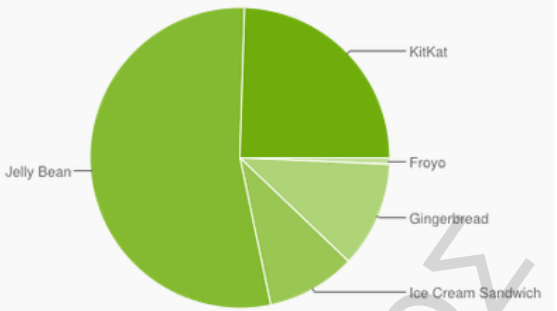
2.1.3. Στατιστικά στοιχεία

Το 2013 οι συσκευές που υποστηρίζουν Android έχουν μεγαλύτερες πωλήσεις από τις συσκευές με Windows, iOS και Mac OS μαζί. Σύμφωνα με στοιχεία του Google Play store - παλαιότερα γνωστό ως Android Market -, της υπηρεσίας που αναπτύσσεται και συντηρείται από τη Google για διανομή εφαρμογών και άλλων υπηρεσιών για την πλατφόρμα Android υπάρχουν περισσότερες από 1.200.000 εφαρμογές (μέτρηση Ιούνιος 2014) και έχει υπολογιστεί ότι περισσότερες από πενήντα δισεκατομμύρια εφαρμογές έχουν κατέβει από το Google Play store.

Σύμφωνα με τα στοιχεία που διατίθενται στην ιστοσελίδα developer.android.com , καθημερινά περισσότερο από ένα εκατομμύριο συσκευές Android ενεργοποιούνται παγκοσμίως, ενώ περισσότερο από 1,5 δισεκατομμύρια εφαρμογών κατεβαίνουν μηνιαίως από τον επίσημο ιστότοπο διάθεσης εφαρμογών, το Google Play .

Στον παρακάτω πίνακα και διάγραμμα παρουσιάζονται πρόσφατα δεδομένα σχετικά με την κατανομή των εκδόσεων του Android, σύμφωνα με το Google Play και τις μετρήσεις που λαμβάνει από τους χρήστες που εισέρχονται στην εφαρμογή Google Play από την κινητή συσκευή τους. Οι εκδόσεις που εμφανίζονται είναι από το Android API level 8 (έκδοση 2.2 Froyo) και έπειτα. Υπολογίζεται όμως ότι οι προηγούμενες εκδόσεις λειτουργούν μόνο στο 1% των κινητών συσκευών. [2]

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%
4.4	KitKat	19	24.5%



Εικόνα 2. Κατανομή των εκδόσεων των Android. (Δεδομένα 9/9/2014).

2.1.4. Σύγκριση έναντι άλλων λειτουργικών

Παρακάτω παρατίθενται αλφαβητικά τα πιο γνωστά λειτουργικά για κινητές συσκευές. Η παρακάτω

παράγραφος επικεντρώνεται στη σύγκριση Android και iOS, αφού αποτελούν τα πιο γνωστά και τα πλέον διαδεδομένα στην αγορά των έξυπνων συσκευών.

- ***Android και iOS***

Το Android της Google και το iOS της Apple είναι λειτουργικά συστήματα τα οποία χρησιμοποιούνται κυρίως για κινητές συσκευές, όπως "έξυπνα τηλέφωνα" και tablets. Το Android μοιάζει περισσότερο με λειτουργικό σύστημα που θα συναντούσε κανείς σε κάποιον υπολογιστή από ότι το iOS, καθώς η διεπιφάνεια και τα βασικά χαρακτηριστικά του είναι ευκολότερο να προσαρμοστούν βάσει του χρήστη της συσκευής. Παρόλα αυτά κάποιες φορές η ομοιόμορφη σχεδίαση του iOS είναι προτιμότερη, καθώς εμφανίζεται ως πιο φιλική προς το χρήστη. [3] Παρακάτω παρατίθεται ένας πίνακας με κάποιες διαφορές και ομοιότητες ανάμεσα στα δύο λειτουργικά συστήματα.

Εταιρεία	Apple Inc.	Google
Οικογένεια Λειτουργικού Συστήματος	OS X, UNIX	Linux
Κατασκευαστής συσκευών	Apple Inc	Google, LG, Samsung, HTC, Sonny, ASUS, Motorola....
Προσαρμόζεται στις ανάγκες του κάθε χρήστη	Ελάχιστα.	Πολύ.
Κυκλοφόρησε	29/07/2007	23/09/2008
Τελευταία έκδοση	7.1 (10 Μαρτίου 2014)	Android 4.4.2 KitKat (Δεκέμβριος 2013)
Γλώσσες προγραμματισμού	C, C++, Objective-C	C, C++, Java
Ανοιχτού / κλειστού κώδικα	Κλειστού με κάποια συστατικά του να είναι ανοιχτού. Ο πυρήνας του είναι κλειστού τύπου, βασισμένος στο ανοιχτού κώδικα Darwin Os	Ανοιχτού, ο πυρήνας του, η διεπιφάνεια χρήστη, ακόμα και κάποιες από τις στάνταρ εφαρμογές του.
"Κατάστημα¨ εφαρμογών	Apple app store, με περισσότερες από ένα εκατομμύριο εφαρμογές.	Google Play, με περισσότερες από ένα εκατομμύριο εφαρμογές. Εφαρμογές Android διατίθενται και

		από ιστότοπους, όπως η Amazon.
Μερίδιο αγοράς	12,9% σε smartphones.	81% σε smartphones.
		Παγκόσμιες συνεργασίες και μεγάλη βάση
Εφαρμογές	Μέχρι πρόσφατα, η Apple κατείχε τα πρωτεία στην έκδοση εφαρμογών, κάτι που όμως περνάει στην κυριαρχία του Android. Όμως μέχρι στιγμής δεν έχει αλλάξει η αξιοπιστία των εφαρμογών της Apple, καθώς εφαρμογές όπως το Instagram είναι προσβάσιμες στους τελικούς χρήστες γρηγορότερα από ότι σε άλλες πλατφόρμες. Επίσης, εφόσον χρειάζεται ένα εύλογο χρονικό διάστημα έτσι ώστε να δοκιμαστεί η προς διάθεση εφαρμογή από την Apple, αναφορικά με την αξιοπιστία της εφαρμογής όταν εγκατασταθεί από το χρήστη, το τελικό προϊόν έχει λιγότερες πιθανότητες να μην χρησιμοποιηθεί, ή να απεγκατασταθεί ή ο χρήστης να μην το χρησιμοποιήσει.	Ανοιχτού κώδικα. Επιτρέπει στους ανεξάρτητους σχεδιαστές/προγραμματιστές, στις εταιρείες και στους χρήστες να τροποποιήσουν ή και να διορθώσουν τον κώδικα, χρησιμοποιώντας την άδεια “Apache Licence”. Εάν θέλει κάποιος να εκδώσει κάποια εφαρμογή μπορεί να το κάνει και εκτός της επίσημης ιστοσελίδας για τις εφαρμογές Android.
Δοκιμή εφαρμογών	SIMULATOR (ΠΡΟΣΟΜΟΙΩΤΗΣ) Αποκλειστικά συσκευές Apple. Εάν η εφαρμογή στοχεύει σε IPhone, τότε μόνο η αντίστοιχη προσομοίωση για αυτήν την	EMULATOR(ΕΞΟΜΟΙΩΤΗΣ) Δίνει τη δυνατότητα να τρέξει η εφαρμογή σε μία πλειάδα συσκευών που χρησιμοποιούν λογισμικό Android. Κάθε φορά πέρα από το API Level που ορίζεται για την

	εικονική συσκευή θα είναι δυνατόν να τρέξει την εφαρμογή.	«εξομοίωση» της εφαρμογής, ορίζεται και η συσκευή που θα στοχεύει. Φυσικά, είναι δυνατόν να τρέξει η εφαρμογή σε όλες τις εικονικές συσκευές που υποστηρίζουν τις αντίστοιχες εκδόσεις που στοχεύει η συγκεκριμένη εφαρμογή.
Περιβάλλον ανάπτυξης εφαρμογών	Apple IDE (Integrated Development Environment) XCODE https://developer.apple.com/xcode/	Είναι δυνατόν να χρησιμοποιηθούν διάφορα περιβάλλοντα ανάπτυξης, π.χ. Eclipse, αρκεί να έχει γίνει εγκατάσταση του Android SDK, το οποίο παρέχει τις απαραίτητες βιβλιοθήκες και τα εργαλεία για την ανάπτυξη, δοκιμή και αποσφαλμάτωση των εφαρμογών. https://developer.android.com/sdk/index.html Το Android Studio (μέχρι στιγμής BETA Edition, v0.8.6), βασισμένο στο περιβάλλον IntelliJ IDEA, αναμένεται να είναι το επίσημο περιβάλλον ανάπτυξης για το Android μόλις ολοκληρωθεί. https://developer.android.com/sdk/installing/studio.html

Η σύγκριση αυτή έχει παρατεθεί λόγω του γεγονότος ότι το Android και το iOS έχουν το μεγαλύτερο μερίδιο της αγοράς.

Η επιλογή ανάμεσα στα δύο λειτουργικά συστήματα και κατ' επέκταση στις συσκευές είναι καθαρά προσωπική. Πολλοί υποστηρίζουν την άποψη ότι παρόλο που το Android είναι πιο ευέλικτο λόγω του ανοικτού κώδικα, έχει χάσει σε επίπεδο εμπειρίας χρήστη, καθώς η

πλατφόρμα διατίθεται σε πολλές διαφορετικές εταιρείες συσκευών, οι οποίες παραμετροποιούν διαφορετικά το λειτουργικό βάσει των συσκευών που διαθέτουν προς εμπορική χρήση. Η Google και η Apple υποστηρίζουν δύο τελείως διαφορετικά επιχειρηματικά μοντέλα στον τομέα των κινητών συσκευών, με την Apple να παίρνει μερίδιο από τις πωλήσεις των συσκευών. Επίσης, αρκετές από τις εφαρμογές που προορίζονται για συσκευές με Android είναι δωρεάν, ενώ για οι αντίστοιχες σε iOS πωλούνται.

- ***BlackBerry***

Οι συσκευές BlackBerry αποτελούν, επίσης, ένα ακόμα μερίδιο της αγοράς στις κινητές συσκευές. Σχεδιάζονται και προωθούνται από την εταιρεία BlackBerry Limited, η οποία ήταν παλαιότερα γνωστή ως Research In Motion Limited (RIM). Η πρώτη συσκευή βγήκε στην αγορά το 1999. Διαφέρουν από τις υπόλοιπες, καθώς οι περισσότερες διαθέτουν ένα φυσικό πληκτρολόγιο QWERTY, ενώ κάποια νεότερα μοντέλα είναι υβρίδιο μεταξύ οθόνης αφής και εικονικού πληκτρολογίου. Το λειτουργικό σύστημα που χρησιμοποιούν τα δύο τελευταία μοντέλα, το Z10και Q10, είναι το BlackBerry 10, το οποίο λανσαρίστηκε στις 30/01/2013. Το λειτουργικό αυτό χαρακτηρίζεται ως Unix-like. Οι χρήστες έχουν πρόσβαση σε εφαρμογές για τις συσκευές τους μέσω του BlackBerry World. Επιπρόσθετα, από τον Οκτώβριο 2011 η RIM ανακοίνωσε επισήμως ότι οι συσκευές της μπορούν να τρέχουν, ακόμα και χωρίς μετατροπές στο τελικό προϊόν, εφαρμογές Android, χρησιμοποιώντας την τελευταία έκδοση του λειτουργικού συστήματος (BlackBerry 10). Οι συσκευές BlackBerry χρησιμοποιούνται από αρκετές υπηρεσίες, καθώς παρέχουν δυνατότητες κρυπτογράφησης. Σήμερα, το μερίδιο αγοράς το οποίο κατέχει το BlackBerry είναι περίπου 1%.

Εικόνα 3. Η συσκευή BlackBerry Q10.

- ***Symbian***

Το Symbian OS δημιουργήθηκε από την εταιρεία Symbian Ltd με τη γλώσσα προγραμματισμού C++. Συσκευές που χρησιμοποιούσαν αυτό το λειτουργικό είναι Nokia, Sony Ericsson, Samsung και άλλες. Ήταν το πλέον δημοφιλές λειτουργικό σύστημα για έξυπνες κινητές συσκευές μέχρι το 2010, όπου το Android πήρε τα πρωτεία στο μερίδιο αγοράς. Πλέον το διαχειρίζεται η Accenture.

Το Nokia 808 PureView, που χρησιμοποιεί την τελευταία έκδοση του λειτουργικό, ονόματι Nokia Belle, είναι επίσημα το τελευταίο έξυπνο κινητό που έχει εγκατεστημένο το Symbian, ενώ πλέον δεν αναπτύσσονται εφαρμογές για την πλατφόρμα. Παρόλα αυτά οι ήδη υπάρχουσες εξακολουθούν να μπορούν να χρησιμοποιηθούν. Για την ανάπτυξη εφαρμογών γινόταν χρήση των Java ME, Adobe Flash Lite και Python, Ruby κ.ά.. Η δημιουργία του λειτουργικού βασίστηκε στις αρχές των περιορισμένων πόρων που υπάρχουν στις κινητές συσκευές και χρησιμοποίησε αρχικά την προσέγγιση microkernel για τον πυρήνα του.

- ***Windows Phone***

Αναπτύχθηκε από τη Microsoft και παρόλο που είναι ο διάδοχος του λειτουργικού για κινητές συσκευές Windows Mobile, δεν υποστηρίζει προς τα πίσω συμβατότητα. Παρουσιάστηκε τον Οκτώβριο του 2010 με μία καινούρια διεπιφάνεια χρήστη. Η τελευταία έκδοση είναι η Windows Phone 8.1 (14/04/2014). Έχει αντικαταστήσει τη χρήση του Symbian στις συσκευές Nokia. Η Microsoft στοχεύει αυτό το λειτουργικό να είναι πιο φιλικό προς τον τελικό χρήστη.

Εικόνα 4. Η συσκευή Nokia Lumia 800 ήταν η πρώτη που προωθήθηκε από τη Nokia με Windows Phone.

Η διεπιφάνεια χρήστη ("Live Tiles") ήταν η έμπνευση για τη δημιουργία της διεπιφάνειας του λειτουργικού συστήματος για προσωπικούς υπολογιστές Windows 8. Χρησιμοποιούνται ως σύνδεσμοι προς τις εφαρμογές και τα αρχεία, οι οποίοι είναι δυναμικοί, καθώς ανανεώνονται εάν υπάρχουν αλλαγές στις διασυνδεόμενες εφαρμογές.

Ένα άλλο χαρακτηριστικό είναι η χρήση "hubs", δηλαδή ο συνδυασμός τοπικών δεδομένων και περιεχομένου που υπάρχει στις ιστοσελίδες κοινωνικής δικτύωσης ή οι επαφές του χρήστη οι οποίες προέρχονται από τους λογαριασμούς ηλεκτρονικού ταχυδρομείου, επαφών που έχει εισάγει ο ίδιος και κοινωνικά δίκτυα. Μεγάλο προσόν, επίσης, είναι ότι όλα τα κινητά τα οποία υποστηρίζουν Windows Phone έχουν προεγκατεστημένη τη σουΐτα του Microsoft Office, η οποία μάλιστα παρέχει διαλειτουργικότητα μεταξύ της έκδοσης που είναι εγκατεστημένη στο κινητό και στην αντίστοιχη σε έναν προσωπικό υπολογιστή. Μάλιστα, μπορεί να γίνει επεξεργασία απευθείας από την κινητή συσκευή. Άλλες εταιρείες που χρησιμοποιούν αυτό το λειτουργικό είναι οι Huawei, HTC, Samsung, ZTE και άλλες. Η πλατφόρμα μέσω της οποίας γίνεται διανομή των

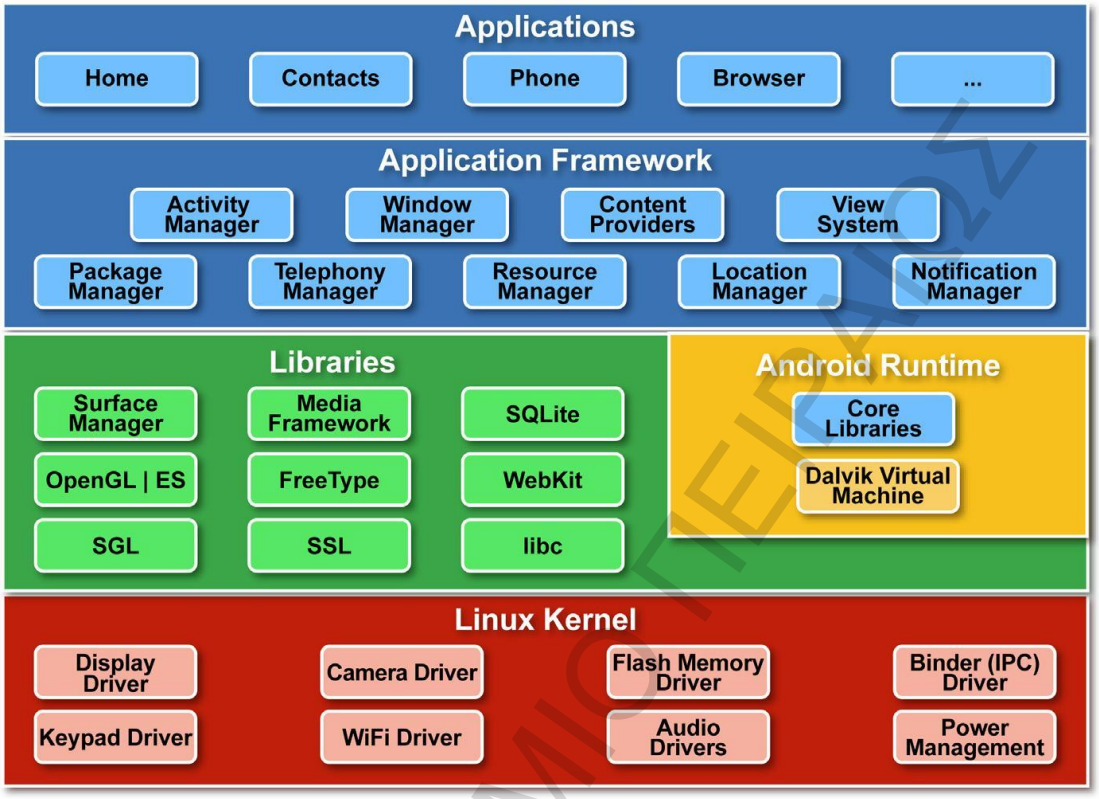
εφαρμογών είναι η Windows Phone Store. Βάσει στατιστικών που αφορούν την προηγούμενη χρονιά (2013), το μερίδιο αγοράς είναι περίπου 3,2%.

- ***Άλλα λειτουργικά συστήματα***

Στην αγορά είναι διαθέσιμα και άλλα λειτουργικά συστήματα για κινητές συσκευές, ενώ παλαιότερα υπήρχαν και άλλα τα οποία πλέον έχει περιέλθει σε αχρηστία. Κάποια από τα πλέον γνωστά είναι:

- το Firefox OS, από το μη κερδοσκοπικό οργανισμό Mozilla Foundation, βασισμένο στο Linux και με χρήση ανοιχτών προτύπων και χρήση HTML5 και Javascript,
- το Ubuntu Touch OS, που αναπτύσσεται από τις Canonical UK Ltd και την κοινότητα του Ubuntu,
- το Sailfish OS από τη Jolla, βασισμένο στο Linux, και
- το Tizen, που υποστηρίζεται από το Linux Foundation, και επιχειρεί να συνδέσει πολλές συσκευές, καθώς διατίθεται για smartphones, tablets, φορητούς υπολογιστές, κάμερες και τηλεοράσεις. Η Samsung έχει ανακοινώσει ότι η σειρά Samsung Z θα υποστηρίζει αυτό το λειτουργικό.

2.2. Η Αρχιτεκτονική του Android



Εικόνα 5. Η στοίβα πρωτοκόλλων του Android.

Το Android είναι μια στοίβα πρωτοκόλλων για κινητά τερματικά, που εμπεριέχει το λειτουργικό σύστημα, ενδιάμεσο λογισμικό (middleware) και άλλες εφαρμογές, μέσα σε μια αρχιτεκτονική. Άρα είναι κατανοητό ότι το Android δεν είναι ακριβώς ένα λειτουργικό σύστημα.

Η αρχιτεκτονική του Android αποτελείται από μία στοίβα λογισμικού (Software Stack), η οποία περιέχει αρκετά επίπεδα ξεκινώντας από τα πιο χαμηλά και φτάνοντας μέχρι το επίπεδο των εφαρμογών. Σχεδιάστηκε κυρίως, αλλά όχι αποκλειστικά, για χρήση σε κινητά τερματικά.

Στο χαμηλότερο επίπεδο (κόκκινη περιοχή) είναι ο πυρήνας Linux έκδοση 2.6.65 (Linux kernel), ο οποίος είναι ειδικά σχεδιασμένος για το Android (μονολιθικός τύπος) και περιέχει οδηγούς για το υλικό, για παράδειγμα Camera Driver, διαχειρίζεται τη μνήμη και τις διεργασίες, τις συνδέσεις δικτύου, τα χαρακτηριστικά των φακέλων χαμηλού επιπέδου, και επιπρόσθετα χειρίζεται τους πόρους και την ενέργεια, λόγω της περιορισμένης παροχής

ενέργειας των κινητών συσκευών. Επί παραδείγματι, επιτρέπει ή όχι την πρόσβαση σε προστατευμένες και μη περιοχές, αλλά διαχειρίζεται και τη μνήμη και τις διεργασίες με τέτοιο τρόπο, ώστε διεργασίες να εκτελούνται ταυτόχρονα χωρίς να παρεμβαίνει η μία στην άλλη. Λειτουργεί επίσης ως ένα ενδιάμεσο επίπεδο μεταξύ της στοίβας λογισμικού και του υλικού μέσω των κλήσεων συστήματος (system calls), επιτρέποντας έτσι την επικοινωνία με το υλικό, όπως κάμερα και μνήμη. Επίσης, ο πυρήνας Linux για το Android περιλαμβάνει ένα μηχανισμό για "επικοινωνία μεταξύ των διαδικασιών", ο οποίος ονομάζεται Binder και επιτρέπει σε μία ή περισσότερες διαδικασίες (processes) να διαμοιράζονται δεδομένα και υπηρεσίες.

Στο αμέσως επόμενο επίπεδο (πράσινη περιοχή), είναι οι βιβλιοθήκες του συστήματος ή εγγενείς βιβλιοθήκες (Native Libraries). Έχουν αναπτυχθεί χρησιμοποιώντας τις γλώσσες προγραμματισμού C και C++. Αυτή η περιοχή είναι υπεύθυνη για το χειρισμό της επίδοσης ευαίσθητων δραστηριοτήτων στη συσκευή. Χρησιμοποιούνται από διάφορα στοιχεία του συστήματος του Android, όπως η απεικόνιση ιστοσελίδων και η ανανέωση της οθόνης (web page rendering και display update). Κάποια στοιχεία αυτής της περιοχής είναι:

- Surface Manager, για την ανανέωση της οθόνης,
- Media Framework, για την αναπαραγωγή αρχείων ήχου και βίντεο,
- WebKit, για την αναπαράσταση και την απεικόνιση ιστοσελίδων,
- SQLite, το οποίο βασίζεται στην SQL, για τη διαχείριση σχεσιακών βάσεων δεδομένων,
- OpenGL, για την απόδοση γραφικών.

Οι προγραμματιστές αποκτούν πρόσβαση στις δυνατότητες αυτού του επιπέδου μέσω του επιπέδου πλαισίου λογισμικού των εφαρμογών (Application Framework).

Σε αυτό το επίπεδο (κίτρινη περιοχή) βρίσκουμε και το επίπεδο εκτέλεσης του Android (Android Runtime), το οποίο υποστηρίζει τη διαδικασία εγγραφής και εκτέλεσης των εφαρμογών, καθώς περιέχει δύο κύρια συστατικά: τις βασικές βιβλιοθήκες της γλώσσας προγραμματισμού Java και την εικονική μηχανή Dalvik. Πιο αναλυτικά, οι βιβλιοθήκες Java, οι οποίες εισάγονται στην αρχή κάθε κλάσης των εφαρμογών, είναι ένα σετ από δυναμικές βιβλιοθήκες, τις οποίες η Java καλεί κατά το χρόνο εκτέλεσης και είναι απαραίτητες, δεδομένου ότι η πλατφόρμα Java δεν εξαρτάται από το λειτουργικό σύστημα στο οποίο εκτελείται, και άρα δεν μπορεί να βασιστεί στις βιβλιοθήκες της κάθε πλατφόρμας. Αντί αυτού, χρησιμοποιεί της βιβλιοθήκες κλάσης, όπου περιέχονται οι απαραίτητης πληροφορίες

για διαδικασίες που είναι κοινές στα σύγχρονα λειτουργικά συστήματα. Αναφορικά με το Android, οι βιβλιοθήκες οι οποίες διατίθενται είναι τέσσερεις και είναι οι βασικές βιβλιοθήκες Java, βιβλιοθήκες σχετικά με την Android εφαρμογή, σχετικές με υπηρεσίες Internet και Web και τέλος βιβλιοθήκες απαραίτητες για τη δοκιμή μερών του κώδικα της εφαρμογής (unit testing) και τα αντίστοιχα πακέτα τα οποία πρέπει να εισαχθούν σε κάθε κλάση είναι java.*, javax.*, android.*, org.* και junit.*. Σημειώνεται ότι για να εισαχθούν αυτά τα πακέτα κλάσεων στην εφαρμογή απαραίτητη είναι η δήλωσή τους στην αρχή κάθε αρχείου .java με τη λέξη import, για παράδειγμα `import android.*;` . Έτσι, γίνεται ακόμα πιο κατανοητή η χρήση της γλώσσας Java ως μία λειτουργική γλώσσα προγραμματισμού, καθώς επαναχρησιμοποιεί κώδικα.

Το δεύτερο πεδίο της στοίβας του Android που αφορά το επίπεδο εκτέλεσης του Android είναι η εικονική μηχανή Dalvik (Dalvik Virtual Machine / VM). Η εικονική μηχανή Dalvik είναι μία από τις διαφορές που έχει το Android σε σχέση με τη Java και τις υπόλοιπες τεχνολογίες. Είναι το λογισμικό εκείνο το οποίο εκτελεί τις Android εφαρμογές. Ο πηγαίος κώδικας Java μεταγλωττίζεται (compiling) με χρήση της εικονικής μηχανής Java (JVM) σε πολλά αρχεία ενδιάμεσου κώδικα (bytecode) Java. Οπότε, εφόσον έχει γραφτεί ο πηγαίος κώδικας και έχει μεταγλωττιστεί, ένα εργαλείο του ADT ή Android Development Tools μετατρέπει τον ενδιάμεσο κώδικα Java σε ένα μόνο αρχείο ενδιάμεσου κώδικα μορφής .dex (classes.dex). Η διαδικασία της μετατροπής του πηγαίου κώδικα (κλάσεων Java) και των υπόλοιπων πόρων σε εφαρμογή κατάλληλη για την πλατφόρμα Android γίνεται από το εργαλείο DX, το οποίο μετατρέπει όλα τα αρχεία σε ένα αρχείο .dex. Όλη η πλεονάζουσα πληροφορία που υπάρχει στα αρχεία του πηγαίου κώδικα βελτιστοποιείται, όπως για παράδειγμα εάν κάποιος πόρος εμφανίζεται σε διάφορες κλάσεις, τότε το αρχείο .dex, μετά τη βελτιστοποίηση, θα περιέχει μόνο μία αναφορά προς αυτόν τον πόρο. Συνεπώς, τα αρχεία .dex είναι μικρότερα σε μέγεθος από τα αντίστοιχα αρχεία κλάσεων. Το αρχείο αυτό μαζί με τους πόρους της εφαρμογής (xml αρχεία, εικόνες, ήχοι κλπ) μετατρέπονται σε ένα αρχείο μορφής .apk (Android Package). Το εργαλείο aapt (Android Asset Packaging Tool) του ADT είναι υπεύθυνο για την εκτέλεση αυτού του βήματος. Το αρχείο .apk που προκύπτει περιλαμβάνει όλες τις απαραίτητες πληροφορίες για την εκτέλεση της εφαρμογής και μπορεί μάλιστα να εκτελεστεί και σε μία συσκευή Android με χρήση του εργαλείου adb.

Αυτό το αρχείο εγκαθίσταται στη συσκευή μαζί με τους υπόλοιπους πόρους της εφαρμογής. Όταν ο χρήστης χρησιμοποιήσει την εφαρμογή αυτή, τότε η εικονική μηχανή Dalvik εκτελεί

το αρχείο classes.dex. Είναι απαραίτητη η χρήση της Dalvik VM έναντι της JVM, καθώς το περιβάλλον εκτέλεσης διαθέτει περιορισμένους πόρους (μπαταρία, μνήμη και ισχύς). Η Dalvik VM είναι σχεδιασμένη για μια τέτοια πλατφόρμα, όπως οι κινητές συσκευές με περιορισμένη διάρκεια μπαταρίας, πιο αργό επεξεργαστή και λιγότερη μνήμη. Πιο κάτω δίνονται αναλυτικότερες πληροφορίες σχετικά με την εικονική μηχανή Dalvik. Η πράσινη και η κίτρινη περιοχή της στοίβας είναι το middleware του Android.

Στο τρίτο επίπεδο (μπλε επίπεδο, πρώτη περιοχή) συναντούμε το πλαίσιο λογισμικού των εφαρμογών (Application Framework), το οποίο είναι ένα επαναχρησιμοποιούμενο λογισμικό, αποτελείται από ένα σύνολο συστημάτων και υπηρεσιών και το οποίο πολλά κινητά τερματικά πιθανόν να χρειαστούν. Επί παραδείγματι, το Σύστημα Γραφικών Στοιχείων (View System) περιλαμβάνει στοιχεία γραφικών, όπως κουμπιά, κουτιά κειμένων (text boxes), εικόνες και άλλα που σχετίζονται με τη διεπιφάνεια χρήστη. Ο Διαχειριστής Πακέτων (Package Manager) παρακολουθεί τα πακέτα των εφαρμογών στη συσκευή επιτρέποντας μία εφαρμογή να χρησιμοποιήσει μία άλλη και να διαμοιραστεί δεδομένα με αυτήν. Ουσιαστικά είναι μία βάση δεδομένων όλων των εγκατεστημένων εφαρμογών της συσκευής. Άλλο στοιχείο είναι ο Διαχειριστής Παραθύρων (Window Manager), που διαχειρίζεται τα παράθυρα των εφαρμογών. Ο Διαχειριστής Πόρων (Resource Manager) διαχειρίζεται μη μεταγλωττίσιμους πόρους, για παράδειγμα τα γραφικά και τους φακέλους που περιέχουν τα strings ή τα layout files. Το στοιχείο Διαχειριστής Δραστηριοτήτων (Activity Manager) διαχειρίζεται τον κύκλο ζωής των εφαρμογών. Επίσης, επειδή συνήθως οι Android activities αντιστοιχούν σε μία μοναδική διεπιφάνεια χρήστη, ο Activity Manager έχει αναλάβει και την πλοήγηση μεταξύ των activities μεταφέροντας τον χρήστη σε διαφορετικές διεπιφάνειες. Οι Πάροχοι Περιεχομένου (Content Providers) είναι ουσιαστικά βάσεις δεδομένων οι οποίες επιτρέπουν στις εφαρμογές να αποθηκεύσουν αλλά και να μοιραστούν δομημένες πληροφορίες. Για παράδειγμα, όπως η εφαρμογή του τηλεφώνου σε μία συσκευή Android χρησιμοποιεί πληροφορίες από μία άλλη εφαρμογή στην οποία είχαν αρχικά εισαχθεί οι πληροφορίες, τις Επαφές. η πληροφορία που χρησιμοποιείται είναι αποθηκευμένη σε έναν Content Provider, ο οποίος και είναι τελικά υπεύθυνος για την πρόσβαση των εφαρμογών σε δεδομένα άλλων εφαρμογών και τη διαμοίραση των πληροφοριών μεταξύ των εφαρμογών. Ο Διαχειριστής Τηλεφωνίας (Telephony Manager) παρέχει πρόσβαση σε πληροφορίες σχετικά με τις υπηρεσίες τηλεφωνίας της συσκευής. Επίσης, χρησιμοποιείται από τις εφαρμογές για την κατάσταση των υπηρεσιών αυτών αλλά και για πρόσβαση σε συγκεκριμένες πληροφορίες

σχετικά με το συνδρομητή, εφόσον ο ίδιος έχει επιτρέψει στην εκάστοτε εφαρμογή να έχει πρόσβαση σε αυτές τις πληροφορίες. Ο Διαχειριστής Τοποθεσίας (Location Manager) είναι υπεύθυνος για την παροχή πληροφοριών σχετικά με την τοποθεσία και την κίνηση της συσκευής Android. Η πληροφορία που παράγεται είναι αντίστοιχη με αυτή ενός GPS. Ο Διαχειριστής Ειδοποιήσεων (Notification Manager) μέσω λογισμικού ειδοποιεί το χρήστη όταν κάτι σημαντικό συμβεί. Οι ειδοποιήσεις αυτές είναι ορατές στην περιοχή μπάρα κατάστασης (Status Bar) της συσκευής στο πάνω μέρος της οθόνης, την οποία διαχειρίζεται το σύστημα, ενώ οι εφαρμογές έχουν πρόσβαση σε αυτήν μέσω του Notification Manager. Είναι σημαντικό ότι παρέχει έναν τρόπο ειδοποίησης των χρηστών για γεγονότα εκτός την διεπιφάνειας της εφαρμογής στην οποία βρίσκονται. Είναι, δε, σχεδόν πάντα ορατή στο χρήστη.

Στο τελευταίο επίπεδο της στοίβας λογισμικού του Android (μπλε επίπεδο) βρίσκεται το επίπεδο των εφαρμογών. Κάθε συσκευή Android έρχεται με κάποιες ενσωματωμένες εφαρμογές, όπως τις Αρχική Οθόνη (Home Screen), Επαφές, δηλαδή τη βάση δεδομένων με τις πληροφορίες των επαφών μας, το Τηλέφωνο, Περιηγητή (Web Browser), Αναγνώστη ηλεκτρονικών μηνυμάτων και άλλες. Όλες οι εφαρμογές που συναντούμε είναι γραμμένες στη γλώσσα προγραμματισμού Java και χρησιμοποιούν τις βιβλιοθήκες κλάσης που αναλύθηκαν στο επίπεδο Android Run Time. Φυσικά, λόγω της φιλοσοφίας του Android, καμία από αυτές τις προεγκατεστημένες εφαρμογές δεν είναι απαραίτητα αυτή που πρέπει να χρησιμοποιήσουμε, καθώς μπορούμε να αντικαταστήσουμε τις εφαρμογές αυτές είτε με κάποια δική μας είτε με κάποια προερχόμενη από ένα τρίτο μέλος, έχοντας και την αντίστοιχη άδεια. [4] [5] [6]

2.1.5. Γιατί Linux ;;;

- Δεν κοστίζει καθώς είναι ελεύθερη η χρήση του.
- Είναι ανοιχτού κώδικα, με δυνατότητα επεξεργασίας και αλλαγών. Κάθε χρήστης μπορεί να τροποποιήσει τον κώδικα, με τη μόνη υποχρέωση να διανείμει τον τροποποιημένο πηγαίο κώδικα.
- Είναι εύρωστο, καθώς έχει περάσει από πληθώρα δοκιμών.

- Είναι ευρέως διαδεδομένο σε όλον τον κόσμο, με τους χρήστες του να αυξάνονται συνεχώς, όπως και των μελών της κοινότητας που ασχολείται με την αναβάθμισή του. [7]

2.1.6. Βασικά Χαρακτηριστικά του Linux

Το Linux είναι ένα ελεύθερου τύπου UNIX (UNIX-like) λειτουργικό που δημιουργήθηκε από το Linus Torvalds και αναπτύχθηκε χάρη στους διάφορους προγραμματιστές που αξιοποιούν και αναπτύσσουν τον πηγαίο κώδικά του. Η άδεια που χρησιμοποιεί είναι η GPL (GNU General Public License). Ολόκληρο το κείμενο της εν λόγω άδειας βρίσκεται στην ηλεκτρονική διεύθυνση http://www.gnu.org/copyleft/gpl.html . [8]

Το Linux μπορεί να λειτουργήσει σε διάφορα υπολογιστικά συστήματα, όπως σε κινητές συσκευές, προσωπικούς υπολογιστές και υπερυπολογιστές. Παρότι είναι συνηθέστερο να χρησιμοποιείται σε υπερσυστήματα, σε κινητά τερματικά ή διακομιστές, τα τελευταία χρόνια με την αύξηση της χρήσης των netbooks που έχουν περιορισμένους πόρους, αλλά και της έκδοσης Ubuntu η οποία είναι εξαιρετικά δημοφιλής στους οικιακούς χρήστες, παρατηρείται άνοδος της χρήσης του σε προσωπικούς υπολογιστές. Υπάρχουν διαφορετικές διανομές Linux με μεγάλες δυνατότητες παραμετροποίησης από τον τελικό χρήστη. Η κάθε διανομή έχει διαφορετικά χαρακτηριστικά, όπως φιλικότητα προς το χρήστη (διανομή Ubuntu), απλότητα συστήματος, ευκολία παραμετροποίησης και άλλα. [9]

Το λειτουργικό Linux αποτελείται από μία στοίβα λογισμικού, όπου το κάθε αντικείμενο είναι σχεδιασμένο να εκτελεί συγκεκριμένες εργασίες. Ο πυρήνας του (Kernel) είναι υπεύθυνος για την επικοινωνία με το υλικό στο οποίο έχει γίνει η εγκατάσταση του λειτουργικού και για τη διαχείριση των συσκευών, της μνήμης, των κλήσεων συστήματος και των διαδικασιών. Οι βιβλιοθήκες συστήματος (System Libraries) είναι πρότυπα τα οποία υλοποιούν τις κλήσεις από τις εφαρμογές προς τον πυρήνα. Είναι ο ενδιάμεσος στην επικοινωνία πυρήνα - προγραμματιστή. Είναι προγράμματα τα οποία χρησιμοποιούνται από τους προγραμματιστές για τη συγγραφή λογισμικού για το λειτουργικό σύστημα και διαθέτουν μεθόδους για τη δημιουργία και τη διαχείριση διαδικασιών, τη διαχείριση φακέλων, δικτύων κ.τ.λ. Τα εργαλεία συστήματος (system tools) έχουν υλοποιηθεί βάσει των βιβλιοθηκών συστήματος και εκτελούν εντολές για τη διαχείριση και την πλοήγηση στους φακέλους, την εκτέλεση άλλων προγραμμάτων, τη διαμόρφωση των δικτύων κ. ά. Τα

εργαλεία ανάπτυξης (development tools) είναι απαραίτητα για τη δημιουργία νέου λογισμικού. Τα εργαλεία συστήματος και ανάπτυξης είναι εμφανή στον τελικό χρήστη.

Κάποια από τα βασικά χαρακτηριστικά του λειτουργικού συστήματος Linux είναι:

- Είναι ανοικτού κώδικα (open source). Ο πηγαίος κώδικας, συμπεριλαμβανομένου του πυρήνα και των οδηγών, του είναι διαθέσιμος προς όλους και η δυναμική του αυξάνεται λόγω της κοινότητας που το υποστηρίζει, το διατηρεί και το εξελίσσει.
- Ο πυρήνας του (Kernel) με καλά δομημένες διεπαφές. Υποστηρίζεται από μία μερίδα της κοινότητας ότι όταν λέμε Linux, εννοούμε τον πυρήνα του, ενώ το λειτουργικό είναι η κάθε έκδοση με διαφορετικά χαρακτηριστικά που φυσικά περιέχει τον πυρήνα Linux. Χαρακτηρίζεται ως modular – monolithic kernel, δηλαδή ο πυρήνας είναι ένα αυτόνομο και ενιαίο πρόγραμμα (monolithic) που ελέγχει το υλικό και τις διεργασίες του υπολογιστή και διαχειρίζεται το υλικό ενεργοποιώντας οδηγούς (modules), δηλαδή drivers. Όλο το λειτουργικό σύστημα σε έναν πυρήνα monolithic πραγματοποιεί τις διεργασίες του σε ένα ενιαίο χώρο διευθύνσεων, στο χώρο του πυρήνα (kernel space), και σε κατάσταση πυρήνα, για λόγους αυξημένης απόδοσης, καθώς η μετάβαση από το χώρο του χρήστη (user space) στο χώρο του πυρήνα παρουσιάζει σημαντικό χρονικό κόστος και μειώνει τις επιδόσεις του συστήματος εκτελώντας πολλές κλήσεις συστήματος.
- Το "κέλυφος" (shell), στις περισσότερες διανομές Linux ονομάζεται bash και είναι ένα πρόγραμμα διερμηνευτής το οποίο μεταφέρει τις εντολές που έχουμε πληκτρολογήσει στο λειτουργικό για να τις εκτελέσει, ενώ μπορεί επίσης να "διαβάσει" εντολές από αρχεία οι οποίες ονομάζονται scripts. Σε παλιότερες εκδόσεις ήταν η μόνη διεπιφάνεια χρήστη, γνωστή ως διεπιφάνεια γραμμής εντολών, ενώ τώρα υπάρχει και η γραφική διεπιφάνεια χρήστη.
- Multi-Tasking: Είναι δυνατόν πολλά προγράμματα να εκτελούνται ταυτόχρονα στο Linux. Αυτά τα προγράμματα μπορεί να αφορούν διαδικασίες του χρήστη, αλλά μπορεί και το λειτουργικό να εκτελεί δικά του προγράμματα παρασκηνιακά.
- Multi-User: Σε ένα σύστημα Linux μπορούμε να έχουμε πολλαπλούς λογαριασμούς χρηστών, αλλά και πολλαπλούς χρήστες συνδεδεμένους ταυτόχρονα. Ο κάθε χρήστης μπορεί να ορίζει το περιβάλλον του όπως επιθυμεί, να έχει πρόσβαση σε πόρους του συστήματος ταυτόχρονα με τους υπόλοιπους χρήστες, αλλά και να έχει λογαριασμό που προστατεύεται από κωδικό, έτσι ώστε να μην έχουν άλλοι χρήστες πρόσβαση στις

εφαρμογές και στα δεδομένα του. Είναι ένας από τους κύριους λόγους για τους οποίους το λειτουργικό Linux θεωρείται ασφαλές. Το χαρακτηριστικό αυτό το μοιράζεται με το Android.

- Multi-Platform: μπορεί να εκτελείται σε διάφορες κεντρικές μονάδες επεξεργαστών.
- Multi-Processing, για τη βελτίωση της απόδοσης του υλικού, είτε επιλέγοντας οι επεξεργαστές να εκτελέσουν διαφορετικές εργασίες είτε να τρέξουν παράλληλα εκτελώντας την ίδια εργασία (asymmetrical multi-processing και symmetrical multi-processing). [10] [11]
- Η αρχιτεκτονική του, που είναι ανεξάρτητη της εκάστοτε τεχνολογίας και εκτελείται το ίδιο ασχέτως του διαφορετικού υλικού (hardware).
- Η ασφάλεια του χρήστη, καθώς παρέχει τη δυνατότητα πιστοποίηση των χρηστών με κωδικούς προστασίας, ελεγχόμενη πρόσβαση σε φακέλους και κρυπτογράφηση δεδομένων.
- Δυναμικό μέγεθος του χώρου που καταλαμβάνει η cache memory.
- Τα εκτελέσιμα μπορούν να φορτώνονται κατά απαίτηση (Load on Demand), ώστε να αξιοποιούνται στο μέγιστο οι υπολογιστικοί πόροι και η μνήμη του συστήματος.
- Dynamically Linked Shared Libraries (dll), είναι βιβλιοθήκες που "φορτώνονται" από τα προγράμματα όταν αυτά ξεκινούν και συνήθως είναι κλάσεις και μέθοδοι που έχουν μετατραπεί σε ένα αρχείο το οποίο χρησιμοποιείται από διάφορα προγράμματα, συνήθως για την ανάπτυξη λογισμικού. Εξ ορισμού χρησιμοποιούνται από τα προγράμματα του Linux. Εάν μία καινούρια έκδοση της βιβλιοθήκης εγκατασταθεί, τα προγράμματα που την χρησιμοποιούσαν πρέπει να είναι σε θέση να χρησιμοποιήσουν την καινούρια έκδοση, ειδάλλως πρέπει η παλαιότερη έκδοση να είναι διαθέσιμη. Επιπρόσθετα, πρέπει να μπορούν να προσπελάσσονται συγκεκριμένες μέθοδοι αυτών από τα προγράμματα κατά την εκτέλεσή τους. Στο Linux τοποθετούνται σε καταλόγους όπως /lib και οι δυναμικές κοινόχρηστες βιβλιοθήκες έχουν κατάληξη .so.
- Στατικές βιβλιοθήκες επίσης.
- Διάφορες μορφές εκτελέσιμων αρχείων.
- Διαφορετικά πρωτόκολλα δικτύου, όπως τα TCP, IPv4, IPv6.
- Διαφορετικά File Systems, μεταξύ των οποίων FAT32, NTFS, BSD ufs, και άλλα.

- Δικαιώματα υπερχρήστη (root access), το οποίο όμως στο Android επιτυγχάνεται η πλήρης πρόσβαση και η αφαίρεση των περιορισμών "σπάζοντας" κατά κάποιο τρόπο το σύστημα. Η διαδικασία αυτή καλείται rooting και επιτρέπει στη συσκευή να υπερβεί τους περιορισμούς που έχουν τεθεί από τον πάροχο ή από τους κατασκευαστές. [12]

Το Android δεν είναι ακριβώς ένα λειτουργικό σύστημα. Είναι μία στοίβα λογισμικού για κινητές συσκευές, η οποία περιλαμβάνει ένα λειτουργικό σύστημα και βασικές εφαρμογές, που "τρέχουν" στην Dalvik Virtual Machine.

Ο πυρήνας του Android είναι βασισμένος στον πυρήνα του Linux. Βασικά χαρακτηριστικά του Android, ανάμεσα σε άλλα, είναι ότι είναι ένα λειτουργικό σύστημα βασισμένο στο πυρήνα Linux (ή κατά το ορθότερο GNU/Linux, καθώς χρησιμοποιεί πολλά προγράμματα και βιβλιοθήκες του GNU), το γραφικό περιβάλλον χρήστη προγραμματίζεται με χρήση των γλωσσών C, C++ και Java, ενώ πρόσφατα έχει αρχίσει να χρησιμοποιείται για τον κώδικα των εφαρμογών και το "πακέτο" HTML5, CSS3 και JavaScript. Οι συσκευές που υποστηρίζουν το Android έχουν επεξεργαστές ARM αρχιτεκτονικής, δηλαδή μία αρχιτεκτονική συνόλου εντολών 32-bit όσον αφορά τους μικροεπεξεργαστές και μικροελεγκτές, σχετικά απλή και κατάλληλη για συσκευές με χαμηλή κατανάλωση ισχύος, όπως οι κινητές συσκευές και τα ενσωματωμένα συστήματα, και έχουν σχετικά μικρό κόστος, ενώ παράλληλα έχουν υψηλή απόδοση. Τέλος, είναι λογισμικό ανοιχτού κώδικα, δηλαδή επιτρέπεται το δωρεάν "κατέβασμα", η εμπορία και η δημιουργία δικιάς μας διανομής (Android-based διανομή). Όμως, οι οδηγοί των συσκευών αυτών είναι κλειστού κώδικα. Παρόλα αυτά, επειδή οι συσκευές χρησιμοποιούν τον πυρήνα του Linux, οι εταιρείες είναι υποχρεωμένες να δίνουν τον κώδικα στο ευρύ κοινό και έτσι έπειτα μπορεί να το επεξεργαστούμε.

Έτσι, το Android ανήκει σε όλους αλλά και σε κανέναν. Υπεύθυνη για την ανάπτυξη του Android είναι η Google, η οποία ίδρυσε την Open Handset Alliance (OHA), που είναι μία κοινοπραξία πολλών μεγάλων εταιρειών τεχνολογίας για την προστασία των ανοιχτών προτύπων, η οποία όμως ουσιαστικά προστατεύει το Android.

Οι διανομές Linux αποτελούνται από τον πυρήνα σε συνδυασμό με συνοδευτικά προγράμματα, όπως βιβλιοθήκες, εργαλεία συστήματος, παραθυρικό περιβάλλον εργασίας και άλλες εφαρμογές απαραίτητες για την εύρυθμη λειτουργία μίας συσκευής (προσωπικός υπολογιστής, κινητό τερματικό). Χαρακτηριστικό των διανομών είναι η μεγάλη δυνατότητα

παραμετροποίησης και ανάλογα με τη φιλοσοφία της κάθε διανομής δίνεται έμφαση στις ευκολίες παραμετροποίησης, στη φιλικότητα προς το χρήστη κτλ.

Είναι όμως όντως το Android μία Linux διανομή; Η απάντηση είναι όχι και ναι. Πολλοί υποστηρίζουν ότι το Android δεν είναι ακόμα μία διανομή. Εντούτοις, εκπληρώνει τις βασικές αρχές μίας διανομής βασισμένης στο πυρήνα του Linux. Οι τρεις αυτές αρχές είναι το απαραίτητο υλικό (hardware) για να εγκατασταθεί ο πυρήνας, είτε είναι κινητή συσκευή είτε προσωπικός ηλεκτρονικός υπολογιστής, έπειτα πρέπει να εγκατασταθεί ο πυρήνας του Linux, ο οποίος και ενώνει το λογισμικό (software) με το υλικό, δηλαδή οι εντολές που δίνουμε μέσω του πυρήνα ενεργοποιούν ανάλογα τις συσκευές του υλικού, και τέλος, το λογισμικό, είτε αφορά οθόνες αφής είτε γραφικό περιβάλλον προσωπικού υπολογιστή. Υποστηρίζεται επίσης ότι το Android είναι μία ακόμα έκδοση "λειτουργικού συστήματος" βασισμένη στο Linux kernel επειδή προσφέρει στον πυρήνα βελτιώνοντάς τον.

Η προσφορά του Android στο Linux όμως είναι μεγαλύτερη. Το ευρύ κοινό γνώρισε το Linux και τα ανοιχτά πρότυπα μέσω του Android. Η διανομή Ubuntu του Linux μπορεί να ήταν επαναστατική, καθώς σήμαινε την έλευση του γραφικού περιβάλλοντος και του user experience στο Linux, όμως το Android σηματοδότησε την έναρξη των παιχνιδιών στο Linux, κάτι το οποίο ήταν προορισμένο μόνο για χρήστες των Microsoft Windows. Αποτέλεσμα είναι το OpenGL ES (Open Graphics Library for Embedded Devices, το οποίο είναι μία διεπαφή -API- για 2D/3D γραφικά ανεξαρτήτως πλατφόρμας και συσκευής, που συνεργάζεται στενά με το υλικό με πολύ καλά αποτελέσματα απόδοσης (rendering) γραφικών και η οποία συνδέεται με τις γλώσσες C, C++, Java, Ruby και άλλες), το οποίο είναι ανοιχτές βιβλιοθήκες που χρησιμοποιεί το Android για τη δημιουργία γραφικών και παιχνιδιών. Επίσης, σημαντική είναι η συνεισφορά του Android στην εκπαίδευση νέων προγραμματιστών, καθώς μέσα από τη δημιουργία εφαρμογών και με τη χρήση εργαλείων (για παράδειγμα Eclipse, Android SDK) εύκολων σχετικά για κάποιον που έχει ελάχιστες γνώσεις προγραμματισμού, έχουν μία πρώτη επαφή με τον προγραμματισμό και τους βοηθά να αναπτύξουν δεξιότητες σχετικές με την εκμάθηση συγγραφής κώδικα, όπως την υλοποίηση ιδεών, τη συνεργασία και τη δημιουργικότητα. [13]

Παρόλο που οι περισσότερες εφαρμογές που σχεδιάζονται για να διατεθούν σε τερματικά Android γράφονται στη γλώσσα προγραμματισμού Java, υπάρχουν σημαντικές διαφορές ανάμεσα στις δύο αυτές διεπαφές προγραμματισμού εφαρμογών (ή API – Application Programming Interface). Πιο αναλυτικά, μία διαφορά έγκειται στο γεγονός ότι το Android

API δε χρησιμοποιεί κατά τη μεταγλώττιση το Java Virtual Machine, αλλά μία άλλη που ονομάζεται Dalvik Virtual Machine, όπου Virtual Machine ένας υποθετικός υπολογιστής, στην ουσία προδιαγραφές οι οποίες μιμούνται έναν πραγματικό (φυσικό) υπολογιστή/συσκευή. Εδώ και στις δύο περιπτώσεις οι VM μιμούνται όχι ένα λειτουργικό αλλά φιλοξενούνται σε ένα λειτουργικό και μιμούνται μία διαδικασία και εκτελούν ακολουθίες bytes (bytecodes). Στο Android API, όμως, αντί των αρχείων .class , τα οποία δημιουργούνται μετά τη μεταγλώττιση στο Java API, διαμορφώνονται σε Dalvik.dex (Dalvik executables / εκτελέσιμα), το οποίο περιέχει όλα τα αρχεία της εφαρμογής.

Επίσης, είναι σημαντικό να σημειωθεί ότι εξαιτίας του γεγονότος ότι το Android είναι βασισμένο στον πυρήνα του Linux, κάθε εφαρμογή είναι απομονωμένη από τις υπόλοιπες εφαρμογές που εκτελούνται, όπως θα αναλυθεί και στη συνέχεια του κειμένου. Εάν είναι απαραίτητο να διαμοιραστούν δεδομένα μεταξύ διαφορετικών εφαρμογών, πρέπει να το κάνει μέσω ενός συστατικού του Android που χειρίζεται τη διαμοίραση δεδομένων, όπως μία Υπηρεσία είναι ένας Πάροχος Περιεχομένου. Τα συστατικά αναφέρονται παρακάτω στο κείμενο.

2.1.7. Η χρήση της γλώσσας Java

Ενώ το λειτουργικό Android βασίζεται στον πυρήνα Linux και άρα στη γλώσσα C, το SDK χρησιμοποιεί τη γλώσσα προγραμματισμού Java για τη δημιουργία των εφαρμογών. Όμως, όπως ήδη έχει τονιστεί, τα αρχεία που προκύπτουν από τον πηγαίο κώδικα (bytecodes) στοχεύουν στην εικονική μηχανή Dalvik και όχι στην εικονική μηχανή Java και είναι σχεδιασμένα για συσκευές με περιορισμένους πόρους. Είναι δυνατή η χρήση και άλλων γλωσσών προγραμματισμού στον πηγαίο κώδικα, ο οποίος με τις κατάλληλες ρυθμίσεις θα σχηματίσει κατάλληλα αρχεία για το Android. Η χρήση της Java, όμως, παρέχει πολλά πλεονεκτήματα έναντι άλλων γλωσσών. Οι αρχικοί στόχοι της Java ήταν μία γλώσσα "απλή και αντικειμενοστραφής, εύρωστη και ασφαλής, ανεξάρτητη πλατφόρμας και φορητή, με εκτελέσιμα αρχεία υψηλής απόδοσης, νηματική και δυναμική". Άρα, η Java είναι αντικειμενοστραφής, δημιουργεί δηλαδή κλάσεις οι οποίες χρησιμοποιούνται ως πρότυπα για τη δημιουργία αντικειμένων. Χρησιμοποιεί ταυτόχρονη (κατανεμημένη) επεξεργασία (concurrent computing), όπου πολλαπλές διεργασίες εκτελούνται ταυτόχρονα αντί σειριακά και όπου πολλαπλές διεργασίες υλοποιούνται είτε ως ξεχωριστά προγράμματα είτε ως ένα

σύνολο διεργασιών ή νημάτων που έχουν δημιουργηθεί από ένα μόνο πρόγραμμα. Δηλαδή, αντί να περιμένει η μία να ολοκληρωθεί η προηγούμενη, εκτελούνται κατά τρόπο όπου συμπίπτουν χρονικά. Υπάρχει ένα ξεχωριστό νήμα για κάθε μία από αυτές τις διεργασίες. Έτσι, κάθε μία διαιρείται σε υπο – διεργασίες που εκτελούνται ταυτόχρονα, αλλά όχι παράλληλα, καθώς στην παράλληλη επεξεργασία η εκτέλεση συμβαίνει την ίδια χρονική στιγμή, για παράδειγμα διαφορετικοί επεξεργαστές που εκτελούν εντολές σε ένα σύστημα, και άρα είναι αδύνατον να συμβεί σε έναν μόνο επεξεργαστή, εφόσον είναι δυνατή η εκτέλεση μόνο μίας επεξεργασίας ανά χρονική στιγμή, ενώ στην ταυτόχρονη μπορεί να αλληλοκαλύπτονται οι χρόνοι των επεξεργασιών, αλλά δεν είναι απαραίτητη η εκτέλεση στον ίδιο χρονικό κύκλο. Η φορητότητα, δηλαδή η εκτέλεση των προγραμμάτων που έχουν αναπτυχθεί με Java σε οποιαδήποτε πλατφόρμα / λειτουργικό σύστημα / υλικό, επιτυγχάνεται με την μετατροπή του πηγαίου κώδικα σε αρχεία bytecode που εκτελούνται σε διαφορετικές πλατφόρμες έναντι αρχείων προορισμένων για κάθε ένα από αυτά τα συστήματα ξεχωριστά. Τα αρχεία αυτά είναι αντίστοιχα αυτών που δημιουργούνται σε γλώσσα μηχανής, αλλά απευθύνονται για την εκτέλεσή τους στην εικονική μηχανή, διαφορετική κάθε φορά ανάλογα με την πλατφόρμα. Οι τελικοί χρήστες εκτελούν τα αρχεία αυτά σε ένα περιβάλλον χρόνου εκτέλεσης Java (Java Runtime Environment, JRE) εγκατεστημένο στη δική τους συσκευή. Παρόλο που θεωρείται πιο αργή διαδικασία από ότι τα εγγενή εκτελέσιμα αρχεία, οι πρόσφατες τροποποιήσεις στην εικονική μηχανή και η μη χρήση δεικτών εντέλει ωφελούν αντί να μειονεκτούν. Είναι ανεξάρτητη πλατφόρμας, καθώς η μεταγλώττιση του πηγαίου κώδικα σε αρχεία που ονομάζονται bytecodes εκτελούνται από οποιαδήποτε συσκευή, σύστημα ή λειτουργικό που χρησιμοποιεί διερμηνευτή Java. Είναι μία ασφαλής γλώσσα, καθώς θεωρείται ευκολότερη στην εκμάθησή της σε σχέση με άλλες γλώσσες, όπως για παράδειγμα η C, και όπου η δέσμευση και η αποδέσμευση μνήμης γίνεται αυτόματα από την Java, δεν περιλαμβάνει δείκτες και έχει μόνο μονή κληρονομικότητα, δηλαδή κληρονομεί μόνο από μία υπερκλάση, και όλα αυτά οδηγούν στην ασφάλεια της γλώσσας. Για τη συλλογή απορριμάτων, που είναι η ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο, υπάρχει ο συλλέκτης απορριμάτων (garbage collector) και είναι μία διαδικασία που γίνεται αυτόματα στη Java, και ενεργοποιείται μόλις η εικονική μηχανή καταλάβει ότι συσσωρεύονται πολλά αντικείμενα στη μνήμη. Οπότε, ο προγραμματιστής δεν ασχολείται με δείκτες και με πιθανά σφάλματα, αλλά και ούτε πότε θα απελευθερωθεί τμήμα της μνήμης. Οι δείκτες που χρησιμοποιούνται σε άλλες γλώσσες, όπως

στη C, είναι πολύ χρήσιμοι για τον προγραμματιστή, καθώς μπορεί να καθορίσει πώς θα λειτουργεί η μνήμη και βοηθούν στη γρήγορη ανταπόκριση των προγραμμάτων, αλλά τα σφάλματα που μπορεί να προκύψουν λόγω κακού χειρισμού των δεικτών είναι κοινά. Άρα, η Java μπορεί να χαρακτηριστεί πιο ασφαλής και γρήγορη λόγω της έλλειψης των δεικτών και του αυτοματοποιημένου χειρισμού του συλλέκτη απορριμάτων. Επιπλέον, με τροποποιήσεις που γίνονται για τη βελτιστοποίηση της εικονικής μηχανής, η ταχύτητα της Java φτάνει ή και ξεπερνά τις αντίστοιχες άλλων γλωσσών υψηλού επιπέδου. Η χρήση της εικονικής μηχανής βοηθάει περαιτέρω στην ασφάλεια του κώδικα, καθώς είναι υπεύθυνη για την επικοινωνία χρήστη – μηχανής. Εάν ο προγραμματιστής γράψει κώδικα με καταστροφικά αποτελέσματα ή ο χρήστης εκτελέσει ένα αρχείο με κακό κώδικα, δεν θα το επιτρέψει η εικονική μηχανή. Για όλα τα παραπάνω, η Java έχει μεγάλη ζήτηση και έχει αρκετές χρήσεις, όπως παιχνίδια, εφαρμογές κινητών εκτός Android, πακέτα λογισμικού για εταιρείες, και άλλα. [12] [14] [15] [16] [17]

2.3. Εργαλεία του Android

Στην παρούσα παράγραφο θα αναφερθούν επιγραμματικά κάποια από τα εργαλεία του Android, τα οποία παρουσιάζονται στην παρούσα διπλωματική.

Το Android Developer Tools (ADT), είναι ένα πρόσθετο για το Eclipse IDE, που παρέχει τη σουΐτα εργαλείων κατάλληλων για την ανάπτυξη εφαρμογών Android. Προσφέρει εργαλεία για τη συγγραφή κώδικα, την αποσφαλμάτωση, τη δημιουργία των τελικών πακέτων των εφαρμογών και την εγκατάσταση, καθώς επίσης παρέχει πρόσβαση στα εργαλεία του SDK (Software Developer Kit), και διαθέτει editors για Java και XML. Μερικά από τα εργαλεία του SDK που παρέχονται μέσω του πρόσθετου ADT είναι:

1. Το Android Debug Bridge (adb), ένα εργαλείο γραμμής εντολών το οποίο επιτρέπει την επικοινωνία με τον εξομοιωτή ή με τη συνδεδεμένη συσκευή μέσα από το περιβάλλον ανάπτυξης. Είναι ένα πρόγραμμα βασισμένο στο μοντέλο πελάτη - διακομιστή, όπου όταν ο πελάτης adb ξεκινά να τρέχει στην συσκευή ανάπτυξης, ελέγχει εάν υπάρχει διακομιστής. Εάν όχι, εγκαθιστά έναν στη θύρα TCP 5037 και αναμένει εντολές από τον πελάτη. Τότε, συνδέεται με εξομοιωτές / συσκευές που υπάρχουν στις θύρες με ζυγό αριθμό στο εύρος τιμών 5555 έως 5585. Κάποια χαρακτηριστικά του adb είναι προσβάσιμα από το ADT, όπως η εγκατάσταση, η μεταφορά φακέλων και άλλα, ενώ

κάποια άλλα, όπως οι εντολές κελύφους, είναι προσβάσιμες μόνο μέσω της γραμμής εντολών.

2. Τον Android Emulator (Εξομοιωτής Android), ο οποίος αναλύεται διεξοδικά παρακάτω.
3. Το Android Asset Packaging Tool (aapt), το οποίο επιτρέπει τη δημιουργία, ενημέρωση και προβολή αρχείων με καταλήξεις .zip, .jar, .apk. Αναλαμβάνει να δημιουργήσει το αρχείο .apk που περιέχει την εφαρμογή Android.
4. Εάν υπάρχουν .aidl διεπαφές στον κώδικα, το εργαλείο aidl τις μετατρέπει σε διεπαφές Java.
5. Το Dalvik Debug Monitor Service ή DDMS, το οποίο βοηθά στην αποσφαλμάτωση παρέχοντας στην "προοπτική" του τις επιλογές της απεικόνισης της οθόνης του τερματικού (εξομοιωτής / συσκευή) και της αλληλεπίδρασης με αυτό, τις τρέχουσες εργασίες, τη δημιουργία "πλαστών" εισερχόμενων κλήσεων και μηνυμάτων, την παροχή γεωγραφικών δεδομένων, την παρακολούθηση των νημάτων, περιέχει το Traceview, στατιστικά δικτύου, την απεικόνιση των φακέλων που περιέχονται στη συσκευή / εξομοιωτή και τη διαχείριση αυτών, μεταξύ άλλων. Στο παρόν κείμενο αναφέρεται και σε άλλες περιπτώσεις η χρήση του DDMS και οι λειτουργίες του.
6. Το εργαλείο dex είναι απαραίτητο για τη μετατροπή των αρχείων .class σε αρχεία ενδιάμεσου κώδικα κατάλληλα για το Android σύστημα, τα Dalvik bytecodes. Τα αρχεία .class δημιουργούνται μετά τη μεταγλώττιση του πηγαίου κώδικα, του αρχείου R.java και, εάν υπάρχουν των αρχείων .aidl από τον Java compiler, ενώ περιέχει επίσης και τις βιβλιοθήκες που έχουν εισαχθεί. Τη δημιουργία των αρχείων .dex θα ακολουθήσει η δημιουργία του τελικού αρχείου .apk, μαζί με τους μεταγλωττίσιμους και μη πόρους της εφαρμογής.
7. Το Lint, από την έκδοση ADT 16, το οποίο ψάχνει για πιθανά σφάλματα, όπως πόροι που δεν έχουν χρησιμοποιηθεί ή παρουσιάζουν λάθη, για παράδειγμα strings τα οποία ορίζονται μόνο στον πηγαίο κώδικα, λάθη στο αρχείο AndroidManifest.xml και άλλα.
8. Το Logcat, που συλλέγει και απεικονίζει τα γεγονότα που συμβαίνουν όταν εκτελείται η εφαρμογή κατά την αποσφαλμάτωσή της. Περιέχεται στο DDMS και δίνει πληροφορίες σχετικά με λάθη, ειδοποιήσεις και άλλα απαραίτητα για την καλύτερη εκτίμηση της απόδοσης της εφαρμογής.

9. Το Monkey, το οποίο είναι ένα πρόγραμμα που τρέχει στην εικονική ή φυσική συσκευή και παράγει ψευδο-τυχαίες ακολουθίες από γεγονότα (events) που δημιουργεί ο χρήστης, όπως clicks, touches, gestures, αλλά και κάποια συμβάντα συστήματος. Επιτρέπει τη δοκιμή της εφαρμογής από το δημιουργό σε ένα σχετικά ασφαλές περιβάλλον, αφού ο δημιουργός μπορεί να θέσει κάποιους περιορισμούς, όπως για παράδειγμα τον αριθμό ή τη συχνότητα των γεγονότων και επιστρέφει πληροφορίες εάν αντιμετωπίσει κάποια σφάλματα ή αν τα όρια που έχουν οριστεί ξεπεραστούν.
10. Το Traceview, που επιτρέπει τη γραφική προβολή των execution logs της εφαρμογής που εξάγονται από τη χρήση της κλάσης Debug. Βοηθά στην αποσφαλμάτωση και εκτίμηση της απόδοσης της εφαρμογής.
11. Τον AVD Manager, ο οποίος επιτρέπει τη δημιουργία και διαχείριση των εικονικών συσκευών.
12. Τον SDK Manager, που αναλαμβάνει το "κατέβασμα" εργαλείων, καινούριων εκδόσεων Android, και άλλων συστατικών δημιουργώντας πακέτα και διευκολύνοντας τον τελικό χρήστη να έχει μία ολοκληρωμένη άποψη των ήδη εγκαταστημένων, νέων και μη εγκατεστημένων εργαλείων / εκδόσεων. Αποτελεί μέρος του ADT και μπορεί να εκτελεστεί μέσω του Eclipse, αλλά και από τη γραμμή εντολών ή από το εκτελέσιμο αρχείο του.

Επίσης, για τους δημιουργούς υπάρχει το Android NDK, το οποίο είναι ένα σετ εργαλείων που επιτρέπει στους προγραμματιστές να αναπτύξουν κάποια τμήματα της εφαρμογής χρησιμοποιώντας τις γλώσσες C και C++. Καθώς οι εγγενείς βιβλιοθήκες του Android (native libraries) του Android είναι υλοποιημένες σε αυτές τις γλώσσες, μπορεί αυτό το σετ να είναι χρήσιμο, εφόσον επαναχρησιμοποιεί ήδη υπάρχοντα τμήματα κώδικα και διαθέσιμες βιβλιοθήκες, όμως δε συστήνεται η χρήση τους από τους προγραμματιστές, καθώς δεν είναι απαραίτητο ότι θα αυξήσει την απόδοση των εφαρμογών. Τουναντίον, μπορεί να μετατρέψει την υλοποίηση των εφαρμογών σε μία περίπλοκη διαδικασία. Γενικώς, θεωρείται καλή πρακτική να αποφεύγεται η χρήση του, εκτός εάν είναι απολύτως απαραίτητο για την εφαρμογή και εάν δεν υπάρχουν αντίστοιχα APIs που να μπορούν να διαθέσουν την απαραίτητη λειτουργικότητα. [2] [4] [5]

2.4. Ασφάλεια

Το Android έχει κληρονομήσει αρκετά χαρακτηριστικά από το Linux, ένα εκ των οποίων είναι και η ασφάλεια που παρέχει. Κάθε εφαρμογή εγκαθίσταται με ένα μοναδικό ID χρήστη και κάθε αρχείο που προέρχεται από την εκάστοτε εφαρμογή είναι ιδιωτικό, δηλαδή δεν μπορούν άλλες εφαρμογές να έχουν πρόσβαση σε αυτό. Επίσης, κάθε εφαρμογή κατά την εκκίνησή της ξεκινά δική της διεργασία. Έτσι, όπως συμβαίνει στα συστήματα Linux, κάθε μία εφαρμογή είναι απομονωμένη από τις υπόλοιπες. Εάν είναι απαραίτητη η διαμοίραση δεδομένων, η εφαρμογή πρέπει να το ζητήσει την άδεια του χρήστη κατά την εγκατάστασή της και να διαμοιραστεί δεδομένα χρησιμοποιώντας συγκεκριμένα συστατικά του Android τα οποία χειρίζονται τη διαμοίραση δεδομένων, όπως οι Υπηρεσίες (Services) και οι Πάροχοι Περιεχομένου (Content Providers), τα οποία θα αναπτυχθούν παρακάτω.

Ένα ακόμη χαρακτηριστικό του Android που συμβαδίζει με την ιδέα της παρεχόμενης ασφάλειας είναι οι Άδειες (ή Δικαιώματα, Permissions). Κάθε εφαρμογή κατά την εγκατάστασή της μπορεί να ζητήσει να δοθούν ή να ορίσει άδειες, όπως την πρόσβαση στις Επαφές ή στο Internet και ο χρήστης επιλέγει εάν θα τις δώσει. Εάν επιλέξει να μην δώσει κάποια άδεια, η εφαρμογή δεν θα εγκατασταθεί. Άρα, μία εφαρμογή που έχει ήδη εγκατασταθεί, έχει ήδη τις άδειες που πιθανόν έχει ζητήσει και οι οποίες δεν μπορούν να αλλάξουν, να αφαιρεθούν ή να αποδοθούν άλλες στη θέση τους. Αυτές οι άδειες δηλώνονται στο αρχείο AndroidManifest.xml, ενώ επιπρόσθετα μπορούν να οριστούν permissions που δεν επιτρέπουν την πρόσβαση σε στοιχεία της εφαρμογής. Αναλυτικά οι άδειες περιγράφονται παρακάτω στο κείμενο.

Όλα τα APKs, δηλαδή τα αρχεία .apk των εφαρμογών, υπογράφονται χρησιμοποιώντας ένα ιδιωτικό "κλειδί", το οποίο έχει ο δημιουργός και το οποίο τον πιστοποιεί και δίνει τη δυνατότητα στις εφαρμογές να χρησιμοποιούν το ίδιο User Id, εφόσον έχει οριστεί στο αρχείο AndroidManifest.xml των εφαρμογών. Τότε και μόνο οι δύο εφαρμογές έχουν το ίδιο User ID και τις ίδιες άδειες.

Κάθε εφαρμογή που εκκινείται σχετίζεται με ένα διαφορετικό λογαριασμό χρήστη, ασχέτως αν η εκκίνηση γίνεται από τον ίδιο χρήστη κάθε φορά, και άρα κάθε μία έχει ένα δικό της σύνολο από άδειες και περιορισμό σε ποιους χώρους της μνήμης έχει πρόσβαση. Εάν μία δεύτερη εφαρμογή εκκινηθεί, για παράδειγμα ένας web browser, τότε σχετίζεται με ένα διαφορετικό λογαριασμό χρήστη. Έστω ότι η δεύτερη εφαρμογή θέλει να έχει πρόσβαση σε δεδομένα της πρώτης, το οποίο σε κάποιο άλλο λειτουργικό είναι εφικτό. Στη συγκεκριμένη

περίπτωση όμως είναι ανέφικτο, εκτός εάν ορίσουμε άδειες τις οποίες ο χρήστης θα αποδεχθεί ή με κάποιους άλλους τρόπους, όπως αποθήκευση των δεδομένων στην SD card, έτσι ώστε να υπάρχει πρόσβαση από όλες τις εφαρμογές. Όμως, κάτι τέτοιο αυτόματα συνεπάγεται και τη μείωση της ασφάλειας που επιθυμεί το σύστημα, καθώς έχει συσχετίσει κάθε εφαρμογή με ξεχωριστό λογαριασμό και έχει τεμαχίσει τον αποθηκευτικό χώρο σε τμήματα που σχετίζονται με τους διαφορετικούς λογαριασμούς, ακριβώς για την εξασφάλιση της ασφάλειας.

Φυσικά, η ασφάλεια της συσκευής και του συστήματος εφιστά την προσοχή του χρήστη, ο οποίος είναι υπεύθυνος για την εγκατάσταση των εφαρμογών. Συστήνεται εφαρμογές που δεν διατίθενται μέσω του Google Play ή δεν διαθέτονται επαρκείς πληροφορίες, να μην εγκαθίστανται και προτού την εγκατάσταση των εφαρμογών να δίδεται προσοχή σε τυχόν άδειες που υπάρχουν. Εάν δεν τις εγκρίνουμε, τότε να μην επιτρέπουμε την εγκατάσταση. Φυσικά, και από τη μεριά του ο δημιουργός μίας εφαρμογής πρέπει να είναι πολύ προσεκτικός σχετικά με τις άδειες που θα ζητήσει. [4] [18]

ΚΕΦΑΛΑΙΟ 3

3.1. Εγκατάσταση του Android SDK και του περιβάλλοντος Eclipse IDE σε λειτουργικό σύστημα Windows

Στο κεφάλαιο που ακολουθεί θα δοθούν οι απαιτήσεις συστήματος και ποιες τεχνολογίες πρέπει να υποστηρίζει ο υπολογιστής για να είναι σε θέση να εγκαταστήσει τα εργαλεία του Android. [2]

3.1.1. Απαιτήσεις συστήματος

Για την ανάπτυξη εφαρμογών κατάλληλων για την πλατφόρμα Android είναι απαραίτητη η εγκατάσταση κάποιων εργαλείων σε συσκευές οι οποίες μπορούν να υποστηρίξουν αυτό το έργο. Έτσι, ο σταθερός ή φορητός ηλεκτρονικός υπολογιστής πρέπει να έχει εγκαταστημένο κάποιο από τα ακόλουθα λειτουργικά συστήματα: Microsoft Windows έκδοση XP ή αργότερη, Mac OSX 10.5.8 ή αργότερη με απαραίτητα εγκατεστημένη κεντρική μονάδα επεξεργασίας βασισμένη στα πρότυπα της Intel, και πολλές διανομές Linux.

Απαραίτητη επίσης, είναι η εγκατάσταση των εργαλείων jdk έκδοση 6 ή νεότερη. Είναι καλύτερα να εγκατασταθεί η έκδοση 6, καθώς αυτή κυρίως υποστηρίζεται από το Android.

Πολλοί δημιουργοί εφαρμογών Android χρησιμοποιούν κάποιο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), όπως είναι το Eclipse, καθώς οι σουίτες αυτές βοηθούν αρκετά λόγω του ότι περιλαμβάνουν επεξεργαστή πηγαίου κώδικα, αποσφαλματωτή (debugger) και άλλα εργαλεία απαραίτητα για τις υπό ανάπτυξη εφαρμογές σε ένα ενιαίο περιβάλλον. Η Google έχει ανακοινώσει ότι μελλοντικά η προτιμώμενη σουίτα για την ανάπτυξη εφαρμογών Android θα είναι το Android Studio, που είναι ακόμα στη φάση δοκιμών.

Το ADT Bundle περιέχει το περιβάλλον ανάπτυξης Eclipse και το πρόσθετο ADT που αφορά την ανάπτυξη εφαρμογών κατάλληλων για Android συστήματα. Περιέχει τις τελευταίες βιβλιοθήκες, εργαλεία ανάπτυξης, αποσφαλμάτωσης, δοκιμής και τεκμηρίωση.

Η πλατφόρμα Android αποτελείται επίσης από εργαλεία τα οποία μπορούν να τρέξουν τις εφαρμογές. Όταν ολοκληρωθεί η φάση της συγγραφής κώδικα και δημιουργηθεί το πακέτο .apk, πρέπει να τρέξουμε την εφαρμογή αυτή. Οι επιλογές είναι οι εξής: είτε σε ένα φυσικό τερματικό, δηλαδή σε μία κινητή συσκευή, όπου με τις κατάλληλες επιλογές μπορούμε να τεστάρουμε το τελικό προϊόν ή να το τρέξουμε σε εικονικές συσκευές, οι οποίες ονομάζονται emulators. Στην περίπτωση της κινητής συσκευής, συνδέουμε με καλώδιο το τερματικό με τον υπολογιστή, πηγαίνουμε στις Ρυθμίσεις της συσκευής και επιλέγουμε USB Debugging,

έτσι ώστε να μας επιτραπεί να τεστάρουμε την εφαρμογή και να γίνει η αποσφαλμάτωση (debugging). Και στις δύο περιπτώσεις υπάρχουν θετικά και αρνητικά. Εάν επιλέξουμε τη λύση της εικονικής συσκευής, η οποία φυσικά θα είναι φθηνότερη, καθώς έχουμε τη δυνατότητα να έχουμε όχι μία συσκευή αλλά δεκάδες, αλλά και να τη διαμορφώσουμε όπως επιθυμούμε αλλάζοντας τα χαρακτηριστικά της, για παράδειγμα το μέγεθος της οθόνης ή της εξωτερικής μνήμης (SD card), θα υπάρχουν τα μειονεκτήματα ότι η εικονική συσκευή είναι αρκετά αργή, δεσμεύοντας μάλιστα μεγάλους πόρους του υπολογιστή, κάποια χαρακτηριστικά, όπως η σύνδεση μέσω USB ή η χρήση Bluetooth δεν είναι δυνατή, και φυσικά καθώς δεν είναι φυσικό τερματικό, είναι πιθανόν να μην εμφανίζεται εντελώς σωστά η εφαρμογή.

Περιλαμβάνει επίσης, εργαλεία για τη διαχείριση των emulators, όπως το Telnet και το DDMS (Dalvik Debug Monitor Service), όπου μπορούμε να εισάγουμε διαφορετικά χαρακτηριστικά συσκευών ή χρηστών, για παράδειγμα ταχύτητα δικτύου, πώς λειτουργεί η συσκευή, εάν φορτίζεται ή εάν έχει χαμηλή στάθμη μπαταρίας, γεωγραφικές συντεταγμένες, μπορεί να στέλνει μηνύματα στην εικονική συσκευή, να επιτρέπει την επικοινωνία μεταξύ πολλαπλών εικονικών τερματικών και άλλα. Περιλαμβάνει, επίσης, το εργαλείο Android Debug Bridge (adb), το οποίο επιτρέπει τη σύνδεση σε μία εικονική ή πραγματική συσκευή με σκοπό τη διαχείριση και την αποσφαλμάτωση της εκάστοτε εφαρμογής.

Το αρχείο .apk (Android application package file) είναι η μορφή του αρχείου που χρησιμοποιείται για τη διανομή και εγκατάσταση εφαρμογών και ενδιάμεσου λογισμικού στην πλατφόρμα του Android, αλλά και σε άλλα συστήματα όπως στο Blackberry 10 με λειτουργικό σύστημα έκδοση 10.2.1, το οποίο, με κάποιες μετατροπές, είναι σε θέση να εκτελεί εφαρμογές Android. Το αρχείο αυτό είναι αντίστοιχο με τα πακέτα λογισμικού MSI για Windows και Deb για λειτουργικά όπως τα Linux Ubuntu. Ο τρόπος δημιουργίας ενός .apk έχει ήδη αναφερθεί στα πλαίσια της παρούσας διπλωματικής. Εν συντομία, όλα τα αρχεία που περιέχει ένα Android project μεταγλωττίζονται και δημιουργείται ένα πακέτο, το αρχείο apk, το οποίο περιέχει όλο των κώδικα, τους πόρους, το αρχείο AndroidManifest.xml, τις άδειες και τα πιστοποιητικά. Είναι ένα συμπιεσμένο αρχείο που βασίζεται στον τύπο αρχείων JAR.

Σε κάθε περίπτωση πρέπει να γίνει εξαγωγή της εφαρμογής έτσι ώστε να μπορεί να χρησιμοποιηθεί από άλλους χρήστες. Αφού γίνουν οι απαραίτητες δοκιμές, γίνεται μέσω του Eclipse, επιλέγοντας την εφαρμογή και πατώντας Εξαγωγή. Πρωτίστως, όμως, πρέπει να

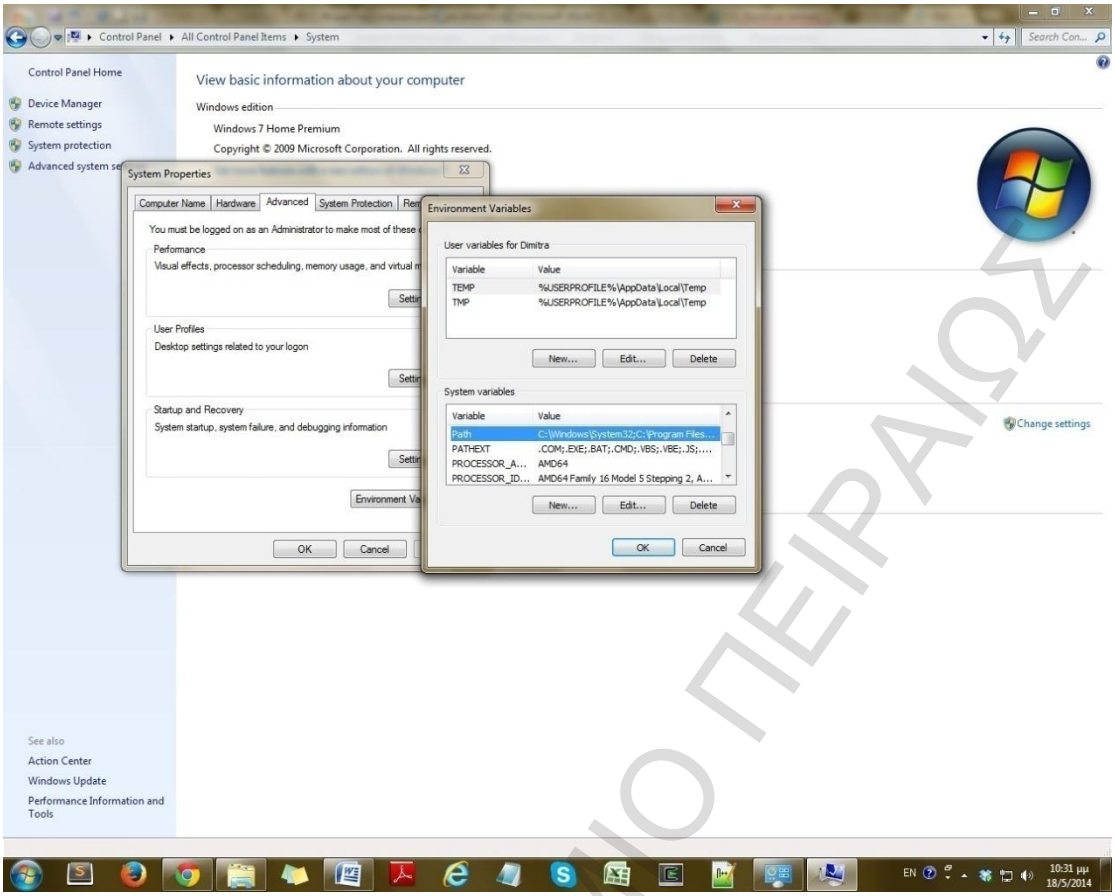
υπάρχει ψηφιακή υπογραφή, η οποία διατίθεται στο δημιουργό της εφαρμογής. Όλες οι εφαρμογές που διατίθενται εμπορικά πρέπει να έχουν αυτό το ψηφιακό πιστοποιητικό, το οποίο ελέγχεται αν είναι σε ενέργεια όταν εγκαθίσταται η εφαρμογή στον τελικό χρήστη. Υπάρχουν εργαλεία, όπως το Keytool και το Jarsigner, για τη δημιουργία αυτών των κλειδιών, το οποίο είναι διαφορετικό από το κλειδί που δημιουργείται για την εφαρμογή κατά τη διαδικασία της αποσφαλμάτωσης. Επίσης, αν και δεν είναι απαραίτητο, προτείνεται στους προγραμματιστές εφαρμογών να χρησιμοποιούν μία Άδεια Χρήσης Τελικού Χρήστη (End-User License Agreement, EULA), που ορίζει τους ίδιους ως δημιουργούς, όπως επίσης κατοχυρώνει το δικαίωμα χρήσης του τελικού χρήστη της εφαρμογής.

3.2. Οδηγίες εγκατάστασης Eclipse

Για την ανάπτυξη εφαρμογών σε Android, είναι απαραίτητη η εγκατάσταση του αντίστοιχου λογισμικού. Καλό είναι πριν από την έναρξη της εγκατάστασης, να υπάρχει σε κάποιο τοπικό αποθηκευτικό μέσο το απαραίτητο λογισμικό, το οποίο διατίθεται δωρεάν. Στην περίπτωση της συγκεκριμένης εργασίας, η εγκατάσταση έγινε σε Windows 7/ 8 και Linux Ubuntu 14.04 LTS και χρησιμοποιήθηκαν:

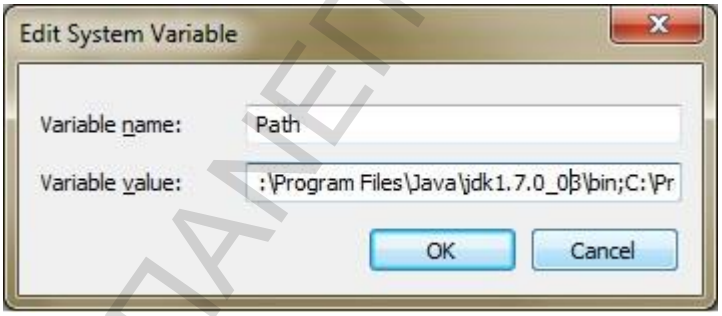
i) jdk1.7.0 Update 13. Το jdk είναι τα αρχικά του Java Developer Kit και είναι το περιβάλλον μέσα στο οποίο μπορεί να τρέξει ένα java πρόγραμμα.
ii) Eclipse IDE έκδοση Juno. Εργαλείο για προγραμματισμό σε Java. Πιο απλά, το περιβάλλον μέσα στο οποίο αναπτύσσονται τα προγράμματα. Το αρχείο είναι σε συμπιεσμένη μορφή (.zip) και πρέπει να αποσυμπιεστεί.
iii) Android SDK (Software Development Kit). Αντίστοιχο με το i), με τη διαφορά ότι είναι για το Android.

Αφού κατεβάσουμε από τον ιστότοπο http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html την κατάλληλη έκδοση σύμφωνα με το λειτουργικό μας σύστημα, ξεκινάμε την εγκατάσταση του JDK ακολουθώντας τα βήματα του οδηγού. Μόλις ολοκληρωθεί, πηγαίνουμε στις μεταβλητές συστήματος, οι οποίες βρίσκονται για το λειτουργικό σύστημα Windows 7 πηγαίνοντας στον Πίνακα Ελέγχου > Σύστημα > Ρυθμίσεις συστήματος για προχωρημένους > επιλέγοντας στο παράθυρο διαλόγου την καρτέλα Για προχωρημένους > Μεταβλητές περιβάλλοντος > Μεταβλητές συστήματος > Path.



Εικόνα 6. Εύρεση στον υπολογιστή των μεταβλητών του συστήματος.

Προσθέτουμε τη διαδρομή για το jdk και συγκεκριμένα για τον υποφάκελο /bin. Για παράδειγμα, *C:\Program Files\Java\jdk1.7.0_13\bin;* . Καλό είναι να μη διαγράψουμε τίποτα από το Path, αλλά να προστεθεί ένα ";" και στη συνέχεια να εισάγουμε τη διαδρομή για το jdk. Επίσης, στο τέλος προσθέτουμε άλλο ένα ";".



Εικόνα 7. Εισαγωγή της διαδρομής του jdk στις μεταβλητές.

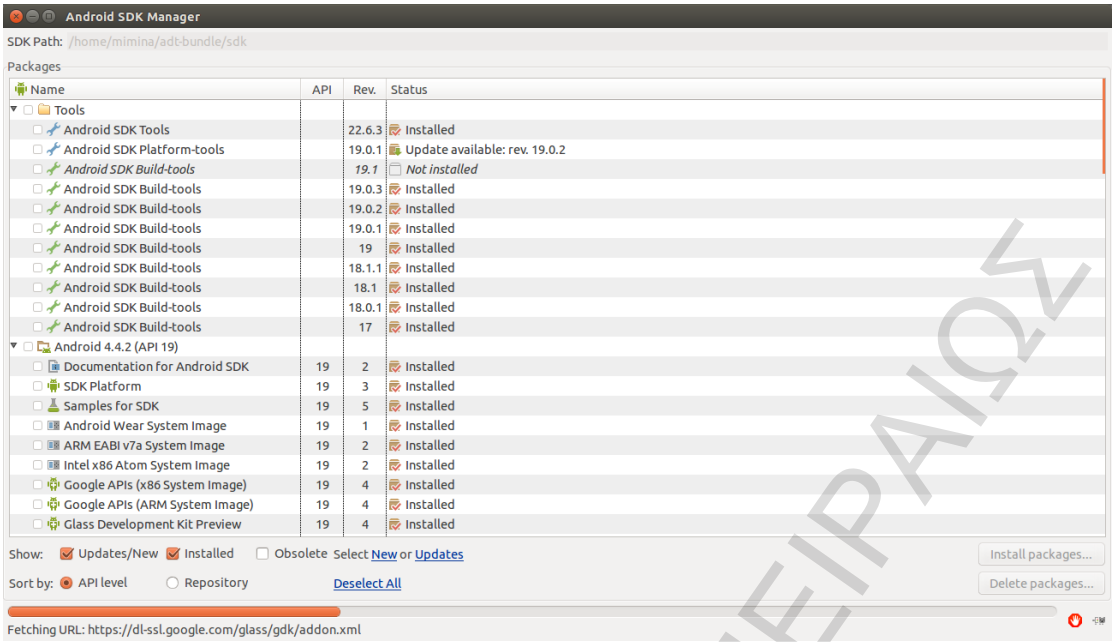
Στη συνέχεια, εφόσον έχουμε κατεβάσει το Eclipse από την τοποθεσία http://www.eclipse.org/downloads/, το αποσυμπιέζουμε.

Εναλλακτικά, μπορούμε να κατεβάσουμε από την τοποθεσία *http://developer.android.com/sdk/index.html?utm_source=weibolife* το Android SDK, το οποίο διαθέτει στο συμπιεσμένο αρχείο και ένα φάκελο με το όνομα Eclipse, ο οποίος περιέχει το Eclipse IDE. Αποσυμπιέζουμε όλο το φάκελο.

Ο φάκελος που αποθηκεύσαμε τοπικά δεν έχει κάποιο setup αρχείο για να το εγκαταστήσουμε. Αντί αυτού, το Eclipse είναι ένα εκτελέσιμο αρχείο (.exe), το οποίο απλώς το τρέχουμε από το φάκελο στον οποίο το αποσυμπιέσαμε. Χρειάζεται, όμως, να οριστεί ένας "χώρος εργασίας" (workspace), ένας φάκελος δηλαδή στον οποίο θα σώζονται τα αρχεία που δημιουργούνται στο Eclipse. Εάν απαιτούνται εξωτερικά αρχεία, όπως αρχεία εικόνας, για κάποια εφαρμογή, πρέπει να εισαχθούν στους φακέλους του project, το οποίο είναι αποθηκευμένο στο workspace.

Το επόμενο βήμα είναι η εγκατάσταση του Android SDK. Απαραίτητο και σε αυτό το Developer Kit είναι να επαναπροσδιορίσουμε τη μεταβλητή συστήματος Path. Θα ακολουθηθεί ο ίδιος τρόπος όπως και για το jdk, με τη διαφορά ότι εδώ ο φάκελος του sdk που πρέπει να προσθέσουμε είναι το tools. Άρα, θα προστεθεί στην Path κάτι παρόμοιο με *C:\adt-bundle-windows-x86_64\sdk\tools,* δηλαδή η διαδρομή του φακέλου που μόλις αποσυμπιέσαμε.

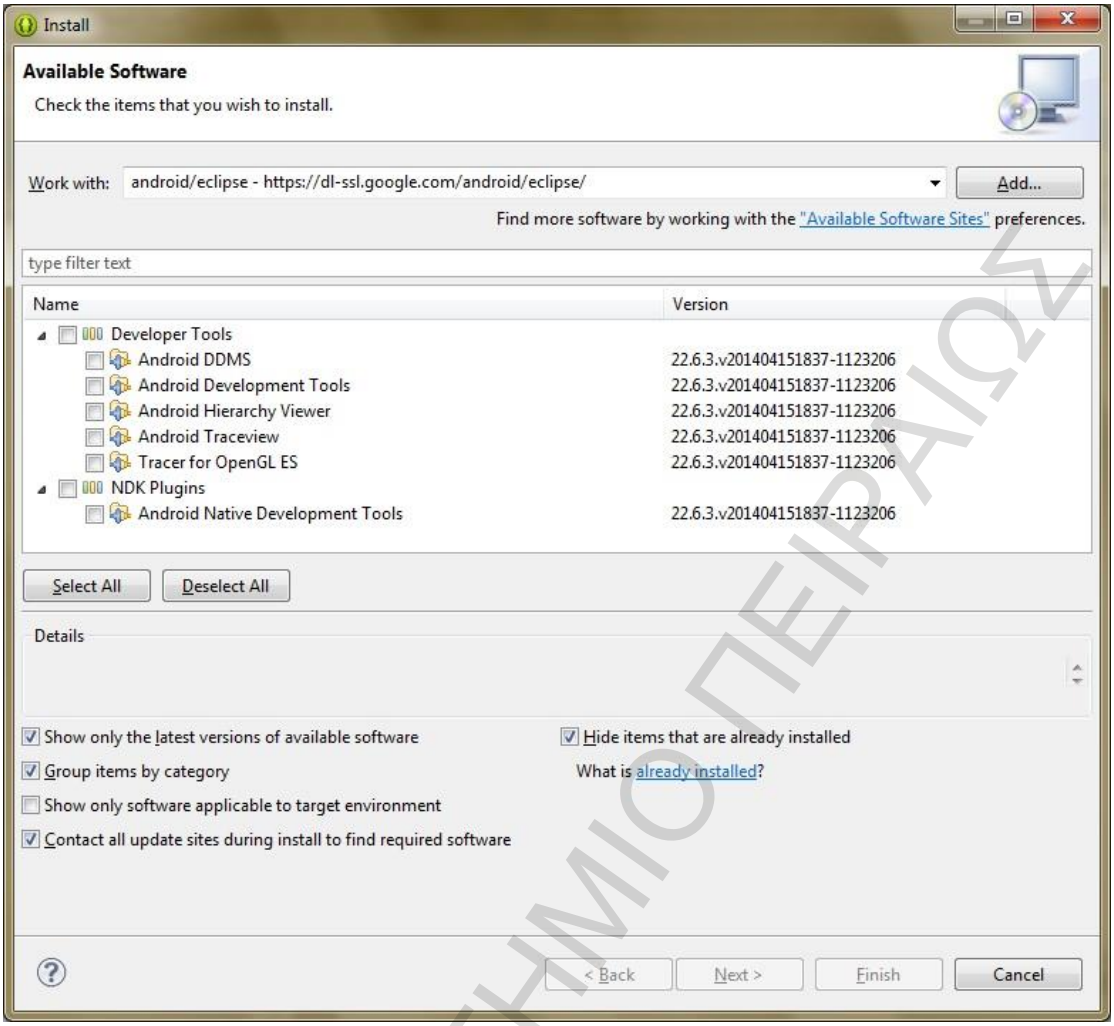
Στη συνέχεια πρέπει να τρέξουμε το Android SDK Manager. Είναι απαραίτητο, καθώς μας επιτρέπει να εγκαταστήσουμε τις διάφορες εκδόσεις του Android API. Από το περιβάλλον του Eclipse IDE επιλέγουμε Window > Android SDK Manager. Εναλλακτικά, τρέχουμε το αρχείο SDK Manager, το οποίο βρίσκεται στη θέση \path\to\Android\android-sdk.



Εικόνα 8. Android SDK Manager.

Είναι καλό, αλλά όχι απαραίτητο να εγκαταστήσουμε όλα τα πακέτα, καθώς πιθανόν να χρειαστεί να δοκιμάσουμε τις εφαρμογές που αναπτύσσουμε σε διαφορετικές εκδόσεις του Android. Επιλέγουμε Install packages... και αποδοχή των licences. Μόλις εγκατασταθούν τα πακέτα που επιλέξαμε, κάνουμε επανεκκίνηση του Eclipse. Ασχέτως αν επιλέξουμε κάποιο ή όλα τα πακέτα, πρέπει να εγκαταστήσουμε το Android Support Library από το SDK Manager. Παρέχει τη λειτουργικότητα που υπάρχει στις τελευταίες εκδόσεις του Android στις προηγούμενες εκδόσεις του.

Επόμενο βήμα είναι να ρυθμίσουμε το Eclipse έτσι ώστε να μπορούμε να αναπτύσσουμε εφαρμογές για την πλατφόρμα Android. Από το περιβάλλον του Eclipse επιλέγουμε Help > Install New Software... και εισάγουμε τη διεύθυνση *https://dl-ssl.google.com/android/eclipse*. Επιλέγουμε τα Developer Tools και NDK Plugins και ακολουθώντας τα βήματα του οδηγού, ολοκληρώνουμε τη διαδικασία.



Εικόνα 9. Ρύθμιση *https://sl-ssl.google.com/android/eclipse*.

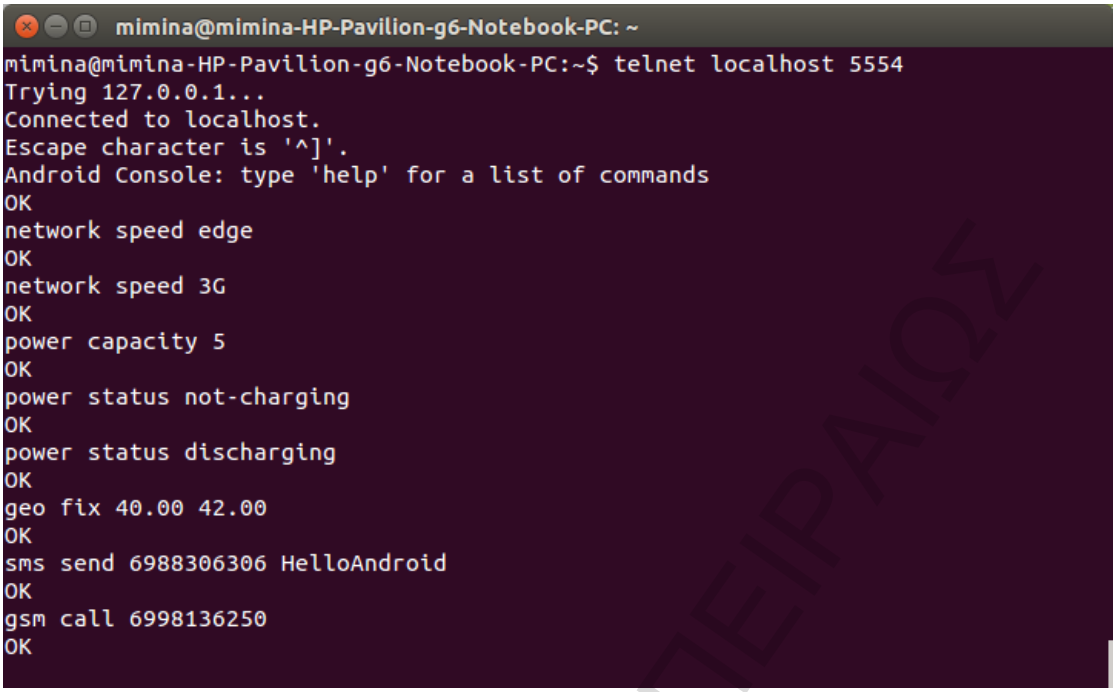
Κατόπιν, πηγαίνοντας από το Eclipse > Window > Preferences στις επιλογές που αφορούν το Android, ελέγχουμε εάν υπάρχει η διαδρομή για το sdk. Εάν όχι, πρέπει να ενημερώσουμε το πεδίο SDK Location. Η διαδρομή έχει τη μορφή /path/to/sdk. Εάν έχουμε κατεβάσει το αρχείο ADT Bundle, κανονικά θα υπάρχει ήδη αυτή η παράμετρος.

3.2.1. Εξομοιωτής του Android

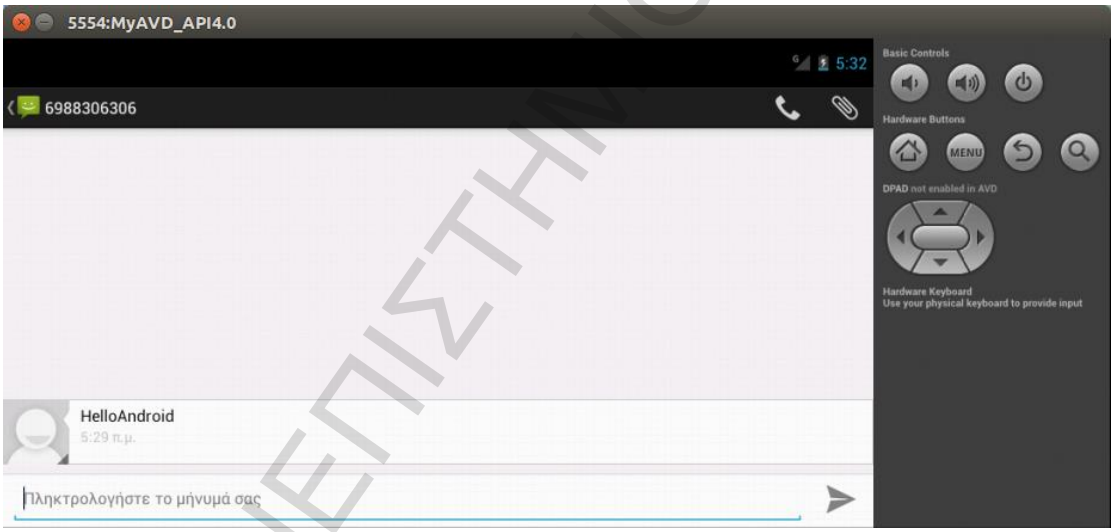
Τέλος, μπορούμε να δημιουργήσουμε τις Android Virtual Devices (AVDs) που θέλουμε είτε μέσα από το Eclipse, είτε τρέχοντας το αρχείο AVD Manager, το οποίο είναι μέρος του ADT Bundle. Είναι ο εξομοιωτής (emulator), ο οποίος προσομοιώνει μία πραγματική συσκευή Android. Οι εικονικές συσκευές επιτρέπουν την εύκολη και ανέξοδη δοκιμή και αποσφαλμάτωση (debugging) των εφαρμογών σε διαφορετικές εκδόσεις Android API και σε

διαφορετικές συσκευές με διαφορετικά στοιχεία, όπως μέγεθος οθόνης και μνήμη. Υπάρχουν κάποια αρνητικά σημεία στη χρήση emulators για τη δοκιμή των εφαρμογών. Πρώτον, καθώς ο εξομοιωτής δεν είναι φυσική συσκευή, υπάρχει μικρή πιθανότητα μετά από μία δοκιμή σε έναν εξομοιωτή, η εφαρμογή να μην εμφανίζεται ή να λειτουργεί όπως περιμένουμε στην αντίστοιχη πραγματική συσκευή. Δεύτερον, η χρήση των εικονικών συσκευών ρίχνει την απόδοση του υπολογιστή, καθώς απαιτείται η χρήση πολλών πόρων του συστήματος για την εκτέλεσή του. Συνεπακόλουθα, επίσης μία εικονική συσκευή είναι αργή και συνήθως πρέπει να περιμένουμε αρκετά μέχρι να εκτελέσει τις εντολές που δίνουμε. Τέλος, κάποια χαρακτηριστικά δεν υποστηρίζονται ή δεν είναι διαθέσιμα από τον εξομοιωτή, όπως για παράδειγμα, η χρήση Bluetooth, η σύνδεση με χρήση USB καλωδίου, οπότε κάποιες εφαρμογές δεν εκτελούνται σε αυτό το περιβάλλον. Όμως, η χρήση εικονικών συσκευών για τη δοκιμή εφαρμογών επιτρέπει την εκτέλεσή τους σε διαφορετικά περιβάλλοντα, όπως εκδόσεις Android, αλλά και διαφορές στο υλικό των συσκευών, όπως μνήμη, χώρος αποθήκευσης, ανάλυση οθόνης και πυκνότητα pixels της οθόνης της συσκευής (screen density, συνήθως αναφέρεται ως dots per inch, dpi). Επίσης, δε χρειάζεται να γίνει η αγορά καμίας συσκευής, οπότε η λύση αυτή προτείνεται ως φθηνότερη. Κάθε μετατροπή που θα γίνει, θα αφορά μόνο την εικονική συσκευή, άρα δε θα υπάρχει περίπτωση να υπάρξει κάποιο πρόβλημα στη συσκευή ή στα δεδομένα της ή στη διαμόρφωσή της.

Κατά τη διαμόρφωση της εικονικής συσκευής, δίνεται η δυνατότητα να προσομοιωθεί η ταχύτητα και άλλα χαρακτηριστικά ενός δικτύου. Μπορούμε να διαμορφώσουμε τη συσκευή έτσι ώστε να προσομοιώνει διαφορετικά επίπεδα μπαταρίας, όπως να φαίνεται ότι η συσκευή φορτίζεται ή ότι δεν έχει αρκετή μπαταρία. Μπορεί να γίνει εισαγωγή γεωγραφικών δεδομένων (γεωγραφικές συντεταγμένες). Επίσης, ο προσομοιωτής δίνει τη δυνατότητα μίμησης αλληλεπίδρασης του χρήστη με το δίκτυο, όπως να λάβει μία τηλεφωνική κλήση ή ένα μήνυμα SMS.



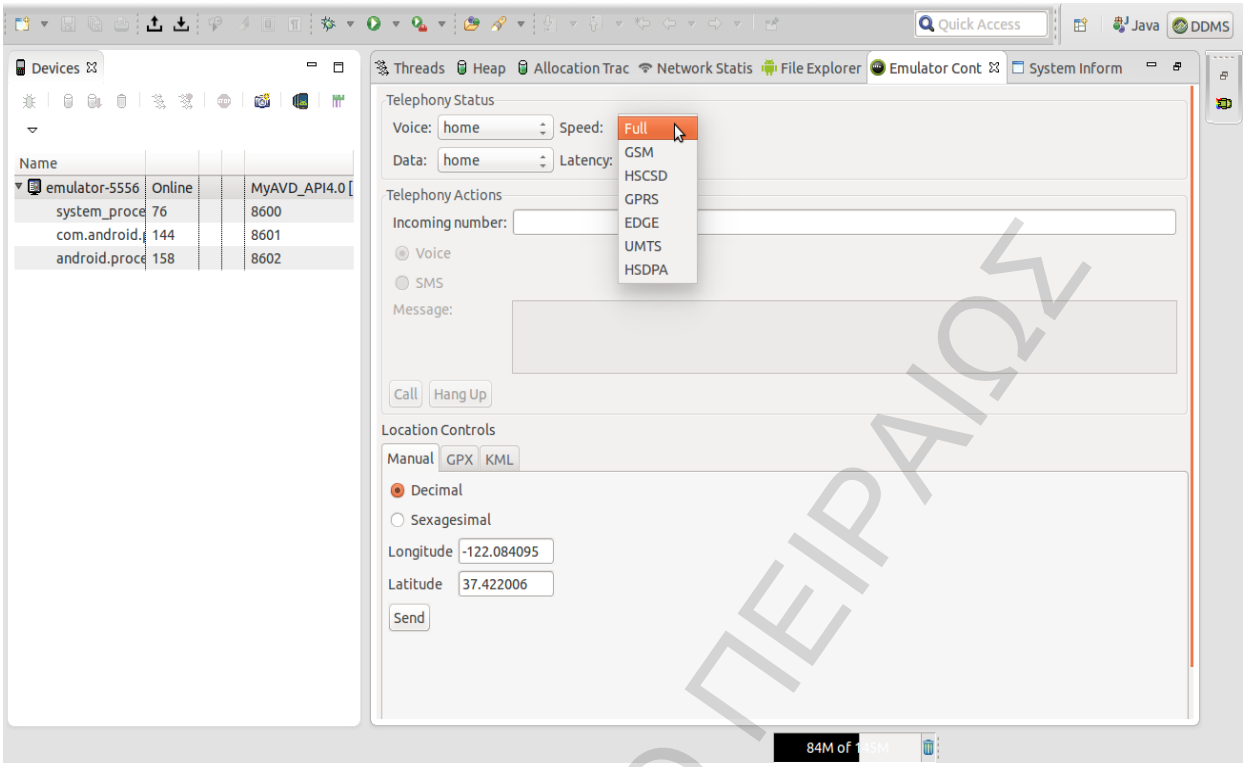
Εικόνα 10. Σύνδεση με την εικονική συσκευή μέσω telnet.



Εικόνα 11. Εκτέλεση της εντολής για αποστολή μηνύματος SMS.

Πιο πάνω οι εικόνες είναι από το τερματικό παράθυρο και από την εικονική συσκευή με την οποία αλληλεπιδρά πληκτρολογώντας κάποιες εντολές, όπως έχουν περιγραφεί πιο πάνω. Πληκτρολογώντας την εντολή `telnet localhost` <αριθμός θύρας εξομοιωτή>, ξεκινά η επικοινωνία με το AVD.

Αντίστοιχα με τις εντολές σε ένα τερματικό παράθυρο με χρήση telnet, μπορεί να χρησιμοποιηθεί η "προοπτική" DDMS του Eclipse:

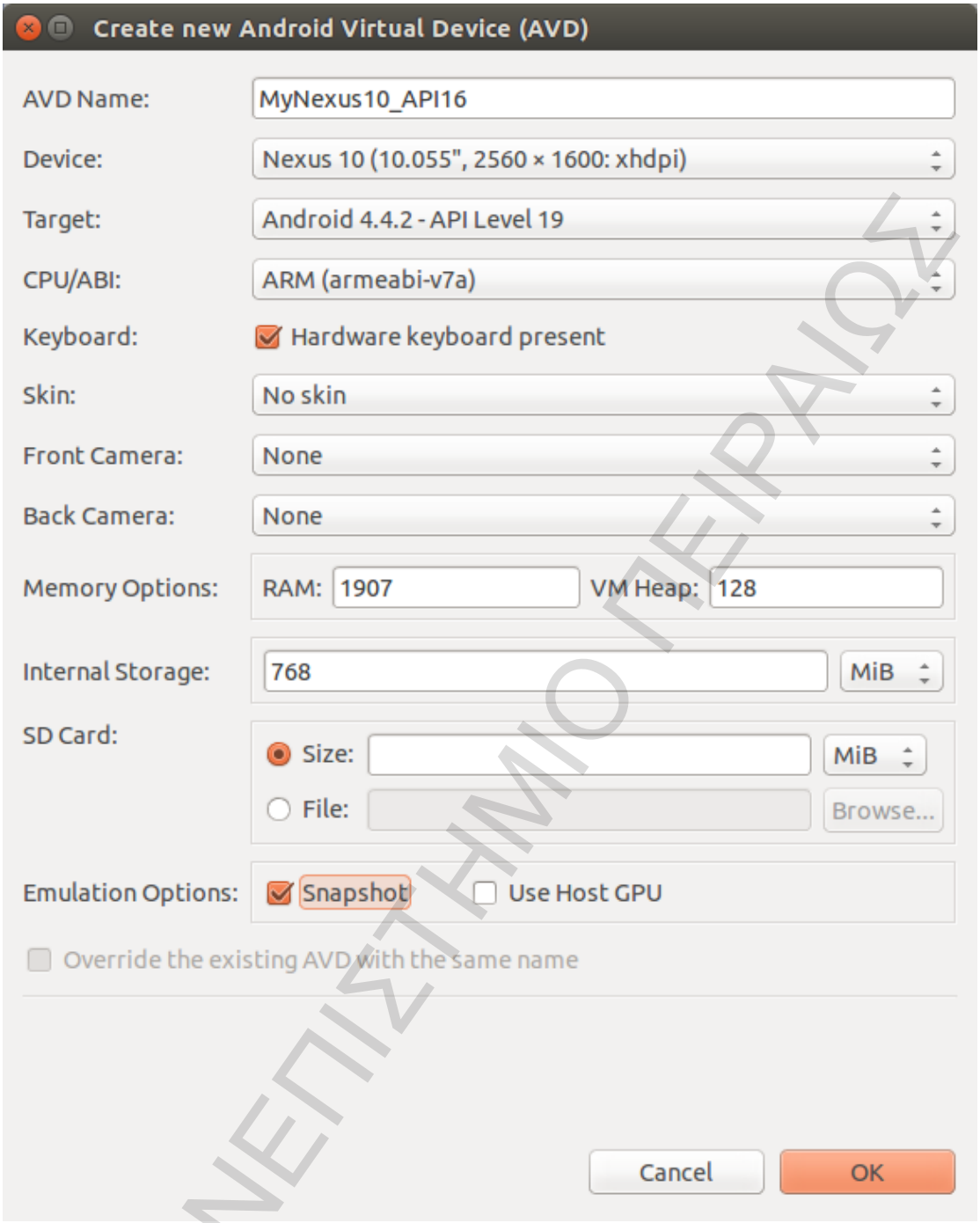


Εικόνα 12. DDMS. Παρέχει τη δυνατότητα παραμετροποίησης της εικονικής συσκευής.

Ακόμα, από εδώ μπορούμε να περιηγηθούμε στα αρχεία που υπάρχουν στην εικονική συσκευή, επιλέγοντας την καρτέλα File Explorer.

Επίσης, δύο εικονικές συσκευές μπορούν να αλληλεπιδράσουν μεταξύ τους, χρησιμοποιώντας τον αριθμό θύρας που εμφανίζεται στο πάνω άκρο της AVD.

Παρακάτω φαίνεται το παράθυρο που ανοίγει για τη δημιουργία εικονικών συσκευών. Πρώτα, επιλέγουμε ένα όνομα, τη συσκευή, την έκδοση επιπέδου API, τον επεξεργαστή, αν θα υπάρχει κάμερα μπροστά ή και πίσω, τη μνήμη RAM, το χώρο αποθήκευσης και την εξωτερική μνήμη. Εάν για τη χρήση CPU επιλεχθεί η αρχιτεκτονική Intel CPI, η εικονική μηχανή θα είναι γρηγορότερη κατά την εκτέλεση έναντι της αντίστοιχης που χρησιμοποιεί ARM CPU, αφού δεν χρειάζεται ο εξομοιωτής να μετατρέπει τις εντολές από την ARM σε κατανοητές από την αρχιτεκτονική Intel / AMD CPU του υπολογιστή.



Εικόνα 13. Πλαίσιο διαλόγου για ρυθμίσεις των virtual devices.

Σημειώνεται ότι για την επιλογή Snapshot, από τη δεύτερη φορά που θα ξεκινήσει η AVD θα εκκινήσει πολύ γρηγορότερα, επειδή θα έχει αποθηκεύσει την κατάσταση που ήταν όταν τερματίστηκε η εικονική συσκευή την τελευταία φορά, ενώ εάν επιλεγεί η Use Host GPU θα χρησιμοποιηθεί η κάρτα γραφικών του υπολογιστή, το οποίο θα έχει ως αποτέλεσμα την γρηγορότερη απεικόνιση (rendering) στην οθόνη της εικονικής συσκευής. Επίσης, ενώ μπορεί να επιλεγούν και τα δύο, εάν κάτι τέτοιο συμβεί ένα μήνυμα σφάλματος θα ενημερώνει ότι

δεν μπορούν να ενεργοποιηθούν και τα δύο ταυτόχρονα. Αφού ξεκινήσει να λειτουργεί η AVD, αλληλεπιδράμε με τη συσκευή είτε με χρήση των πλήκτρων που τυχόν υπάρχουν στην εικονική συσκευή, είτε και με το πληκτρολόγιο και το ποντίκι του υπολογιστή.

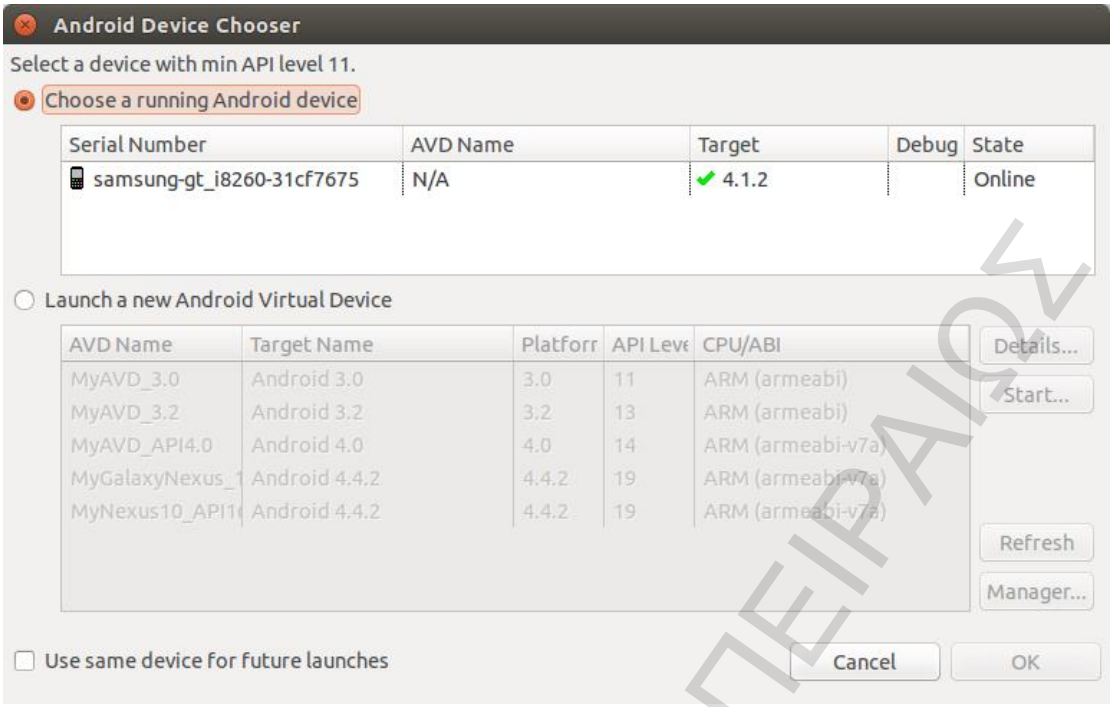
Μπορούμε να δημιουργήσουμε όσες εικονικές συσκευές θέλουμε. Όπως έχει ήδη ειπωθεί, οι εικονικές συσκευές είναι ένας εύκολος και ανέξοδος τρόπος να δοκιμάσουμε τις εφαρμογές σε όσο το δυνατόν περισσότερα περιβάλλοντα - εκδόσεις Android API. Καθώς είναι δυνατή η εκκίνηση πολλών AVDs παράλληλα, μπορούμε να δοκιμάζουμε την εφαρμογή ταυτόχρονα σε συσκευές με διαφορετική διαμόρφωση.

Τέλος, εκτός από τον προεπιλεγμένο εξομοιωτή που υπάρχει στο ADT Bundle, υπάρχουν διαθέσιμοι και άλλοι emulators.

3.2.2. Χρήση κινητής συσκευής κατά τη φάση δοκιμών μίας εφαρμογής

Εναλλακτικά, είναι δυνατή η χρήση μίας συσκευής που λειτουργεί με λειτουργικό σύστημα Android. Όπως έχει ήδη τονιστεί, η χρήση πραγματικών συσκευών έναντι εικονικών συσκευών έχει αρνητικά και θετικά στοιχεία, με πιο θετικό όλων ότι σε κάποιες περιπτώσεις ενώ η εφαρμογή εμφανίζεται να λειτουργεί κανονικά σε μία εικονική συσκευή, σε μία αναλογική με τα ίδια χαρακτηριστικά μπορεί να μη λειτουργεί κανονικά.

Για τη χρήση κατά τη διαδικασία των δοκιμών, συνδέουμε τη συσκευή με τον υπολογιστή μέσω ενός καλωδίου USB. Πιθανόν αναλόγως του λειτουργικού συστήματος του υπολογιστή να χρειαστεί εγκατάσταση των κατάλληλων οδηγών λογισμικού (σε λειτουργικό σύστημα Windows). Έπειτα, ρυθμίζεται η κινητή συσκευή έτσι ώστε να επιτρέπεται η αποσφαλμάτωση μέσω USB (USB debugging). Η επιλογή αυτή στις τελευταίες εκδόσεις (Android 4.0 και πιο πρόσφατες) εμφανίζεται στις Ρυθμίσεις > Επιλογές Προγραμματιστή, ενώ στις πλαιότερες στις Ρυθμίσεις > Εφαρμογές > Ανάπτυξη. Για να "τρέξει" η εφαρμογή, από το Eclipse, επιλέγουμε Run και εμφανίζεται ένα παράθυρο όπου μας ζητάει να επιλέξουμε τη συσκευή για να τρέξει η εφαρμογή. Επιλέγουμε το κινητό και η εφαρμογή εγκαθίσταται στην κινητή συσκευή, ενώ ταυτόχρονα έχουμε εικόνα του logcat.



Εικόνα 14. Επιλογή συσκευής για αποσφαλμάτωση.

Μπορεί, επίσης, να γίνει η διαδικασία αυτή από τη γραμμή εντολών, πηγαίνοντας στο φάκελο του project και εκτελώντας `ant debug` και έπειτα adb install bin/NameOfProject.apk. Είτε επιλέξουμε μία εικονική είτε μία φυσική συσκευή, μπορούμε να τρέξουμε την εφαρμογή κατά τη φάση δοκιμών, Όμως, για να μπορεί να είναι διαθέσιμη στο κοινό, πρέπει να κάνουμε export την εφαρμογή. Επιλέγουμε Export από το μενού του Eclipse. Κατόπιν, πρέπει να "υπογραφεί" η εφαρμογή δημιουργώντας ένα keystore. Είναι καλή πρακτική να χρησιμοποιείται το ίδιο keystore για όλες τις εφαρμογές που δημιουργούμε. Μόλις ολοκληρωθεί το export της εφαρμογής, είναι διαθέσιμη για εγκατάσταση σε οποιαδήποτε Android συσκευή.

3.3. Δομικά στοιχεία μίας εφαρμογής

Τέσσερα είναι τα βασικά συστατικά μίας εφαρμογής και απαρτίζονται από τις εξής κλάσεις:

1. Η κλάση Δραστηριότητα (Activity). Η κύρια κλάση. Παρέχει τη γραφική διεπιφάνεια χρήστη, δια μέσω της οποίας ο χρήστης λαμβάνει και αποστέλλει πληροφορίες στην εφαρμογή. Τα υπόλοιπα τρία συστατικά δεν παρέχουν διεπιφάνειες χρήστη, καθώς δουλεύουν "πίσω" από τη διεπιφάνεια. Κυρίως ασχολείται με την αλληλεπίδραση με το χρήστη, ενώ συνήθως υλοποιεί μία εστιασμένη διεργασία (task), με την οποία

αλληλεπιδρά ο χρήστης. Μία εφαρμογή μπορεί να έχει αρκετές Δραστηριότητες, εκ των οποίων μία θα είναι η κύρια. Για τη δημιουργία της διεπιφάνειας και την αλληλεπίδραση με το χρήστη χρησιμοποιούν Views, που παρουσιάζονται πιο κάτω.

2. Οι Υπηρεσίες (Services). Υποστηρίζουν μακρόχρονες εργασίες ή εργασίες που γίνονται πίσω από το προσκήνιο. Λόγω του ότι δρουν παρασκηνιακά και δεν αλληλεπιδρούν με το χρήση, δεν περιλαμβάνουν διεπιφάνεια χρήστη. Όμως, υποστηρίζουν ποικιλοτρόπως τις διάφορες απομακρυσμένες λειτουργίες. Χρησιμοποιούνται για εργασίες που περιλαμβάνουν την πρόσβαση σε απομακρυσμένους πόρους, όπως το κατέβασμα αρχείων από απομακρυσμένους servers, ή το συγχρονισμό δεδομένων που είναι αποθηκευμένα σε μία κινητή συσκευή με μία Βάση Δεδομένων που υπάρχει στο "σύννεφο" (Cloud), τη διάδραση με ένα Πάροχο Περιεχομένου ή ακόμα και να παρέχει ασφαλείς τρόπους διεκπεραίωσης συναλλαγών με websites ηλεκτρονικού εμπορίου. Συνήθως εκκινούνται από άλλα συστατικά, κυρίως Δραστηριότητες και συνεχίζουν να εκτελούνται, ακόμα και εάν ο χρήστης μεταφερθεί σε άλλη Δραστηριότητα ή εφαρμογή. Χωρίζονται σε Started Services και Bound Services. Εκτός του να ορίσουμε εμείς τις δικές μας Υπηρεσίες, υπάρχουν έτοιμες, όπως η Location Manager Service, που μπορούμε να χρησιμοποιήσουμε, αρκεί να έχουμε το κατάλληλο permission. Άλλα συστατικά μπορούν να συνδεθούν (bind) με μία Υπηρεσία και να αλληλεπιδράσουν μεταξύ τους χρησιμοποιώντας "επικοινωνία μεταξύ των διαδικασιών" (Inter-Process Communication, IPC) για την ασφαλή επικοινωνία τους, και η οποία αναλύεται σε επόμενη παράγραφο.

   Μπορεί να είναι είτε:

   - Started Service, όταν ένα συστατικό, π.χ. μία Δραστηριότητα, εκκινεί την Υπηρεσία καλώντας τη μέθοδο `startService()`. Εφόσον έχει εκκινηθεί, η Υπηρεσία μπορεί να "τρέχει" παρασκηνιακά επ' αόριστον, ακόμα και εάν το συστατικό που την εκκίνησε έχει καταστραφεί. Συνήθως εκτελεί μία και μόνη εργασία και δεν επιστρέφει κάποιο αποτέλεσμα στον καλούντα. Όταν εκτελεστεί η εργασία αυτή, τότε η Υπηρεσία τερματίζεται μόνη της. Παράδειγμα αυτής της κατηγορίας είναι το ανέβασμα και το κατέβασμα αρχείων μέσα ενός δικτύου (uploading και downloading).

- Bound Service, όπου το συστατικό συνδέεται με την Υπηρεσία καλώντας τη μέθοδο `bindService()`. Παρέχει διεπαφή πελάτη – εξυπηρετητή (client – server), όπου τα συστατικά που συνδέονται με την Bound Service αλληλεπιδρούν με αυτήν, στέλνουν αιτήματα και παίρνουν αποτελέσματα ακόμα και εάν είναι σε διαφορετικά διεργασία χρησιμοποιώντας IPC. Μία Υπηρεσία αυτού του τύπου "τρέχει" μόνο όσο ένα άλλο συστατικό είναι συνδεδεμένο με αυτήν, ενώ πολλαπλά συστατικά μπορούν να συνδεθούν ταυτόχρονα με την Υπηρεσία. Σε κάθε περίπτωση, η τελευταία θα τερματιστεί μόλις και το τελευταίο συστατικό αποσυνδεθεί από αυτήν.

3. Οι Δέκτες Εκπεμπόμενων Προθέσεων (Broadcast Receivers). Λαμβάνουν και αποκρίνονται σε γεγονότα τα οποία λαμβάνουν χώρα στη συσκευή. Μπορεί να χρησιμοποιήσει το Διαχειριστή Ειδοποιήσεων (Notification Manager) για να ειδοποιήσει το χρήστη.
4. Οι Πάροχοι Περιεχομένου (Content Providers). Είναι το στοιχείο το οποίο επιτρέπει σε εύρος εφαρμογές να αποθηκεύουν και να διαμοιράζονται δεδομένα. Έχουν διεπιφάνεια που μοιάζει με Βάση Δεδομένων, αλλά είναι περισσότερα από μία Βάση. Το Android παρέχει τη Βάση SQLite, η οποία συνεργάζεται πολλές φορές με τον Πάροχο Περιεχομένου, αποθηκεύοντας η πρώτη τα δεδομένα τα οποία θα προσπελαστούν με τη χρήση του Παρόχου. [2] [4] [5] [18]

3.4. Άλλα βασικά στοιχεία μίας εφαρμογής

Άλλα βασικά στοιχεία είναι:

- Τα αντικείμενα Πρόθεσης (Intent), τα οποία είναι βασικά ένα μήνυμα που ορίζει ότι θέλουμε κάτι να γίνει, μία περιγραφή της ενέργειας αυτής. Περιγράφουν την επιθυμητή ενέργεια (action), τα δεδομένα (data), την κατηγορία των δεδομένων (category) και άλλες εντολές. Το Android βρίσκει το συστατικό που ανταποκρίνεται στην πρόθεση, δημιουργεί ένα αντικείμενο αυτού και το διαθέτει στο αντικείμενο Πρόθεσης. Επιτρέπουν στα συστατικά της εφαρμογής να ζητήσουν από κάποιο άλλο συστατικό της ίδιας ή άλλης εφαρμογής να κάνει κάτι, όπως να ξεκινήσουν την εφαρμογή για την κάμερα. Εκκινούν Δραστηριότητες μέσω άλλων Δραστηριοτήτων. Μέσα στο σώμα του κώδικα, ορίζεται ποια ενέργεια πρέπει να εκτελεστεί και τα

δεδομένα που απαιτούνται για να εκτελεστεί η ενέργεια αυτή. Ένα συστατικό μπορεί να χρησιμοποιήσει ένα αντικείμενο Φίλτρου Προθέσεως (Intent Filter), το οποίο ορίζει σε ποιο Intent μία Activity, Service ή Broadcast Receiver μπορεί να αποκριθεί και ορίζει τι μπορεί να κάνει ή τι μπορεί να διαχειριστεί το συστατικό που το περιέχει. Μπορεί να οριστεί στατικά στο αρχείο AndroidManifest.xml ή δυναμικά στον πηγαίο κώδικα. Ένα αντικείμενο Intent είναι μία πρόθεση να γίνει κάτι, ενώ ένα αντικείμενο Intent Filter αποτελεί μία περιγραφή του τι είδους προθέσεις είναι δυνατόν να εξυπηρετηθούν. Ένα συστατικό θεωρείται ότι αντιστοιχεί στην Πρόθεση, εφόσον έστω και ένα από τα Φίλτρα Προθέσεως ταυτίζεται με την Πρόθεση. Πιο κάτω περιγράφονται αναλυτικότερα οι Προθέσεις.

- Οι ειδοποιήσεις (Notifications). Είναι τα μικρά εικονίδια τα οποία εμφανίζονται στην μπάρα καταστάσεων. Ο χρήστης πατώντας σε κάποιο από αυτά, είναι σε θέση να αλληλεπιδράσει με κάποια εφαρμογή.
- Οι όψεις (Views). Δημιουργούν τη διεπιφάνεια χρήστη (User Interface, UI). Η βασική κλάση για τα widgets, που χρησιμοποιούνται για τη δημιουργία διαδραστικών στοιχείων UI, όπως κουμπιά, πεδία κειμένου, κτλ. Είναι επίσης υπεύθυνες για το χειρισμό των events που δημιουργούνται όταν ο χρήστης αλληλεπιδράσει / πατήσει κάποιο στοιχείο της διεπιφάνειας, αλλά και για την αποθήκευση της τρέχουσας κατάστασης της διεπιφάνειας (saving state).
- Το αρχείο AndroidManifest.xml. Δεν βρίσκεται στο φάκελο res/ , όπως τα υπόλοιπα αρχεία με κατάληξη .xml, αλλά στον κεντρικό φάκελο κάθε εφαρμογής (root directory). Παρέχει στο Android βασικές πληροφορίες της εφαρμογής της οποίας είναι μέρος και τις οποίες το σύστημα πρέπει να γνωρίζει προτού εκτελέσει τον κώδικα της εφαρμογής. Ανάμεσα στις πληροφορίες που περιέχει είναι το όνομα του Java πακέτου της εφαρμογής και είναι μοναδικό για κάθε εφαρμογή, περιγράφει τα συστατικά από τα οποία αποτελείται η εφαρμογή, δηλαδή τις Activities, τους Broadcast Receivers, τις Services και τους Content Providers, καθώς και τις κλάσεις και άλλα στοιχεία, όπως τα Intents, τα οποία υλοποιούνται από κάθε ένα από αυτά τα συστατικά, έτσι ώστε το σύστημα να γνωρίζει ποια συστατικά και υπό ποιες συνθήκες θα εκτελεστούν, τα permissions που πρέπει να έχει η εφαρμογή και αυτά που απαιτεί από άλλες για να έχουν πρόσβαση στο API της εφαρμογής, αλλά περιέχει και μεταδεδομένα (metadata),

όπως η ελάχιστη έκδοση Android API που πρέπει να έχει η συσκευή, έτσι ώστε να εκτελεστεί σωστά ο κώδικας της εφαρμογής και άλλα.

Συγκεκριμένα, οι άδειες (permissions) ορίζονται από την εφαρμογή ή δηλώνει ότι ζητάει κάποια άδεια χρησιμοποιώντας τις ετικέτες `<permission>` και `<uses-permission>`. Μερικές άδειες παραχωρούνται αυτόματα από το σύστημα, ενώ άλλες αυτόματα απορρίπτονται. Συνήθως, εμφανίζονται στο χρήστη πριν από την εγκατάσταση της εφαρμογής σε ένα παράθυρο διαλόγου, όπου απαιτείται η έγκρισή του. Εάν ο χρήστης απορρίψει κάποια από τις άδειες της εκάστοτε εφαρμογής, τότε η εφαρμογή αυτή δεν θα εγκατασταθεί. Ο έλεγχος των αδειών γίνεται μόνο κατά την εγκατάσταση και όχι αργότερα. Είναι πιθανό κάποιος χρήστης να θεωρήσει ότι οι άδειες που απαιτούνται για την εγκατάσταση θα επηρεάσουν άλλες εφαρμογές ή είναι αρκετά παρεισφρητικές, το οποίο ίσως έχει ως αποτέλεσμα τη μη εγκατάσταση της εφαρμογής από το χρήστη ή ακόμα και αρνητική κριτική.

Εικόνα 15. Εμφάνιση των αδειών της εφαρμογής "Learn 50 Languages". Ο χρήστης πρέπι να συμφωνεί για να προχωρήσει στην εγκατάσταση.

Οι άδειες που υπάρχουν στο αρχείο AndroidManifest.xml αναφέρονται σε ποιες από τις δυνατότητες του υλικού της συσκευής, για παράδειγμα η κάμερα, ή σε ποιες από αυτές των άλλων εφαρμογών, π.χ. της εφαρμογής Επαφές, χρειάζεται η εφαρμογή να έχει πρόσβαση. Αναφέρει αυτές τις άδειες, έτσι ώστε να εγκατασταθεί ομαλά η εφαρμογή, για να μπορεί να αλληλεπιδρά με το υπόλοιπο σύστημα. Η παραπάνω εικόνα είναι αυτή που εμφανίζεται εφόσον το σύστημα έχει ανιχνεύσει όλα τα `<uses-permission>` και έχει καθορίσει σε ποιες ακριβώς δυνατότητες αναφέρεται η εφαρμογή, και έπειτα παρουσιάζει αυτό ακριβώς το παράθυρο. Μόλις ο χρήστης επιλέξει την Εγκατάσταση, δηλαδή συμφωνεί να δώσει πρόσβαση στις δυνατότητες που του έχουν παρατεθεί, το Android χρησιμοποιώντας αυτά τα `<uses-permission>` δημιουργεί ένα λογαριασμό στο υποκείμενο Linux λειτουργικό

σύστημα, το οποίο είναι μέρος του Android. Έτσι, δημιουργείται ένας λογαριασμός χρήστη για αυτήν την εφαρμογή.

Πιο κάτω παρατίθεται η AndroidManifest.xml μίας εφαρμογής που δημιουργείται με χρήση του οδηγού (wizard) του Eclipse για μία εφαρμογή Android. Αρχικά μέσα σε μία ετικέτα `<manifest>` παρατίθεται το πεδίο `xmlns:android`, το οποίο πρέπει πάντα να είναι `"https://schemas.android.com/apk/res/android"`. Μέσα σε αυτήν την ετικέτα δίνονται εμφωλευμένες όλες οι υπόλοιπες πληροφορίες. Βλέπουμε τις πληροφορίες για το όνομα του πακέτου που χρησιμοποιείται από την εφαρμογή, το οποίο πρέπει να είναι μοναδικό, έτσι ώστε να αποφεύγονται "συγκρούσεις" με άλλες εφαρμογές. Μάλιστα, εάν ένα αντικείμενο υπάρχει σε διαφορετικό πακέτο, τότε πρέπει να δηλώνεται ολόκληρο όνομα του πακέτου που το περιέχει, για παράδειγμα, `com.example.allidokimi.Object`. Συνήθως, το όνομα ενός πακέτου είναι ανάποδα το domain name του δημιουργού. Υπάρχουν, επίσης, πληροφορίες για την έκδοση της εφαρμογής (`android:versionCode`), η οποία ξεκινάει από το "1" και κάθε φορά που υπάρχει καινούρια έκδοση, προστίθεται ένα νούμερο στο νούμερο της έκδοσης, ασχέτως αν η έκδοση αυτή διαφέρει αρκετά ή όχι από την προηγούμενη, και τα API επίπεδα που απαιτούνται. Στο πεδίο `<uses-sdk>` είναι οι ετικέτες για τη μικρότερη έκδοση που απαιτείται για να δουλεύει σωστά η εφαρμογή (`android:minSdkVersion`), που λειτουργεί ως φίλτρο στο Google Play, καθώς δεν είναι δυνατόν να εγκατασταθεί μία εφαρμογή σε συσκευή με παλαιότερη έκδοση από αυτήν που ορίζεται στην AndroidManifest, ενώ η έκδοση που ορίζεται στο πεδίο `android:targetSdkVersion` είναι η έκδοση στην οποία η εφαρμογή έχει αναπτυχθεί και δοκιμαστεί και στην οποία στοχεύει. Εάν δεν έχει οριστεί, η προεπιλεγμένη τιμή ισούται με την τιμή που έχει αποδοθεί στο `minSdkVersion`. Καθώς κάθε API level έχει περισσότερα χαρακτηριστικά, αλλά και έχει διορθώσει πιθανά σφάλματα που υπήρχαν σε προηγούμενες εκδόσεις, είναι καλή πρακτική να δηλώνεται η τελευταία έκδοση, έτσι ώστε να γίνεται χρήση όλων των αλλαγών και των βελτιώσεων της πλατφόρμας. Εάν η συσκευή δεν αντιστοιχεί στο API επίπεδο που έχει δηλωθεί, το Android μπορεί να εφαρμόσει προς τα πίσω ή προς τα μπροστά συμβατότητα. Μέσα στην ετικέτα `<application> </application>` εμπεριέχονται μεταδεδομένα για τις τοποθεσίες του εικονιδίου και του string για το

όνομα της εφαρμογής, ενώ μπορεί να περιέχει ορισμούς για τα συστατικά Android της εφαρμογής. Έτσι, έχει εμφωλευμένη μία ετικέτα με το όνομα `<activity>` με πληροφορίες για τη Δραστηριότητα που έγινε μέσω του οδηγού, όπως το όνομά της και την τοποθεσία του string για το όνομα. Εάν το όνομα της Δραστηριότητας εμφανιζόταν ως `".MainActivity"` , τότε σίγουρα θα ήταν μέρος του πακέτου το οποίο έχει ήδη δηλωθεί. Μέσα σε αυτήν ετικέτα υπάρχει μία ακόμα εμφωλευμένη, το `<intent-filter>`, που είναι το Φίλτρο Προθέσεως που ήδη έχει περιγραφεί. Οι δύο γραμμές κώδικα που βρίσκονται μέσα στο Φίλτρο Προθέσεως είναι απαραίτητες, καθώς ορίζει ότι αυτή η Δραστηριότητα είναι η κύρια (main activity) και ότι μπορεί να θεωρηθεί ως ένα πιθανό σημείο εισόδου στην εφαρμογή (MAIN) και να είναι διαθέσιμη στην εκκίνηση του Android (LAUNCHER). Εδώ, επίσης, δηλώνεται εάν παρέχει άδεια ή εάν χρειάζεται κάποια άδεια χρησιμοποιώντας τις ετικέτες `<permission>` και `<uses-permission>`, αντίστοιχα. Εάν υπήρχαν και άλλες Δραστηριότητες, προστίθενται και αυτές στο αρχείο AndroidManifest.xml. Παρόμοια με την ετικέτα `<activity>`, μπορεί να υπάρχουν ετικέτες `<service>`, `<receiver>` και `<provider>` για να δηλωθούν οι Υπηρεσίες, οι Δέκτες Εκπεμπόμενων Προθέσεων και οι Πάροχοι Περιεχομένου. Ανάλογα με την εφαρμογή μπορεί να προστεθούν και άλλες ετικέτες.

Παράδειγμα AndroidManifest.xml

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

   package="com.example.dokimi"

   android:versionCode="1"

   android:versionName="1.0" >



<uses-sdk
```

```
    android:minSdkVersion="8"
    android:targetSdkVersion="19" />


    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.dokimi.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name=
                        "android.intent.category.LAUNCHER"  />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

1. Οι πόροι, όπως οι εικόνες και τα αρχεία με κατάληξη xml, βρίσκονται σε διαφορετικό φάκελο από ότι τα αρχεία του πηγαίου κώδικα. Ήδη έχει αναφερθεί η χρήση των πόρων, πώς αναφέρονται μέσα στον πηγαίο κώδικα ή σε κάποιο αρχείο xml, αλλά και τα θετικά της χρήσης εξωτερικών πόρων και όχι στοιχείων μέσα στον πηγαίο κώδικα. Παρακάτω παρατίθεται ένας πίνακας που δείχνει σε ποιο φάκελο αποθηκεύεται κάθε πόρος, ανάλογα με τον τύπο του:

Πόρος	Περιγραφή	Φάκελος
Εικόνες (Drawables)	Εικόνες, όπως αρχεία .png ή .jpeg, ή xml αρχεία που περιγράφουν ένα αντικείμενο Drawable	/res/drawables
Τιμές (Values)	Ορίζουν τα strings, πίνακες που περιέχουν strings, τα στυλ, τις διαστάσεις, τα χρώματα, και άλλα στοιχεία. Κάθε μία από αυτές τις ομάδες έχει το δικό της αρχείο, για παράδειγμα τα strings βρίσκονται στο αρχείο /res/vlaues/strings.xml.	/res/values
Στυλ και Θέματα (Styles and Themes)	Είναι αρχεία που ορίζουν την εμφάνιση της εφαρμογής.	/res/values
Διατάξεις (Layouts)	Τα αρχεία xml που περιγράφουν διατάξεις, σχετίζονται με τη διεπιφάνεια χρήστη και άρα με Δραστηριότητες και fragments.	/res/layout
Γραφικά στοιχεία (Animations)	Χρησιμοποιούνται για να προσδώσουν στην εφαρμογή στοιχεία τα οποία βοηθούν τους χρήστες να καταλάβουν καλύτερα την εφαρμογή και να βελτιώσουν την αλληλεπίδραση με αυτήν.	/res/animator
Ακατέργαστα δεδομένα (Raw data)	Τα δεδομένα αυτά περιλαμβάνουν αρχεία, όπως βίντεο και προσπελαύνονται μέσω ενός αντικειμένου InputStream.	/res/raw
Στοιχεία μενού (Menus)	Είναι μία διεπιφάνεια για τη διαχείριση στοιχείων μενού.	/res/menu

2. Τα IDs των πόρων και το αρχείο R.java. Κάθε πόρος λαμβάνει ένα ID από το Android. Ο φάκελος /gen περιέχει το αρχείο R.java, που εμπεριέχει αυτές τις παραγόμενες τιμές. Αυτές οι αναφορές είναι στατικές τιμές integer (static integer values). Εάν προστεθεί ένας καινούριος πόρος, αυτόματα δημιουργείται η τιμή του ID του και αναφέρεται στο R.java. Κανονικά δεν πρέπει να υπάρχουν αλλαγές ή τροποποιήσεις από το δημιουργό της εφαρμογής σε αυτό το αρχείο, καθώς παράγεται αυτόματα. Το Android σύστημα παρέχει μεθόδους, έτσι ώστε να υπάρχει πρόσβαση στα IDs των πόρων. Επί παραδείγματι, για πρόσβαση σε κάποιο string με το ID `R.string.myString` στον πηγαίο κώδικα, υπάρχει η μέθοδος `getString(R.string.myString)`.
3. Εκτός από τους πόρους που διαθέτει ο δημιουργός, το ίδιο το Android παρέχει και αυτό πόρους. Ονομάζονται πόροι του συστήματος (system resources) και ξεχωρίζουν από τους υπόλοιπους λόγω του τρόπου με τον οποίο αναφέρονται, για παράδειγμα `android.R.string.someString`.

Τα συστατικά αυτά είναι απαραίτητα για τη δημιουργία εφαρμογών. Το Android τα υλοποιεί και τα χρησιμοποιεί όπως κρίνει απαραίτητο, ενώ κάθε μία από αυτές τις συνιστώσες έχει το δικό της σκοπό αλλά και το δικό της API.

Συνοπτικά, μία εφαρμογή πέρα από τον πηγαίο κώδικα αποτελείται και από άλλα απαραίτητα συστατικά, όπως τα αρχεία layout, string, menu και animation. Ο δημιουργός της εφαρμογής πρέπει να ορίσει αυτούς τους πόρους, οι οποίοι βρίσκονται στο φάκελο res/ του αρχείου. Για παράδειγμα, για να οριστεί ένα string, πηγαίνουμε στο φάκελο res/values, όπου χρησιμοποιώντας στοιχεία της γλώσσας xml, γράφουμε `<string name="hello">Hello</string>`, το οποίο περιέχει το όνομα του string, αλλά και το ίδιο το string. Μπορούν να περιλαμβάνουν μορφοποίηση όπως τα HTML tags. Άλλα αρχεία που βρίσκονται στο φάκελο res/ και τα οποία χρησιμοποιούν επίσης στοιχεία της γλώσσας xml, μπορούν να αναφερθούν σε αυτό το string χρησιμοποιώντας την έκφραση `@string/<string_name>` (εδώ `@string/hello`), ενώ στον πηγαίο κώδικα στα αρχεία με κατάληξη .java αναφέρονται ως `R.string.string_name`. Μπορεί επίσης να ορίσει αυτά τα στοιχεία μέσα στον πηγαίο κώδικα ή να συνδυάσει και τους δύο τρόπους. Γενικά, όμως, η προσέγγιση του ορισμού των πόρων σε αρχεία xml είναι προτιμότερη. Τα πλεονεκτήματα του να υπάρχουν οι πόροι αυτοί σε ξεχωριστά αρχεία είναι κυρίως δύο:

μπορούν να αλλαχθούν χωρίς να χρειάζεται εκ νέου να μεταγλωττιστεί ο πηγαίος κώδικας και διαφορετικοί πόροι μπορούν να δημιουργηθούν βάσει των προτιμήσεων χρήστη, των διαφορετικών συσκευών ή διαφορετικών ρυθμίσεων και να χρησιμοποιηθούν χωρίς να επηρεάζεται ο πηγαίος κώδικας και κατά επέκταση η εφαρμογή. [2] [4] [5] [18]

3.5. Βήματα για τη δημιουργία μίας εφαρμογής

Η δημιουργία μίας εφαρμογής αποτελείται από τα παρακάτω βήματα:

Η διεπιφάνεια χρήστη (UI), η οποία ορίζει πώς εμφανίζεται η εφαρμογή στη συσκευή, ορίζεται στα αρχεία που χρησιμοποιούν στοιχεία της γλώσσας xml και έχουν την κατάληξη .xml και βρίσκονται μέσα στο φάκελο res/ του προς εξαγωγή αρχείου. Υπάρχουν εργαλεία τα οποία χρησιμεύουν στο να δημιουργηθεί η διεπιφάνεια χρήστη οπτικά, χωρίς να χρειάζεται η γνώση της γλώσσας xml, και μετατρέπουν αυτή τη διεργασία σε κώδικα. Άλλα στοιχεία που εμφανίζονται στο φάκελο res/ μπορούν να έχουν πρόσβαση στους πόρους της διεπιφάνειας χρησιμοποιώντας την έκφραση `@file/resource_name`, π.χ. `@string/app_name`. Εάν, για παράδειγμα, υπάρχουν πολλαπλές επιλογές, όπως στην περίπτωση διαφορετικών μεγεθών οθόνης ή διαφορετικών ρυθμίσεων χρήστη για τη γλώσσα, η επιλογή γίνεται κατά το χρόνο εκτέλεσης (run time) βάσει της παραμετροποίησης της συσκευής, οπότε και επιλέγεται το αρχείο το οποίο θα απεικονιστεί.

Στα αρχεία .java για να παραχθούν στη γλώσσα java οι πόροι αυτοί, γίνεται η αναφορά τους μέσα στον πηγαίο κώδικα ως εξής: `R.file.resource_name`, π.χ. `R.layout.help`. Κατά τη μεταγλώττιση, οι πόροι που έχουν αναφερθεί μέσα στον πηγαίο κώδικα χρησιμοποιούνται για την παραγωγή της κλάσης R.java, η οποία δημιουργείται αυτόματα καθώς εισάγουμε τους διάφορους πόρους μέσα στα αρχεία java. Η συγκεκριμένη κλάση δεν πρέπει να τροποποιείται. Τα πεδία που βρίσκονται σε αυτήν την κλάση αποτελούν αναφορές στους πόρους της εφαρμογής, οι οποίοι είχαν οριστεί αρχικά στα αρχεία .xml.

Κατά την υλοποίηση των κλάσεων της εφαρμογής, πρέπει να υπάρχει τουλάχιστον μία κλάση που επεκτείνει (`extends`) την κλάση Activity. Στο σώμα αυτής της κλάσης γίνεται η αρχικοποίηση της εφαρμογής καλώντας τη μέθοδο `onCreate()`. Συνήθως, η τυπική ροή εργασίας (workflow) της παραπάνω μεθόδου είναι: επαναφορά της προηγούμενης κατάστασης της εφαρμογής (`savedInstanceState`), δηλαδή της κατάστασης που είχε η εφαρμογή όταν ο χρήστης τη χρησιμοποίησε τελευταία φορά, ορισμός της διεπιφάνειας

χρήστη (`setContentView`), κάνοντας αναφορά στο αρχείο layout και δημιουργώντας έτσι αντικείμενα Java σε αντιστοιχία με τα στοιχεία που ορίζονται στα αρχεία xml και στο φάκελο res/ ως πόροι, αρχικοποίηση στοιχείων διεπιφάνειας χρήστη (`findViewById`) και σύνδεση των στοιχείων αυτών με τον κώδικα, έτσι ώστε να είναι σε θέση να αλληλεπιδρούν με το χρήστη, για παράδειγμα χρησιμοποιώντας listeners.

Στη συνέχεια, το σύστημα δημιουργεί ένα πακέτο με τα συστατικά (components) και τους πόρους της εφαρμογής. Το πακέτο αυτό είναι το αρχείο με κατάληξη .apk και είναι το εκτελέσιμο αρχείο της εφαρμογής και περιλαμβάνει αρχεία .dex, μη μεταγλωττίσιμους πόρους (εικόνες), αρχεία με άλλους πόρους (.xml), και φυσικά, το αρχείο AndroidManifest.xml της εφαρμογής. Τα εργαλεία που παρέχονται στο SDK δημιουργούν αυτό το αρχείο αντλώντας πληροφορίες που υπάρχουν στο αρχείο AndroidManifest.xml. Οι πληροφορίες που απαιτούνται είναι ποικίλες και μπορεί να είναι μεταξύ των: όνομα της εφαρμογής, μία λίστα με τα συστατικά (components) της, αλλά και άλλα χαρακτηριστικά, όπως άδειες (permissions), χαρακτηριστικά του υλικού της συσκευής, ελάχιστο επίπεδο της απαιτούμενης πλατφόρμας Android ή η έκδοση στην οποία στοχεύει η εκάστοτε εφαρμογή, όπως αναφέρθηκε πιο πάνω.

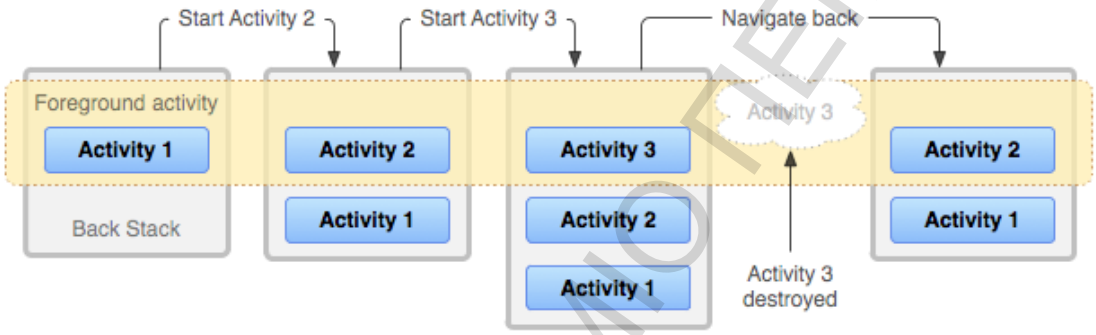
Έπειτα, απαραίτητο είναι να εκδοθεί ένα "κλειδί" το οποίο θα πιστοποιεί το δημιουργό της εφαρμογής. Το Android απαιτεί όλες οι εφαρμογές να έχουν αυτό το ψηφιακό πιστοποιητικό, το οποίο έχει ο δημιουργός στην κατοχή του, έτσι ώστε να πιστοποιείται ο δημιουργός. Παρόλο που είναι μία απλή διαδικασία, είναι υποχρεωτικό για όλες τις εφαρμογές να έχουν την ψηφιακή υπογραφή του δημιουργού τους.

Το τελευταίο βήμα είναι η εγκατάσταση του .apk πακέτου και η χρήση της εφαρμογής είτε σε φυσική είτε σε εικονική συσκευή, έτσι ώστε να γίνουν οι απαραίτητες δοκιμές και η αποσφαλμάτωση. Πλέον το προϊόν έχει δοκιμαστεί πλήρως και είναι έτοιμο για πιθανή εμπορική χρήση.

3.6. Η κλάση Activity

Η κλάση Activity αποτελεί ένα από τα τέσσερα θεμελιώδη συστατικά μίας εφαρμογής Android. Είναι υπεύθυνη για την αλληλεπίδραση με το χρήση, καθώς προσφέρει τη διεπιφάνεια μέσω της οποίας ο χρήστης επικοινωνεί με το σύστημα. Συνήθως, κάθε activity υποστηρίζει μία στοχευμένη δράση που μπορεί να κάνει ο χρήστης, όπως να δει ένα μήνυμα

ηλεκτρονικού ταχυδρομείου ή μία οθόνη σύνδεσης (login screen). Οπότε, μία εφαρμογή τυπικά απαρτίζεται από πολλές επιμέρους activities, η καθεμία εκ των οποίων έχει ένα συγκεκριμένο στόχο, και είναι συνδεδεμένες μεταξύ τους. Μία εξ αυτών είναι η κύρια (main activity), η οποία εμφανίζεται όταν ο χρήστης ξεκινά την εφαρμογή. Εάν ο χρήστης εκτελέσει κάποια άλλη εργασία μέσα στην εφαρμογή, μία άλλη activity θα έρθει στο προσκήνιο, ενώ η προηγούμενη θα σταματήσει, αλλά το σύστημα θα τη διατηρήσει σε μία στοίβα ("back stack"). Καθώς ο χρήστης αλληλεπιδρά με την εφαρμογή μέσω διαφόρων activities, κάθε φορά αυτή που ξεκινά μπαίνει μπροστά σε αυτήν τη στοίβα. Όταν πατηθεί το πλήκτρο Πίσω (Back) της συσκευής, η τελευταία activity φεύγει από αυτήν τη στοίβα και καταστρέφεται, ενώ η προηγουμένως χρησιμοποιούμενη επανέρχεται στο προσκήνιο.



Εικόνα 16. Διάγραμμα του BackStack.

Μία activity μπορεί να ξεκινήσει activities που ανήκουν σε άλλες εφαρμογές. Για παράδειγμα, εάν θέλουμε να στείλουμε ένα γραπτό μήνυμα σε μία επαφή από την εφαρμογή Επαφές της συσκευής, εάν έχουν συμπεριληφθεί κάποια δεδομένα μέσα σε ένα Intent, μπορεί μία άλλη activity που ανήκει σε άλλη εφαρμογή να είναι σε θέση να διαχειριστεί αυτήν τη διεργασία, π.χ. η εφαρμογή Μηνύματα, ή εάν είναι περισσότερες της μίας εφαρμογής στη συσκευή που υποστηρίζουν αυτό το intent, τότε ο χρήστης επιλέγει ποια εφαρμογή θα χρησιμοποιήσει. Μόλις ολοκληρωθεί αυτή η διεργασία, τότε η προηγούμενη activity επανέρχεται και φαίνεται ότι η δεύτερη activity είναι μέρος της εφαρμογής Επαφές. Παρόλο που οι activities ανήκουν σε διαφορετικές εφαρμογές, το Android τις τοποθετεί στο ίδιο task. Ένα task είναι ένα σύνολο από activities με τις οποίες ο χρήστης αλληλεπιδρά όταν χρειάζεται να κάνει μία συγκεκριμένη εργασία και οι οποίες μπορεί να ανήκουν σε μία εφαρμογή ή να επεκτείνονται σε περισσότερες της μίας. Τα περισσότερα tasks ξεκινούν από την Αρχική Οθόνη. Αυτές οι activities μπαίνουν στο backstack και λειτουργούν με τη λογική που περιγράφηκε προηγουμένως.

Άρα, στη στοίβα task backstack όταν μία εφαρμογή ξεκινά, μπαίνει στη στοίβα ως μέρος του τρέχοντος task, ενώ όταν καταστρέφεται, δηλαδή όταν η εφαρμογή τερματιστεί ή ο χρήστης πατήσει το πλήκτρο της Αρχικής Οθόνης ή ακόμα μπορεί το Android να τερματίσει την εφαρμογή λόγω μείωσης των διαθέσιμων πόρων, η activity βγαίνει από αυτήν τη στοίβα.

Οι αλλαγές στην κατάσταση μίας activity πραγματοποιούνται καλώντας τις μεθόδους επανάκλησης του κύκλου ζωής της activity (lifecycle callback methods). Κάθε μία από αυτές παρέχει τη δυνατότητα να γίνουν διαφορετικές εργασίες κατάλληλες για την αλλαγή στην κατάσταση της activity. Επί παραδείγματι, όταν γίνει κλήση στην κατάλληλη μέθοδο για να σταματήσει, η activity θα αποδεσμεύσει αντικείμενα, όπως συνδέσεις δικτύου.

Για να δημιουργηθεί μία activity πρέπει να επεκταθεί (extends) η κλάση Activity ή μία υποκλάση αυτής, και να υλοποιηθούν οι απαραίτητες μέθοδοι callback, τις οποίες καλεί το σύστημα όταν αλλάζει η κατάσταση αυτής της activity, π.χ. όταν δημιουργείται ή καταστρέφεται. Οι πιο σημαντικές μέθοδοι callback είναι οι `onCreate()` και `onPause()`, οι οποίες αναλύονται παρακάτω. [18]

3.7. Ο κύκλος ζωής μίας Activity

Οι activities δημιουργούνται, αναστέλλονται, συνεχίζουν και καταστρέφονται ως μέρος του κύκλου ζωής τους κατά τη διάρκεια που μία εφαρμογή εκτελείται. Οι καταστάσεις αυτές αλλάζουν χωρίς απαραίτητα η εφαρμογή να έχει τον έλεγχο των αλλαγών, καθώς πολλές φορές μία αλλαγή εξαρτάται από άλλους παράγοντες, όπως: η συμπεριφορά του χρήστη, εάν για παράδειγμα πατήσει το πλήκτρο της Αρχικής Οθόνης, ή το ίδιο το σύστημα, καθώς μπορεί μία activity να είναι σε κατάσταση αναστολής και το Android να την καταστρέφει λόγω των πεπερασμένων πόρων που διαθέτει, ακόμα και αν αργότερα θα χρειαστεί να τη δημιουργήσει εκ νέου.

Όταν μία activity αλλάζει κατάσταση, ενημερώνεται με τις μεθόδους επανάκλησης (callback methods). Μέσα στον πηγαίο κώδικα οι μέθοδοι αυτές μπορούν να υπερσκελιστούν (overriden) εφόσον αλλάζει η κατάσταση της activity. Παρακάτω δίνονται οι βασικές lifecycle callback methods, όπως εμφανίζονται μέσα στον κώδικα:

```
Activity 's Lifecycle Callback Methods
```

```
public class ExampleActivity extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      //The activity is being created.

     //It is about to start.
   }
   @Override
   protected void onStart(){
      super.onStart();
      //The activity is about to become visible.

     //It is about to start.
   }
   @Override
   protected void onResume(){
      super.onResume();
      //The activity has become visible (it is now “resumed”).

     //Visible. User interaction.
   }
   @Override
   protected void onPause(){
      super.onPause();
      //Another activity is taking focus (this activity is  //about
      to be “paused”).

     //Maybe partially visible. User does not interact. Can
     // be terminated. Before Android 3.0 they could be
     // terminated once they enter this stage.
```

```
    }
    @Override
    protected void onRestart(){
        super.onRestart();
        //The activity is being called back. After this, the
        // onStart() is called.

    }
    @Override
    protected void onStop(){
        super.onStop();
        //The activity is no longer visible (it is now stopped”)

      //Not visible. Can be terminated by Android.
    }
    @Override
    protected void onDestroy(){
        super.onDestroy();
        //The activity is about to be destroyed.

    }
}
```

Η υλοποίηση κάθε μίας από αυτές τις μεθόδους είναι απαραίτητο να καλεί την υπερκλάση, όπως φαίνεται παραπάνω (`super.onCreate(...)`).

Η activity θα έχει διαφορετική συμπεριφορά αναλόγως σε ποιο στάδιο είναι. Οπότε, για να υποστηρίξει όλα τα πιθανά σενάρια, το Android γνωστοποιεί τις αλλαγές στον κύκλο ζωής στην activity καλώντας κάθε φορά μία από τις παραπάνω μεθόδους.

Στην εικόνα που παρατίθεται παρακάτω, εμφανίζεται πώς αλληλοσυνδέονται μεταξύ τους οι μέθοδοι επανάκλησης, καθώς και η σειρά των σταδίων και των μεθόδων. Όπως παρατηρούμε

και από την εικόνα (Εικόνα 17), οι εφαρμογές δε λειτουργούν αυτοτελώς, αλλά σε συνεργασία με το Android σύστημα.

Επίσης, το γραφικό περιβάλλον της activity δεν είναι ορατό καθόλη τη διάρκεια του κύκλου ζωής της, αλλά και κατά τη διάρκεια που είναι ορατό σε κάποιες καταστάσεις είναι πιθανό ο χρήστης να μην μπορεί να αλληλεπιδράσει με την εφαρμογή. Στο παραπάνω απόσπασμα κώδικα μέσα στα σχόλια περιγράφεται πότε η εφαρμογή είναι ορατή και διαθέσιμη για αλληλεπίδραση με το χρήστη. Κατά τη διάρκεια που ο χρήστης αλληλεπιδρά με τη συγκεκριμένη activity, μπορεί να προκληθεί κάποιος γεγονός (event) και μία καινούρια activity ξεκινάει με δικό της κύκλο ζωής και δικές της μεθόδους επανάκλησης. Εάν συμβεί κάτι τέτοιο, η σειρά που θα ακολουθήσουν οι μέθοδοι callback είναι η εξής: θα γίνει κλήση στην `onPause()` της πρώτης activity ενώ είναι ακόμα στο προσκήνιο και η δεύτερη θα ξεκινά. Μετά από αυτήν την κλήση, η πρώτη activity δε θα είναι ορατή, ενώ το Android καλεί τη μέθοδο `onStop()`. Πλέον ο χρήστης μπορεί να αλληλεπιδρά με τη δεύτερη activity που έχει διαφορετικό κύκλο ζωής και μεθόδους επανάκλησης. Εάν η πρώτη activity κληθεί εκ νέου, πρέπει να έρθει πάλι στο προσκήνιο, καλώντας την μέθοδο onRestart(), μετά onStart() και ούτω καθεξής.

Εάν ο χρήστης φύγει από την activity, για παράδειγμα πατήσει το πλήκτρο Πίσω, το Android ολοκληρώνει τη διαδικασία της απομάκρυνσής της καλώντας διαδοχικά τις μεθόδους `onPause()`, `onStop()` και `onDestroy()`. Σε κάποιες περιπτώσεις μπορεί να μη γίνει κλήση στη μέθοδο onStop() και κατ' επέκταση στην `onDestroy()`, λόγω του ότι το σύστημα μπορεί να έχει τερματίσει την εφαρμογή για λόγους οικονομίας πόρων.

Σε κάθε περίπτωση η μέθοδος `onCreate()` θα είναι αυτή που θα αρχικοποιήσει την activity και η οποία θα πρέπει να διεκπεραιώσει τέσσερεις εργασίες:

Παράδειγμα onCreate()

```
void onCreate(Bundle savedInstanceState){
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   final Button btn = (Button) findViewById(R.id.btn);
}
```

πρώτα να καλέσει τη `super.onCreate()`, για να αρχίσει την αρχικοποίηση το Android και να επιστρέψει στην activity πληροφορίες, όπως εάν είχε καταστραφεί η activity, για παράδειγμα εάν η συσκευή είχε περιστραφεί (Περιστροφή Οθόνης), δεύτερον, να ορίσει το γραφικό περιβάλλον χρήστη περνώντας ως παράμετρο το ID ενός αρχείου layout, τρίτον, να δημιουργήσει αναφορές σε στοιχεία γραφικού περιβάλλοντος, για παράδειγμα κουμπιά, τα οποία βρίσκονται στα αρχεία με κατάληξη xml, και, τέλος, διαμορφώνει τα στοιχεία αυτά όπως απαιτείται έτσι ώστε να βρίσκονται στον πηγαίο κώδικα. Είναι απαραίτητο να οριστεί η μέθοδος `onCreate()`, καθώς σε άλλη περίπτωση ο κώδικας δε θα μεταγλωττιστεί.

Η χρήση της λέξης super είναι απαραίτητη, καθώς λόγω της κληρονομικότητας που υπάρχει στη Java, θα εκτελεστούν πρώτα οι μέθοδοι της υποκλάσης που έχουν υπερσκελιστεί (`override`) και ύστερα της υπερκλάσης της. Εάν επιθυμούμε να εκτελεστεί ένα σκέλος της κλάσης "προγόνου", τότε είναι απαραίτητη η χρήση της λέξης `super` στην αρχή της μεθόδου. Δίνουμε, δηλαδή πρόσβαση να εκτελεστεί η μέθοδος της υπερκλάσης αντί της κλάσης στην οποία αναφέρεται.

Οι υπόλοιπες μέθοδοι της κλάσης Activity δρουν ως εξής:

1. `onStart()` : Η Δραστηριότητα είναι σχεδόν ορατή. Οι ενέργειες που μπορεί να γίνουν είναι για παράδειγμα η ανανέωση των mails σε μία εφαρμογή ηλεκτρονικού ταχυδρομείου ή η αναζήτηση γεωγραφικών δεδομένων.
2. `onResume()` : Η Δραστηριότητα είναι ορατή και έτοιμη για να ξεκινήσει η αλληλεπίδραση με το χρήστη. Τυπικές ενέργειες είναι να ξεκινήσουν συμπεριφορές που σχετίζονται με το "προσκήνιο", όπως να ξεκινήσει ένα γραφικό στοιχείο animation.
3. `onPause()` : Η Δραστηριότητα σχεδόν έχει χάσει την προσοχή ή μπορεί και να αλλάζει Δραστηριότητα. Σταματούν συμπεριφορές που σχετίζονται με το προσκήνιο, όπως τα γραφικά σε ένα παιχνίδι, ενώ εάν ο χρήστης έχει κάνει κάποια επεξεργασία, τότε η κατάσταση αυτή σώζεται και αφορά δεδομένα που θα αποθηκευτούν, για παράδειγμα φωτογραφίες που έχουν τραβηχτεί με τη χρήση της κάμερας της συσκευής. Είναι δυνατόν το σύστημα να τερματίσει (kill) τη δραστηριότητα αφού επιστρέψει η μέθοδος, οπότε είναι πιθανόν οι επόμενες μέθοδοι (`onStop()` και `onDestroy()`) να μην κληθούν και ο κώδικας που περιέχουν να μην εκτελεστεί. Άρα, η `onPause()` είναι η τελευταία μέθοδος που σίγουρα θα κληθεί. Επομένως,

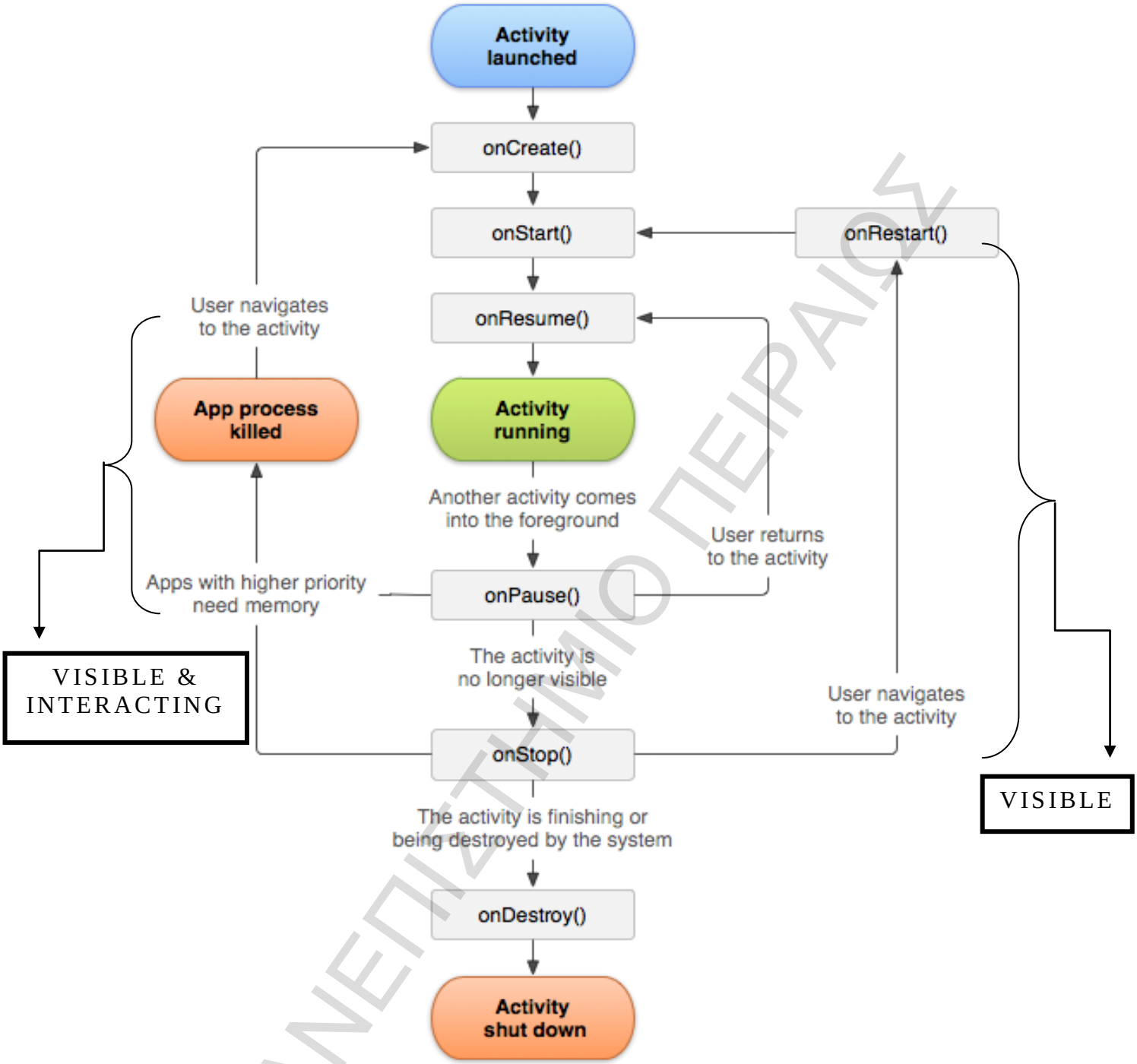
συνίσταται στο σώμα αυτής της μεθόδου να υπάρχει κώδικας που να αποθηκεύει σημαντικά δεδομένα.

4. `onRestart()` : Καλείται εάν η Activity έχει σταματήσει και θα ξεκινήσει εκ νέου. Ακολουθείται πάντα από την `onStart()`.
5. `onStop()` : Η Δραστηριότητα δεν είναι πλέον ορατή στο χρήστη. Μπορεί να επανεκκινήσει αργότερα. Αποθηκεύεται η κατάσταση της Δραστηριότητας, έτσι ώστε να αποκατασταθεί όταν η Δραστηριότητα επανεκκινήσει και καλεστεί η `onStart()`. Προσοχή, καθώς μπορεί η μέθοδος να μην κληθεί, εάν το σύστημα έχει τερματίσει την εφαρμογή, όπως για παράδειγμα λόγω χαμηλής στάθμης μπαταρίας της συσκευής. Οπότε, σε αυτή τη φάση δεν αποθηκεύονται δεδομένα, όπως φωτογραφίες ή videos, τα οποία σώζονται σε αποθηκευτικά μέσα, αλλά πρέπει ήδη να έχει γίνει στην `onPause()`.
6. `onDestroy()` : Η Δραστηριότητα είναι έτοιμη να τερματιστεί. Σε αυτό το στάδιο όλοι οι πόροι που έχουν χρησιμοποιηθεί αποδεσμεύονται. Μπορεί, επίσης, να μην κληθεί.

Όπως είναι κατανοητό, οι μέθοδοι αυτές ορίζουν τρεις καταστάσεις κατά τον κύκλο ζωής μίας δραστηριότητας. Η πρώτη είναι όλη τη διάρκεια του κύκλου ζωής της, που περιλαμβάνει όλες τις μεθόδους. Η δεύτερη είναι η ορατή κατάσταση μεταξύ της `onStart()` και της `onStop()`, όπου ο χρήστης βλέπει τη δραστηριότητα και μπορεί να ξεκινήσει την αλληλεπίδραση με αυτή, ενώ οι απαραίτητοι πόροι είναι δεσμευμένοι για αυτήν τη δραστηριότητα. Επίσης, μπορεί να γίνουν πολλές κλήσεις κατά τη διάρκεια του κύκλου ζωής της activity, καθώς αυτή εισέρχεται ή βγαίνει από την ορατή κατάσταση. Τέλος, υπάρχει η κατάσταση όπου η δραστηριότητα είναι στο προσκήνιο, καθώς είναι ορατή και ο χρήστης αλληλεπιδρά με αυτήν, καθώς είναι η κατάσταση στην οποία η δραστηριότητα βρίσκεται μπροστά από όλες τις άλλες. Σε αυτήν την κατάσταση, η οποία είναι μεταξύ της `onResume()` και της `onPause()` είναι καλό να μην υπάρχουν μεγάλα blocks κώδικα ή κώδικας δύσκολος κατά την εκτέλεσή του, καθώς είναι συχνή η είσοδος και η έξοδος από αυτήν την κατάσταση, για παράδειγμα όταν ένα κουτί διαλόγου μπαίνει στο προσκήνιο ή εμφανίζεται μία ειδοποίηση από μία άλλη εφαρμογή, και άρα η εκτέλεση κώδικα που επιτρέπει την εύκολη μετάβαση από τη μία κατάσταση στην άλλη συντελεί στην καλύτερη εμπειρία χρήστη. Επίσης, είναι πιθανόν η δραστηριότητα να τερματιστεί από το σύστημα σε

μερικές περιπτώσεις, όπως για παράδειγμα εάν είναι δεσμευμένοι όλοι οι πόροι του συστήματος και δεν υπάρχει κανένας διαθέσιμος.

Μπορεί να γίνει κλήση των `onDestroy()` και `onCreate()`, εάν κατά το χρόνο εκτέλεσης αλλάξουν κάποιες ρυθμίσεις στη συσκευή, όπως η γλώσσα ή ο προσανατολισμός οθόνης. Σε αυτές τις περιπτώσεις, το Android επαναδημιουργεί την τρέχουσα δραστηριότητα με τις κατάλληλες αλλαγές, αφού πρώτα καλέσει τις δύο αυτές μεθόδους, το οποίο επιτρέπει στην εφαρμογή να προσαρμοστεί σύμφωνα με τις ρυθμίσεις που έχουν αλλάξει χρησιμοποιώντας τους καινούριους πόρους, αλλά να μην τερματίσει την εφαρμογή.



Εικόνα 17. Κύκλος ζωής μίας Activity.

Οι μεταβολές αυτές μπορεί να συμβαίνουν παράλληλα με αυτές μίας άλλης δραστηριότητας. Για παράδειγμα, εάν μία δραστηριότητα καλεί μία άλλη, τότε η πρώτη παραμένει ορατή και η μέθοδος `onStop()` θα κληθεί, μόνο εφόσον έχει κληθεί η `onResume()` της δεύτερης. Οπωσδήποτε πρέπει να γίνει η δήλωση της activity στο αρχείο manifest της εφαρμογής, έτσι ώστε να είναι δυνατή η πρόσβαση από το σύστημα. Η δήλωσή της γίνεται όπως φαίνεται στο παρακάτω απόσπασμα κώδικα:

Δήλωση της Activity στη manifest.

```
<activity android:name=".ExampleActivity"
android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

Από το παράδειγμα φαίνεται ότι το χαρακτηριστικό `android:name`, το οποίο απαιτείται κατά τη δήλωση, είναι το όνομα της activity και το οποίο δεν πρέπει να αλλαχθεί για τη σωστή λειτουργία της εφαρμογής. Εδώ χρησιμοποιούνται και η ετικέτα `<intent-filter>`, που έχει ήδη αναλυθεί, για τη δήλωση του τρόπου με τον οποίο άλλα στοιχεία της εφαρμογής θα αλληλεπιδράσουν με την activity. Η χρήση του `<action android:name="android.intent.action.MAIN" />` σηματοδοτεί ότι αυτή η δραστηριότητα είναι αυτή που αποτελεί την κεντρική είσοδο της εφαρμογής, δηλαδή είναι η πρώτη δραστηριότητα που εκτελείται όταν ο χρήστης ξεκινά την εφαρμογή, ενώ η χρήση του `<category android:name="android.intent.category.LAUNCHER" />` υποδεικνύει ότι το εικονίδιο αυτής της δραστηριότητας, ή σε περίπτωση που δεν υπάρχει, το εικονίδιο που δηλώνεται στην AndroidManifest.xml στην ετικέτα application, πρέπει να τοποθετηθεί στη διεπιφάνεια χρήστη η οποία επιτρέπει στο χρήστη να προσαρμόσει την αρχική οθόνη της συσκευής του, να εκκινήσει εφαρμογές, αλλά και να εκτελέσει άλλες εργασίες, όπως οι τηλεφωνικές κλήσεις, και η οποία ονομάζεται Launcher (Εκκινητής). Απαραίτητα αυτά τα δύο στοιχεία πρέπει να υπάρχουν μαζί, έτσι ώστε να υπάρχει η

δραστηριότητα στον Εκκινητή και πρέπει μόνο μία δραστηριότητα να περιέχει αυτές τις τιμές για τα στοιχεία action και category. [18]

ΚΕΦΑΛΑΙΟ 4

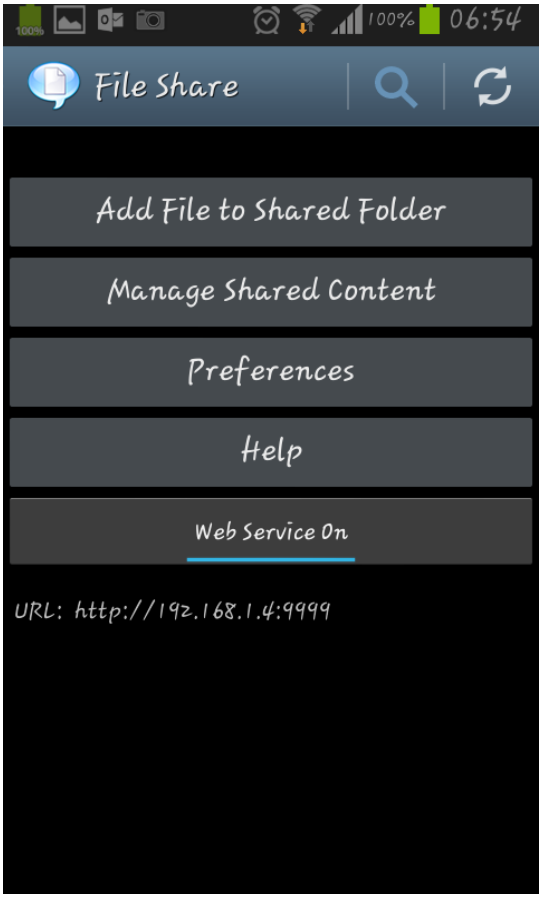
4.1. Η ΕΦΑΡΜΟΓΗ

Η εφαρμογή η οποία αναπτύχθηκε στη διάρκεια της παρούσας διπλωματικής εργασίας ονομάζεται FileShare και επιτρέπει στους χρήστες να διαμοιράζονται αρχεία τα οποία έχουν στην κινητή συσκευή Android με οποιαδήποτε συσκευή είτε είναι Android, είτε iPhone, είτε ένας προσωπικός υπολογιστής, αρκεί η συσκευή να έχει τη δυνατότητα να φυλλομετρεί ιστοσελίδες και να κατεβάζει δεδομένα από το δίκτυο.

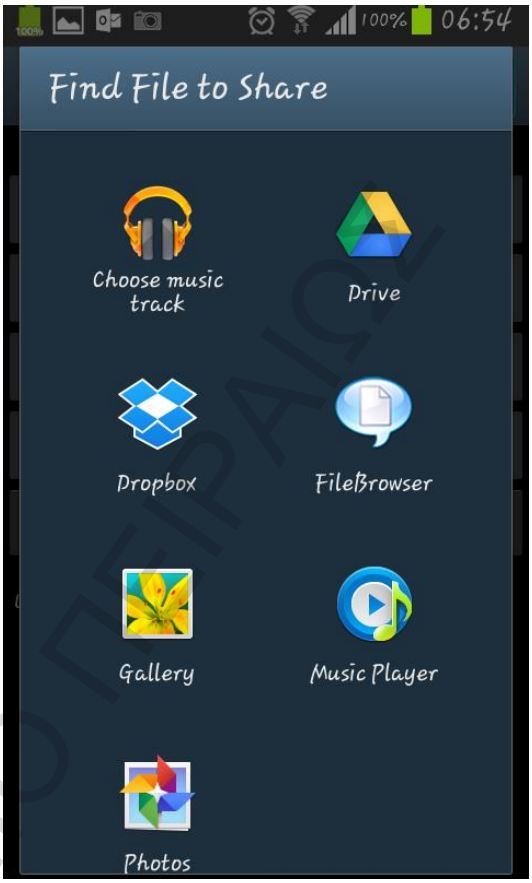
Ο χρήστης δημιουργεί φακέλους προς διαμοίραση (Shared Folders) και τοποθετεί περιεχόμενο σε αυτούς τους φακέλους. Για να μπορεί να τους δει μέσω μίας άλλης συσκευής, θα πρέπει το Android τερματικό να έχει σύνδεση στο διαδίκτυο. Τότε θα πάρει μία LAN IP διεύθυνση. Στη συνέχεια, για να μπορεί να έχει πρόσβαση στα περιεχόμενα του φακέλου του, θα πρέπει να μεταβεί στην IP διεύθυνση που παρέχεται από την εφαρμογή χρησιμοποιώντας κάποιο από τα προγράμματα περιήγησης. Πληκτρολογώντας στη γραμμή διευθύνσεων την IP διεύθυνση της συσκευής Android, μεταβαίνει στους φακέλους του και στα περιεχόμενά τους. Μέσω της διεπιφάνειας αυτής ο χρήστης μπορεί επίσης να ανεβάσει αρχεία τα οποία θα είναι διαθέσιμα στο φάκελο File Share της εφαρμογής.

Η εφαρμογή παρέχει τη δυνατότητα στο χρήστη να δημιουργήσει πολλαπλούς φακέλους τους οποίους στη συνέχεια μπορεί να διαμοιράζεται, και επίσης επιτρέπει την προσπέλαση των αρχείων και το κατέβασμά τους από τις διαφορετικές συσκευές ταυτόχρονα.

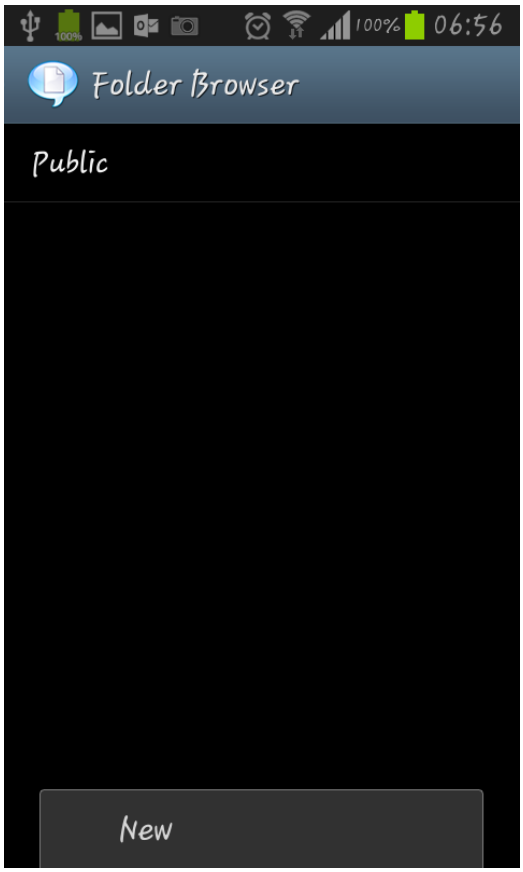
Παρακάτω παρατίθενται εικόνες κατά τη χρήση της εφαρμογής. Για τις εικόνες από την κινητή συσκευή χρησιμοποιήθηκε η συσκευή Samsung Galaxy Core και η εφαρμογή EasyScreenshot.



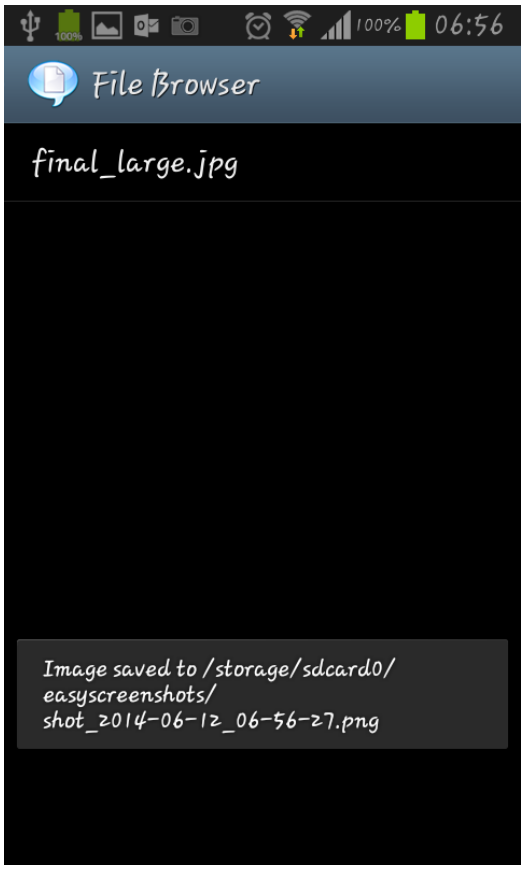
Εικόνα 18. Η αρχική οθόνη της εφαρμογής. Διακρίνονται το Action Bar, οι επιλογές που δίνονται και το URL που έχει λάβει η συσκευή και με το οποίο μπορούμε να συνδεθούμε από άλλη κινητή συσκευή ή από υπολογιστή.



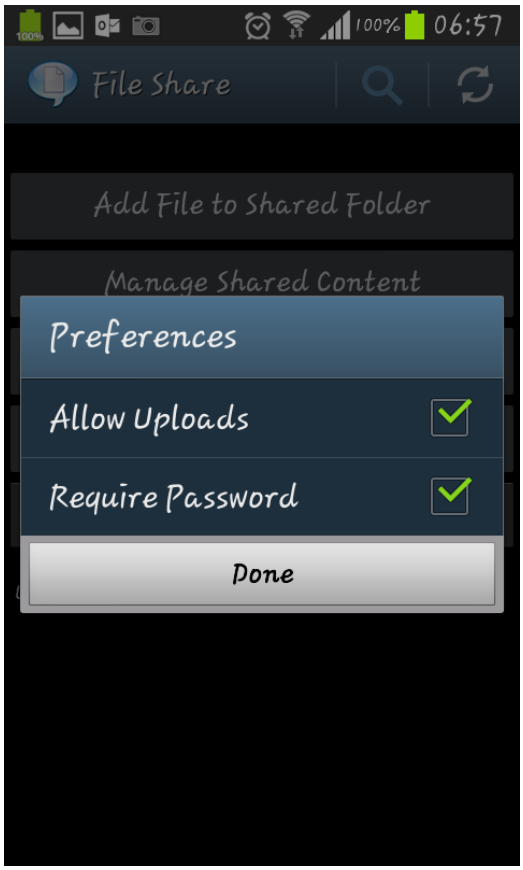
Εικόνα 19. Μετά από το πάτημα του κουμπιού Προσθήκη Αρχείου στον Κοινόχρηστο Φάκελο, ανοίγουν οι επιλογές για την Προσθήκη. Γίνεται μετάβαση είτε σε άλλη Δραστηριότητα είτε σε άλλη εφαρμογή.



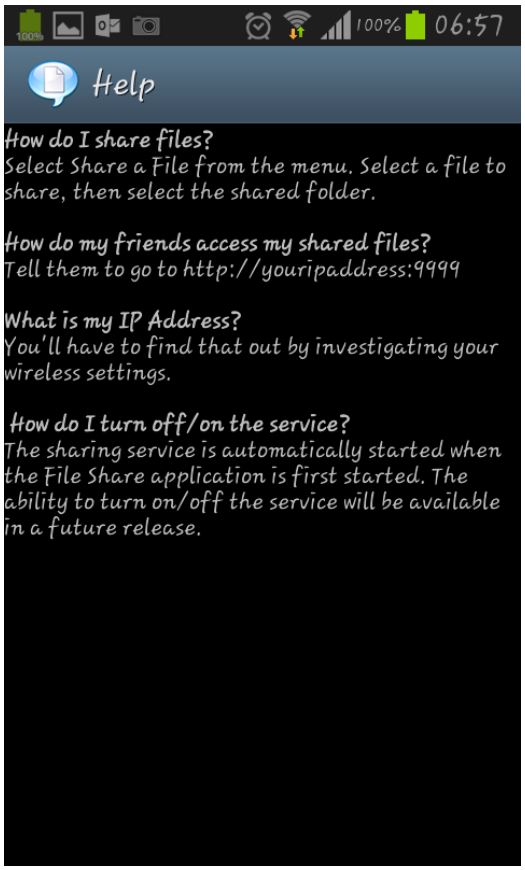
Εικόνα 20. Ο κοινόχρηστος φάκελος. Πατώντας το κουμπί του Μενού της συσκευής, εμφανίζεται η επιλογή New για δημιουργία νέου φακέλου.



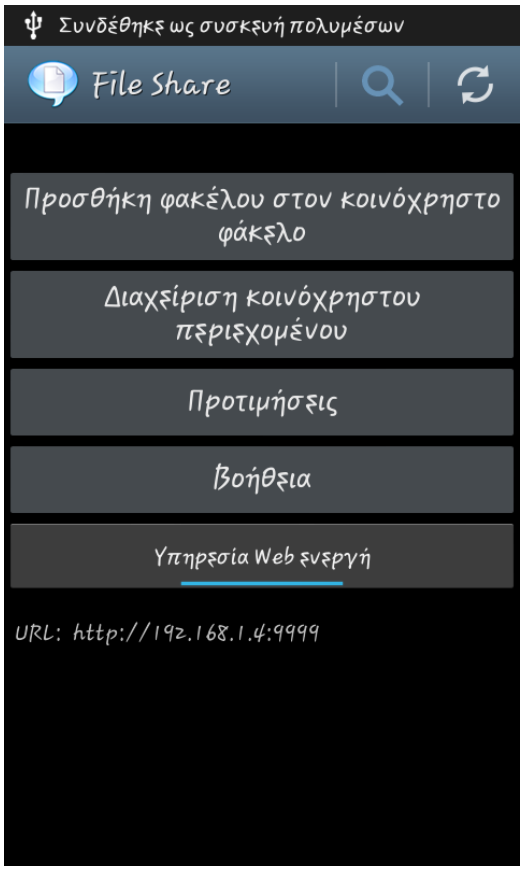
Εικόνα 21. Αποθήκευση αρχείου. Εμφανίζει πού έχει αποθηκευθεί το αρχείο τοπικά.



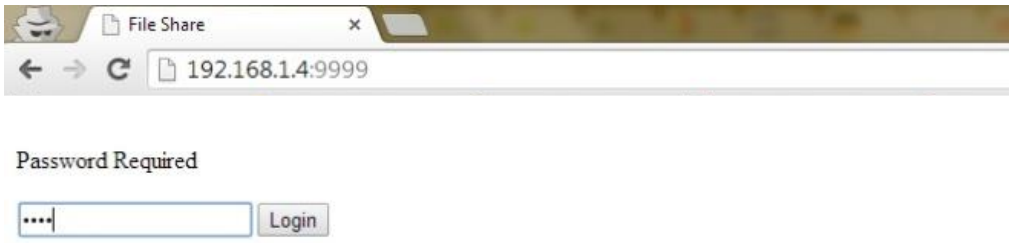
Εικόνα 22. Οι Προτιμήσεις.



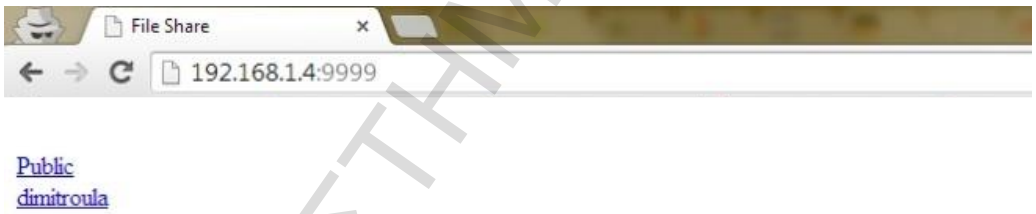
Εικόνα 23. Η Βοήθεια της εφαρμογής.



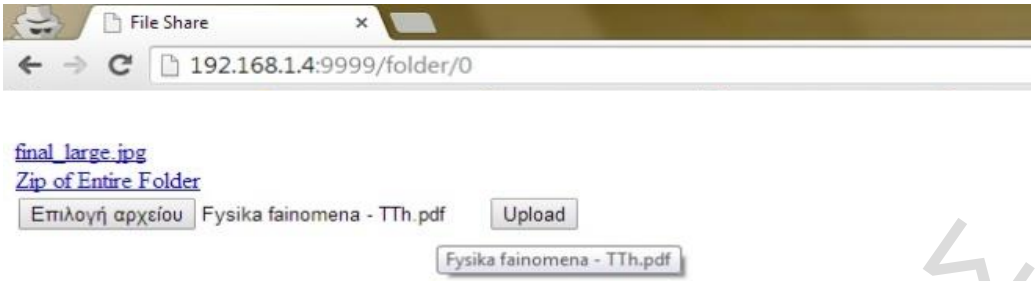
Εικόνα 22. Λειτουργία στα Ελληνικά.



Εικόνα 24. Πληκτρολογώντας την IP που δίνεται στην εφαρμογή, μπαίνουμε στην ιστοσελίδα της. Εφόσον έχει επιλεχθεί κωδικός πρόσβασης, εισάγεται σε αυτό το σημείο.



Εικόνα 23. Οι φάκελοι που υπάρχουν στην εφαρμογή.



Εικόνα 25. Εισαγωγή αρχείου από τον υπολογιστή. Εμφανίζονται όλα τα αρχεία που υπάρχουν στον τρέχοντα φάκελο και η δυνατότητα να αποθηκευτούν τα αρχεία σε μορφή συμπιεσμένου φακέλου.

4.1.1. Apache Software License

Είναι μία ελεύθερη και δωρεάν (free) άδεια λογισμικού για ελεύθερο λογισμικό / λογισμικό ανοιχτού κώδικα (ΕΛ / ΛΑΚ), που έχει γραφτεί από το Ίδρυμα Λογισμικού Apache (Apache Software Foundation, ASF). Η άδεια αυτή απαιτεί τη διατήρηση των δικαιωμάτων πνευματικής ιδιοκτησίας και δήλωση ορισμού των δικαιωμάτων και των υποχρεώσεων ανάμεσα στα συμβαλλόμενα μέλη (αποποίηση ευθύνης). Οι δωρεάν ή/και ελεύθερες άδειες λογισμικού επιτρέπουν στο χρήστη του λογισμικού να το χρησιμοποιήσει για οποιοδήποτε σκοπό, να το διανείμει, να το τροποποιήσει, και τέλος να αναδιανείμει τις τροποποιημένες εκδόσεις του λογισμικού υπό τους όρους της άδειας, χωρίς να πρέπει να λάβει υπόψη του άλλα δικαιώματα. Βεβαίως, κυκλοφορούν αρκετές διαφορετικές άδειες με διαφορετικούς όρους, οι οποίοι όμως κινούνται σε παρόμοια πλαίσια. Ο πηγαίος κώδικας έχει αναπτυχθεί και διανεμηθεί κάνοντας χρήση άδειας Apache, η οποία επιτρέπει στα τροποποιημένα μέρη να χρησιμοποιήσουν άλλη άδεια, αλλά στα μέρη που δεν έχουν αλλαχθεί πρέπει να

παραμείνει η ίδια. Όπου έχουν υπάρξει αλλαγές πρέπει να σημειωθούν μέσα στον κώδικα. Μέσα στον πηγαίο κώδικα της εφαρμογής οι αλλαγές αναφέρονται μέσα σε σχόλια.

Παρόλο που υπάρχουν αρκετές άδειες, η Apache Software License v2.0 προτιμάται για προϊόντα της πλατφόρμας Android. Μάλιστα, η πλειονότητα του λογισμικού για την πλατφόρμα χρησιμοποιεί αυτήν την άδεια. Το κείμενο της άδειας Apache βρίσκεται στην ηλεκτρονική διεύθυνση www.apache.org/licenses/LICENSE-2.0.html. [8] [19]

4.2. Εισαγωγή των αρχείων

Ο πηγαίος κώδικας της εφαρμογής βρίσκεται στον ιστότοπο https://code.google.com/p/android-fileshare/ . Αναπτύχθηκε σε περιβάλλον Eclipse IDE χρησιμοποιώντας το πρόσθετο ADT για την πλατφόρμα Android και σε λειτουργικά συστήματα Windows 7 / 8 και Linux Ubuntu 14.04.

Για την εκτέλεση του πηγαίου κώδικα φτιάχτηκε ένα καινούριο project στο Eclipse με ονομασία πακέτου `com.dimitra.dipl.fileshare.` Μετά το κατέβασμα των αρχείων της εφαρμογής, πρέπει να προστεθεί στο path το αρχείο commons-fileupload-1.3.jar , το οποίο κατέβασα από τον ιστότοπο http://commons.apache.org/proper/commons-fileupload/ . Αφού κατέβει, αποσυμπιέστηκε και προστέθηκε στο path της εφαρμογής, επιλέγοντας το project και Build Path > Configure Build Path. Ανοίγει το παράθυρο διαλόγου που φαίνεται στην παρακάτω εικόνα, και επιλέγω να προσθέσω τη διαδρομή για το jar αρχείο και για το bin, έτσι ώστε να μπορέσω να το εισάγω. Πατώντας Add External Jars... στην καρτέλα Libraries, εισάγω αυτές τις δύο διαδρομές.

4.2.1. Οι Προθέσεις

Σε προηγούμενη παράγραφο περιγράφηκαν τα Φίλτρα Προθέσεως που υπάρχουν στο αρχείο AndroidManifest.xml της εφαρμογής. Εδώ θα αναλυθούν οι Προθέσεις. Οι Προθέσεις είναι μία δομή δεδομένων η οποία εξυπηρετεί τουλάχιστον δύο σκοπούς: προσδιορίζει μία ενέργεια που θα εκτελεστεί και συμβολίζει ένα γεγονός του συστήματος το οποίο έχει συμβεί και θέλουμε να ενημερωθούν άλλα συστατικά Android για αυτό. Για παράδειγμα, οι Προθέσεις χρησιμοποιούνται για να ξεκινήσει μία Δραστηριότητα, έστω η εφαρμογή Τηλέφωνο. Για να λειτουργήσει, πρέπει να χρησιμοποιήσει δεδομένα από την εφαρμογή Επαφές. Άρα, θέλει να επιλέξει μία Επαφή. Η Πρόθεση περιγράφει ακριβώς αυτή τη θέληση.

Μία Πρόθεση δημιουργείται από ένα συστατικό το οποίο θέλει να κάνει κάποια εργασία. Λαμβάνεται από κάποια Δραστηριότητα ή όποιο άλλο συστατικό που μπορεί να διεκπεραιώσει αυτήν την εργασία.

Υπάρχουν διάφορα πεδία Προθέσεως, τα οποία περιγράφουν τον τύπο της πληροφορίας που περιέχει η Πρόθεση, Αυτά είναι:

- το πεδίο ενέργεια (action field). Ένα string που προσδιορίζει ή ονομάζει την ενέργεια που θα γίνει. Παραδείγματα: ACTION_MAIN για να ξεκινήσει την Activity ως την αρχική της εφαρμογής ή ως ένα από τα πιθανά σημεία εισόδου στην εφαρμογή και δε λαμβάνει δεδομένα, ACTION_EDIT για ενέργεια που σχετίζεται με την επεξεργασία δεδομένων, ACTION_DIAL για κλήση ενός αριθμού όπως ορίζεται από τα δεδομένα, ACTION_SYNC για συγχρονισμό δεδομένων.

1. το πεδίο δεδομένα (data field). Είναι δεδομένα που σχετίζονται με την Πρόθεση, τα δεδομένα στα οποία θα ενεργήσει η Πρόθεση, και είναι μορφοποιημένα ως URI. Για παράδειγμα, τα στοιχεία ενός ατόμου στη βάση δεδομένων της επαφής.
2. το πεδίο κατηγορία (category field). Είναι επιπρόσθετες πληροφορίες για το συστατικό που θα διαχειριστεί την Πρόθεση. Συνήθως οι Προθέσεις δεν απαιτούν το πεδίο κατηγορία. Κάποιες από τις πιο κοινές κατηγορίες είναι οι: CATEGORY_BROWSABLE, όπου η Δραστηριότητα που στοχεύει θα εκκινήσει χρησιμοποιώντας μία εφαρμογή φυλλομετρητή διαδικτύου (web browser) και δείχνει δεδομένα μέσω κάποιου συνδέσμου, και CATEGORY_LAUNCHER, που όπως έχει ήδη περιγραφεί, η Δραστηριότητα με την οποία σχετίζεται είναι η αρχική ενός task και είναι διαθέσιμη κατά την εκκίνηση της εφαρμογής και βρίσκεται στον Εκκινητή (Launcher).
3. το πεδίο τύπος (type field). Ορίζει τον τύπο MIME των data της Πρόθεσης. Αυτό το URI αντικείμενο είναι η αναφορά στα δεδομένα και μπορεί να αναφέρεται ως MIME type, ή Internet Media Type, ή και Content-type. Είναι ένα αναγνωριστικό που χρησιμοποιείται στο Internet για να υποδείξει τον τύπο δεδομένων που περιέχει ένα αρχείο, μία προδιαγραφή για τη μορφοποίηση μηνυμάτων από μη ASCII χαρακτήρες, έτσι ώστε να μπορούν να αποσταλούν μέσω Internet. Αρχικά είχε δημιουργηθεί για μηνύματα ηλεκτρονικού ταχυδρομείου που χρησιμοποιούσαν το πρωτόκολλο SMTP και τα αρχικά του σήμαιναν Multipurpose Internet Mail Extensions, και ταυτοποιούσε αρχεία ανάλογα με το format τους, ενώ σήμερα χρησιμοποιείται και με άλλα και έχει

αλλάξει η αρχική του ονομασία. Έχει τη μορφή type / subtype. Παραδείγματα MIME τύπων: διάφορα πρότυπα εικόνας (image/*, image/png, image/jpg), διάφορα πρότυπα κειμένου (text/html, text/plain). Αυτή η μορφή σημαίνει ότι ένα query με αυτό το URI θα επιστρέψει ένα αντικείμενο που ανταποκρίνεται στο type και θα περιέχει στοιχεία που ανταποκρίνονται στο subtype. Εάν δεν έχει οριστεί ο τύπος, το Android θα το συμπεράνει αυτόματα. Εκτός από πρότυπες μορφές mime type, το Android μπορεί να χρησιμοποιήσει και ειδικές μορφές, όπως αυτή που παρατίθεται παρακάτω.

Μέσα στην AndroidManifest.xml της εφαρμογής βρίσκονται οι εξής γραμμές κώδικα:

```
<data android: mimeType="vnd.android.cursor.item/
vnd.dimitra.dipl.sharedfolder" />
```

Αφορούν την ονομασία του MIME. Είναι πιο περίπλοκα από τα πρότυπα mimes που παρέχονται. Το type τους έχει τη μορφή vnd.android.cursor.dir ή vnd.android.cursor.item ανάλογα με το αν αφορούν πολλά αντικείμενα ή ένα. Το subtype αλλάζει και μπορεί να χρησιμοποιηθεί κάποιο ήδη ορισμένο, όπως `vnd.android.cursor.item/phone_v2`, που θα δημιουργήσει ένα αντικείμενο για τηλεφωνικό νούμερο, ή μπορεί ο δημιουργός να ορίσει δικό του subtype. Το vnd. σχετίζεται με προϊόντα τα οποία είναι διαθέσιμα στο ευρύ κοινό και προέρχεται από τη λέξη vendor, η οποία σε αυτήν την περίπτωση θεωρείται ίσης σημασίας με τη λέξη producer. Οποιοσδήποτε που επιθυμεί να ανταλλάξει αρχεία σχετικά με κάποιο προϊόν λογισμικού μπορεί να το δηλώσει με αυτόν τον τρόπο. Η δήλωση αυτή ανήκει στον "πωλητή" ή "παραγωγό" και οποιαδήποτε στιγμή μπορεί να επιλέξει να ασκήσει δικαιώματα ιδιοκτησίας. Συγκεκριμένα για το Android, το χαρακτηριστικό mimeType ορίζεται ως [application]/vnd.[android.package-archive] και χρησιμοποιείται στο κατέβασμα του .apk αρχείου.

Είναι συχνά σημαντικό να ορίζεται ο τύπος δεδομένων (MIME type) μαζί με το URI. Έτσι, ακόμα και εάν το URI μίας Δραστηριότητας που παίζει κάποιο αρχείο ήχου πιθανόν να μην μπορεί να δείξει αρχεία εικόνας, ορίζοντας τον MIME τύπο των δεδομένων, βοηθάμε το Android να βρει το κατάλληλο συστατικό που θα λαμβάνει την Πρόθεση και να παρουσιάσει τα δεδομένα στο χρήστη με το κατάλληλο format.

4. το πεδίο extras (extras field). Επιπρόσθετες πληροφορίες. Χρήσιμες για παροχή εκτεταμένων πληροφοριών στο συστατικό. Για παράδειγμα, για αποστολή

μηνυμάτων ηλεκτρονικού ταχυδρομείου γίνεται χρήση της ενέργειας ACTION_SEND και του extra EXTRA_SUBJECT για εισαγωγή θέματος στο μήνυμα.

5. το πεδίο σημαία (flags field). Καθορίζει πώς διαχειρίζεται η Πρόθεση. Παραδείγματα: FLAG_ACTIVITY_NO_HISTORY, έτσι ώστε να μην μπει η Δραστηριότητα στη στοίβα ιστορικού, GLAG_DEBUG_LOG _RESOLUTION, το οποίο λέει στο Android να εμφανίζει περισσότερες πληροφορίες όταν γίνεται η επεξεργασία της Πρόθεσης.
6. το πεδίο συστατικό (component field). Ορίζει το συστατικό και την κλάση που θα χρησιμοποιηθεί για την Πρόθεση. Συνήθως καθορίζεται σε σχέση με τις υπόλοιπες πληροφορίες της Πρόθεσης, δηλαδή τα πεδία action, data / type και category, τα οποία τα συνδυάζει με κάποιο συστατικό που μπορεί να διαχειριστεί το Intent. Ορίζοντας αυτό το χαρακτηριστικό, όλα τα υπόλοιπα πεδία ορίζονται προαιρετικά. Χρησιμοποιείται όταν υπάρχει μόνο ένα συστατικό που μπορεί να λάβει την Πρόθεση.

Τρία από αυτά τα πεδία χρησιμοποιούνται για να συνδέσουν την Πρόθεση με το κατάλληλο συστατικό: η ενέργεια, ο τύπος και η κατηγορία. Βάσει αυτών των πληροφοριών γίνεται αναζήτηση (query) στον Package Manager για το συστατικό το οποίο είναι σε θέση να διαχειριστεί την Πρόθεση.

Το Android πρέπει να γνωρίζει τη Δραστηριότητα που στοχεύει η Πρόθεση. Αυτό γίνεται είτε με άμεσο τρόπο, ορίζοντας το συστατικό που στοχεύει, είτε εμμέσως, βασιζόμενο στις Προθέσεις που χρησιμοποιούνται και στις Δραστηριότητες που είναι εγκατεστημένες στη συσκευή.

Παρακάτω παρατίθεται ένα κομμάτι κώδικα της εφαρμογής που περιέχει ένα αντικείμενο Πρόθεσης.

Fileshare.java

```
private final View.OnClickListener mAddFileListener = new
View.OnClickListener() {
    public void onClick(View v) {
      Intent pickFileIntent = new Intent();
      pickFileIntent.setAction(Intent.ACTION_GET_CONTENT);
```

```
        pickFileIntent.addCategory(Intent.CATEGORY_OPENABLE);
        pickFileIntent.setType("*/*");
        Intent chooserIntent = Intent.createChooser(pickFileIntent,
            getText(R.string.choosefile_title));
        startActivityForResult(chooserIntent, PICK_FILE_REQUEST);
    }
};
```

Ορίζει για την Πρόθεση `pickFileIntent` τα πεδία ενέργεια, κατηγορία και τύπος.

4.2.2. Το αρχείο AndroidManifest.xml

Παρακάτω παρατίθεται όλο το αρχείο που υπάρχει στην εφαρμογή:

AndroidManifest.xml

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

<manifest
xmlns:android="http://schemas.android.com/apk/res/android"

      package="com.dimitra.dipl.fileshare"

      android:versionCode="4" android:versionName="1.0c Beta">

   <uses-sdk

         android:minSdkVersion="11"

         android:targetSdkVersion="18"/>

        <uses-permission
       android:name="android.permission.INTERNET"/>

         <uses-permission
```

```
android:name="android.permission.ACCESS_WIFI_STATE"/>

<application android:icon="@drawable/ic_launcher"

             android:label="@string/app_name" >

    <activity android:name="FileShare"

              android:label="@string/app_name" >

          <intent-filter>

           <action android:name="android.intent.action.MAIN"

           />

           <category

           android:name="android.intent.category.

           LAUNCHER" />

          </intent-filter>

    </activity>

    <activity android:name="FileBrowser"

              android:label="FileBrowser">

          <intent-filter>

           <action android:name="android.intent.action.

               GET_CONTENT" />

           <action android:name="android.intent.action.VIEW"
```

```
                />
                <category
                android:name="android.intent.category.DEFAULT" />
                <category
                android:name="android.intent.category.
                OPENABLE" />
                <data android:mimeType=
                "vnd.android.cursor.item/
                vnd.dimitra.dipl.file" />
                <data android:mimeType=
                "vnd.android.cursor.dir/
                vnd.dimitra.dipl.file" />
            </intent-filter>
        </activity>

        <activity android:name="SharedFolderBrowser"
                android:label="Folder Browser">
            <intent-filter>
             <category android:name=
```

```
            "android.intent.category.DEFAULT" />
            <action android:name="android.intent.action.PICK"
            />
            <data android:mimeType=
            "vnd.android.cursor.item/
            vnd.dimitra.dipl.sharedfolder" />
          </intent-filter>
          <intent-filter>
            <action android:name="android.intent.action.MAIN"
            />
            <category android:name=
                 "android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.dir/
            vnd.dimitra.dipl.sharedfolder" />
          </intent-filter>
        </activity>
        <activity android:name="SharedFolderDeleter"
                 android:label="Delete Folder">
          <intent-filter>
```

```
            <category
            android:name="android.intent.category.
            DEFAULT" />
            <action
            android:name="android.intent.action.
            DELETE" />
            <data android:mimeType="vnd.android.cursor.item/
                  vnd.dimitra.dipl.sharedfolder" />
        </intent-filter>
    </activity>
    <activity android:name="SharedFolderCreator"
            android:label="Create Folder">
        <intent-filter>
        <category
            android:name="android.intent.category.
            DEFAULT" />
        <action
        android:name="android.intent.action.INSERT" />
        <data android:mimeType=
```

```
            "vnd.android.cursor.dir/
            vnd.dimitra.dipl.sharedfolder" />
            </intent-filter>
        </activity>
        <activity android:name="SharedFileBrowser"
                android:label="File Browser">
            <intent-filter>
            <category
            android:name="android.intent.category.DEFAULT" />
            <action android:name="android.intent.action.VIEW"
            />
            <data android:mimeType="vnd.android.cursor.item/
                    vnd.dimitra.dipl.sharedfolder" />
            </intent-filter>
        </activity>
        <activity android:name="Help" android:label="Help">
            <intent-filter>
            <category
            android:name="android.intent.category.DEFAULT" />
```

```
            <action
            android:name="android.intent.action.MAIN"/>

            </intent-filter>

        </activity>

        <provider android:name="FileProvider"

                  android:authorities=

                  "com.dimitra.dipl.filesharer.FileProvider"

                  android:exported="true" />

        <provider android:name="FileSharingProvider"

            android:authorities=

            "com.dimitra.dipl.filesharer.FileSharingProvider"

            android:exported="true" />

        <service android:enabled="true"

                  android:permission="android.permission.INTERN
                  ET"

                  android:name="FileSharingService"

                  android:exported="true">

            <intent-filter>

            <action android:name=
```

```
                "com.dimitra.dipl.filesharer.

                IFileSharingService"/>

                </intent-filter>

            </service>

        </application>



        <uses-permission

            android:name="android.permission.

            WRITE_EXTERNAL_STORAGE">

        </uses-permission>

</manifest>
```

Έχει ήδη περιγραφεί τι ορίζει το αρχείο AndroidManifest.xml σε μία εφαρμογή. Για την προκειμένη ορίζει:

- το όνομα του πακέτου, εδώ `com.dimitra.dipl.fileshare`. Κάθε εφαρμογή πρέπει να έχει το δικό της μοναδικό όνομα πακέτου και συνήθως είναι το domain name αντίστροφα και το όνομα της εφαρμογής.
- την ελάχιστη έκδοση (`API level`) που απαιτείται (`minSdkVersion="11"`), καθώς και αυτήν στην οποία στοχεύει (`android:targetSdkVersion="18"`). Οι συγκεκριμένες αναφορές αφορούν τις εκδόσεις Android 3.0.x Honeycomb και Android 4.3 Jellybean, αντίστοιχα.

- τα permissions (άδειες) που πρέπει να έχει η εφαρμογή για την ομαλή λειτουργία της. Κατά την εγκατάσταση και μόνο τότε, ο χρήστης αποφασίζει αν θα δοθούν. Εάν όχι, η εφαρμογή δεν εγκαθίσταται. Περιγράφονται μέσα στις ετικέτες `<uses-permission> </uses-permission>`. Αφορούν τις άδειες για να επιτρέπεται στην εφαρμογή να ανοίξει θύρες (sockets) για πρόσβαση σε δίκτυο (`android.permission.INTERNET`), να έχει πρόσβαση σε πληροφορίες για δίκτυα Wi-Fi (`android.permission.ACCESS_WIFI_STATE`), και να αποθηκεύσει δεδομένα σε φακέλους που θα φτιάξει στον εξωτερικό αποθηκευτικό χώρο (`android.permission.WRITE_EXTERNAL_STORAGE`). Το όνομα της άδειας είναι είτε μία άδεια που ορίζεται από την εφαρμογή, μία άδεια που ορίστηκε από άλλη εφαρμογή ή κάποια από τις ήδη ορισμένες από το σύστημα, όπως αυτές που χρησιμοποιούνται εδώ.

- μέσα σε ετικέτες `<activity>` ορίζει εκτός της main activity και όσες κλάσεις ακόμα είναι υποκλάσεις της Activity ή της ListActivity. Αυτές είναι οι: FileShare (main activity), FileBrowser, Help, SharedFileBrowser, SharedFolderBrowser, SharedFolderCreator, και SharedFolderDeleter.

- μέσα σε ετικέτα `<service>` ορίζει την Υπηρεσία FileSharingService και μέσα σε ετικέτα `<provider>` τους Παρόχους Περιεχομένου FileProvider και FileSharingProvider.

- μέσα σε κάθε ορισμό Δραστηριότητας και στον ορισμό της Υπηρεσίας, ορίζει τα Φίλτρα Προθέσεως (Intent Filters). Είναι οι τύποι Προθέσεως στους οποίους τα συστατικά μπορούν να αποκριθούν. Δηλώνει τις δυνατότητες του συστατικού που το περιέχει, δηλαδή τι μπορούν να κάνουν μία Δραστηριότητα ή μία Υπηρεσία ή τι τύπους μπορεί να χειριστεί ένας Δέκτης Εκπεμπόμενων Προθέσεων. Ανάλογα με το τι έχει οριστεί, "φιλτράρει" τις Προθέσεις σε αυτές τις οποίες δέχονται και διαχειρίζονται τα συστατικά ή όχι. Από όλα τα συστατικά, δεν μπορεί να οριστεί μόνο σε έναν Πάροχο Περιεχομένου. Έτσι, υπάρχουν:
    - η ενέργεια (action) MAIN. Δηλώνει ότι το στοιχείο μέσα στο οποίο ορίζεται είναι το κύριο σημείο για να ξεκινήσει η εφαρμογή και δεν περιμένει κανένα δεδομένο Προθέσεως.

- η κατηγορία (category) LAUNCHER. Δηλώνει ότι πρέπει να είναι διαθέσιμη κατά την εκκίνηση του Android συστήματος.
- η ενέργεια GET_CONTENT. Θα επιστρέψει στο στοιχείο που την καλεί δεδομένα που έχουν επιλεγεί από το χρήστη.
- η ενέργεια VIEW. Παρουσιάζει δεδομένα στο χρήστη.
- η κατηγορία DEFAULT. Με τον ορισμό ως DEFAULT, επιτρέπεται να δέχεται έμμεσες Προθέσεις, δηλαδή Προθέσεις οι οποίες δεν ορίζουν ένα συγκεκριμένο συστατικό δηλώνοντας το όνομα της κλάσης του, αλλά μία γενική ενέργεια όπου ένα συστατικό από μία άλλη εφαρμογή θα διαχειριστεί.
- η κατηγορία OPENABLE, για να επιστρέφονται αρχεία τα οποία θα αναπαρασταθούν ως file stream.
- τα δεδομένα `mimeType="vnd.android.cursor.item/vnd.dimitra.dipl.file"`. Αφορούν τον τύπο των δεδομένων της Πρόθεσης. Εάν δεν οριστεί mime type, ο τύπος αυτός θα παράγεται από τα ίδια τα δεδομένα, ενώ δηλώνοντάς το, επιλέγεται συγκεκριμένος τύπος. Το *vnd.android.cursor.item/vnd.dimitra.dipl.file* είναι ένα URI όπου ένα Cursor που περιλαμβάνει ακριβώς ένα αντικείμενο (*vnd.android.cursor.item*) το οποίο μπορεί να ανακτηθεί και περιλαμβάνει ένα αρχείο της εφαρμογής (item). Μαζί με την ενέργεια GET_CONTENT θα επιστρέψει δεδομένα στο στοιχείο που την καλεί τα οποία έχουν επιλεχθεί από το χρήστη, αλλά το στοιχείο αυτό θα έχει επιλέξει τον τύπο των δεδομένων που επιθυμεί. Το Cursor αφορά έναν πίνακα που αναπαριστά τη Βάση δεδομένων. Όταν επιλέγονται δεδομένα που πρέπει να επιστραφούν, τότε η βάση δημιουργεί ένα αντικείμενο Cursor και επιστρέφει την αναφορά.
- τα δεδομένα `mimeType="vnd.android.cursor.dir/vnd.dimitra.dipl.file`, που διαφέρουν με το προηγούμενο στο ότι είναι ένα URI όπου ένα Cursor περιλαμβάνει μηδέν ή περισσότερα αντικείμενα (directory).

- άλλα mimeType είναι το *vnd.android.cursor.item/vnd.dimitra.dipl.sharedfolder* στις activities SharedFolderBrowser, SharedFolderDeleter και SharedFileBrowser, και το *vnd.android.cursor.dir/vnd.dimitra.dipl.sharedfolder* στην activity SharedFolderCreator.
- πιο κάτω βρίσκεται η ενέργεια PICK. Ο χρήστης διαλέγει κάποιο από τα δεδομένα. Αυτή η πληροφορία (το URI) επιστρέφεται.
- η ενέργεια DELETE. Η πληροφορία που έχει επιλεχθεί διαγράφεται.
- η ενέργεια INSERT. Στη Δραστηριότητα που έχει ανατεθεί, εισάγει ένα κενό στοιχείο. Εδώ δημιουργεί / εισάγει ένα καινούριο κενό φάκελο. Επιστρέφει το URI του καινούριου δεδομένου.

Παράδειγμα Intent που υπάρχουν μέσα στον πηγαίο κώδικα και έχουν δηλωθεί στην AndroidManifest.xml:

SharedFolderBrowser.java

```
protected void onListItemClick(ListView l, View v, int position,
        long id) {
    if (getIntent().getAction().equals(Intent.ACTION_PICK))  {
      Uri uri = Uri.withAppendedPath(FileSharingProvider.Folders
          .CONTENT_URI, "" + id);
      Log.d(TAG, "Uri for folder = " + uri.toString());
      setResult(RESULT_OK, new Intent().setData(uri));
      finish();
    } else {
      /* show files in the folder */
      Uri uri = Uri.withAppendedPath(FileSharingProvider.Folders
          .CONTENT_URI, "" + id);
      Intent intent = new Intent();
      intent.setData(uri);
      intent.setAction(Intent.ACTION_VIEW);
```

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

Το σημείο που δηλώνεται το Φίλτρο Προθέσεως για τη συγκεκριμένη κλάση μέσα στη manifest είναι, όπου ορίζεται ως έμμεση Πρόθεση:

AndroidManifest.xml

```
<activity android:name="SharedFolderBrowser"
          android:label="Folder Browser">
          <intent-filter>
            <category
android:name="android.intent.category.DEFAULT" />
           <action android:name="android.intent.action.PICK" />
           <data
android:mimeType="vnd.android.cursor.item/vnd.dimitra
                                .dipl.sharedfolder" />
        </intent-filter>
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category
android:name="android.intent.category.DEFAULT" />
           <data
android:mimeType="vnd.android.cursor.dir/vnd.dimitra
                                .dipl.sharedfolder" />
       </intent-filter>
</activity>
```

- οι ετικέτες <provider> δηλώνουν κλάσεις Παρόχου Περιεχομένου, που είναι υποκλάσεις της κλάσης ContentProvider, η οποία παρέχει δομημένη πρόσβαση σε δεδομένα που διαχειρίζεται η εφαρμογή. Όπως οι Δραστηριότητες πρέπει απαραίτητα

να δηλώνονται μέσα στο αρχείο AndoidManifest.xml με την ετικέτα `<activity>`, έτσι δηλώνονται και όλοι οι Πάροχοι που υπάρχουν στην εφαρμογή. Αλλιώς, το σύστημα δεν γνωρίζει πως υπάρχουν και δεν τους εκτελεί. Πάροχοι Περιεχομένου που χρησιμοποιούνται από την εφαρμογή, αλλά είναι μέρος του πηγαίου κώδικα μίας άλλης εφαρμογής δεν πρέπει να δηλώνονται. Ορίζονται ως Πάροχοι οι κλάσεις FileProvider και FileSharingProvider, οι οποίες είναι υποκλάσεις της κλάσης ContentProvider. Όπως φαίνεται και στα αντίστοιχα αρχεία java:

```
public class FileProvider extends ContentProvider
```

```
public class FileSharingProvider extends ContentProvider
```

Σε κάθε μία από αυτές τις δηλώσεις για τους Παρόχους εμφανίζονται τα εξής χαρακτηριστικά:

- το όνομα του Παρόχου (`name`),
- το χαρακτηριστικό `android:authorities`, μία λίστα από ένα ή περισσότερα URIs τα οποία αποδίδουν δεδομένα που παρέχονται από τον Πάροχο. Πρέπει να είναι μοναδικό, εφόσον χρησιμοποιείται για να συνδέεται κάθε κλάση Παρόχου με αυτό που δηλώνεται στη manifest, έτσι ώστε η εφαρμογή αλλά και άλλες εφαρμογές που πιθανόν να έχουν πρόσβαση στη δική μας μπορούν να χειρίζονται δεδομένα μέσω του Παρόχου Περιεχομένου με τη μορφή ενός URI, οπότε χρησιμοποιείται και εδώ η τακτική της ανάποδης αναφοράς στο domain name του δημιουργού και της υποκλάσης του ContentProvider που υλοποιεί τον πάροχο (εδώ `com.dimitra.dipl.filesharer.FileProvider` και `com.dimitra.dipl.filesharer.FileSharingProvider`).

  Προσπελάσσονται παρόμοια με τα http urls, δηλαδή content://<authority-name>/<data-in-the-provider>, οπότε εδώ έχουμε σε κάθε μία από τις κλάσεις:

  - στο FileSharingProvider.java: `public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/folders")` και `public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/files")`.

- Στο FileProvider.java: `public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/files").`

Σε κάθε ένα από αυτά τα αρχεία δηλώνονται και τα mimeTypes που θα χρησιμοποιηθούν για να επιστραφούν τα δεδομένα. Έτσι, παρατηρούμε στο αρχείο FileSharingProvider.java:

FileSharingProvider.java

```
public interface Folders {

    public static final Uri CONTENT_URI = Uri.parse("content://" +
AUTHORITY + "/folders");

    /* #Dimitra# */
    public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.dimitra.dipl.sharedfolder";
    public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.dimitra.dipl.sharedfolder";
...
}

  /* File related constants */
public interface Files {
    public static final Uri CONTENT_URI = Uri.parse("content://" +
AUTHORITY + "/files");

    /* #Dimitra# */
    public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.dimitra.dipl.sharedfile";
    public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.dimitra.dipl.sharedfile";
```

```
...
}
```

- Ανάλογα με το αν θέλουμε να επιστραφεί ένα αρχείο (item) ή ένας κατάλογος από αρχεία (dir), και αν αυτό θα είναι ένας φάκελος ή αρχεία ξεχωριστά, έχουμε δύο διεπαφές (Interfaces).
- το χαρακτηριστικό `android:exported` με τιμή `"true"`. Ορίζει εάν ο Πάροχος είναι διαθέσιμος για χρήση σε άλλες εφαρμογές. Εάν του έχει αποδοθεί η τιμή true, τότε είναι διαθέσιμος και κάθε εφαρμογή μπορεί να έχει πρόσβαση χρησιμοποιώντας το URI του Παρόχου για την προσπέλασή του, ενώ υπόκειται στις άδειες που έχουν οριστεί για αυτόν τον Πάροχο. Αντίθετα, αν η τιμή είναι false, ο provider είναι διαθέσιμος μόνο σε εφορμογές που έχουν το ίδιο user ID με την εφαρμογή που περιέχει τον Πάροχο. Η προεπιλεγμένη τιμή είναι true για εφαρμογές που στη manifest τους στα χαρακτηριστικά `android:minSdkVersion` και `android:targetSdkVersion` έχουν αποδώσει την τιμή “16” (Android 4.1 JELLYBEAN) ή χαμηλότερη, ενώ αυτές που έχουν τιμή μεγαλύτερη ή ίση του “17” (Android 4.2 JELLYBEAN MR1) είναι false. Επίσης, είναι δυνατόν να έχει δηλωθεί η τιμή false και μέσω permissions να υπάρχει περιορισμένη πρόσβαση.

- Η ετικέτα `<service>` δηλώνει μία υπηρεσία , που είναι υποκλάση (extends) της κλάσης Service. Οι Υπηρεσίες δεν έχουν γραφική διεπιφάνεια χρήστη, ενώ υλοποιούν εργασίες που τρέχουν παρασκηνιακά χωρίς τη διάδραση με το χρήστη ή παρέχουν λειτουργικότητα που μπορεί να χρησιμοποιηθεί από άλλες εφαρμογές. Κάθε μία από αυτές πρέπει οπωσδήποτε να δηλώνεται μέσα στη manifest. Εάν δεν δηλωθούν, το σύστημα δεν μπορεί να τις αναγνωρίσει και να τις εκτελέσει. Στον πηγαίο κώδικα δηλώνεται η εξής Υπηρεσία:

```
public class FileSharingService extends Service
```

- Καθώς οι Υπηρεσίες τρέχουν στο κύριο νήμα, εάν πρέπει να χρησιμοποιηθούν αρκετοί υπολογιστικοί πόροι, είναι καλή πρακτική η Υπηρεσία να τρέχει στο δικό της νήμα. Στον κώδικα της εφαρμογής υπάρχει:

FileSharingService.java

```
private Thread mWebServerThread;   /* δήλωση του νήματος */

. . .

    mWebServerThread = new Thread() {    /* αρχικοποίηση του
νήματος */
      @Override
      public void run() {
        mWebServer.runWebServer();
      }
    };
    mWebServerThread.start();
. . .

   public void onDestroy() {
        super.onDestroy();
        if (mWebServer != null) {
             mWebServerThread.interrupt();
             try {
                  mWebServerThread.join();
             } catch (InterruptedException e) {

             }
        }
   }
```


Τα χαρακτηριστικά που δηλώνονται στη manifest μέσα στην ετικέτα της Υπηρεσίας είναι:

- Το `android:enabled="true"` για να μπορεί η Υπηρεσία να αρχικοποιηθεί (instantiated) από το σύστημα. Η boolean τιμή true είναι η προεπιλεγμένη. Καθώς είναι ένα χαρακτηριστικό που μπορεί να αποδοθεί και στο στοιχείο `<application>`, το οποίο εφόσον είναι true, εφαρμόζεται σε όλα τα συστατικά, συμπεριλαμβανομένων και των τυχόν Υπηρεσιών, πρέπει απαραιτήτως να δηλώνεται ως true και μέσα στο `<service>` για να είναι δυνατή η αρχικοποίηση της Υπηρεσίας, ενώ εάν έστω και ένα έχει την τιμή false, δεν θα εκτελεστεί η Υπηρεσία.
- Το χαρακτηριστικό `android:permission="android.permission.INTERNET"`. Είναι το όνομα της άδειας που πρέπει να υπάρχει για να αρχίσει μία Υπηρεσία ή να κάνει τη σύνδεση με αυτή. Εάν υπάρχει κάποια από τις μεθόδους `startService()`, `bindService()` ή `stopService()` δεν έχει αυτήν την άδεια, τότε η μέθοδος δε θα λειτουργήσει και το αντικείμενο Πρόθεσης δε θα αποδοθεί στην Υπηρεσία. Εάν δεν έχει οριστεί άδεια μέσα στην ετικέτα `<service>` της manifest, τότε θα χρησιμοποιηθεί η άδεια που έχει οριστεί στο στοιχείο `<application>`, ενώ αν δεν έχει δηλωθεί καμία από αυτές, τότε η Υπηρεσία δεν προστατεύεται από κάποια άδεια.

  Στο παρακάτω απόσπασμα κώδικα βλέπουμε πώς χρησιμοποιείται η `startService()` για την εκκίνηση της Υπηρεσίας, περνώντας ως παράμετρο μία Πρόθεση στην οποία ορίζεται η ενέργεια (`setAction()`), η οποία έχει ως παράμετρο το όνομα του aidl αρχείου.

FileShare.java

```
/* Startup the FileSharingService, unless the last state was off.
*/
final SharedPreferences sharedPreferences = getSharedPreferences(
    FileSharingService.PREFS_NAME, MODE_PRIVATE);
    ToggleButton serviceButton =
          (ToggleButton)findViewById(R.id.service);
    if (sharedPreferences.getBoolean(
         FileSharingService.PREFS_SERVICE_ON_STARTUP, true)) {
```

```
            Intent serviceIntent = new Intent();
            serviceIntent.setAction(
               "com.dimitra.dipl.filesharer.IFileSharingService");
            startService(serviceIntent);
            serviceButton.setChecked(true);
        } else {
          serviceButton.setChecked(false);
        }

        /* Setup toggling the service. */
serviceButton.setOnCheckedChangeListener(new
OnCheckedChangeListener(){
              @Override
          public void onCheckedChanged(CompoundButton arg0,
                boolean newValue) {
                Intent serviceIntent = new Intent();
               serviceIntent.setAction(
                 "com.dimitra.dipl.filesharer.IFileSharingService");
                    if (newValue == true) {
                        startService(serviceIntent);
                    } else {
                        stopService(serviceIntent);
                    }
                     SharedPreferences.Editor editor =
                     sharedPreferences.edit();
                editor.putBoolean(FileSharingService
                    .PREFS_SERVICE_ON_STARTUP, newValue);
                 editor.commit();
          }
});
```

- το χαρακτηριστικό `android:name="FileSharingService"`, που είναι το όνομα της κλάσης που υλοποιεί την Υπηρεσία.
- το `android:exported="true"`. Όπως και στον Πάροχο, δηλώνει εάν άλλες εφαρμογές μπορούν να χρησιμοποιήσουν και να αλληλεπιδράσουν με αυτήν την Υπηρεσία. Η boolean τιμή true σημαίνει ότι μπορούν, ενώ εάν ήταν false, η service είναι διαθέσιμη μόνο σε συστατικά της τρέχουσας εφαρμογής ή σε όσες έχουν το ίδιο user ID. Η προεπιλεγμένη τιμή εδώ εξαρτάται από το εάν υπάρχουν Φίλτρα Προθέσεως. Εάν όχι, μπορεί να κληθεί μόνο με χρήση του ονόματος της κλάσης της και άρα προορίζεται για χρήση μόνο από συστατικά της εφαρμογής που την υλοποιεί, εφόσον δεν είναι δυνατόν να γνωρίζει άλλος το μοναδικό όνομα της κλάσης. Επομένως, η προεπιλεγμένη τιμή θα είναι false, εφόσον δεν υπάρχει κανένα Intent Filter. Εάν όμως έχει οριστεί έστω και ένα, τότε προορίζεται και για χρήση από άλλες εφαρμογές και η προεπιλεγμένη τιμή θα είναι true. Επίσης, είναι δυνατόν μέσω permissions να περιοριστεί η πρόσβαση από εξωτερικά στοιχεία.
- μέσα στην ετικέτα της Υπηρεσίας δηλώνεται και ένα ακόμα Φίλτρο Πρόθεσης:

```
<intent-filter>
      <action android:name="com.dimitra.dipl.filesharer
                           .IFileSharingService"/>
</intent-filter>
```

- Το όνομα που δηλώνεται είναι το όνομα του αρχείου IFileSharingService.java. Είναι το όνομα της ενέργειας που έχει οριστεί στην Πρόθεση όπου χρησιμοποιείται η Υπηρεσία (εμφανίζεται στο προηγούμενο απόσπασμα κώδικα ως παράμετρος της μεθόδου `setAction()`). Σύμφωνα με την τεκμηρίωση για το Android, για ενέργειες που ορίζονται από το δημιουργό της εφαρμογής, είναι καλύτερο να χρησιμοποιείται το όνομα του πακέτου και της κλάσης, έτσι ώστε να είναι μοναδικό χαρακτηριστικό.

4.2.3. Το αρχείο με κατάληξη .aidl

Ανοίγοντας τον πηγαίο κώδικα της εφαρμογής στο φάκελο res/ υπάρχει ένα αρχείο με κατάληξη .aidl. Το αρχείο είναι το εξής:

IFileSharingService.aidl

```
package com.dimitra.dipl.fileshare;

interface IFileSharingService {

      int getPort();

}
```

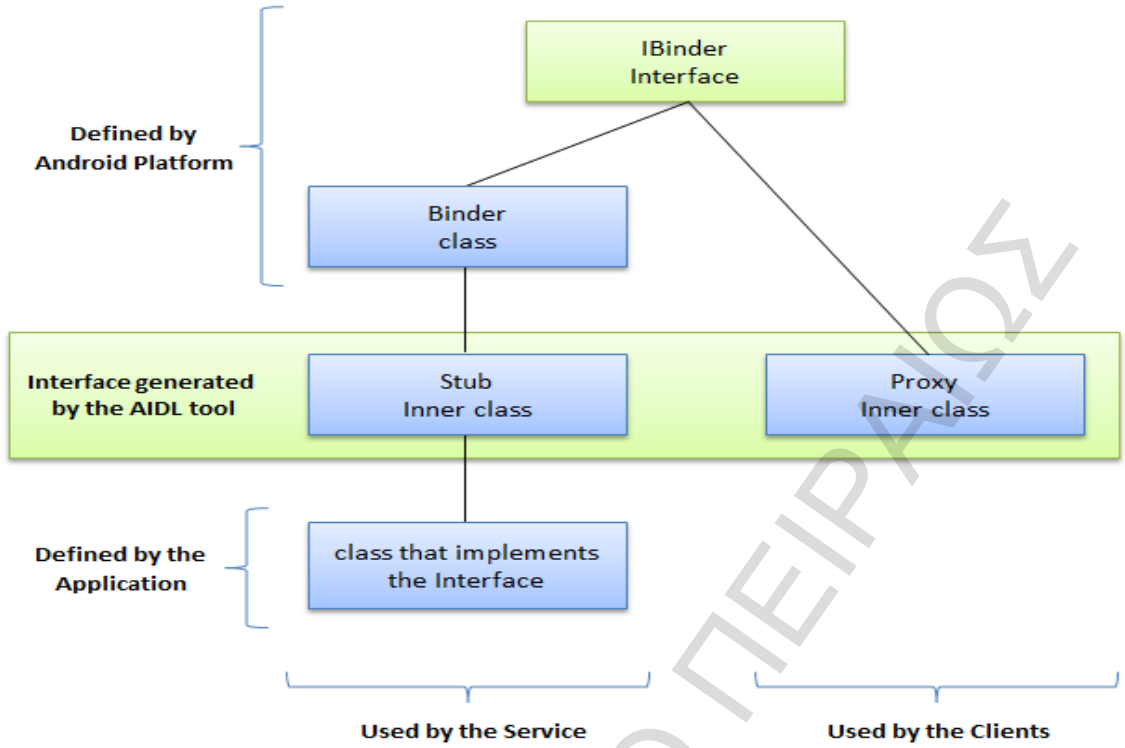
Η κατάληξη του αρχείου προέρχεται από τα αρχικά των λέξεων Android Interface Definition Language, ενώ υπάρχουν και άλλες Interface Definitions Languages (IDLs), με τις οποίες μοιάζει. Μας επιτρέπει να καθορίσουμε τη διεπαφή που ο πελάτης (client) και η υπηρεσία (service) θα χρησιμοποιήσουν, έτσι ώστε να μπορέσουν να επικοινωνήσουν μεταξύ τους χρησιμοποιώντας "επικοινωνία μεταξύ των διαδικασιών" (Inter-Process Communication, IPC). Η IPC είναι ένα σύνολο μεθόδων για την ανταλλαγή δεδομένων ανάμεσα σε πολλαπλά νήματα σε μία ή περισσότερες διαδικασίες (processes), οι οποίες μπορεί να "τρέχουν" σε μία ή περισσότερες συσκευές συνδεδεμένες σε κάποιο δίκτυο. Οι λόγοι για τους οποίους επιλέγουμε αυτή τη συνεργασία των διαδικασιών ποικίλλουν και περιλαμβάνουν τη διαμοίραση πληροφορίας, την υπολογιστική ενίσχυση των συσκευών, τη χρήση "κοινής μνήμης" (shared memory, όπου πολλά προγράμματα ή νήματα έχουν πρόσβαση στην ίδια μνήμη για να μπορούν να επικοινωνούν μεταξύ τους και να αποφεύγεται η χρήση για τις ίδιες λειτουργίες περισσότερης μνήμης), τη διευκόλυνση των χρηστών και άλλα.

Στην πλατφόρμα Android κανονικά μία διαδικασία δεν μπορεί να έχει πρόσβαση στη μνήμη μίας άλλης διαδικασίας για να προστατεύουν τα περιεχόμενά τους από την πρόσβαση από άλλες διαδικασίες. Εάν, όμως, κρίνεται απαραίτητο να υπάρχει επικοινωνία και μεταφορά δεδομένων μεταξύ των διαδικασιών, το Android χρησιμοποιεί Remote Procedure Call (RPC) για την πραγματοποίηση IPC και μέσω της AIDL παράγεται η διεπαφή που θα πραγματοποιηθεί η επικοινωνία. Οπότε, για να γίνει αυτό δυνατό, πρέπει να μετατρέψουν τα

αντικείμενά τους σε αντικείμενα τα οποία μπορεί το λειτουργικό σύστημα να κατανοήσει και να προσπεράσουν αυτό το εμπόδιο. Έπειτα, μπορούν να ανταλλάξουν αυτά τα δεδομένα μέσω μηχανισμών IPC, και συγκεκριμένα μηχανισμούς marshalling και demarshalling, που μετατρέπουν δεδομένα από τη μορφή στην οποία βρίσκονται σε μορφή κατάλληλη για την κατανόησή τους από το Linux σύστημα και τη μεταφορά τους. Επειδή ο κώδικας για τη διαχείριση αυτής της διαδικασίας είναι δύσκολος, σε αυτήν την περίπτωση χρησιμοποιείται η AIDL.

Πρέπει να σημειωθεί ότι η χρήση της AIDL κρίνεται απαραίτητη μόνο όταν επιτρέπουμε σε πελάτες (clients) από άλλες εφαρμογές να έχουν πρόσβαση στην υπηρεσία μας για να υπάρξει επικοινωνία μεταξύ των διαδικασιών (IPC) και θέλουμε να χειριστούμε πολυνηματικές διαδικασίες στην υπηρεσία μας. Εάν θέλουμε IPC, όμως κρίνεται ότι η Υπηρεσία μας δεν χρειάζεται να έχει πρόσβαση ταυτόχρονα και να μη διαχειριστούμε πολυνηματικές διαδικασίες, προτείνεται η χρήση της κλάσης Messenger, η οποία είναι πιο απλή, καθώς εκτελεί όλες τις κλήσεις σε ένα νήμα δημιουργώντας μία ουρά κλήσεων και άρα δε συνεπάγεται πολλαπλά νήματα. Εάν δε χρειάζεται να υπάρχει ταυτόχρονη IPC μεταξύ διαφορετικών εφαρμογών, μπορούμε να δημιουργήσουμε αυτή την διεπιφάνεια επικοινωνίας υλοποιώντας ένα αντικείμενο της κλάσης Binder. Απαραίτητα πριν την υλοποίηση της AIDL πρέπει να είμαστε σε θέση να κατανοήσουμε τις Bound Services, αλλά και να γνωρίζουμε σε ποιο νήμα θα γίνεται η κλήση των μεθόδων. Οι Bound Services είναι η υλοποίηση της κλάσης Service που επιτρέπουν σε άλλες εφαρμογές να συνδέονται και να επικοινωνούν με αυτήν. Αντιστοιχεί με τη μεριά του server σε ένα μοντέλο client – server. Επιτρέπει σε συστατικά, όπως οι Δραστηριότητες, να συνδέονται με την Υπηρεσία, να αποστέλλουν αιτήματα (requests) και να λαμβάνουν αποκρίσεις (responses) και να επικοινωνούν μεταξύ των διαδικασιών (IPC). Σε κάθε περίπτωση, είναι σημαντική η γνώση και η κατανόηση της AIDL, αφού παρέχουν έναν αντικειμενοστραφή IPC χειρισμό για την επικοινωνία των εφαρμογών με τις Bound Services που λειτουργούν σε διαφορετικές διαδικασίες, ενώ σε περιπτώσεις που χρειάζονται περισσότερα του ενός νήματα κρίνεται απαραίτητη η διεπαφή AIDL.

Παρακάτω δίδεται μία απεικόνιση της επικοινωνία μεταξύ των διαδικασιών χρησιμοποιώντας AIDL και πώς λειτουργεί από τη μεριά του client και της Υπηρεσίας (Server):



Εικόνα 26. Πώς λειτουργεί η IPC στο Android χρησιμοποιώντας AIDL.

Πριν να γίνουν κλήσεις σε μία διεπαφή AIDL, πρέπει να είμαστε σίγουροι ποιο νήμα είναι αυτό που πραγματοποιεί την κλήση, καθώς ο τρόπος εκτέλεσής τους θα είναι διαφορετικός ανάλογα με το αν είναι τοπική ή απομακρυσμένη η διεργασία που έχει κάνει την κλήση.

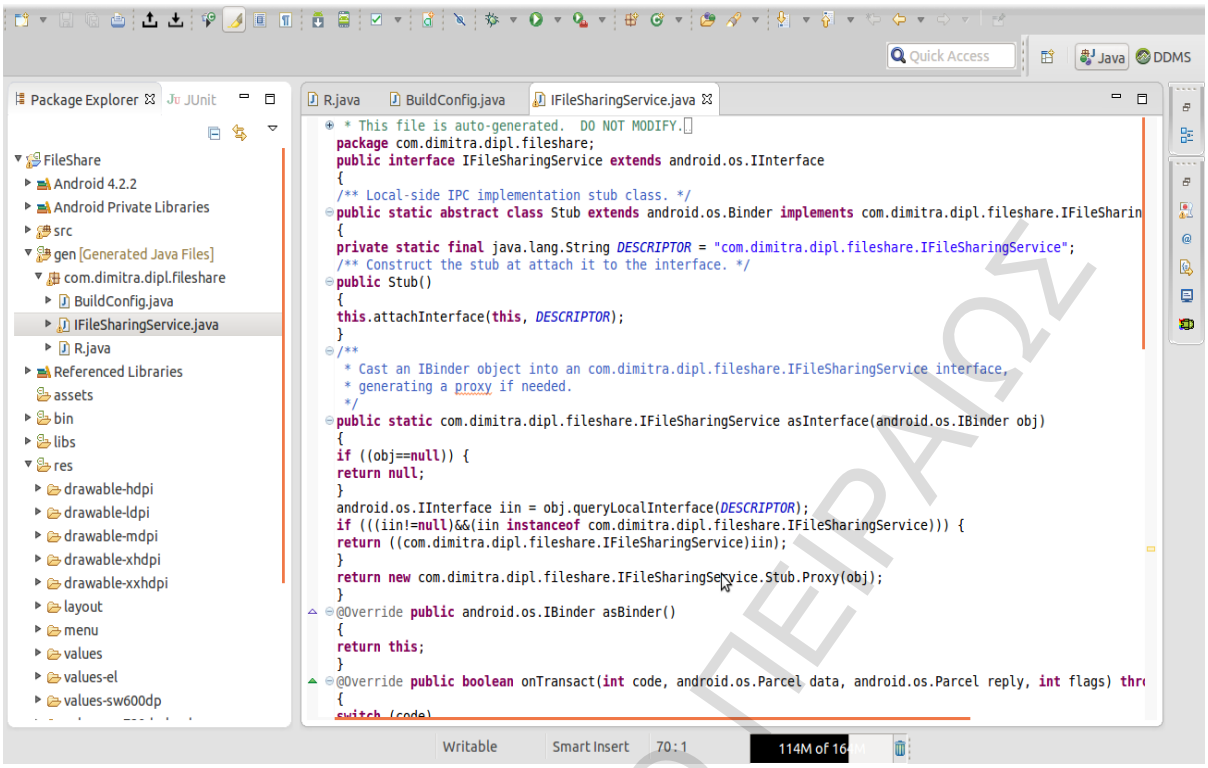
Το συντακτικό που χρησιμοποιείται για την δημιουργία αρχείων AIDL μοιάζει πολύ με αυτό που χρησιμοποιείται για τη δημιουργία interfaces με χρήση της γλώσσας προγραμματισμού Java. Όμως, δεν είναι ακριβώς το ίδιο.

Οι διαφορές είναι ότι δεν υπάρχουν στατικά πεδία (static fields), ενώ οι μη πρωταρχικοί παράμετροι, δηλαδή ό, τι δεν είναι int, float, byte κτλ, πρέπει να χαρακτηρίζονται από κάποια "κατεύθυνση" (direction), οι οποίες είναι οι : in, προεπιλεγμένο και μόνη επιλογή για τους πρωταρχικούς (primitive) τύπους, όπου τα δεδομένα μεταφέρονται από αυτόν που καλεί στο δέκτη, out, όπου επιστρέφονται στον καλούντα, και inout, το οποίο χρησιμοποιείται σπανιότερα και αφορά τη μεταφορά και προς τις δύο μεριές, και κυρίως όταν ο δέκτης αλλάζει αυτήν την παράμετρο. Ενδείκνυται να γίνεται σωστή χρήση αυτών των "κατευθύνσεων", καθώς η μεταφορά παραμέτρων από ένα μέρος του προγράμματος σε ένα άλλο (marshalling parameters) είναι μία σχετικά ακριβή διαδικασία. Είναι σημαντικό να γίνει

αυτή η επεξεργασία, καθώς όταν αρχίζει η μεταφορά δεδομένων, πρέπει να μετριαστεί η ποσότητα αυτών. Η χρήση κατευθύνσεων είναι χαρακτηριστικό και άλλων IDLs. Άλλη διαφορά είναι ότι μπορούν οι μέθοδοι και οι AIDL διεπαφές να οριστούν ως "oneway", το οποίο τροποποιεί τη συμπεριφορά των απομακρυσμένων κλήσεων και όπου δεν εμποδίζεται αυτός που καλεί, απλώς στέλνονται τα δεδομένα και επιστρέφει αμέσως. Δηλαδή, συνήθως όταν γίνεται κλήση κάποιας μεθόδου, ακόμα και εάν δεν επιστρέφεται κάτι, ο καλών εμποδίζεται για τη διάρκεια της κλήσης σε μία τυπική διεπαφή Java. Εδώ απλώς γίνεται η συναλλαγή δεδομένων και αμέσως επιστρέφει. Τέλος, δεν υπάρχει η δυνατότητα χρήσης εξαιρέσεων (Exceptions) και ενώ στις διεπαφές Java μπορεί μία διεπαφή να κληρονομήσει από μία άλλη, στην AIDL δεν υπάρχει αντίστοιχη δυνατότητα, αλλά μπορεί να περαστεί μία άλλη διεπαφή AIDL ως παράμετρος μεθόδου της διεπαφής. Συνήθως χρησιμοποιούνται κατά την υλοποίηση ασύγχρονων μεθόδων επανάκλησης (asynchronous one-way callbacks). Όπως, όμως, συμβαίνει και στις Java διεπαφές, η aidl διεπαφή περιέχει τον ορισμό και την υπογραφή των μεθόδων και όχι την υλοποίηση του σώματός τους.

Σημαντική παρατήρηση είναι ότι κάθε αρχείο .aidl ορίζει μία και μόνο, διαφορετική διεπαφή AIDL. Οπότε εάν παραχθούν παραπάνω από μία διεπαφές, και εφόσον μία μπορεί να χρησιμοποιηθεί από κάποια άλλη, πρέπει να είναι σίγουρο ότι μπορεί να την αναγνωρίσει. Αυτό γίνεται εισάγοντας (`import` ...) το αρχείο της διεπαφής σε μία άλλη. Επίσης, θα παράγουν διαφορετικά αρχεία στο φάκελο /gen.

Ένα αρχείο διεπαφής AIDL μεταγλωττίζεται από τον aidl compiler και παράγει ένα αρχείο .java, το οποίο μπαίνει στο φάκελο /gen της εφαρμογής, όπως φαίνεται πιο κάτω:



Εικόνα 27. Το παραγόμενο αρχείο IFile SharingService.java στο φάκελο /gen. Μέρος του κώδικά του εμφανίζεται στην εικόνα.

Δηλώνονται και εδώ τιμές επιστροφής (return values ή void) και παράμετροι. Οι πρωταρχικοί / αρχέγονοι τύποι δεδομένων Java (Java primitive types) που υποστηρίζονται είτε ως παράμετροι είτε ως τύποι επιστροφής είναι : boolean, boolean[], int, int[], byte, byte[], long, long[], float, float[], double, double[] και char[]. Επίσης μπορεί να επιστραφούν CharSequences και Strings. Ακόμα μπορεί να χρησιμοποιηθούν Λίστες (List), των οποίων τα στοιχεία πρέπει απαραίτητα να είναι είτε πρωταρχικοί τύποι δεδομένων, είτε άλλες διεπαφές AIDL ή Parcelables που έχουν δηλωθεί ή μπορεί να οριστεί ως μία γενική (generic) Λίστα, δηλαδή να χρησιμοποιήσει τύπους, κλάσεις και διεπαφές, ως παραμέτρους όταν ορίζει άλλες κλάσεις, διεπαφές ή μεθόδους, για παράδειγμα `List<String>`. Επίσης, μπορεί να χρησιμοποιηθούν Maps (java.util.Map), όπου επίσης πρέπει το κάθε στοιχείο να είναι ένας από τους υποστηριζόμενους τύπους δεδομένων, όπως αναλύθηκε προηγουμένως. Τέλος, είναι δυνατόν να ορίσουμε τους δικούς μας τύπους δεδομένων, οι οποίοι θα είναι παράμετροι ή τιμές επιστροφής, αρκεί να υλοποιούν την κλάση Parcelable και τις μεθόδους της. Οπότε, δημιουργούμε μία τέτοια κλάση και έχουμε instances αυτής ως παραμέτρους και τιμές επιστροφής. Την κατάλληλη στιγμή το Android θα καλέσει αυτές τις μεθόδους είτε στο Stub είτε στο Proxy που θα δημιουργηθεί στην aidl διεπαφή. Οτιδήποτε άλλο εισάγεται (import...),

ακόμα και αν αποτελεί μέρος του ίδιου πακέτου με τη διεπαφή AIDL. Οι μέθοδοι μπορεί να έχουν μηδέν ή περισσότερες παραμέτρους και επιστρέφουν μία τιμή ή void. Επίσης, δεν υποστηρίζονται στατικά πεδία. Το αρχείο αποθηκεύεται στον κατάλογο src/ και παράγεται μία διεπαφή IBinder, κατάλληλη για απομακρυσμένη πρόσβαση, στον κατάλογο gen/ με το ίδιο όνομα και κατάληξη .java. Ο μεταγλωττιστής AIDL είναι υπεύθυνος για αυτήν τη μετατροπή και παράγει Java διεπαφές που αντιστοιχούν στα αρχεία .aidl. Επίσης, περιέχουν ένα Stub (`extends android.os.IBinder`) που θα χρησιμοποιηθεί από το συστατικό που θα υλοποιήσει τη διεπαφή. Τέλος, δημιουργείται ένα Proxy που θα χρησιμοποιηθεί από τον client, εφόσον υπάρχει σύγχρονη επικοινωνία με την Υπηρεσία και την Υπηρεσία / Δραστηριότητα. Αυτό το αρχείο περιλαμβάνει μία υποκλάση Stub που είναι η υλοποίηση της αρχικής διεπαφής και δηλώνει όλες τις μεθόδους της διεπαφής και κάποιες βοηθητικές ακόμη, όπως την `asInterface()` .

IfileSharingService.java

```
/*
 * This file is auto-generated.  DO NOT MODIFY.
 * Original file: /home/mimina/workspace (another
copy)/FileShare/src/com/dimitra/dipl/fileshare/IFileSharingService.
aidl
 */
package com.dimitra.dipl.fileshare;
public interface IFileSharingService extends android.os.IInterface
{
/** Local-side IPC implementation stub class. */
public static abstract class Stub extends android.os.Binder
implements com.dimitra.dipl.fileshare.IFileSharingService
{
private static final java.lang.String DESCRIPTOR =
"com.dimitra.dipl.fileshare.IFileSharingService";
/** Construct the stub at attach it to the interface. */
public Stub()
```

```
{
this.attachInterface(this, DESCRIPTOR);
}
/**
 * Cast an IBinder object into an
 * com.dimitra.dipl.fileshare.IFileSharingService interface,
 * generating a proxy if needed.
 */
public static com.dimitra.dipl.fileshare.IFileSharingService
asInterface(android.os.IBinder obj)
{
if ((obj==null)) {
return null;
}
android.os.IInterface iin = obj.queryLocalInterface(DESCRIPTOR);
if (((iin!=null)&&(iin instanceof
com.dimitra.dipl.fileshare.IFileSharingService))) {
return ((com.dimitra.dipl.fileshare.IFileSharingService)iin);
}
return new
com.dimitra.dipl.fileshare.IFileSharingService.Stub.Proxy(obj);
}
@Override public android.os.IBinder asBinder()
{
return this;
}
@Override public boolean onTransact(int code, android.os.Parcel
data, android.os.Parcel reply, int flags) throws
android.os.RemoteException
{
switch (code)
```

```
{
case INTERFACE_TRANSACTION:
{
reply.writeString(DESCRIPTOR);
return true;
}
case TRANSACTION_getPort:
{
data.enforceInterface(DESCRIPTOR);
int _result = this.getPort();
reply.writeNoException();
reply.writeInt(_result);
return true;
}
}
return super.onTransact(code, data, reply, flags);
}
private static class Proxy implements
com.dimitra.dipl.fileshare.IFileSharingService
{
private android.os.IBinder mRemote;
Proxy(android.os.IBinder remote)
{
mRemote = remote;
}
@Override public android.os.IBinder asBinder()
{
return mRemote;
}
public java.lang.String getInterfaceDescriptor()
{
```

```
return DESCRIPTOR;
}
@Override public int getPort() throws android.os.RemoteException
{
android.os.Parcel _data = android.os.Parcel.obtain();
android.os.Parcel _reply = android.os.Parcel.obtain();
int _result;
try {
_data.writeInterfaceToken(DESCRIPTOR);
mRemote.transact(Stub.TRANSACTION_getPort, _data, _reply, 0);
_reply.readException();
_result = _reply.readInt();
}
finally {
_reply.recycle();
_data.recycle();
}
return _result;
}
}
static final int TRANSACTION_getPort =
(android.os.IBinder.FIRST_CALL_TRANSACTION + 0);
}
public int getPort() throws android.os.RemoteException;
}
```

Για την υλοποίηση της διεπαφής που έχει προκύψει από το .aidl, γίνεται extend το Binder που έχει προκύψει, εδώ το IFileSharingService.Stub, και υλοποιώ τις μεθόδους που έχουν κληρονομηθεί από το αρχείο .aidl.

FileSharingService.java

```
public class FileSharingService extends Service {



   private int mPort;

   //   δημιουργία του mBinder

   private final IFileSharingService.Stub mBinder = new

         IFileSharingService.Stub() {

      //   υλοποίηση των μεθόδων που έχουν κληρονομηθεί από το

      //   .aidl αρχείο

      public int getPort() {

          return mPort;

      }

   };

//   ολοκλήρωση της διεπαφής

. . .

   public void onCreate() {

      . . .

       mPort = settings.getInt("port", DEFAULT_PORT);

      . . .
```

```
    }

. . .

    //   δημιουργία της onBind()

    public IBinder onBind(Intent intent) {

        if (IfileSharingService.class.getName()
                .equals(intent.getAction())) {

          //   επιστροφή της διεπαφής
          return mBinder;
        }
        return null;
    }
}
```

Το mBinder είναι αντικείμενο (instance) της κλάσης Stub, που είναι ένας Binder, δηλαδή ένας μηχανισμός για επικοινωνία μεταξύ των διαδικασιών και επιτρέπει σε μία ή περισσότερες διαδικασίες (processes) να διαμοιράζονται δεδομένα και υπηρεσίες, και ορίζει την RPC (Remote Procedure Call) διεπαφή για την Υπηρεσία. Τρέχει στον πυρήνα του Linux. Οι clients μέσω αυτού του Binder μπορούν να επικοινωνήσουν με την Υπηρεσία. Σημειώνεται ότι: οι εισερχόμενες κλήσεις δεν είναι σίγουρο ότι θα εκτελεστούν στο κυρίως νήμα, οπότε χρειάζεται εξαρχής να γνωρίζουμε ότι πιθανόν να χρειαστεί multithreading για την εξασφάλιση των κλήσεων, οι κλήσεις RPC είναι σύγχρονες, και άρα εάν χρειαστεί αρκετός χρόνος για να ολοκληρωθεί ένα request, δεν πρέπει να καλείται από το κυρίως νήμα, καθώς υπάρχει περίπτωση το Android να εμφανίσει το μήνυμα “Η εφαρμογή δεν ανταποκρίνεται”, οπότε η διαδικασία υλοποιείται σε ξεχωριστό νήμα στον client, και όλες οι εξαιρέσεις (Exceptions) δεν υλοποιούνται στη μεριά του καλούντος (caller). Τέλος, οι εφαρμογές σπανίως χρησιμοποιούν απευθείας έναν Binder, αλλά μέσω Stubs και Proxies. Αυτά μετατρέπουν τα δεδομένα του καλούντα (client) σε parcels, τα μεταφέρουν στη μεριά

της Υπηρεσίας και τα ανασυνθέτουν χρησιμοποιώντας μηχανισμούς marshalling και demarshalling.

Μόλις ολοκληρωθεί η διεπαφή της Υπηρεσίας, την παρέχουμε στους clients κάνοντας extend την κλάση Service και υλοποιώντας τη μέθοδο `onBind()`, για να επιστρέψει ένα αντικείμενο IBinder της κλάσης που υλοποιεί το Stub. Όταν ένας client προσπαθεί να συνδεθεί με αυτήν την Υπηρεσία, καλεί την `bindService()`, για να συνδεθεί με την Υπηρεσία χωρίς να επιστρέφει κάποια τιμή στον client, και η `onServiceConnected()`, που υλοποιείται στον client, λαμβάνει το αντικείμενο mBinder που έχει επιστραφεί στην `onBind()` της Υπηρεσίας. Ακόμα και εάν συνδεθούν περισσότεροι του ενός clients, η `onBind()` της Υπηρεσίας καλείται μόνο μία φορά από το σύστημα για να επιστρέψει το αντικείμενο IBinder, μόλις ο πρώτος client συνδεθεί, και αποδίδει το ίδιο αντικείμενο στους υπόλοιπους που θα συνδεθούν μετά από αυτόν, χωρίς να καλέσει ξανά τη μέθοδο. Μόλις ο τελευταίος client αποσυνδεθεί από την Υπηρεσία, το σύστημα αυτόματα την καταστρέφει. Επίσης, εάν ο client και η service βρίσκονται σε διαφορετικές εφαρμογές, τότε πρέπει και οι δύο να έχουν πρόσβαση στις μεθόδους AIDL που έχουν οριστεί. Άρα, ένα αντίγραφο του .aidl αρχείου είναι απαραίτητο να υπάρχει στον κατάλογο src/ του client για τη δημιουργία της διεπαφής android.os.Binder. [2] [4] [18] [20] [21] [22]

Για την αποστολή δεδομένων μέσω μίας διεπαφής IPC, η κλάση πρέπει να υλοποιεί μία διεπαφή Parcelable, καθώς επιτρέπει στο Android να αποσυνθέτει τα αντικείμενα σε πρωταρχικούς τύπους τα οποία έπειτα είναι σε θέση να μεταφερθούν στις διαδικασίες (marshalling). Μέσα στο αρχείο της διεπαφής που δημιουργήθηκε από το .aidl αρχείο, υπάρχουν αντικείμενα της κλάσης Parcel, που είναι υπεύθυνα για το marshalling, τη μετάδοση και το demarshalling των παραμέτρων. Για να υποστηριχθεί το πρωτόκολλο Parcelable, η κλάση πρέπει να υλοποιεί τη διεπαφή Parcelable και το αρχείο .aidl δηλώνει την κλάση που υλοποιεί τη διεπαφή Parcelable. Παράδειγμα χρήσης Parcelable παρατίθεται παρακάτω:

MyMessage.java

```
package package com.dimitra.dokimiAidl;



import android.os.Parcel;
```

```
import android.os.Parcelable;

public class MyMessage implements Parcelable{
   //πρέπει να έχει ένα στατικό πεδίο CREATOR.
   //Ένα αντικείμενο που υλοποιεί τη διεπαφή
   //Parcelable.Creator.
   public static final Parcelable.Creator <MyMessage> CREATOR =
       new Parcelable.Creator<MyMessage>(){ ... };
            ...
   //πρέπει να υλοποιεί τη μέθοδο αυτή.
   public void writeToParcel(Parcel parcel, int flags){ ... };
}
```

Και η AIDL διεπαφή:

MyMessage.aidl

```
package com.dimitra.dokimiAidl;

parcelable MyMessage;
```

4.2.4. Χρήση της SQLite

Η SQLite είναι μία βιβλιοθήκη που υλοποιεί Βάσεις Δεδομένων SQL, που είναι ανοιχτού κώδικα. Είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που περιέχεται σε μία C προγραμματιστική βιβλιοθήκη. Λειτουργεί όπως και η SQL, χρησιμοποιώντας τις ίδιες εκφράσεις και συντακτικό. Όμως, χρειάζεται λιγότερη μνήμη -περίπου 250KByte- κατά το χρόνο εκτέλεσης. Σε αντίθεση με άλλα αντίστοιχα συστήματα διαχείρισης βάσης δεδομένων, η SQLite δεν είναι μία ξεχωριστή διεργασία που προσπελάζεται από μία εφαρμογή πελάτη, αλλά αποτελεί ενσωματωμένο μέρος της. Ο πηγαίος κώδικάς της αποτελεί κοινό κτήμα (public domain). Καθώς υπάρχει σε κάθε συσκευή Android, δε χρειάζεται να κάνουμε κάτι

για να την εγκαταστήσουμε κα δε χρειάζεται δικαιώματα διαχειριστή βάσης. Χρειάζεται μόνο να δηλώσουμε τη δημιουργία και την ανανέωση της ΒΔ και έπειτα το Android τη διαχειρίζεται αυτόματα. Τα πακέτα κλάσεων που χρησιμοποιούνται είναι τα: android.database.*; και android.database.sqlite.*; [23]

Για τη δημιουργία και ανανέωση της βάσης δημιουργούμε μία κλάση που επεκτείνει (extends) την SQLiteOpenHelper, δηλαδή είναι υποκλάση της. Στη μέθοδο – δημιουργό (constructor) γίνεται κλήση στη μέθοδο `super()`, όπου ορίζονται το όνομα και η έκδοση της βάσης. Ύστερα, υπερσκελίζονται (`override`) οι μέθοδοι `onCreate()` και `onUpgrade()` για τη δημιουργία και την ανανέωση της βάσης. Και οι δύο μέθοδοι λαμβάνουν ένα αντικείμενο SQLiteDatabase σαν παράμετρο.

CookiesDatabaseOpenHelper.java

```
package com.dimitra.dipl.fileshare;



import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;



public class CookiesDatabaseOpenHelper extends SQLiteOpenHelper {



   public CookiesDatabaseOpenHelper(Context context) {

      super(context, "cookies.db", null, 1);

   }
```

```
    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL("CREATE TABLE cookies(name STRING, value
            STRING, expiry INTEGER)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub
    }
}
```

Η κλάση SQLiteDatabase παρέχει μεθόδους για ανοίξουμε, υποβάλλουμε ερωτήματα, να ανανεώσουμε και να κλείσουμε τη ΒΔ. Οι μέθοδοι αυτές είναι οι `insert()`, `update()`, `delete()`, `execSQL()`, `query()` και `rawQuery()`.

FileSharingProvider.java

```
/**
* This class helps open, create, and upgrade the database file.
```

```
*/
private static class DatabaseHelper extends SQLiteOpenHelper {
   DatabaseHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE " + FOLDERS_TABLE_NAME + " ("
         + Folders.Columns._ID + " INTEGER PRIMARY KEY,"
         + Folders.Columns.DISPLAY_NAME + " TEXT,"
         + Folders.Columns.PASSWORD + " TEXT"
         + ");");
      db.execSQL("CREATE TABLE " + FILES_TABLE_NAME + " ("
         + Files.Columns._ID + " INTEGER PRIMARY KEY,"
         + Files.Columns.FOLDER_ID + " INTEGER,"
         + Files.Columns.DISPLAY_NAME + " TEXT,"
         + Files.Columns._DATA + " TEXT"
```

```
        + ");");
    db.execSQL("INSERT INTO " + FOLDERS_TABLE_NAME + " VALUES(
        0,'Public', null)");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion,
        int newVersion) {
    Log.w(TAG, "Upgrading database from version " + oldVersion
        + " to "
        + newVersion + ", which will destroy all old data");
    /* Move existing tables out of the way. */
    db.execSQL("ALTER TABLE " + FOLDERS_TABLE_NAME
        + " RENAME TO " + FOLDERS_TABLE_NAME + "_TEMP");
    db.execSQL("ALTER TABLE " + FILES_TABLE_NAME
        + " RENAME TO " + FILES_TABLE_NAME + "_TEMP");
    /* Create the new tables. */
    createTables(db, FOLDERS_TABLE_NAME, FILES_TABLE_NAME);
```

```
/* Copy data from old tables to new tables. */
Cursor c = db.query(
   FOLDERS_TABLE_NAME + "_TEMP",null, null, null, null,
      null, null);
while (c.moveToNext()) {
   ContentValues values = new ContentValues();
   values.put(Folders.Columns._ID, c.getInt(c.getColumnIndex(
      Folders.Columns._ID)));
   values.put(Folders.Columns.DISPLAY_NAME,
   c.getString(c.getColumnIndex(
      Folders.Columns.DISPLAY_NAME)));
   values.put(Folders.Columns.PASSWORD, c.getString(
      c.getColumnIndex(Folders.Columns.PASSWORD)));
   db.insert(FOLDERS_TABLE_NAME,
      Folders.Columns.DISPLAY_NAME, values);
}
c.close();
c = db.query(FILES_TABLE_NAME + "_TEMP",null, null, null,
```

```
            null, null, null);
        while (c.moveToNext()) {
            ContentValues values = new ContentValues();
            values.put(Files.Columns._ID, c.getInt(c.getColumnIndex(
                Files.Columns._ID)));
            values.put(Files.Columns.FOLDER_ID,
c.getInt(c.getColumnIndex(
                Files.Columns.FOLDER_ID)));
            values.put(Files.Columns.DISPLAY_NAME, c.getString(
                c.get ColumnIndex(Files.Columns.DISPLAY_NAME)));
            values.put(Files.Columns._DATA, c.getString(c.getColumnIndex(
                Files.Columns._DATA)));
            db.insert(FILES_TABLE_NAME, Files.Columns.DISPLAY_NAME,
                values);
        }
        c.close();
        /* Delete the temp tables. */
        db.execSQL("DROP TABLE " + FOLDERS_TABLE_NAME + "_TEMP");
        db.execSQL("DROP TABLE " + FILES_TABLE_NAME + "_TEMP");
```

```
}


private static void createTables(
   SQLiteDatabase db, String foldersTable, String filesTable) {
      db.execSQL("CREATE TABLE " + foldersTable + " ("
         + Folders.Columns._ID + " INTEGER PRIMARY KEY,"
         + Folders.Columns.DISPLAY_NAME + " TEXT,"
         + Folders.Columns.PASSWORD + " TEXT,"
         + "UNIQUE (" + Folders.Columns.DISPLAY_NAME + ")"
         + ");");
      db.execSQL("CREATE TABLE " + filesTable + " ("
         + Files.Columns._ID + " INTEGER PRIMARY KEY,"
         + Files.Columns.FOLDER_ID + " INTEGER,"
         + Files.Columns.DISPLAY_NAME + " TEXT,"
         + Files.Columns._DATA + " TEXT,"
         + "UNIQUE (" + Files.Columns.FOLDER_ID + ","
         + Files.Columns._DATA + ")"
         + ");");
```

```
        }
    }


    private DatabaseHelper mOpenHelper;


    @Override
    public boolean onCreate() {
        mOpenHelper = new DatabaseHelper(getContext());
            return true;
    }
}
```

Ένα ερώτημα στη βάση (query) επιστρέφει ένα αντικείμενο Cursor, το οποίο αντιπροσωπεύει το αποτέλεσμα του ερωτήματος. Επειδή επιστρέφει μία γραμμή της βάσης, δρα αποτελεσματικά, καθώς το Android δε χρειάζεται να "φορτώσει" τη μνήμη με όλα τα δεδομένα. Το αντικείμενο Cursor πρέπει απαραιτήτως να τερματιστεί και γίνεται η χρήση της μεθόδου `close()`.

FileSharingProvider.java

```
Cursor c = qb.query(db, projection, selection, selectionArgs, null,
null, sortOrder);

c.close();
```

Η εφαρμογή υλοποιεί επίσης τις μεθόδους `getReadableDatabase()` και `getWriteableDatabase()` οι οποίες δίνουν πρόσβαση στο αντικείμενο SQLiteDatabase.

FileSharingProvider.java

```
SQLiteDatabase db = mOpenHelper.getReadableDatabase();
```

FileSharingService.java

```
mWebServer = new WebServer(this, settings,

new CookiesDatabaseOpenHelper(this).getWritableDatabase(),
mPort);
```

4.2.5. Υποστηρικτικές Βιβλιοθήκες και Build Paths

Στον πηγαίο κώδικα της εργασίας έχουν εισαχθεί υποστηρικτικές βιβλιοθήκες (support libraries). Οι υποστηρικτικές βιβλιοθήκες του Android είναι ένα σετ βιβλιοθηκών που επιτρέπουν την προς τα πίσω συμβατότητα μεταξύ των διεπαφών προγραμματισμού εφαρμογών (framework APIs) του Android. Η χρήση τους επιτρέπει στο σύστημα να χρησιμοποιεί χαρακτηριστικά των αντίστοιχων εκδόσεων Android, ακόμα και αν λειτουργεί με κάποια παλαιότερη έκδοση. Ένα παράδειγμα της χρήσης τους είναι η Μπάρα Ενεργειών (Action Bar), που εισήχθη στην έκδοση Android 3.0. Εάν η συσκευή έχει άλλη έκδοση, η εφαρμογή είναι δυνατόν να εμφανίσει την Μπάρα, αρκεί κατά τη διαδικασία υλοποίησης της εφαρμογής ο δημιουργός να έχει εισάγει την κατάλληλη υποστηρικτική βιβλιοθήκη. Άρα, είναι κατανοητό ότι η εισαγωγή των υποστηρικτικών βιβλιοθηκών είναι η βέλτιστη πρακτική, έτσι ώστε η εφαρμογή να στοχεύσει σε ένα μεγάλο εύρος εκδόσεων που υποστηρίζονται από τα τερματικά. Βοηθά, επίσης, στην καλύτερη απεικόνιση της εφαρμογής, καθώς δίνει πρόσβαση σε όλα τα χαρακτηριστικά που έχουν υλοποιηθεί. Καθώς κάθε μία από τις βιβλιοθήκες υποστηρίζει ένα Android API επίπεδο, παρέχει τα αντίστοιχα χαρακτηριστικά που σχετίζονται με αυτό το επίπεδο, οπότε είναι σημαντικό να κατανοήσουμε το τι μπορεί να προσφέρει κάθε μία από αυτές για τη βέλτιστη χρήση τους. Σύμφωνα με τον ιστότοπο

developer.android.com, είναι καλό μία εφαρμογή να εμπεριέχει τις βιβλιοθήκες v4 support και v7 appcompat, αφού υποστηρίζουν πολλές εκδόσεις του Android, παρέχουν APIs για τη διεπιφάνεια χρήστη, την Μπάρα Ενεργειών, τη σύνδεση με δίκτυα, τη διαχείριση δεδομένων μεταξύ άλλων, και μπορούν να χρησιμοποιηθούν από συσκευές που υποστηρίζουν έκδοση 1.6 και μεγαλύτερη. Για να μπορούν να χρησιμοποιηθούν οι βιβλιοθήκες αυτές στο περιβάλλον Eclipse, πρέπει να έχει εγκατασταθεί το πακέτο Android Support Library μέσα από το περιβάλλον του SDK Manager, το οποίο περιέχει τις υποστηρικτικές βιβλιοθήκες στη διαδρομή <sdk>/extras/android/support/v4, να δημιουργήσουμε μέσα στα αρχεία της εφαρμογής ένα φάκελο με το όνομα libs/, να αντιγραφεί το αρχείο jar που έχουμε κατεβάσει από τη διαδρομή που είναι αποθηκευμένο μέσα στο libs/, και τέλος να προστεθεί στο Build Path πατώντας το αρχείο και επιλέγοντας Build Path > Add to Build Path.

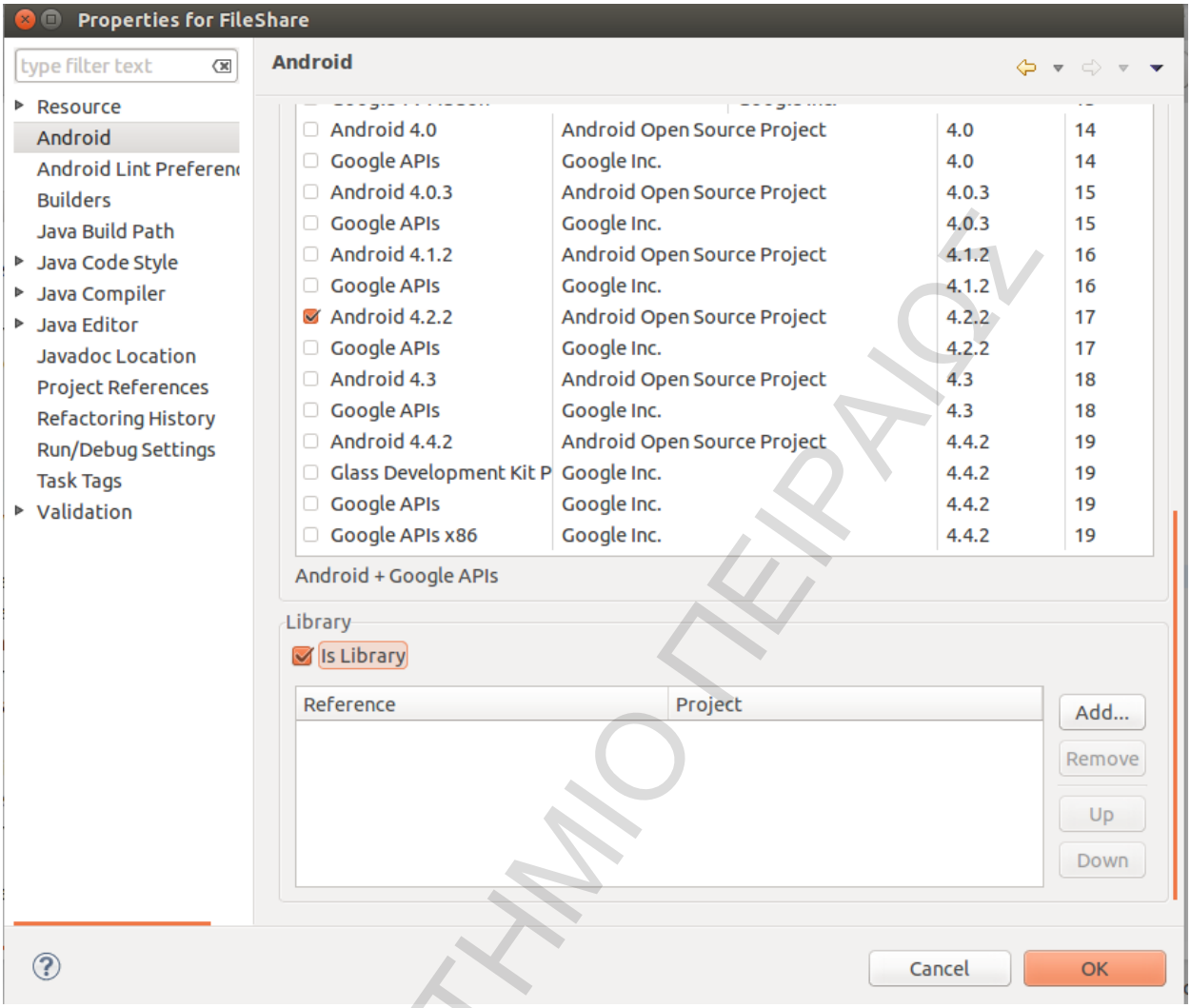
Επίσης, στην εφαρμογή έχει εισαχθεί το αρχείο jar commons-fileupload από την ιστοσελίδα http://commons.apache.org/proper/commons-fileupload/ , το οποίο, σύμφωνα με τον ιστότοπο, είναι ένα πακέτο που προσθέτει δυνατότητα μεταφοράς αρχείων στις εφαρμογές, εφόσον υπάρχει μέσα στον κώδικα εντολή που υλοποιεί τη μέθοδο POST για HTTP requests με content type “multipart/form-data”. Τότε μπορεί το fileupload να αναλύσει αυτήν την εντολή και να εξάγει αποτελέσματα. Μέσα στον κώδικα υλοποιείται στο αρχείο WebServer.java. Όπως αναφέρεται στην ιστοσελίδα http://commons.apache.org/, το ίδρυμα Apache Software Foundation, το οποίο παρέχει τα Apache Commons, σκοπεύει με τη χρήση των Commons να παρέχει επαναχρησιμοποιήσιμο, ανοιχτού τύπου λογισμικό Java. Παρακάτω παρατίθεται το απόσπασμα κώδικα που υλοποιεί τη μέθοδο POST και το content type.

WebServer.java

```
private String getUploadForm(String folderId) {
    if(mSharedPreferences.getBoolean(
          FileSharingService.PREFS_ALLOW_UPLOADS,false)) {
       return "<form method=\"POST\" action=\"/folder/"
          + folderId + "\" "
          + "enctype=\"multipart/form-data\"> "
          + "<input type=\"file\" name=\"file\" size=\"40\"/> "
```

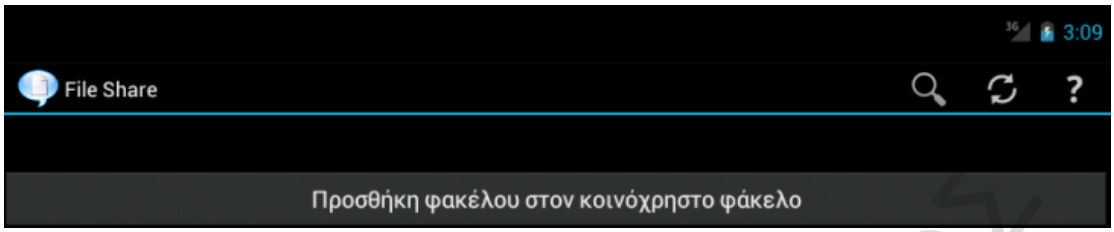
```
            + "<input type=\"submit\" value=\"Upload\"/>";
    }
    return "";
}
```

Οι εφαρμογές Android μπορούν να χρησιμοποιήσουν τον κώδικα που υπάρχει στα αρχεία jar (Java libraries). Αυτές οι βιβλιοθήκες επιτρέπουν την αποθήκευση πηγαίου κώδικα, αλλά και πόρων για να επαναχρησιμοποιηθούν από άλλες εφαρμογές εισάγοντάς τις στο φάκελο libs/ της εφαρμογής. Αυτομάτως προστίθεται στο classpath της εφαρμογής. Οι βιβλιοθήκες αυτές δεν μπορούν να εκτελεστούν ως αυτόνομες εφαρμογές, αλλά πρέπει να αποτελούν μέρος μίας άλλης, ενώ ο πηγαίος κώδικάς τους και, εάν υπάρχουν, οι πόροι τους περιλαμβάνονται στο τελικό αποτέλεσμα, δηλαδή μεταγλωττίζονται και είναι μέρος του αρχείου .apk της εφαρμογής στην οποία έχουν εισαχθεί. Αρχεία jar όπως τα v4 support και v7 appcompat, καθώς το commons-fileupload, χρησιμοποιούνται ευρέως, αφού διευκολύνουν τη λειτουργικότητα της εφαρμογής αλλά και ελαττώνουν τον πηγαίο κώδικα που πρέπει να αναπτύξει ο προγραμματιστής. Η δημιουργία μίας τέτοιας βιβλιοθήκης είναι εύκολη, καθώς μπορεί να περιέχει μεταξύ άλλων πόρους και κλάσεις Java. Ο προγραμματιστής πρέπει να δηλώσει ότι η εργασία αυτή θα είναι μία βιβλιοθήκη δημιουργώντας μία καινούρια εφαρμογή στο Eclipse και διαλέγοντας στον οδηγό την επιλογή Mark this project as a library. Εναλλακτικά, μπορεί να δημιουργήσει πρώτα την εφαρμογή και πηγαίνοντας στις Προτιμήσεις της εργασίας να διαλέξει στην καρτέλα Android την επιλογή isLibrary, έτσι ώστε να μετατραπεί σε βιβλιοθήκη. Στην ίδια καρτέλα μπορεί να προσθέσει τυχόν βιβλιοθήκες που επιθυμεί να αποτελούν μέρος της εκάστοτε εφαρμογής.



Εικόνα 28. Ορισμός εργασίας (project) στο Android ως βιβλιοθήκης.

4.2.6. Το Action Bar



Εικόνα 29. Το Action Bar της εφαρμογής.

Η Μπάρα Ενεργειών, πιο γνωστή ως Action Bar, είναι τοποθετημένη στην κορυφή της activity. Παρέχει στο χρήστη κάποιες λειτουργίες και πληροφορίες, όπως η ονομασία της εφαρμογής, το εικονίδιό της, ενέργειες που ο χρήστης προκαλεί, επιπλέον views και άλλα διαδραστικά αντικείμενα, αλλά και ως "πλοηγός" μέσα στην εφαρμογή. Το Action Bar είναι ενεργοποιημένο για συσκευές Android με έκδοση 11 (API level) ή Android 3.0. Ο χρήστης μπορεί να το απενεργοποιήσει. Εφαρμογές με target SDK version (ορίζεται στην AndroidManifest.xml της εφαρμογής) χαμηλότερο του API level 11 χρησιμοποιούν εναλλακτικά το μενού επιλογών (options menu), εφόσον υπάρχει το αντίστοιχο κουμπί στη συσκευή. Το action bar είναι εξ αρχής ενεργοποιημένο και άρα ορατό, ενώ το options menu είναι ορατό μόνο εφόσον πατηθεί το κουμπί των επιλογών και άρα ο χρήστης μπορεί να μην γνωρίζει ότι υπάρχουν διαθέσιμες επιλογές. Εάν η συσκευή έχει έκδοση μικρότερη του Android 3.0, υπάρχουν κάποιες επιλογές για να λειτουργήσει το action bar, όπως για παράδειγμα η χρήση της υποστηρικτικής βιβλιοθήκης (support library) v7 για εκδόσεις 2.1 και πάνω, που αναλύθηκε σε προηγούμενη παράγραφο. Ό, τι τοποθετείται στο action bar ονομάζεται ενέργεια (action). Μπορούμε να δημιουργήσουμε αυτές τις ενέργειες είτε αναφέροντάς τις άμεσα (explicitly) στον πηγαίο κώδικα, είτε να τις ορίσουμε στο φάκελο των πόρων σε ένα .xml αρχείο. Όπως έχει ήδη αναφερθεί στα πλαίσια της παρούσας διπλωματικής εργασίας, ό,τι εισάγουμε σε ένα αρχείο πόρων πρέπει να το αναφέρουμε στον πηγαίο κώδικα. Το θετικό είναι ότι εάν αλλαχθούν οι πόροι, δε χρειάζεται ο κώδικας να μεταγλωττιστεί εκ νέου και διαφορετικοί πόροι μπορούν να δημιουργηθούν βάσει των προτιμήσεων χρήστη, των διαφορετικών συσκευών ή διαφορετικών ρυθμίσεων και να χρησιμοποιηθούν χωρίς να επηρεάζεται ο πηγαίος κώδικας και κατά επέκταση η εφαρμογή.
Εάν ορίσουμε τις ενέργειες αυτές σε ένα .xml αρχείο, όπως είναι συνηθέστερο, θα το τοποθετήσουμε στο φάκελο /res/menu του project. Αυτόματα τα εργαλεία του Android θα

δημιουργήσουν αναφορές για την καθεμία ενέργεια στο αρχείο R.java, έτσι ώστε οι πόροι του μενού να είναι διαθέσιμοι.

Παρακάτω παρατίθεται το αρχείο xml το οποίο έχει αναπτυχθεί στα πλαίσια της παρούσας διπλωματικής:

/res/menu/activity_main_actions.xml

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

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<!-- Search / will display always -->

<item android:id="@+id/action_search"

      android:icon="@drawable/ic_action_search"

      android:title="@string/action_search"

      android:showAsAction="always"

      android:actionViewClass="android.widget.SearchView"/>

<!-- Refresh -->

<item android:id="@+id/action_refresh"

      android:icon="@drawable/ic_action_refresh"

      android:title="@string/action_refresh"

      android:showAsAction="always" />

<!-- Help -->

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

```
        android:icon="@drawable/ic_action_help"
        android:title="@string/action_help"
        android:showAsAction="ifRoom"/>
</menu>
```

Στο πιο πάνω αρχείο η επιλογή `showAsAction` ορίζει πώς θα φαίνεται μία ενέργεια. Παραδείγματος χάρη, εάν έχει οριστεί ως always, τότε θα εμφανίζεται πάντοτε στο action bar. Υπάρχουν οι επιλογές: `never`, `always`, `ifRoom`, `withText`, `collapseActionView`. Οι τρεις τελευταίες επιλογές αφορούν τις περιπτώσεις όπου η ενέργεια θα είναι ορατή στο action bar μόνο εάν υπάρχει αρκετός χώρος στην οθόνη, εάν η ενέργεια περιλαμβάνει εικονίδιο και κείμενο έτσι ώστε να εμφανιστούν και τα δύο και όχι μόνο το εικονίδιο, και η μπάρα ενεργειών να συμπτυχθεί σε ένα κουμπί ενεργειών, αντίστοιχα.

Για να υπάρξει αναφορά στο action bar μίας εφαρμογής, πρέπει να κληθεί η μέθοδος `getActionBar()`. Μία activity προσθέτει τις ενέργειες που έχουν οριστεί μέσω της μεθόδου `onCreateOptionsMenu()`, η οποία καλείται μία μόνο φορά. Για να χρησιμοποιηθεί ένα menu στην activity, πρέπει να μετατραπεί ο πόρος που είναι σε μορφή XML σε αντικείμενο μενού μέσα στον πηγαίο κώδικα, το οποίο γίνεται με χρήση της κλάσης android.view.MenuInflater και της μεθόδου `getMenuInflater()`.

FileShare.java (#ActionBar)

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar
   // if it is present.
   MenuInflater inflater = getMenuInflater();
```

```
    getMenuInflater().inflate(R.menu.activity_main_actions, menu);
    //associate searchable configuration w/ the SearchView
    SearchManager searchManager = (SearchManager)
            getSystemService(Context.SEARCH_SERVICE);
    SearchView searchView = (SearchView)
            menu.findItem(R.id.action_search).getActionView();
    searchView.setSearchableInfo(searchManager
            .getSearchableInfo(getComponentName()));
    return super.onCreateOptionsMenu(menu);
}
```

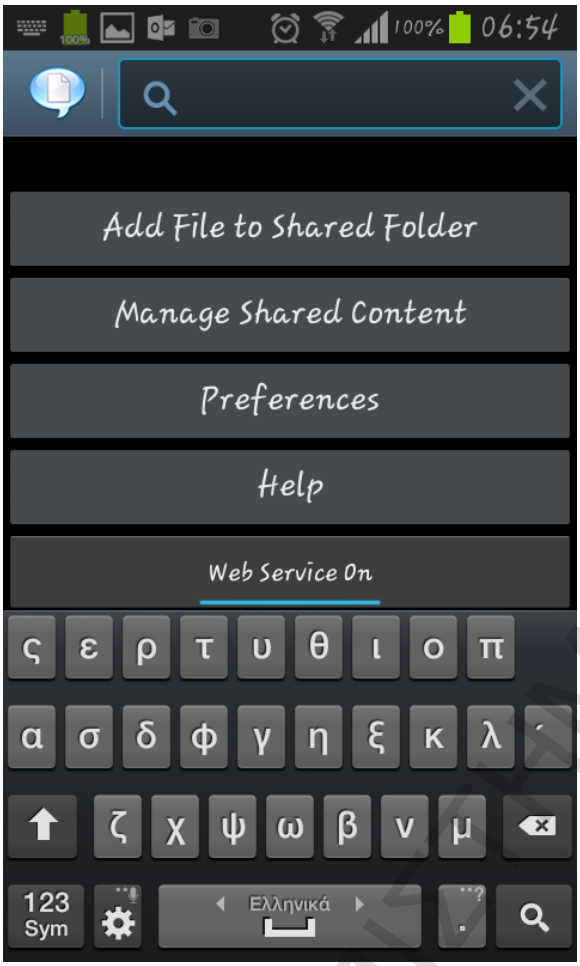
Εάν μία ενέργεια επιλεγεί, καλείται η μέθοδος `onOptionsItemSelected()`, η οποία παίρνει ως παράμετρο την ενέργεια που έχει επιλεχθεί. Η μέθοδος `getItemId()` της κλάσης Menu επιτρέπει την εύρεση των στοιχείων menu βάσει του id τους ως πόρους. Το id αφορά το γνώρισμα (attribute) των πόρων στα αρχεία xml, το οποίο και είναι μοναδικό, και ορίζεται ως string.

FileShare.java

```
/**
* On selecting action bar icons
**/
@Override
```

```
public boolean onOptionsItemSelected(MenuItem item){

      int itemId = item.getItemId();

      if (itemId == R.id.action_search) {

      //search action

          return true;

      } else if (itemId == R.id.action_refresh) {

         //refresh

         //added after the ProgressBar

         refreshMenuItem = item;

        //load the date from server

         new SyncData().execute();

         return true;

      } else if (itemId == R.id.action_help) {

         return true;

      } else {

        return super.onOptionsItemSelected(item);

      }

}
```

Τα εικονίδια που χρησιμοποιήθηκαν σε αυτήν την εφαρμογή δημιουργήθηκαν με το Eclipse, χρησιμοποιώντας τις επιλογές File > New > Other... > Android > Android Icon Set και ακολουθώντας τα βήματα του οδηγού. Είναι δυνατή η χρήση εξωτερικών αρχείων, τα οποία θα εισαχθούν στο φάκελο των πόρων.



Εικόνα 30. Χρήση της Εύρεσης.

4.2.7. Λειτουργία στα ελληνικά

Στο φάκελο των πόρων υπάρχουν δύο αρχεία με το όνομα strings.xml, τα οποία είναι σε διαφορετικούς φακέλους (/res/values και /res/values-el). Το αρχείο /res/values/string.xml είναι το αρχείο που περιέχει τις τιμές των strings που υπάρχουν στην εφαρμογή. Εάν χρειάζεται να υπάρχει η δυνατότητα υποστήριξης διαφορετικών γλωσσών στην εφαρμογή, το αρχείο αυτό μεταφράζεται και ανάλογα με τις ρυθμίσεις που έχει ο χρήστης στη συσκευή του (ρυθμίσεις γλώσσας), αυτόματα επιλέγεται η γλώσσα βάσει των ρυθμίσεων αυτών.

Εάν πρέπει να μεταφραστούν τα strings, μπαίνουν σε ένα καινούριο αρχείο με την ίδια ονομασία, αλλά σε διαφορετικό φάκελο, ο οποίος έχει την ονομασία values-xx, όπου xx είναι η γλώσσα για την οποία έχει γίνει η μετάφραση.

Μέρος του αρχείου /res/values-el/strings.xml

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

<resources>

<string name="app_name">File Share</string>

<string name="manage_button">Διαχείριση κοινόχρηστου
περιεχομένου</string>

<string name="addfile_button">Προσθήκη αρχείου στον κοινόχρηστο
φάκελο</string>

<string name="choosefile_title">Εύρεση αρχείου προς
διαμοίραση</string>

<string name="empty_directory">Κενό</string>

<string name="port">Θύρα:9999</string>

.

.

.

</resources>
```

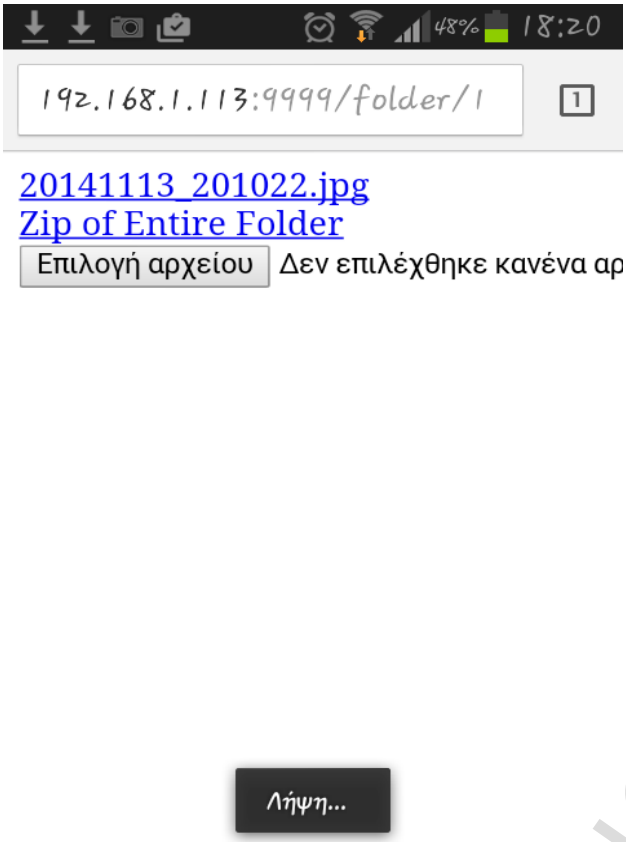
4.2.8. Εκτέλεση της εφαρμογής

Για τη δοκιμασία της εφαρμογής γίνεται η χρήση εξομοιωτών και της φυσικής συσκευής Samsung Galaxy Core. Κατά την εγκατάσταση του .apk στη συσκευή, δίνεται η εξής εικόνα στο console του Eclipse:

```
[2014-06-15 02:30:35 - FileShare] Android Launch!
[2014-06-15 02:30:35 - FileShare] adb is running normally.
[2014-06-15 02:30:35 - FileShare] Performing com.dimitra.dipl.fileshare.FileShare activity launch
[2014-06-15 02:30:38 - FileShare] Uploading FileShare.apk onto device '31cf7675'
[2014-06-15 02:30:39 - FileShare] Installing FileShare.apk...
[2014-06-15 02:30:44 - FileShare] Success!
[2014-06-15 02:30:44 - FileShare] Starting activity com.dimitra.dipl.fileshare.FileShare on device 31cf7675
[2014-06-15 02:30:45 - FileShare] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.d
[2014-06-15 02:33:45 - FileShare] ...............................
```

Εικόνα 31. Εικόνα από την κονσόλα του Eclipse.

Για χρήση πραγματικής συσκευής, είναι απαραίτητο να συνδεθεί το κινητό τερματικό με τον υπολογιστή που υλοποιείται η εφαρμογή μέσω θύρας USB και να διαλέξουμε την επιλογή USB Debugging (Εντοπισμός σφαλμάτων, σε συσκευή με προεπιλεγμένη την ελληνική γλώσσα) από τις Ρυθμίσεις της συσκευής. Προσοχή στη διαδικασία αυτή, καθώς πιθανόν η συσκευή να τεθεί εκτός εγγύησης. Έπειτα με συνδεδεμένη τη συσκευή, απλώς επιλέγουμε την εφαρμογή και πατάμε Run. Το Eclipse την εγκαθιστά αυτόματα στη συσκευή. Από την κονσόλα βλέπουμε σε ποια συσκευή (ή και εξομοιωτή) έχει εγκατασταθεί επιτυχώς.



Εικόνα 32. Με την IP που λαμβάνω έχω πρόσβαση από την κινητή συσκευή με τη χρήση ενός περιηγητή.

Χρησιμοποιώντας την IP διεύθυνση που βλέπω στην εφαρμογή και ένα πρόγραμμα φυλλομετρητή έχω πρόσβαση στα αρχεία. Εμφανίζεται η επιλογή Zip of Entire Folder για το κατέβασμα όλων των αρχείων του φακέλου.

ΚΕΦΑΛΑΙΟ 5

5.1. Συμπεράσματα

Σε αυτό το κεφάλαιο συνοψίζονται τα συμπεράσματα που προέκυψαν με το πέρας της παρούσας διπλωματικής εργασίας.

Η συγγραφή και η ενασχόληση με το Android μπορεί να προσφέρει ένα πλήθος θετικών επακόλουθων. Για να συνοψίσουμε μερικά, ένας νέος προγραμματιστής κάνοντας χρήση των εργαλείων του Android εισάγεται στον αντικειμενοστραφή προγραμματισμό και ενδυναμώνει δεξιότητες που σχετίζονται με αυτόν, όπως την επίλυση προβλημάτων. Καθώς η συγγραφή κώδικα και η εκτέλεση των εφαρμογών γίνεται σε ένα ασφαλές και ανέξοδο περιβάλλον, εφόσον χρησιμοποιείται λογισμικό μηδενικού κόστους, αλλά σε ασφαλές περιβάλλον, γίνεται κατανοητό ότι έχει ιδιαίτερη σημασία για την εκπαίδευση νέων προγραμματιστών και δημιουργών εφαρμογών, αλλά και για την απόκτηση γνώσεων σχετικών με τις σύγχρονες τεχνολογίες, όπως για την εξέλιξη γνώσεων που υπάρχουν ήδη.

Ωστόσο, παρατηρούνται θετικές αντιδράσεις και από το ευρύ καταναλωτικό κοινό που έρχεται σε επαφή με αυτές τις τεχνολογίες. Πιθανόν για κάποιους να είναι η είσοδος στη χρήση των νέων τεχνολογιών, η οποία προσεγγίζεται σε χαμηλότερο κόστος από άλλες, για παράδειγμα υπολογιστές, σταθερές συνδέσεις Internet κλπ. Ένα εξίσου σημαντικό όφελος αποτελεί το γεγονός ότι το λογισμικό που παρέχεται απαιτεί από το χρήστη να το εγκαταστήσει. Άρα. ο χρήστης μαθαίνει πλέον να αυτονομείται στη χρήση νέων τεχνολογιών. Επίσης, έρχεται σε επαφή με το ελεύθερο λογισμικό και πλέον αποκτά οντότητα μέσα στην κοινότητα που υποστηρίζει αυτές τις τεχνολογίες, καθώς είναι σε θέση να κρίνει εάν κάποια από αυτές έχει επιφέρει θετικά αποτελέσματα στην καθημερινή του ζωή,

Σε αυτό το σημείο έχουν γίνει κατανοητά τα εξής:

- Η δημιουργία λογισμικού για κινητές συσκευές Android είναι μία σχετικά εύκολη διαδικασία, η οποία όμως απαιτεί την κατανόηση της λειτουργίας της τεχνολογίας αυτής και των δυνατοτήτων που προσφέρει..
- Οι τεχνολογίες που μπορούν να το υποστηρίξουν βασίζονται σε ανοιχτά πρότυπα. Η χρήση τους πλέον διαδίδεται όλο και περισσότερο και στους προσωπικούς υπολογιστές.
- Αποκτήσαμε πρακτική εμπειρία μελετώντας, αναλύοντας και αναπτύσσοντας περαιτέρω τη συγκεκριμένη εφαρμογή, η οποία αφορά ένα καίριο ζήτημα σχετικά με την ασφαλή διαμοίραση αρχείων χρησιμοποιώντας. Πολύ σημαντική ήταν η εξοικείωση με τον

αντικειμενοστραφή προγραμματισμό και τις γλώσσες Java, AIDL και xml, τα ανοιχτά πρότυπα και την κατανόηση της λειτουργίας και της προσφοράς Unix-like συστημάτων.

- Τέλος, το σημαντικότερο όλων πιστεύουμε ότι αποτελεί το τελικό προϊόν που παρέχει ο προγραμματιστής στον καταναλωτή και το οποίο με την εμπέδωση και έμπρακτη εφαρμογή όσων μελετήσαμε, συμβάλλει τελικά στη συνολική εμπειρία χρήστη.

5.2. Προτάσεις περαιτέρω ανάπτυξης

Η εφαρμογή που αναλύθηκε υλοποιεί ένα πολύ σημαντικό ζήτημα για το Android σύστημα: τη διαμοίραση αρχείων. Πλέον, μιλώντας για υπολογιστικό νέφος και διαμοιρασμό πόρων, και έχοντας ως βάση την ήδη έτοιμη εφαρμογή, μπορούμε να εφαρμόσουμε την αποκτηθείσα γνώση προσβλέποντας σε αυτόν το στόχο, το οποίο πιθανόν να αποτελέσει μελλοντική εργασία.

Ορισμοί

API (Application Programming Interface, Διεπαφή / Διασύνδεση Προγραμματισμού Εφαρμογών): ένα σύνολο από πρωτόκολλα, εργαλεία, κλάσεις, δομές δεδομένων και διαδικασίες για τη ανάπτυξη προϊόντων λογισμικού. Είναι, επίσης, η διεπαφή που ένα υπολογιστικό σύστημα, μία βιβλιοθήκη ή μία διαδικτυακή εφαρμογή / υπηρεσία παρέχει προκειμένου να μπορούν να γίνουν αιτήσεις από και προς αυτό/ή από άλλα προγράμματα. Διατυπώνει το σύνολο των λειτουργιών και υπηρεσιών που παρέχει το υπολογιστικό σύστημα / βιβλιοθήκη / υπηρεσία σε άλλα προγράμματα, χωρίς να χρειάζεται να γίνει κάποια αναφορά στον κώδικα που τα υλοποιεί.

Στο Android το API level είναι μία τιμή integer που είναι μοναδική και αντιστοιχεί σε ένα και μόνο API framework στην πλατφόρμα Android, μέσω του οποίου μπορούν οι εφαρμογές να αλληλεπιδράσουν με το σύστημα. Το API framework αποτελείται από πακέτα και κλάσεις, στοιχεία και χαρακτηριστικά xml για το αρχείο manifest και για τη δήλωση και πρόσβαση στους πόρους, ένα σύνολο Προθέσεων και ένα σύνολο αδειών. Παρέχει προς τα πίσω συμβατότητα, με τις νέες εκδόσεις να παρέχουν αυξημένη λειτουργικότητα, ενώ χαρακτηριστικά που δεν υποστηρίζονται πλέον ή έχουν αντικατασταθεί, στις περισσότερες περιπτώσεις δεν αφαιρούνται για την ομαλή λειτουργία των εφαρμογών. Άρα, ενώ κάθε έκδοση υποστηρίζει ένα και μόνο API level, στην πραγματικότητα υποστηρίζει και όλες τις προηγούμενες εκδόσεις.

Framework: πλαίσιο (λογισμικού / εφαρμογής). Στα αντικειμενοστραφή συστήματα αφορά ένα σύνολο κλάσεων το οποίο περιέχει ένα "αφηρημένο, γενικό και επαναχρησιμοποιήσιμο πρότυπο", το οποίο μπορεί επιλεκτικά να αλλάξει, εάν προστεθεί κώδικας ειδικά γραμμένος για τη συγκεκριμένη εφαρμογή, έτσι ώστε η τελική εφαρμογή και κατά συνέπεια και ο κώδικας αυτής να έχει αποκτήσει δικά του χαρακτηριστικά. Ένα πλαίσιο λογισμικού περιλαμβάνει μεταγλωττιστές, βιβλιοθήκες, εργαλεία και διεπαφές προγραμματισμού εφαρμογών (APIs). Επειδή η διαδικασία συγγραφής κώδικα είναι χρονοβόρα, είναι δυνατόν να χρησιμοποιηθεί ένας υπάρχων κώδικας, ο οποίος μπορεί να επεκταθεί, χωρίς όμως να παρουσιάζει σημαντικές αλλαγές με το αρχικό πρόγραμμα. Είναι ένας "σκελετός" πάνω στον οποίο θα δημιουργηθούν πιο ουσιαστικά και πιο ευέλικτα οι μετέπειτα εφαρμογές.

Kernel: ο πυρήνας του λειτουργικού συστήματος. Ένα πρόγραμμα υπολογιστή που αλληλεπιδρά με το υλικό του υπολογιστή, καθώς το συνδέει με το λογισμικό. Ο κώδικας είναι

γραμμένος σε γλώσσα χαμηλού επιπέδου που είναι πολύ κοντά σε γλώσσα μηχανής (π.χ. C) και συνδέεται με την κατανομή του χρόνου και της μνήμης στα προγράμματα, καθώς και τη διαχείριση των αρχείων και της επικοινωνίας μεταξύ υλικού - λογισμικού χρησιμοποιώντας τις κλήσεις συστήματος (system calls). Λόγω της σημαντικότητάς του, συνήθως βρίσκεται σε προστατευμένη περιοχή της μνήμης όπου δεν επιτρέπεται η εγγραφή (kernel space σε αντίθεση με το user space που αφορά τη διατιθέμενη περιοχή μνήμης που έχει ο χρήστης. Ο πυρήνας Linux για τα Unix-οειδή λειτουργικά συστήματα της οικογένειας Linux, διανέμεται υπό τη Γενική Άδεια Δημόσιας Χρήσης GNU έκδοση 2(GNU General Public License). Το Android βασίζεται στον πυρήνα Linux 2.6.25 και πλέον χρησιμοποιεί νεότερες εκδόσεις αυτού (π.χ. έκδοση 4.4.2 KitKat τρέχει πυρήνα 3.10) και εν αντιθέσει με τις υπόλοιπες διανομές Linux δεν παρέχει δικαιώματα διαχειριστή (root access) σε προστατευμένες περιοχές της μνήμης (μονολιθικός τύπος, τύπος τροποποιημένου πυρήνα Linux). Μικροπυρήνες (microkernels) έχουν αναπτυχθεί καθώς είναι ευκολότερη η συντήρησή τους, όπου εκτελούνται μόνο οι κυριότερες διεργασίες, όπως η διαχείριση της μνήμης, αλλά συστήματα που υποστηρίζονται από αυτούς πιθανόν να έχουν χαμηλότερη απόδοση λόγω της συχνότητας των κλήσεων συστήματος.

Middleware: ενδιάμεσο λογισμικό. Το περιβάλλον των κινητών συσκευών αποτελεί πρόκληση λόγω διαφόρων παραμέτρων, όπως περιορισμένη πόροι, μικρή μνήμη, διαφορά από συσκευή σε συσκευή, ιδιαίτερα στην πλατφόρμα Android, όπου πολλές διαφορετικές συσκευές λειτουργούν με Android αλλά παραμετροποιημένο διαφορετικά για κάθε μία από αυτές κτλ. Αυτά τα θέματα μπορεί να αποτελέσουν τροχοπέδη στην ανάπτυξη εφαρμογών μετατρέποντάς την σε μία χρονοβόρα και περίπλοκη διαδικασία. Το ενδιάμεσο λογισμικό διευκολύνει την ανάπτυξη των εφαρμογών απαλείφοντας κάποια από αυτά τα θέματα. Καθώς το περιβάλλον των κινητών συσκευών απαιτεί κάποια συγκεκριμένα χαρακτηριστικά, όπως προσβασιμότητα, προσαρμοστικότητα, αξιοπιστία, να μπορεί να επικοινωνεί άμεσα και να είναι καθολικά εφαρμόσιμο, για το λόγο αυτό είναι απαραίτητη η ύπαρξη του middleware. Συνήθως περιλαμβάνει λειτουργίες όπως την ασφάλεια, την αποθήκευση και το συγχρονισμό των δεδομένων, την απομακρυσμένη κλήση διαδικασίας (Remote Procedure Call, RPC) κ.ά. Στο Android οι υπηρεσίες εκτελούνται γρήγορα, καθώς οι βιβλιοθήκες στο ενδιάμεσο λογισμικό είναι σε γλώσσα μηχανής. Επίσης, επειδή εκτελούν διαδικασίες που σχετίζονται με το διαφορετικό περιβάλλον της κάθε συσκευής, δεν απασχολούν το πλαίσιο λογισμικού των εφαρμογών (Application Framework). Επίσης, περιέχει εκτός από τις βιβλιοθήκες (πράσινη

περιοχή), και το επίπεδο χρόνου εκτέλεσης (κίτρινη περιοχή) της στοίβας πρωτοκόλλων του Android, που διαθέτει τις βασικές βιβλιοθήκες και την εικονική μηχανή Dalvik.

Αρχείο .jar: Με την κατάληξη αυτή ορίζονται τα αρχεία που χρησιμοποιούνται για τη διανομή Java κλάσεων, μεταδεδομένων και πόρων, όπως εικόνες. Το format αυτό είναι βασισμένο στο ZIP format για τα συμπιεσμένα αρχεία όπου ένα αρχείο αυτής της μορφής περιέχει ένα ή περισσότερα διαφορετικών format. Ένα αρχείο .jar μπορεί να χρησιμοποιηθεί είτε για διανομή του πηγαίου κώδικα και των αρχείων που περιέχονται στην εφαρμογή είτε ήδη υπάρχοντα .jar να μπουν στην εφαρμογή βάζοντάς τα στο classpath, και χρησιμοποιώντας τις κλάσεις που περιέχουν. Η προέκταση jar προέρχεται από τις λέξεις Java Archive.

Διεπαφή (Interface): Η διεπαφή ορίζει τη συμπεριφορά που θα έχουν τα αντικείμενα που την χρησιμοποιούν, και ενώ συνήθως στη Java ορίζεται σε ένα ξεχωριστό αρχείο, μοιάζει με κλάση, αλλά είναι ένας τύπος αναφοράς, που περιέχει σταθερές (constants), υπογραφές μεθόδων, δηλαδή μέθοδοι που δεν έχουν σώμα, και ορίζονται ως στατικές (static) ή προεπιλεγμένες (default) και εμφωλευμένους τύπους. Δεν μπορούν να χρησιμοποιηθούν ως κλάσεις από άλλες κλάσεις και να δημιουργήσουν αντικείμενα. Μπορούν, όμως, να υλοποιηθούν από κλάσεις (implements) ή να επεκταθούν (extends) από άλλες διεπαφές.
Οι διεπαφές AIDL που ορίζονται στον κώδικα της εφαρμογής FileShare μοιάζουν με αυτές της γλώσσας προγραμματισμού Java, όμως διαφέρουν σε κάποια σημεία τα οποία έχουν αναλυθεί στην παρούσα εργασία. Μετατρέπονται σε διεπαφές Java όταν μεταγλωττίζονται από τον AIDL compiler.

Εικονική Μηχανή Java (JVM): Λογισμικό που εξαρτάται από την πλατφόρμα / συσκευή / επεξεργαστή που εκτελείται και για κάθε είδος από αυτά υπάρχει διαφορετική έκδοση, για παράδειγμα Dalvik Virtual Machine για συσκευές Android, ή εκδόσεις για Windows, Linux, παιχνιδομηχανές κτλ. *Ο πηγαίος κώδικας που έχει γραφτεί στη γλώσσα προγραμματισμού Java μεταγλωττίζεται με χρήση του μεταγλωττιστή javac, ο οποίος παράγει αρχεία με κατάληξη .class, που είναι τα αρχεία κώδικα byte ή bytecodes. Όταν η εφαρμογή θα εκτελεστεί, ανεξαρτήτως συσκευής ή πλατφόρμας, πρέπει να έχει εγκατασταθεί η εικονική μηχανή Java ή JVM. Αυτή θα διαβάσει τα αρχεία .class, θα τα μεταφράσει σε γλώσσα μηχανής, η οποία με τη σειρά της πρέπει να υποστηρίζεται από το εκάστοτε λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να μπορεί να εκτελεστεί. Πιο πρόσφατα, καινούριες εφαρμογές της JVM*

είναι σε θέση να μεταγλωττίζουν εκ των προτέρων τμήματα bytecodes σε κώδικα μηχανής (εγγενής κώδικας ή native code) για βελτίωση στην ταχύτητα.

Ελεύθερο Λογισμικό: Οι χρήστες έχουν την ελευθερία να εκτελούν για οποιοδήποτε σκοπό, να αντιγράφουν, να διανέμουν τα αντίγραφά του, να μελετούν τον τρόπο λειτουργίας του προγράμματος και να το προσαρμόζουν στις εκάστοτε ανάγκες τους, να τροποποιούν και να βελτιώνουν το εν λόγω λογισμικό και να δημοσιεύουν τις βελτιώσεις, ώστε να επωφεληθεί η κοινότητα. Δε χρειάζεται να υπάρχει εξουσιοδότηση ή πληρωμή για κάποια άδεια. Ολόκληρος ο ορισμός του Ελεύθερου Λογισμικού υπάρχει στη διεύθυνση http://www.gnu.org/philosophy/free-sw.html .

Κοινό κτήμα (public domain): Το έργο το οποίο δεν υπάγεται σε καθεστώς πνευματικής ιδιοκτησίας, είτε γιατί ο δημιουργός του έχει παραιτηθεί του δικαιώματος αυτού, είτε γιατί η χρονική διάρκεια προστασίας της πνευματικής του ιδιοκτησία έχει παρέλθει εξαιτίας του χρόνου που έχει περάσει από τη δημιουργία τους, είτε γιατί δεν εννοείται ότι υπάγεται σε προστασία πνευματικής ιδιοκτησίας, όπως για παράδειγμα οι νόμοι της φυσικής. Καθώς το δικαίωμα αυτό μπορεί να διαφέρει από χώρα σε χώρα, μπορεί κάποιο πνευματικό έργο σε μία χώρα να αποτελεί κοινό κτήμα, ενώ σε μία άλλη να προστατεύεται από νόμους πνευματικής ιδιοκτησίας.

Πλατφόρμα: Περιλαμβάνει το υλικό, το λογισμικό και τις αντίστοιχες βιβλιοθήκες.

Πρότυπο ανοιχτού κώδικα: Λογισμικό ανοιχτού κώδικα, του οποίο ο πηγαίος κώδικας είναι διαθέσιμος και η άδεια του οποίου αναφέρει ότι ο δημιουργός (copyright holder) παρέχει το δικαίωμα σε τρίτους της μελέτης, μεταβολής και διανομής του κώδικα για οποιοδήποτε σκοπό.

Πρότυπο κλειστού κώδικα: Η άδεια που δίνεται από το δημιουργό στο χρήστη τον υποχρεώνει να χρησιμοποιεί το λογισμικό υπό συγκεκριμένες συνθήκες και δεν μπορεί να το χρησιμοποιήσει, το λογισμικό ή τον πηγαίο κώδικά του, για μελέτη, να το αλλάξει / επεξεργαστεί, να το αναδιαθέσει / διαμοιράσει, ή να το αναλύσει. Συνήθως ο πηγαίος κώδικας λογισμικού κλειστού τύπου δε διατίθεται στο ευρύ κοινό.

Βιβλιογραφία

- [1] www.androidgreece.gr
- [2] developer.android.com
- [3] "Google's iron grip on Android: Controlling open source by any means necessary". Ars Technica. Retrieved 2013-12-08.
- [4] http://www.vogella.com/
- [5] Hello, Android (3rd edition): Introducing Google's Mobile Development Platform, by Ed Burnette
- [6] Charlie Collins, Michael Galpin and Matthias Kaeppler (2011), Android in Practice, Manning Publications.
- [7] http://swift.siphos.be/linux_sea/whatislinux.htm
- [8] www.apache.org/licenses
- [9] http://www.tldp.org/
- [10] Linux®: Asymmetric Multiprocessing (ASMP) Versus Symmetric Multiprocessing (SMP), V1.0, Ted Peters
- [11] http://www.ibm.com/developerworks/library/l-linux-smp/
- [12] Operating System Concepts 9th edition, Abraham Silberschatz. "Chapter 4: Threads"
- [13] Χρήστος Καρασούλης - Android in Greece - Διαφορές Linux Kernel και Android Linux Kernel
- [14] Todd Greanier, Java, Εισαγωγή στη Σύγχρονη Τεχνολογία, εκδόσεις: Μ. Γκιούρδας.
- [15] Rogers Cadenhead & Laura Lemay, Πλήρες Εγχειρίδιο της Java™ 2, εκδότης: Μ. Γκιούρδας.
- [16] http://www.oracle.com/
- [17] https://docs.oracle.com/javase/tutorial/
- [18] https://class.coursera.org/android-001/lecture/13
- [19] source.android.com/source/licences.html
- [20] https://class.coursera.org/posa-002/lecture/179 "AIDL Syntax and Supported DataTypes"
- [21] https://class.coursera.org/posa-002/lecture/181 "Implementing AIDL Interfaces"
- [22] http://www.slideshare.net/jserv/android-ipc-mechanism

- [23] www.sqlite.org

Κώδικες που μελετήθηκαν

- http://code.google.com/p/android-fileshare/source/browse/tags/android-fileshare-1-0b-beta/AndroidManifest.xml
- http://www.androidhive.info/2013/11/android-working-with-action-bar/
- http://www.vogella.com/tutorials/AndroidActionBar/article.html
- http://www.vogella.com/code
- http://developer.android.com/guide/topics/ui/actionbar.html
- docs.oracle.com/javase/tutorial/java/index.html

Προγράμματα που χρησιμοποιήθηκαν

- ADT Bundle
- Easy Screenshot for Android
- Eclipse
- Gimp (Linux)
- Illustrator
- Libre Office Writer
- Microsoft OfficeWord
- Photoshop

Παράρτημα Ι
Κώδικας της Εφαρμογής

FileShare\src\com\dimitra\dipl\fileshare\CookiesDatabaseOpenHelper.java

```
package com.dimitra.dipl.fileshare;

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

public class CookiesDatabaseOpenHelper extends SQLiteOpenHelper {

  public CookiesDatabaseOpenHelper(Context context) {
    super(context, "cookies.db", null, 1);
  }

  @Override
  public void onCreate(SQLiteDatabase database) {
    database.execSQL(
      "CREATE TABLE cookies(name STRING, value STRING, expiry INTEGER)");
  }

  @Override
  public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
    // TODO Auto-generated method stub

  }

}
```

FileShare\src\com\dimitra\dipl\fileshare\FileBrowser.java

```
package com.dimitra.dipl.fileshare;


import java.util.Stack;


import com.dimitra.dipl.fileshare.R;


import android.app.AlertDialog;

import android.app.Dialog;

import android.app.ListActivity;

import android.content.Intent;

import android.database.Cursor;
```

```
import android.database.MatrixCursor;
import android.database.MergeCursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.OpenableColumns;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
import android.widget.AdapterView.AdapterContextMenuInfo;



/**
 * Browser and picker for files on device.
 *
 * @author Nav Jagpal (nav@gmail.com)
 *
 * Current distribution was modified by
 * @author  Dimitra Mazaraki
 * dimitramaz@yahoo.gr
 *
 * The changes / additions made by Dimitra Mazaraki are marked by #Dimitra#
 */
public class FileBrowser extends ListActivity {

  /* Keep the paths so we can go back */
  private Stack<String> mPaths = new Stack<String>();

  private static final int EMPTY_DIRECTORY_DIALOG = 0;
```

```
private static final int MENU_SHARE_ALL = 0;

@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  /* If no data is supplied, get default from the FileProvider */
  if (getIntent().getData() == null) {
    getIntent().setData(FileProvider.CONTENT_URI);
  }

  String currentPath = FileProvider.getPath(getIntent().getData());

  /* Special case for "/". Wouldn't need this is we have a class to
   * do path joining for us */
  if (currentPath.equals("/"))
    currentPath = "";
  mPaths.push(currentPath);

  /* Populate list with the root content */
  Cursor c = managedQuery(getIntent().getData(), null, null, null, null);
  changeCursor(c);

  registerForContextMenu(getListView());
}

@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
  TextView textView = (TextView) v;

  /* Path will contain the directory or file the user actually wants to see */
  String path = "";
```

```
String fileName = (String) textView.getText();

/* This may be a ".." signal to go back */
if (fileName.equals("..")) {
  path = mPaths.pop();
  if (mPaths.size() > 0)
    path = mPaths.pop();
  fileName = "";
} else {
  /* Normal processing for non ".." names */
  path = mPaths.peek() + "/" + fileName;
}

/* Special processing for "/" */
if (path.equals("/"))
  path = "";

/* Create a content uri using the path. Normally this is a numeric id, but the file provider
 * is different.
 */
Uri uri = Uri.withAppendedPath(getIntent().getData(), path);
if (getContentResolver().getType(uri).equals(FileProvider.CONTENT_ITEM_TYPE)) {
  /* User has selected a file */
  setResult(RESULT_OK, new Intent().setData(uri));
  finish();
} else {
  /* A directory */
  Cursor c = managedQuery(uri, null, null, null, null);

  /* No files, show dialog */
  if (c.getCount() == 0) {
    this.showDialog(EMPTY_DIRECTORY_DIALOG);
```

```
    } else {
      /* We have some files. Add current path to stack so we can get back
       * Add ".." if we're not looking at the root
       */
      mPaths.push(path);
      if (mPaths.size() > 1) {
        MatrixCursor matrixCursor = new MatrixCursor(
            new String[] {"_id", OpenableColumns.DISPLAY_NAME});
        matrixCursor.addRow(new Object[] {-1, ".."});
        c = new MergeCursor(new Cursor[] {matrixCursor, c});
      }
      changeCursor(c);
    }
  }
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenu.ContextMenuInfo menuInfo) {
  menu.add(0, MENU_SHARE_ALL, 0, R.string.share_all);
}

@Override
public boolean onContextItemSelected(MenuItem item) {
  AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
  switch (item.getItemId()) {
    case MENU_SHARE_ALL:
      Cursor c = (Cursor) getListView().getItemAtPosition(
          info.position);
      c.moveToPosition(info.position);
      String path = mPaths.peek() + "/" + c.getString(
          c.getColumnIndex(OpenableColumns.DISPLAY_NAME));
```

```
      Uri uri = Uri.withAppendedPath(getIntent().getData(), path);
      setResult(RESULT_OK, new Intent().setData(uri));
      finish();
      break;
    }
    return false;
  }

  @Override
  protected Dialog onCreateDialog(int id) {
    switch (id) {
      case EMPTY_DIRECTORY_DIALOG:
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setCancelable(true).setMessage(R.string.empty_directory);
        return builder.create();
    }
    return null;
  }

  private void changeCursor(Cursor cursor) {
    ListAdapter adapter = new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_1, cursor,
        new String[] { OpenableColumns.DISPLAY_NAME },
        new int[] { android.R.id.text1 });
    setListAdapter(adapter);
  }

}
```

FileShare\src\com\dimitra\dipl\fileshare\FileProvider.java

```
package com.dimitra.dipl.fileshare;
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.List;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;
import android.os.ParcelFileDescriptor;
import android.provider.OpenableColumns;

/**
 * @author Nav Jagpal (nav@gmail.com)
 *
 *
 * Current distribution was modified by
 * @author  Dimitra Mazaraki
 * dimitramaz@yahoo.gr
 *
 * The changes / additions made by Dimitra Mazaraki are marked by #Dimitra#
 */

public class FileProvider extends ContentProvider {

        /* #Dimitra# */
  public static final String AUTHORITY = "com.dimitra.dipl.filesharer.FileProvider";
  public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY  + "/files");
```

```
  public static final String CONTENT_TYPE = "vnd.android.cursor.dir/vnd.dimitra.dipl.file";
  public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.dimitra.dipl.file";

  @Override
  public boolean onCreate() {
    // TODO Auto-generated method stub
    return false;
  }

  @Override
  public String getType(Uri uri) {
    String path = getPath(uri);
    File file = new File(path);
    if (file.isDirectory())
      return CONTENT_TYPE;
    else if (file.isFile())
      return CONTENT_ITEM_TYPE;

    return null;
  }

  @Override
  public Uri insert(Uri uri, ContentValues values) {
    // TODO Auto-generated method stub
    return null;
  }

  @Override
  public int delete(Uri uri, String selection, String[] selectionArgs) {
    throw new UnsupportedOperationException("Deletes are not supported.");
  }
```

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
    String[] selectionArgs, String sortOrder) {

  return getCursorForFiles(getPath(uri));
}

@Override
public int update(Uri uri, ContentValues values, String selection,
    String[] selectionArgs) {
  throw new UnsupportedOperationException("Updates are not supported.");
}

/* Add a row to the matrix cursor */
private void addRow(Collection<Object[]> fileList, File file, int id,
                    String path) {
  String fileName = file.getName();
  int fileSize = (int)file.length();
  fileList.add(new Object[] {id, fileName, fileSize, path});
}

/* Return a cursor for files in the specified path */
private Cursor getCursorForFiles(String path) {
  /* _id doesn't really mean anything but the system really wants it */
  String[] columns = {
     "_id",
     OpenableColumns.DISPLAY_NAME,
     OpenableColumns.SIZE,
     "_data"
  };
```

```
  File baseDir = new File(path);
  LinkedList<Object[]> fileList = new LinkedList<Object[]>();
  if (baseDir.isDirectory()) {
    File[] files = baseDir.listFiles();
    int id = 0;
    for (File file: files) {
      addRow(fileList, file, id, file.getAbsolutePath());
      id++;
    }
  } else if (baseDir.isFile()) {
    addRow(fileList, baseDir, 0, baseDir.getAbsolutePath());
  }

  // Sort listing by display name.
  Collections.sort(fileList, new Comparator<Object[]>() {
        public int compare(Object[] r1, Object[] r2) {
                return ((String) r1[1]).compareTo((String) r2[2]);
        }
  });

  MatrixCursor c = new MatrixCursor(columns);
  for (Object[] row : fileList) {
        c.addRow(row);
  }
  return c;
}

@Override
public ParcelFileDescriptor openFile(Uri uri, String mode)
  throws FileNotFoundException, SecurityException  {
  return openFileHelper(uri, mode);
}
```

```
  /* Construct a file path based on the uri */
  public static String getPath(Uri uri) {
    List<String> segments = uri.getPathSegments();

    String path = "";
    for (String segment: segments) {
      if (segment.equals("files"))
        continue;
      path += "/" + segment;
    }
    if (path == "")
      path = "/";

    return path;
  }


}
```

FileShare\src\com\dimitra\dipl\fileshare\FileShare.java

```
package com.dimitra.dipl.fileshare;

import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.UnknownHostException;

import com.dimitra.dipl.fileshare.R;

import android.annotation.SuppressLint;
```

```
import android.app.ActionBar;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.SearchManager;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.database.Cursor;
import android.database.SQLException;
import android.net.Uri;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.provider.OpenableColumns;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.SearchView;
import android.widget.TextView;
import android.widget.ToggleButton;
```

```
/**
 * @author Nav Jagpal (nav@gmail.com)
 *
 *
 * Current distribution was modified by
 * @author  Dimitra Mazaraki
 * dimitramaz@yahoo.gr
 *
 * The changes / additions made by Dimitra Mazaraki are marked by #Dimitra#
 */

public class FileShare extends Activity implements ActionBar.OnNavigationListener{

private static final String TAG = "FileSharer";

  /* startActivity request codes */
  private static final int PICK_FILE_REQUEST = 1;
  private static final int PICK_FOLDER_REQUEST = 2;

  /* Used to keep track of the currently selected file */
  private Uri mFileToShare;

  /*
   * #Dimitra#
   *
   * these 2 are added after following the
   * action bar
   */
  //action bar --mimina

  private ActionBar actionBar;
```

```
  //Refresh  menu item
  private MenuItem refreshMenuItem;

  private static final int DIALOG_PASSWORD = 0;

  private final View.OnClickListener mAddFileListener = new View.OnClickListener() {
    public void onClick(View v) {
      Intent pickFileIntent = new Intent();
      pickFileIntent.setAction(Intent.ACTION_GET_CONTENT);
      pickFileIntent.addCategory(Intent.CATEGORY_OPENABLE);
      pickFileIntent.setType("*/*");
      Intent chooserIntent = Intent.createChooser(pickFileIntent,
          getText(R.string.choosefile_title));
      startActivityForResult(chooserIntent, PICK_FILE_REQUEST);
    }
  };

  private final View.OnClickListener mManageContentListener = new
View.OnClickListener() {
    public void onClick(View v) {
      Intent manageIntent = new Intent();
      manageIntent.setAction(Intent.ACTION_MAIN);
      manageIntent.setType(FileSharingProvider.Folders.CONTENT_TYPE);
      startActivity(manageIntent);
    }
  };

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);
```

```
/* #Dimitra# */
//get the Action Bar
ActionBar actionBar = getActionBar();

/* Startup the FileSharingService, unless the last state was off. */
final SharedPreferences sharedPreferences = getSharedPreferences(
      FileSharingService.PREFS_NAME, MODE_PRIVATE);
ToggleButton serviceButton = (ToggleButton) findViewById(R.id.service);
if (sharedPreferences.getBoolean(
            FileSharingService.PREFS_SERVICE_ON_STARTUP, true)) {
      Intent serviceIntent = new Intent();
      serviceIntent.setAction("com.dimitra.dipl.filesharer.IFileSharingService");
      startService(serviceIntent);
      serviceButton.setChecked(true);
} else {
      serviceButton.setChecked(false);
}

/* Setup toggling the service. */
serviceButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean newValue) {
                  Intent serviceIntent = new Intent();
               serviceIntent.setAction("com.dimitra.dipl.filesharer.IFileSharingService");
                  if (newValue == true) {
                     startService(serviceIntent);
                  } else {
                           stopService(serviceIntent);
                  }
               SharedPreferences.Editor editor = sharedPreferences.edit();
               editor.putBoolean(FileSharingService.PREFS_SERVICE_ON_STARTUP,
```

```
newValue);
                    editor.commit();
                }
    });

    /* Add the add file to shared folder */
    Button addFileButton = (Button) findViewById(R.id.addfile);
    addFileButton.setOnClickListener(mAddFileListener);

    /* Manage content button */
    Button manageButton = (Button) findViewById(R.id.manage);
    manageButton.setOnClickListener(mManageContentListener);

    /* Setup the status text */
    TextView ipTextView = (TextView) findViewById(R.id.url);
    ipTextView.setText("http://" + getIPAddress(this) + ":9999");

    /* Preferences button */
    Button preferencesButton = (Button) findViewById(R.id.preferences);
    preferencesButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        AlertDialog.Builder builder = new AlertDialog.Builder(FileShare.this);

        builder.setTitle("Preferences");
        builder.setMultiChoiceItems(new CharSequence[] { "Allow Uploads",
        "Require Password" }, new boolean[] {
            sharedPreferences.getBoolean(
                FileSharingService.PREFS_ALLOW_UPLOADS, false),
                sharedPreferences.getBoolean(
                    FileSharingService.PREFS_REQUIRE_LOGIN, false) },
                    new DialogInterface.OnMultiChoiceClickListener() {
          public void onClick(DialogInterface dialog,
```

```
            int item, boolean value) {
          if (item == 0) {
            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.putBoolean(FileSharingService.PREFS_ALLOW_UPLOADS,
                value);
            editor.commit();
          } else if (item == 1) {
            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.putBoolean(FileSharingService.PREFS_REQUIRE_LOGIN,
                value);
            editor.commit();
            /* If user has enabled the password feature,
             * show the password dialog. */
            if (value == true) {
              showDialog(DIALOG_PASSWORD);
            }
          }
        }
      });
      builder.setPositiveButton("Done", null);
      builder.show();
    }
  });

  /* Help button */
  Button helpButton = (Button) findViewById(R.id.help);
  helpButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
      Intent helpIntent = new Intent();
      helpIntent.setClass(FileShare.this, Help.class);
      startActivity(helpIntent);
    }
```

```
    });

    /* Display a "What's New" dialog if necessary. */
    PackageManager pm = getPackageManager();
    try {
      PackageInfo pi = pm.getPackageInfo("com.dimitra.dipl.fileshare", 0);
      if (!sharedPreferences.getBoolean(pi.versionName, false)) {
        sharedPreferences.edit().putBoolean(pi.versionName, true).commit();
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.whats_new_title);
        builder.setMessage(R.string.whats_new);
        builder.setPositiveButton("OK", null);
        builder.show();
      }
    } catch (NameNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

  /* #Dimitra# */
  //the Action Bar method #1
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
     // Inflate the menu; this adds items to the action bar if it is present.
     MenuInflater inflater = getMenuInflater();
          getMenuInflater().inflate(R.menu.activity_main_actions, menu);

          //associate searchable configuration w/ the SearchView
          SearchManager searchManager = (SearchManager)
getSystemService(Context.SEARCH_SERVICE);
          SearchView searchView = (SearchView) menu.findItem(R.id.action_search)
```

```
                        .getActionView();
        searchView.setSearchableInfo(searchManager
                        .getSearchableInfo(getComponentName()));

   return super.onCreateOptionsMenu(menu);
}

/**
 * On selecting action bar icons
 **/
/* #Dimitra# */
//the Action Bar method #2
@Override
public boolean onOptionsItemSelected(MenuItem item){
        int itemId = item.getItemId();
        if (itemId == R.id.action_search) {
                //search action
                  return true;
        } else if (itemId == R.id.action_refresh) {
                //refresh
                //added after the orogressBar
                refreshMenuItem = item;
                //load the date from server
                new SyncData().execute();
                  return true;
        } else if (itemId == R.id.action_help) {
                return true;
        } else {
                return super.onOptionsItemSelected(item);
        }
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (resultCode != RESULT_OK)
    return;

  switch (requestCode) {
    case PICK_FILE_REQUEST:
      /* Store this file somewhere */
      mFileToShare = data.getData();

      /* Now pick a folder */
      Intent pickFolder = new Intent();
      pickFolder.setAction(Intent.ACTION_PICK);
      pickFolder.setType(FileSharingProvider.Folders.CONTENT_ITEM_TYPE);
      startActivityForResult(pickFolder, PICK_FOLDER_REQUEST);
      break;
    case PICK_FOLDER_REQUEST:
      addFileToFolder(mFileToShare, data.getData());
      break;

  }
}

/**
 * Adds a file to a shared folder. If the provided file is actually a
 * folder, all files under that folder will be added to the shared folder.
 * This includes files from sub-directories as well.
 *
 * @param file Uri for file.
 * @param folder Uri for shared folder.
 */
```

```
private void addFileToFolder(Uri file, Uri folder) {
  /* The URI could be a folder. If it is, assume we want all files under that
   * folder.
   */
  if (getContentResolver().getType(file).equals(
      FileProvider.CONTENT_TYPE)) {
    Cursor c = managedQuery(
        file, new String[] {OpenableColumns.DISPLAY_NAME},
        null, null, null);
    while (c.moveToNext()) {
      String filename = c.getString(c.getColumnIndex(
          OpenableColumns.DISPLAY_NAME));
      Uri uri = Uri.withAppendedPath(
          file, filename);
      addFileToFolder(uri, folder);
    }
  } else {
    try {
      FileSharingProvider.addFileToFolder(getContentResolver(), file, folder);
    } catch (SQLException exception) {
      Log.w(TAG, "Error adding file " + file + " to folder " + folder);
    }
  }
}

public static String getIPAddress(Context context) {
  WifiManager wifiManager = (WifiManager) context
  .getSystemService(Context.WIFI_SERVICE);
  android.net.wifi.WifiInfo info = wifiManager.getConnectionInfo();
  ByteArrayOutputStream bos = new ByteArrayOutputStream();
  DataOutputStream dos = new DataOutputStream(bos);
```

```
  byte[] intByte;
  try {
    dos.writeInt(info.getIpAddress());
    dos.flush();
    intByte = bos.toByteArray();
  } catch (IOException e) {
    Log.e(TAG, "Problem converting IP address");
    return "unknown";
  }

  // Reverse int bytes.. damn, this is a hack.
  byte[] addressBytes = new byte[intByte.length];
  for (int i = 0; i < intByte.length; i++) {
    addressBytes[i] = intByte[(intByte.length - 1) - i];
  }

  InetAddress address = null;
  try {
    address = InetAddress.getByAddress(addressBytes);
  } catch (UnknownHostException e) {
    Log.e(TAG, "Problem determing IP address");
    return "unknown";
  }
  return address.getHostAddress();
}

@Override
protected Dialog onCreateDialog(int id) {
  Dialog dialog;
  switch (id) {
    case DIALOG_PASSWORD:
      dialog = new Dialog(FileShare.this);
```

```
    dialog.setContentView(R.layout.password_dialog);
    dialog.setTitle(R.string.set_password_title);
    final EditText passwordText = (EditText) dialog
    .findViewById(R.id.password);
    passwordText.setText(getSharedPreferences(FileSharingService.PREFS_NAME,
       MODE_PRIVATE).getString(FileSharingService.PREFS_PASSWORD, ""));
    Button okButton = (Button) dialog.findViewById(R.id.ok_button);
    final Dialog passwordDialog = dialog;
    okButton.setOnClickListener(new View.OnClickListener() {
     public void onClick(View view) {
       SharedPreferences preferences = getSharedPreferences(
          FileSharingService.PREFS_NAME, MODE_PRIVATE);
       String newPassword = passwordText.getText().toString();
       if (!newPassword.equals(preferences.getString(
          FileSharingService.PREFS_PASSWORD, ""))) {
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString(FileSharingService.PREFS_PASSWORD, passwordText
           .getText().toString());
        editor.commit();
        /* The password has changed, delete all cookies. */
        new CookiesDatabaseOpenHelper(FileShare.this).getWritableDatabase()
        .delete("cookies", null, null);
       }
       passwordDialog.dismiss();
     }
    });
    break;
   default:
    dialog = null;
  }
  return dialog;
}
```

```
@Override
public boolean onNavigationItemSelected(int itemPosition, long itemId) {
	// TODO Auto-generated method stub
	return false;
}

/**
 * Async task to load
 * the data from server
 */
private class SyncData extends AsyncTask<String, Void, String> {


	@SuppressLint("NewApi")
	@Override
	protected void onPreExecute(){
		//set the progress bar view
		refreshMenuItem.setActionView(R.layout.action_progressbar);

		refreshMenuItem.expandActionView();
	}

	@Override
	protected String doInBackground(String... params) {
		// TODO Auto-generated method stub
		//not making a real request in this prj
		//for now we just use a timer to wait for sometime
		try{
			Thread.sleep(3000);
		}catch(InterruptedException ie){
			ie.printStackTrace();
```

```
                }
                return null;
        }

        @SuppressLint("NewApi")
        @Override
        protected void onPostExecute(String result){
                refreshMenuItem.collapseActionView();
                //remove the progressbar view
                refreshMenuItem.setActionView(null);
        }

}
}
```

FileShare\src\com\dimitra\dipl\fileshare\FileSharingProvider.java

```
package com.dimitra.dipl.fileshare;

import java.util.HashMap;

import android.content.ContentProvider;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
```

```
import android.provider.BaseColumns;
import android.provider.OpenableColumns;
import android.util.Log;



/**
 * @author Nav Jagpal (nav@gmail.com)
 *
 *
 * Current distribution was modified by
 * @author  Dimitra Mazaraki
 * dimitramaz@yahoo.gr
 *
 * The changes / additions made by Dimitra Mazaraki are marked by #Dimitra#
 */

public class FileSharingProvider extends ContentProvider {

        /*  #Dimitra# */
  private static final String AUTHORITY = "com.dimitra.dipl.filesharer.FileSharingProvider";

  private static final String TAG = "FileSharingProvider";

  private static final String DATABASE_NAME = "file_sharer.db";
  private static final int DATABASE_VERSION = 2;
  private static final String FOLDERS_TABLE_NAME = "folders";
  private static final String FILES_TABLE_NAME = "files";

  /* Internal codes for dealing with different types */
  private static final int FOLDERS = 1;
  private static final int FILE = 2;
  private static final int FOLDER = 3;
```

```
  private static final int FILES = 4;

  private static HashMap<String, String> sFoldersProjectionMap;
  private static HashMap<String, String> sFilesProjectionMap;

  private static final UriMatcher sUriMatcher;

  /* Folder related constants */
  public interface Folders {

    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY +
"/folders");

    /* #Dimitra# */
    public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.dimitra.dipl.sharedfolder";
    public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.dimitra.dipl.sharedfolder";

    public interface Columns extends BaseColumns, OpenableColumns {
      public static final String PASSWORD = "password";
    }
  }

  /* File related constants */
  public interface Files {
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/files");

    /* #Dimitra# */
    public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.dimitra.dipl.sharedfile";
    public static final String CONTENT_ITEM_TYPE =
```

```
"vnd.android.cursor.item/vnd.dimitra.dipl.sharedfile";

  public interface Columns extends BaseColumns, OpenableColumns {
    public static final String _DATA = "_data";
    public static final String FOLDER_ID = "folder_id";
  }
 }

 /**
  * This class helps open, create, and upgrade the database file.
  */
 private static class DatabaseHelper extends SQLiteOpenHelper {

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

  @Override
  public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + FOLDERS_TABLE_NAME + " ("
        + Folders.Columns._ID + " INTEGER PRIMARY KEY,"
        + Folders.Columns.DISPLAY_NAME + " TEXT,"
        + Folders.Columns.PASSWORD + " TEXT"
        + ");");
    db.execSQL("CREATE TABLE " + FILES_TABLE_NAME + " ("
        + Files.Columns._ID + " INTEGER PRIMARY KEY,"
        + Files.Columns.FOLDER_ID + " INTEGER,"
        + Files.Columns.DISPLAY_NAME + " TEXT,"
        + Files.Columns._DATA + " TEXT"
        + ");");
    db.execSQL("INSERT INTO " + FOLDERS_TABLE_NAME + " VALUES(0, 'Public',
null)");
```

```
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
      + newVersion + ", which will destroy all old data");

  /* Move existing tables out of the way. */
  db.execSQL("ALTER TABLE " + FOLDERS_TABLE_NAME
      + " RENAME TO " + FOLDERS_TABLE_NAME + "_TEMP");
  db.execSQL("ALTER TABLE " + FILES_TABLE_NAME
      + " RENAME TO " + FILES_TABLE_NAME + "_TEMP");

  /* Create the new tables. */
  createTables(db, FOLDERS_TABLE_NAME, FILES_TABLE_NAME);

  /* Copy data from old tables to new tables. */
  Cursor c = db.query(
      FOLDERS_TABLE_NAME + "_TEMP",
      null, null, null, null, null, null);
  while (c.moveToNext()) {
    ContentValues values = new ContentValues();
    values.put(Folders.Columns._ID, c.getInt(c.getColumnIndex(
        Folders.Columns._ID)));
    values.put(Folders.Columns.DISPLAY_NAME, c.getString(c.getColumnIndex(
        Folders.Columns.DISPLAY_NAME)));
    values.put(Folders.Columns.PASSWORD, c.getString(c.getColumnIndex(
        Folders.Columns.PASSWORD)));
    db.insert(FOLDERS_TABLE_NAME, Folders.Columns.DISPLAY_NAME, values);
  }

  c.close();
```

```
    c = db.query(
        FILES_TABLE_NAME + "_TEMP",
        null, null, null, null, null, null);
    while (c.moveToNext()) {
      ContentValues values = new ContentValues();
      values.put(Files.Columns._ID, c.getInt(c.getColumnIndex(
          Files.Columns._ID)));
      values.put(Files.Columns.FOLDER_ID, c.getInt(c.getColumnIndex(
          Files.Columns.FOLDER_ID)));
      values.put(Files.Columns.DISPLAY_NAME, c.getString(c.getColumnIndex(
          Files.Columns.DISPLAY_NAME)));
      values.put(Files.Columns._DATA, c.getString(c.getColumnIndex(
          Files.Columns._DATA)));
      db.insert(FILES_TABLE_NAME, Files.Columns.DISPLAY_NAME, values);
    }
    c.close();

    /* Delete the temp tables. */
    db.execSQL("DROP TABLE " + FOLDERS_TABLE_NAME + "_TEMP");
    db.execSQL("DROP TABLE " + FILES_TABLE_NAME + "_TEMP");
  }

  private static void createTables(
      SQLiteDatabase db, String foldersTable, String filesTable) {
    db.execSQL("CREATE TABLE " + foldersTable + " ("
        + Folders.Columns._ID + " INTEGER PRIMARY KEY,"
        + Folders.Columns.DISPLAY_NAME + " TEXT,"
        + Folders.Columns.PASSWORD + " TEXT,"
        + "UNIQUE (" + Folders.Columns.DISPLAY_NAME + ")"
        + ");");
    db.execSQL("CREATE TABLE " + filesTable + " ("
        + Files.Columns._ID + " INTEGER PRIMARY KEY,"
```

```
        + Files.Columns.FOLDER_ID + " INTEGER,"
        + Files.Columns.DISPLAY_NAME + " TEXT,"
        + Files.Columns._DATA + " TEXT,"
        + "UNIQUE (" + Files.Columns.FOLDER_ID + ","
        + Files.Columns._DATA + ")"
        + ");");
  }
}

private DatabaseHelper mOpenHelper;

@Override
public boolean onCreate() {
  mOpenHelper = new DatabaseHelper(getContext());
  return true;
}

@Override
public String getType(Uri uri) {
  switch (sUriMatcher.match(uri)) {
    case FOLDERS:
      return Folders.CONTENT_TYPE;
    case FOLDER:
      return Folders.CONTENT_ITEM_TYPE;
    case FILES:
      return Files.CONTENT_TYPE;
    case FILE:
      return Files.CONTENT_ITEM_TYPE;
    default:
      throw new IllegalArgumentException("Unknown URI " + uri);
  }
}
```

```
 @Override
 public Uri insert(Uri uri, ContentValues values) {
  // Validate the requested uri
  if (getType(uri).equals(Folders.CONTENT_TYPE)) {

   if (values.containsKey(Folders.Columns.DISPLAY_NAME) == false) {
    throw new IllegalArgumentException("Name required");
   }

   SQLiteDatabase db = mOpenHelper.getWritableDatabase();
   long rowId = db.insert(FOLDERS_TABLE_NAME,
Folders.Columns.DISPLAY_NAME, values);
   if (rowId > 0) {
    Uri folderUri = ContentUris.withAppendedId(Folders.CONTENT_URI, rowId);
    getContext().getContentResolver().notifyChange(folderUri, null);
    return folderUri;
   }

  } else if (getType(uri).equals(Files.CONTENT_TYPE)) {
   if (!values.containsKey(Files.Columns.FOLDER_ID)) {
    throw new IllegalArgumentException("Folder id required");
   }
   if (!values.containsKey(Files.Columns.DISPLAY_NAME)) {
    throw new IllegalArgumentException("File name required");
   }
   if (!values.containsKey(Files.Columns._DATA)) {
    throw new IllegalArgumentException("Data URI required");
   }

   SQLiteDatabase db = mOpenHelper.getWritableDatabase();
   long rowId = db.insert(FILES_TABLE_NAME, Files.Columns.DISPLAY_NAME,
```

```
values);
    if (rowId >= 0) {
      Uri fileUri = ContentUris.withAppendedId(Files.CONTENT_URI, rowId);
      getContext().getContentResolver().notifyChange(fileUri, null);
      Log.i(TAG, "Inserted row " + fileUri.toString());
      return fileUri;
    }
  }
  throw new SQLException("Failed to insert row into " + uri);
 }

 @Override
 public int delete(Uri uri, String selection, String[] selectionArgs) {
  SQLiteDatabase db = mOpenHelper.getWritableDatabase();
  int rowsDeleted = 0;
  if (getType(uri).equals(Files.CONTENT_TYPE)) {
   rowsDeleted += db.delete(FILES_TABLE_NAME, selection, null);
  } else if (getType(uri).equals(Folders.CONTENT_TYPE)) {
   rowsDeleted += db.delete(FOLDERS_TABLE_NAME, selection, null);
  }
  return rowsDeleted;
 }

 @Override
 public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
    String sortOrder) {
  SQLiteQueryBuilder qb = new SQLiteQueryBuilder();

  switch (sUriMatcher.match(uri)) {
   case FOLDERS:
    qb.setTables(FOLDERS_TABLE_NAME);
    qb.setProjectionMap(sFoldersProjectionMap);
```

```
      break;
    case FOLDER:
      qb.setTables(FOLDERS_TABLE_NAME);
      qb.setProjectionMap(sFoldersProjectionMap);
      qb.appendWhere(Folders.Columns._ID + "=" + uri.getPathSegments().get(1));
      break;
    case FILES:
      qb.setTables(FILES_TABLE_NAME);
      qb.setProjectionMap(sFilesProjectionMap);
      break;
    case FILE:
      qb.setTables(FILES_TABLE_NAME);
      qb.setProjectionMap(sFilesProjectionMap);
      qb.appendWhere(Files.Columns._ID + "=" + uri.getPathSegments().get(1));
      break;
    default:
      throw new IllegalArgumentException("Unknown URI " + uri);
  }

  // Get the database and run the query
  SQLiteDatabase db = mOpenHelper.getReadableDatabase();
  Cursor c = qb.query(db, projection, selection, selectionArgs, null, null, sortOrder);

  // Tell the cursor what uri to watch, so it knows when its source data changes
  c.setNotificationUri(getContext().getContentResolver(), uri);
  return c;
}

@Override
public int update(Uri uri, ContentValues values, String selection,
    String[] selectionArgs) {
  // TODO Auto-generated method stub
```

```
  return 0;
}

/**
 * Add a file to a shared folder
 *
 * @param cr
 * @param file
 * @param folder
 * @return The Uri of the new shared item.
 */
public static final Uri addFileToFolder(ContentResolver cr, Uri file, Uri folder) {
  Log.i(TAG, "Adding file to folder " + file + " " + folder);
  /* Get file path */
  Cursor c = cr.query(file, null, null, null, null);
  int nameIndex = c.getColumnIndexOrThrow(OpenableColumns.DISPLAY_NAME);
  c.moveToFirst();
  String name = c.getString(nameIndex);

  /* Get folder id */
  int folderId = Integer.parseInt(folder.getPathSegments().get(1));

  ContentValues values = new ContentValues();
  values.put(Files.Columns._DATA, file.toString());
  values.put(Files.Columns.DISPLAY_NAME, name);
  values.put(Files.Columns.FOLDER_ID, folderId);
  return cr.insert(Files.CONTENT_URI, values);
}

/**
 * Deletes folders
 *
```

```
   * @param cr
   * @param folder
   * @return Whether or not the folder was deleted.
   */
  public static final boolean deleteFolder(ContentResolver cr, Uri folder) {
    int folderId = Integer.parseInt(folder.getPathSegments().get(1));
    cr.delete(Files.CONTENT_URI, Files.Columns.FOLDER_ID + "=" + folderId, null);
    cr.delete(Folders.CONTENT_URI, Folders.Columns._ID + "=" + folderId, null);
    return true;
  }

  static {
    sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    sUriMatcher.addURI(AUTHORITY, "folders", FOLDERS);
    sUriMatcher.addURI(AUTHORITY, "folders/#", FOLDER);
    sUriMatcher.addURI(AUTHORITY, "files/#", FILE);
    sUriMatcher.addURI(AUTHORITY, "files", FILES);

    sFoldersProjectionMap = new HashMap<String, String>();
    sFoldersProjectionMap.put(Folders.Columns._ID, Folders.Columns._ID);
    sFoldersProjectionMap.put(Folders.Columns.DISPLAY_NAME,
Folders.Columns.DISPLAY_NAME);
    sFoldersProjectionMap.put(Folders.Columns.PASSWORD,
Folders.Columns.PASSWORD);

    sFilesProjectionMap = new HashMap<String, String>();
    sFilesProjectionMap.put(Files.Columns._ID, Files.Columns._ID);
    sFilesProjectionMap.put(Files.Columns.FOLDER_ID, Files.Columns.FOLDER_ID);
    sFilesProjectionMap.put(Files.Columns.DISPLAY_NAME,
Files.Columns.DISPLAY_NAME);
    sFilesProjectionMap.put(Files.Columns._DATA, Files.Columns._DATA);
  }
```

```
}
```

FileShare\src\com\dimitra\dipl\fileshare\FileSharingService.java

```
// Copyright 2009 Google Inc.
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//      http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
// See the License for the specific language governing permissions and
// limitations under the License.

package com.dimitra.dipl.fileshare;

import java.io.IOException;

import com.dimitra.dipl.fileshare.IFileSharingService;
import com.dimitra.dipl.fileshare.R;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
```

```
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.IBinder;
import android.util.Log;

public class FileSharingService extends Service {

  static final String PREFS_NAME = "FileSharerServicePrefs";
  static final String PREFS_ALLOW_UPLOADS = "ALLOW_UPLOADS";
  public static String PREFS_REQUIRE_LOGIN = "REQUIRE_LOGIN";
  public static String PREFS_PASSWORD = "PASSWORD";
  public static final String PREFS_SERVICE_ON_STARTUP =
"SERVICE_ON_STARTUP";



  private static final int DEFAULT_PORT = 9999;

  private static final String TAG = "FileSharerService";

  private WebServer mWebServer;

  private Thread mWebServerThread;

  private int mPort;

  /* aidl */
  private final IFileSharingService.Stub mBinder = new IFileSharingService.Stub() {
    public int getPort() {
      return mPort;
    }
  };
```

```
 @Override
 public void onCreate() {
  super.onCreate();

  SharedPreferences settings = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
  mPort = settings.getInt("port", DEFAULT_PORT);
  try {
   mWebServer = new WebServer(this, settings,
     new CookiesDatabaseOpenHelper(this).getWritableDatabase(), mPort);
   mWebServer
   .setOnTransferStartedListener(new WebServer.TransferStartedListener() {
    public void started(Uri uri) {
     NotificationManager nm = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
     Notification notification = new Notification(R.drawable.folder,
       null, System.currentTimeMillis());
     PendingIntent pendingIntent = PendingIntent.getActivity(
       FileSharingService.this, 0, new Intent(
         FileSharingService.this, FileShare.class), 0);
     notification.setLatestEventInfo(FileSharingService.this,
       "File Share", "Transfer Started", pendingIntent);
     nm.notify(1, notification);
    }
   });
  } catch (IOException e) {
   Log.e(TAG, "Problem creating server socket " + e.toString());
  }

  mWebServerThread = new Thread() {
   @Override
   public void run() {
    mWebServer.runWebServer();
```

```
    }
  };
  mWebServerThread.start();
  Log.i(TAG, "Started webserver");
 }

 @Override
 public void onDestroy() {
        super.onDestroy();
        if (mWebServer != null) {
                mWebServerThread.interrupt();
                try {
                        mWebServerThread.join();
                } catch (InterruptedException e) {

                }
        }
 }

 @Override
 public IBinder onBind(Intent intent) {
  if (IFileSharingService.class.getName().equals(intent.getAction())) {
   return mBinder;
  }
  return null;
 }

}
```

FileShare\src\com\dimitra\dipl\fileshare\Help.java

```
package com.dimitra.dipl.fileshare;

import com.dimitra.dipl.fileshare.R;

import android.app.Activity;
import android.os.Bundle;

public class Help extends Activity {

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    setContentView(R.layout.help);
  }

}
```

FileShare\src\com\dimitra\dipl\fileshare\IFileSharingService.aidl

```
package com.dimitra.dipl.fileshare;

interface IFileSharingService {

     int getPort();

}
```

FileShare\src\com\dimitra\dipl\fileshare\SharedFileBrowser.java

```
package com.dimitra.dipl.fileshare;
```

```
import android.app.ListActivity;
import android.content.ContentUris;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.AdapterView;
import android.widget.ListAdapter;
import android.widget.SimpleCursorAdapter;

import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;

import com.dimitra.dipl.fileshare.R;

public class SharedFileBrowser extends ListActivity {

  private static final String TAG = "FileShareFilePicker";
  private static final int MENU_DELETE = 0;
  private static final int MENU_SHARE_URL = 1;

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
```

```
    int folderId =
Integer.parseInt(getIntent().getData().getPathSegments().get(1));
    String where = FileSharingProvider.Files.Columns.FOLDER_ID + "="
+ folderId;
    Cursor c = managedQuery(FileSharingProvider.Files.CONTENT_URI,
null, where, null,
            FileSharingProvider.Files.Columns.DISPLAY_NAME + " ASC");
    changeCursor(c);

    // Inform the list we provide context menus for items
    getListView().setOnCreateContextMenuListener(this);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);

    return true;
  }



  @Override
  public boolean onPrepareOptionsMenu(Menu menu) {
    super.onPrepareOptionsMenu(menu);
    final boolean haveItems = getListAdapter().getCount() > 0;

    // If there are any notes in the list (which implies that one of
    // them is selected), then we need to generate the actions that
    // can be performed on the current selection.  This will be a
```

```
combination
    // of our own specific actions along with any extensions that
can be
    // found.
    if (haveItems) {
      // This is the selected item.
      Uri uri = ContentUris.withAppendedId(getIntent().getData(),
getSelectedItemId());

      // Add delete option if we have something selected, and the
delete option
      // isn't already there.
      if (menu.findItem(MENU_DELETE) == null && getSelectedItemId()
>= 0) {
        menu.add(0, MENU_DELETE, 0, R.string.delete);
      } else {
        menu.removeItem(MENU_DELETE);
      }

      // ... is followed by whatever other actions are available...
      Intent intent = new Intent(null, uri);
      intent.addCategory(Intent.CATEGORY_ALTERNATIVE);
      menu.addIntentOptions(Menu.CATEGORY_ALTERNATIVE, 0, 0, null,
null, intent, 0,
          null);

    } else {
      menu.removeGroup(Menu.CATEGORY_ALTERNATIVE);
    }

    return true;
```

```
  }

  @Override
  public void onCreateContextMenu(ContextMenu menu, View view,
ContextMenuInfo menuInfo) {
    AdapterView.AdapterContextMenuInfo info;
    try {
      info = (AdapterView.AdapterContextMenuInfo) menuInfo;
    } catch (ClassCastException e) {
      Log.e(TAG, "bad menuInfo", e);
      return;
    }

    Cursor cursor = (Cursor)
getListAdapter().getItem(info.position);
    if (cursor == null) {
      Log.i(TAG, "Item not available");
      // For some reason the requested item isn't available, do
nothing
      return;
    }

    // Setup the menu header
    int nameIndex =
cursor.getColumnIndexOrThrow(FileSharingProvider.Files.Columns.DISPL
AY_NAME);
    menu.setHeaderTitle(cursor.getString(nameIndex));

    // Add a menu item to delete the folder
    menu.add(0, MENU_DELETE, 0, R.string.delete);
```

```
    // Add a menu item to share a link to the folder.
    menu.add(0, MENU_SHARE_URL, 0, R.string.share_url);
  }

  @Override
  public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info;
    try {
      info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
    } catch (ClassCastException e) {
      Log.e(TAG, "bad menuInfo", e);
      return false;
    }

    switch (item.getItemId()) {
      case MENU_DELETE: {
        deleteSharedFile(info.id);
        return true;
      }
      case MENU_SHARE_URL: {
        Intent shareURLIntent = new Intent();
        shareURLIntent.setAction(Intent.ACTION_SEND);
        shareURLIntent.setType("text/plain");
        shareURLIntent.putExtra(Intent.EXTRA_TEXT,
getShareURL(info.id, this));
        Intent chooserIntent = Intent.createChooser(
            shareURLIntent, getText(R.string.shareurl_title));
        startActivity(chooserIntent);
        return true;
      }
```

```
    }
    return false;
  }

  public static String getShareURL(long fileId,
      Context context) {
    String where = FileSharingProvider.Files.Columns._ID + "=" +
fileId;
    Cursor c = context.getContentResolver().query(
        FileSharingProvider.Files.CONTENT_URI,
        new String[]
{FileSharingProvider.Files.Columns.DISPLAY_NAME},
        where, null, null);
    c.moveToFirst();
    String encodedName = "";

    // This really shouldn't happen, but if it does, we can just
skip
    // the name since the web server depends on the id, not the
name.
    try {
      encodedName = URLEncoder.encode(c.getString(0), "UTF8");
    } catch (UnsupportedEncodingException e) {
      Log.e(TAG, "Problem encoding display name " + c.getString(0));
    }

    return "http://" + FileShare.getIPAddress(context) + ":9999" +
    "/file/" + fileId + "/" + encodedName;
  }

  @Override
```

```
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
      case MENU_DELETE:
        deleteSharedFile(getSelectedItemId());
        break;
    }

    return false;
  }

  private void changeCursor(Cursor cursor) {
    ListAdapter adapter = new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_1, cursor,
        new String[] {
FileSharingProvider.Files.Columns.DISPLAY_NAME },
        new int[] { android.R.id.text1 });
    setListAdapter(adapter);
  }

  private void deleteSharedFile(long fileId) {
    String where = FileSharingProvider.Files.Columns._ID + "=" +
fileId;

getContentResolver().delete(FileSharingProvider.Files.CONTENT_URI,
where, null);

getContentResolver().notifyChange(FileSharingProvider.Files.CONTENT_
URI, null);
  }

}
```

FileShare\src\com\dimitra\dipl\fileshare\SharedFolderBrowser.java

```
package com.dimitra.dipl.fileshare;

import com.dimitra.dipl.fileshare.R;

import android.app.ListActivity;
import android.content.ContentUris;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.AdapterView;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;



public class SharedFolderBrowser extends ListActivity {

  private static final String TAG = "FileShareBrowser";

  private static final int MENU_DELETE = 0;
  private static final int MENU_CREATE = 1;
  private static final int MENU_SHARE_URL = 2;
```

```
  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    /* If no data is supplied, get default from the FileProvider */
    if (getIntent().getData() == null) {
      getIntent().setData(FileSharingProvider.Folders.CONTENT_URI);
    }

    Cursor c = managedQuery(getIntent().getData(), null, null, null,
null);
    changeCursor(c);

    // Inform the list we provide context menus for items
    getListView().setOnCreateContextMenuListener(this);
  }

  @Override
  protected void onListItemClick(ListView l, View v, int position,
long id) {
    if (getIntent().getAction().equals(Intent.ACTION_PICK))  {
      Uri uri =
Uri.withAppendedPath(FileSharingProvider.Folders.CONTENT_URI, "" +
id);
      Log.d(TAG, "Uri for folder = " + uri.toString());
      setResult(RESULT_OK, new Intent().setData(uri));
      finish();
    } else {
      /* show files in the folder */
      Uri uri =
Uri.withAppendedPath(FileSharingProvider.Folders.CONTENT_URI, "" +
```

```
id);
      Intent intent = new Intent();
      intent.setData(uri);
      intent.setAction(Intent.ACTION_VIEW);
      startActivity(intent);
    }
  }

  private void changeCursor(Cursor cursor) {
    ListAdapter adapter = new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_1, cursor,
        new String[] {
FileSharingProvider.Folders.Columns.DISPLAY_NAME },
        new int[] { android.R.id.text1 });
    setListAdapter(adapter);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);

    /* Always give option to create */
    menu.add(0, MENU_CREATE, 0, getText(R.string.create));

    return true;
  }

  @Override
  public boolean onPrepareOptionsMenu(Menu menu) {
    super.onPrepareOptionsMenu(menu);
    final boolean haveItems = getListAdapter().getCount() > 0;
```

```
    // If there are any notes in the list (which implies that one of
    // them is selected), then we need to generate the actions that
    // can be performed on the current selection.  This will be a
combination
    // of our own specific actions along with any extensions that
can be
    // found.
    if (haveItems) {
      // This is the selected item.
      Uri uri = ContentUris.withAppendedId(getIntent().getData(),
getSelectedItemId());

      // Build menu...  always starts with the DELETE action...
      Intent[] specifics = new Intent[1];
      specifics[0] = new Intent(Intent.ACTION_DELETE, uri);
      MenuItem[] items = new MenuItem[1];

      // ... is followed by whatever other actions are available...
      Intent intent = new Intent(null, uri);
      intent.addCategory(Intent.CATEGORY_ALTERNATIVE);
      menu.addIntentOptions(Menu.CATEGORY_ALTERNATIVE, 0, 0, null,
specifics, intent, 0,
          items);

    } else {
      menu.removeGroup(Menu.CATEGORY_ALTERNATIVE);
    }

    return true;
  }
```

```
  @Override
  public void onCreateContextMenu(ContextMenu menu, View view,
ContextMenuInfo menuInfo) {
    AdapterView.AdapterContextMenuInfo info;
    try {
      info = (AdapterView.AdapterContextMenuInfo) menuInfo;
    } catch (ClassCastException e) {
      Log.e(TAG, "bad menuInfo", e);
      return;
    }

    Cursor cursor = (Cursor)
getListAdapter().getItem(info.position);
    if (cursor == null) {
      Log.i(TAG, "Item not available");
      // For some reason the requested item isn't available, do
nothing
      return;
    }

    // Setup the menu header
    int nameIndex = cursor.getColumnIndexOrThrow(
        FileSharingProvider.Folders.Columns.DISPLAY_NAME);
    menu.setHeaderTitle(cursor.getString(nameIndex));

    // Add a menu item to delete the folder
    menu.add(0, MENU_DELETE, 0, R.string.delete);

    // Add a menu item to share a URL to the folder.
```

```
    menu.add(0, MENU_SHARE_URL, 0, R.string.share_url);
  }

  @Override
  public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info;
    try {
      info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
    } catch (ClassCastException e) {
      Log.e(TAG, "bad menuInfo", e);
      return false;
    }

    switch (item.getItemId()) {
      case MENU_DELETE: {
        // Delete the note that the context menu is for
        Intent deleteIntent = new Intent();
        deleteIntent.setAction(Intent.ACTION_DELETE);
        deleteIntent.setData(Uri.withAppendedPath(
            FileSharingProvider.Folders.CONTENT_URI, "" + info.id));
        startActivity(deleteIntent);
        return true;
      }
      case MENU_SHARE_URL: {
        Intent shareURLIntent = new Intent();
        shareURLIntent.setAction(Intent.ACTION_SEND);
        shareURLIntent.setType("text/plain");
        shareURLIntent.putExtra(Intent.EXTRA_TEXT,
getShareURL(info.id));
        Intent chooserIntent = Intent.createChooser(
```

```
            shareURLIntent, getText(R.string.shareurl_title));
        startActivity(chooserIntent);
        return true;
      }
    }
    return false;
  }

  public String getShareURL(long folderId) {
    return "http://" + FileShare.getIPAddress(this) + ":9999" +
    "/folder/" + folderId;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
      case MENU_DELETE:
        Intent deleteIntent = new Intent();
        deleteIntent.setAction(Intent.ACTION_DELETE);
        deleteIntent.setData(Uri.withAppendedPath(
            FileSharingProvider.Folders.CONTENT_URI, "" +
getSelectedItemId()));
        startActivity(deleteIntent);
        break;
      case MENU_CREATE:
        Intent createIntent = new Intent();
        createIntent.setAction(Intent.ACTION_INSERT);

createIntent.setType(FileSharingProvider.Folders.CONTENT_TYPE);
        startActivity(createIntent);
        break;
```

```
    }

    return false;
  }


}
```

FileShare\src\com\dimitra\dipl\fileshare\SharedFolderCreator.java

```
package com.dimitra.dipl.fileshare;

import com.dimitra.dipl.fileshare.R;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.ContentValues;
import android.database.SQLException;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

/**
 * Creates Shared Folders
 */
public class SharedFolderCreator extends Activity {

  private static final int DIALOG_ERROR = 0;

  @Override
```

```
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    setContentView(R.layout.createfolder);

    /* Simple cancel button */
    Button cancelButton = (Button) findViewById(R.id.cancel_button);
    cancelButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        setResult(RESULT_CANCELED);
        finish();
      }
    });

    /* Grab name from the UI views and insert using the provder */
    Button createButton = (Button) findViewById(R.id.ok_button);
    createButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        /* Get params */
        EditText editText = (EditText)
findViewById(R.id.foldername);
        String folderName = editText.getText().toString();

        /* Do the insert */
        ContentValues values = new ContentValues();
        values.put(FileSharingProvider.Folders.Columns.DISPLAY_NAME,
folderName);
        try {
        getContentResolver().insert(
            FileSharingProvider.Folders.CONTENT_URI, values);
        } catch (SQLException e) {
```

```
            showDialog(DIALOG_ERROR);
            return;
          }
          setResult(RESULT_OK);
          finish();
        }
      });
  }

  @Override
  protected Dialog onCreateDialog(int id) {
    Dialog dialog = null;
    switch (id) {
      case DIALOG_ERROR:
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(R.string.error_creating_folder)
          .setTitle(R.string.error_title);
        dialog = builder.create();
        break;
    }
    return dialog;
  }

}
```

FileShare\src\com\dimitra\dipl\fileshare\SharedFolderDeleter.java

```
package com.dimitra.dipl.fileshare;

import com.dimitra.dipl.fileshare.R;

import android.app.Activity;
```

```
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

/**
 * Deletes Shared Folders
 */
public class SharedFolderDeleter extends Activity {

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    setContentView(R.layout.deletefolder);

    /* Populate UI elements with folder information */
    Uri folderUri = getIntent().getData();
    Cursor c = managedQuery(folderUri, null, null, null, null);
    c.moveToFirst();

    /* Get the name and the id */
    int nameIndex = c.getColumnIndexOrThrow(
        FileSharingProvider.Folders.Columns.DISPLAY_NAME);
    int folderIdIndex = c.getColumnIndexOrThrow(
        FileSharingProvider.Folders.Columns._ID);
    String name = c.getString(nameIndex);
    int folderId = c.getInt(folderIdIndex);
```

```
    /* Find out how many files we'll be removing from the folder */
    String where = FileSharingProvider.Files.Columns.FOLDER_ID + "="
+ folderId;
    c = managedQuery(
        FileSharingProvider.Files.CONTENT_URI, null, where, null,
null);
    int num_files = c.getCount();

    TextView nameText = (TextView) findViewById(R.id.foldername);
    nameText.setText(name);

    TextView sizeText = (TextView) findViewById(R.id.foldersize);
    sizeText.setText("" + num_files);

    /* Simple cancel button */
    Button cancelButton = (Button) findViewById(R.id.cancel_button);
    cancelButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        setResult(RESULT_CANCELED);
        finish();
      }
    });

    /* Delete button */
    Button deleteButton = (Button) findViewById(R.id.ok_button);
    deleteButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        FileSharingProvider.deleteFolder(
            getContentResolver(), getIntent().getData());
        setResult(RESULT_OK);
        finish();
```

```
      }
    });
  }
}
```

FileShare\src\com\dimitra\dipl\fileshare\StreamingZipEntity.java

```
package com.dimitra.dipl.fileshare;

import android.content.ContentResolver;
import android.database.Cursor;
import android.net.Uri;

import org.apache.http.entity.AbstractHttpEntity;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PipedInputStream;
import java.io.PipedOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

/**
 * HttpEntity that streams the contents of a shared folder as a ZIP
file.
 *
 * When you access the FileShare from a PC,
 * you can download the whole file as a .zip
 */
public class StreamingZipEntity extends AbstractHttpEntity {
```

```
  private ContentResolver mContentResolver;
  private String mFolderId;
  private boolean mFinished;
  private static final int BUFFER_SIZE = 1024;

  public StreamingZipEntity(ContentResolver contentResolver, String
folderId) {
    mContentResolver = contentResolver;
    mFolderId = folderId;
    mFinished = false;
  }

  public InputStream getContent() throws IOException,
IllegalStateException {
    PipedInputStream in = new PipedInputStream();
    final PipedOutputStream out = new PipedOutputStream(in);
    new Thread(
      new Runnable() {
        public void run() {
          try {
            writeTo(out);
          } catch (IOException e) {
            e.printStackTrace();
          }
        }
      }
    ).start();
    return in;
  }

  public long getContentLength() {
```

```
    return -1;
  }

  public boolean isRepeatable() {
    return false;
  }

  public boolean isStreaming() {
    return mFinished;
  }

  @Override
  public void consumeContent() {
    mFinished = true;
  }

  public void writeTo(OutputStream out) throws IOException {
    Cursor c = mContentResolver.query(
        FileSharingProvider.Files.CONTENT_URI,
        new String[] {
            FileSharingProvider.Files.Columns.DISPLAY_NAME,
            FileSharingProvider.Files.Columns._DATA
        },
        FileSharingProvider.Files.Columns.FOLDER_ID + "=?",
        new String[] {mFolderId}, null);
    ZipOutputStream zipOut = new ZipOutputStream(out);
    byte[] buf = new byte[BUFFER_SIZE];
    while (c.moveToNext()) {
      String filename = c.getString(

c.getColumnIndex(FileSharingProvider.Files.Columns.DISPLAY_NAME));
```

```
      String data = c.getString(

c.getColumnIndex(FileSharingProvider.Files.Columns._DATA));
      zipOut.putNextEntry(new ZipEntry(filename));
      InputStream input =
mContentResolver.openInputStream(Uri.parse(data));
      int len;
      while ((len = input.read(buf)) > 0) {
        zipOut.write(buf, 0, len);
      }
      zipOut.closeEntry();
      input.close();
    }
    zipOut.finish();
    mFinished = true;
  }
}
```

FileShare\src\com\dimitra\dipl\fileshare\WebServer.java

```
package com.dimitra.dipl.fileshare;

import org.apache.commons.fileupload.MultipartStream;
import org.apache.http.Header;
import org.apache.http.HttpException;
import org.apache.http.HttpRequest;
import org.apache.http.HttpResponse;
import org.apache.http.HttpVersion;
import org.apache.http.RequestLine;
import org.apache.http.entity.StringEntity;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.impl.DefaultHttpServerConnection;
```

```
import org.apache.http.message.BasicHttpEntityEnclosingRequest;
import org.apache.http.message.BasicHttpResponse;
import org.apache.http.params.BasicHttpParams;

import java.util.Random;
import java.util.StringTokenizer;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.URLDecoder;
import java.nio.channels.ClosedByInterruptException;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;

import android.content.ContentValues;
import android.content.Context;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.provider.OpenableColumns;
import android.util.Log;

public class WebServer {
```

```
  private static final String TAG = "FileSharer WebServer";

  private int mPort;

  private ServerSocketChannel mServerSocketChannel;

  private Context mContext;

  private SharedPreferences mSharedPreferences;

  private SQLiteDatabase mCookiesDatabase;

  public interface TransferStartedListener {
    public void started(Uri uri);
  }

  private TransferStartedListener mTransferStartedListener;

  /* How long we allow session cookies to last. */
  private static final int COOKIE_EXPIRY_SECONDS = 3600;

  /* Start the webserver on specified port */
  public WebServer(Context context,
      SharedPreferences sharedPreferences, SQLiteDatabase
cookiesDatabase,
      int port) throws IOException {
    mPort = port;
    mServerSocketChannel = ServerSocketChannel.open();
    mServerSocketChannel.socket().setReuseAddress(true);
    mServerSocketChannel.socket().bind(new
```

```
InetSocketAddress(mPort));
    mContext = context;
    mSharedPreferences = sharedPreferences;
    mCookiesDatabase = cookiesDatabase;
    deleteOldCookies();
  }

  /* Returns port we're using */
  public int getPort() {
    return mPort;
  }

  public void setOnTransferStartedListener(TransferStartedListener
listener) {
    mTransferStartedListener = listener;
  }

  public void runWebServer() {
    while (true) {
      Log.i(TAG, "Running main webserver thread");
      try {
     SocketChannel channel = mServerSocketChannel.accept();
        final Socket socket = channel.socket();
        Log.d(TAG, "Socket accepted");
        Thread t = new Thread() {
          @Override
          public void run() {
            handleRequest(socket);
          }
        };
        t.start();
```

```
      } catch (ClosedByInterruptException e) {
      Log.i(TAG, "Received interrupt to shutdown.");
      return;
      } catch (IOException e) {
        Log.e(TAG, "Unexpected error, shutting down. " +
e.toString());
        return;
      }
    }
  }

  /* Handles a single request. */
  public void handleRequest(Socket socket) {
    try {
      DefaultHttpServerConnection serverConnection = new
DefaultHttpServerConnection();
      serverConnection.bind(socket, new BasicHttpParams());
      HttpRequest request = serverConnection.receiveRequestHeader();
      RequestLine requestLine = request.getRequestLine();

      /* First make sure user is logged in if that is required. */
      boolean loggedIn = false;
      if
(mSharedPreferences.getBoolean(FileSharingService.PREFS_REQUIRE_LOGI
N,
          false)) {
        /* Does the user have a valid cookie? */
        Header cookiesHeader = request.getFirstHeader("Cookie");
        if (cookiesHeader != null) {
          String cookies = cookiesHeader.getValue();
          String cookie = cookies.substring(cookies.indexOf("id=")
```

```
                + "id=".length());
            loggedIn = isValidCookie(cookie);
          }
        } else {
          loggedIn = true;
        }

        if (!loggedIn) {
          /* Could be the result of the login form. */
          if (requestLine.getUri().equals("/login")) {
            handleLoginRequest(serverConnection, request,
requestLine);
          } else {
            sendLoginForm(serverConnection, requestLine);
          }
        } else if (requestLine.getUri().equals("/")) {
          Log.i(TAG, "Sending shared folder listing");
          sendSharedFolderListing(serverConnection);
        } else if (requestLine.getMethod().equals("GET")
            && requestLine.getUri().startsWith("/folder")) {
          Log.i(TAG, "Sending list of shared files");
          sendSharedFilesList(serverConnection, requestLine);
        } else if (requestLine.getUri().startsWith("/zip")) {
          Log.i(TAG, "Sending zip file.");
          sendFolderContent(serverConnection, requestLine);
        } else if (requestLine.getUri().startsWith("/file")) {
          Log.i(TAG, "Sending file content");
          sendFileContent(serverConnection, requestLine);
        } else if (requestLine.getMethod().equals("POST")) {
          Log.i(TAG, "User is uploading file");
          handleUploadRequest(serverConnection, request, requestLine);
```

```
      } else if (requestLine.getUri().startsWith("/playlist")) {
        Log.i(TAG, "User is requesting playlist");
        sendPlaylist(serverConnection, requestLine);
      } else {
        Log.i(TAG, "No action for " + requestLine.getUri());
        sendNotFound(serverConnection);
      }
      serverConnection.flush();
      serverConnection.close();
    } catch (IOException e) {
      Log.e(TAG, "Problem with socket " + e.toString());
    } catch (HttpException e) {
      Log.e(TAG, "Problemw with HTTP server " + e.toString());
    }
  }

  private void handleLoginRequest(DefaultHttpServerConnection
serverConnection,
      HttpRequest request, RequestLine requestLine) throws
HttpException,
      IOException {

    BasicHttpEntityEnclosingRequest enclosingRequest = new
BasicHttpEntityEnclosingRequest(
        request.getRequestLine());
    serverConnection.receiveRequestEntity(enclosingRequest);

    InputStream input = enclosingRequest.getEntity().getContent();
    InputStreamReader reader = new InputStreamReader(input);

    StringBuffer form = new StringBuffer();
```

```
    while (reader.ready()) {
      form.append((char) reader.read());
    }
    String password = form.substring(form.indexOf("=") + 1);
    if (password.equals(mSharedPreferences.getString(
        FileSharingService.PREFS_PASSWORD, ""))) {
      HttpResponse response = new BasicHttpResponse(new
HttpVersion(1, 1), 302,
          "Found");
      response.addHeader("Location", "/");
      response.addHeader("Set-Cookie", "id=" + createCookie());
      response.setEntity(new StringEntity(getHTMLHeader() +
"Success!"
          + getHTMLFooter()));
      serverConnection.sendResponseHeader(response);
      serverConnection.sendResponseEntity(response);
    } else {
      HttpResponse response = new BasicHttpResponse(new
HttpVersion(1, 1), 401,
          "Unauthorized");
      response.setEntity(new StringEntity(getHTMLHeader()
          + "<p>Login failed.</p>" + getLoginForm() +
getHTMLFooter()));
      serverConnection.sendResponseHeader(response);
      serverConnection.sendResponseEntity(response);
    }
  }

  private String createCookie() {
    Random r = new Random();
    String value = Long.toString(Math.abs(r.nextLong()), 36);
```

```
    ContentValues values = new ContentValues();
    values.put("name", "id");
    values.put("value", value);
    values.put("expiry", (int) System.currentTimeMillis() / 1000
        + COOKIE_EXPIRY_SECONDS);
    mCookiesDatabase.insert("cookies", "name", values);
    return value;
  }

  private boolean isValidCookie(String cookie) {
    Cursor cursor = mCookiesDatabase.query("cookies", new String[] {
"value" },
        "name = ? and value = ? and expiry > ?", new String[] {
"id", cookie,
            "" + (int) System.currentTimeMillis() / 1000 }, null,
null, null);
    boolean isValid = cursor.getCount() > 0;
    cursor.close();
    return isValid;
  }

  private void deleteOldCookies() {
    mCookiesDatabase.delete("cookies", "expiry < ?", new String[] {
""
        + (int) System.currentTimeMillis() / 1000 });
  }

  private void sendNotFound(DefaultHttpServerConnection
serverConnection)
      throws UnsupportedEncodingException, HttpException,
IOException {
```

```
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 404,
        "NOT FOUND");
    response.setEntity(new StringEntity("NOT FOUND"));
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  private void handleUploadRequest(
      DefaultHttpServerConnection serverConnection, HttpRequest
request,
      RequestLine requestLine) throws IOException, HttpException,
      UnsupportedEncodingException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
        "OK");
    String folderId = getFolderId(requestLine.getUri());
    processUpload(folderId, request, serverConnection);
    String header = getHTMLHeader();
    String form = getUploadForm(folderId);
    String footer = getHTMLFooter();
    String listing = getFileListing(Uri.withAppendedPath(
        FileSharingProvider.Folders.CONTENT_URI, folderId));
    response.setEntity(new StringEntity(header + listing + form +
footer));
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  private void sendFileContent(DefaultHttpServerConnection
serverConnection,
```

```
      RequestLine requestLine) throws IOException, HttpException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
        "OK");
    String fileId = getFileId(requestLine.getUri());

addFileEntity(Uri.withAppendedPath(FileSharingProvider.Files.CONTENT
_URI,
        fileId), response);
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  /**
   * Sends a ZIP file containing all files from the shared folder.
   *
   * @param serverConnection
   * @param requestLine
   * @throws IOException
   * @throws HttpException
   */
  private void sendFolderContent(DefaultHttpServerConnection
serverConnection,
      RequestLine requestLine) throws IOException, HttpException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
        "OK");
    String folderId = getFolderId(requestLine.getUri());
    addFolderZipEntity(folderId, response);
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
```

```
  }

  private void addFolderZipEntity(String folderId, HttpResponse
response) {
    response.addHeader("Content-Type", "application/zip");
    response.setEntity(new StreamingZipEntity(
        mContext.getContentResolver(), folderId));
  }

  private void sendPlaylist(DefaultHttpServerConnection
serverConnection,
      RequestLine requestLine) throws IOException, HttpException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
    "OK");

    Cursor c = mContext.getContentResolver().query(
        FileSharingProvider.Files.CONTENT_URI,
        new String[] {FileSharingProvider.Files.Columns._ID,
            FileSharingProvider.Files.Columns.DISPLAY_NAME},
            null, null, null);
    String playlist = "";
    while (c.moveToNext()) {
      long id = c.getLong(c.getColumnIndex(
          FileSharingProvider.Files.Columns._ID));
      String name = c.getString(c.getColumnIndex(
          FileSharingProvider.Files.Columns.DISPLAY_NAME));
      if (name.endsWith(".mp3")) {
        playlist += SharedFileBrowser.getShareURL(id, mContext) +
"\n";
      }
```

```
    }
    c.close();
    response.addHeader("Content-Type", "audio/x-mpegurl");
    response.addHeader("Content-Length", "" + playlist.length());
    response.setEntity(new StringEntity(playlist));
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  private void sendSharedFilesList(
      DefaultHttpServerConnection serverConnection, RequestLine
requestLine)
      throws UnsupportedEncodingException, HttpException,
IOException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
        "OK");
    String folderId = getFolderId(requestLine.getUri());
    String header = getHTMLHeader();
    String form = getUploadForm(folderId);
    String footer = getHTMLFooter();
    String listing = getFileListing(Uri.withAppendedPath(
        FileSharingProvider.Folders.CONTENT_URI, folderId));
    response.setEntity(new StringEntity(header + listing + form +
footer));
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  private void sendLoginForm(DefaultHttpServerConnection
serverConnection,
```

```
      RequestLine requestLine) throws UnsupportedEncodingException,
      HttpException, IOException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
        "OK");
    response.setEntity(new StringEntity(getHTMLHeader()
        + "<p>Password Required</p>" + getLoginForm() +
getHTMLFooter()));
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  private void sendSharedFolderListing(
      DefaultHttpServerConnection serverConnection)
      throws UnsupportedEncodingException, HttpException,
IOException {
    HttpResponse response = new BasicHttpResponse(new HttpVersion(1,
1), 200,
        "OK");
    response.setEntity(new StringEntity(getHTMLHeader() +
getFolderListing()
        + getHTMLFooter()));
    serverConnection.sendResponseHeader(response);
    serverConnection.sendResponseEntity(response);
  }

  @SuppressWarnings("deprecation")
  public void processUpload(String folderId, HttpRequest request,
      DefaultHttpServerConnection serverConnection) throws
IOException,
      HttpException {
```

```
    /* Find the boundary and the content length. */
    String contentType = request.getFirstHeader("Content-
Type").getValue();
    String boundary =
contentType.substring(contentType.indexOf("boundary=")
        + "boundary=".length());
    BasicHttpEntityEnclosingRequest enclosingRequest = new
BasicHttpEntityEnclosingRequest(
        request.getRequestLine());
    serverConnection.receiveRequestEntity(enclosingRequest);

    InputStream input = enclosingRequest.getEntity().getContent();
    MultipartStream multipartStream = new MultipartStream(input,
boundary
        .getBytes());
    String headers = multipartStream.readHeaders();

    /* Get the filename. */
    StringTokenizer tokens = new StringTokenizer(headers, ";",
false);
    String filename = null;
    while (tokens.hasMoreTokens() && filename == null) {
      String token = tokens.nextToken().trim();
      if (token.startsWith("filename=")) {
        filename =
URLDecoder.decode(token.substring("filename=\"".length(),
            token.lastIndexOf("\"")), "utf8");
      }
    }
```

```
    File uploadDirectory = new File("/sdcard/fileshare/uploads");
    if (!uploadDirectory.exists()) {
      uploadDirectory.mkdirs();
    }

    /* Write the file and add it to the shared folder. */
    File uploadFile = new File(uploadDirectory, filename);
    FileOutputStream output = new FileOutputStream(uploadFile);
    multipartStream.readBodyData(output);
    output.close();

    Uri fileUri = Uri.withAppendedPath(FileProvider.CONTENT_URI,
uploadFile
        .getAbsolutePath());
    Uri folderUri = Uri.withAppendedPath(
        FileSharingProvider.Folders.CONTENT_URI, folderId);

FileSharingProvider.addFileToFolder(mContext.getContentResolver(),
        fileUri, folderUri);
  }

  public String getHTMLHeader() {
    return "<html><head><title>File Share</title></head><body>";
  }

  public String getHTMLFooter() {
    return "</body></html>";
  }

  private String getFolderListing() {
    /* Get list of folders */
```

```
    Cursor c = mContext.getContentResolver().query(
        FileSharingProvider.Folders.CONTENT_URI,
        null, null, null, null);
    int nameIndex = c

.getColumnIndexOrThrow(FileSharingProvider.Folders.Columns.DISPLAY_N
AME);
    int idIndex = c

.getColumnIndexOrThrow(FileSharingProvider.Folders.Columns._ID);
    String s = "";
    while (c.moveToNext()) {
      String name = c.getString(nameIndex);
      int id = c.getInt(idIndex);
      s += folderToLink(name, id) + "<br/>";
    }
    c.close();
    return s;
  }

  /**
   * Returns a form that allows users to upload files.
   */
  private String getUploadForm(String folderId) {
    if
(mSharedPreferences.getBoolean(FileSharingService.PREFS_ALLOW_UPLOAD
S,
        false)) {
      return "<form method=\"POST\" action=\"/folder/" + folderId +
"\" "
          + "enctype=\"multipart/form-data\"> "
```

```
          + "<input type=\"file\" name=\"file\" size=\"40\"/> "
          + "<input type=\"submit\" value=\"Upload\"/>";
    }
    return "";
  }

  private String getFolderId(String firstline) {
    Pattern p = Pattern.compile("/(?:folder|playlist|zip)/(\\d+)");
    Matcher m = p.matcher(firstline);
    boolean b = m.find(0);
    if (b) {
      return m.group(1);
    }
    return null;
  }

  private String getLoginForm() {
    return "<form method=\"POST\" action=\"/login\""
        + " enctype=\"application/x-www-form-urlencoded\""
        + "/><input type=\"password\" name=\"password\"/>"
        + "<input type=\"submit\" value=\"Login\"/></form>";
  }

  private String getFileId(String firstline) {
    Pattern p = Pattern.compile("/file/(\\d+)");
    Matcher m = p.matcher(firstline);
    boolean b = m.find(0);
    if (b) {
      return m.group(1);
    }
    return null;
```

```
  }

  private String getFileListing(Uri uri) {
    int folderId = Integer.parseInt(uri.getPathSegments().get(1));
    Uri fileUri = FileSharingProvider.Files.CONTENT_URI;
    String where = FileSharingProvider.Files.Columns.FOLDER_ID + "="
+ folderId;
    Cursor c = mContext.getContentResolver().query(
        fileUri, null, where, null, null);
    int nameIndex = c

.getColumnIndexOrThrow(FileSharingProvider.Files.Columns.DISPLAY_NAM
E);
    int idIndex = c

.getColumnIndexOrThrow(FileSharingProvider.Files.Columns._ID);
    String s = "";
    boolean hasMusic = false;
    while (c.moveToNext()) {
      String name = c.getString(nameIndex);
      int id = c.getInt(idIndex);
      s += fileToLink(name, id) + "<br/>";
      if (name.endsWith(".mp3")) {
        hasMusic = true;
      }
    }
    c.close();
    if (hasMusic) {
      s += getPlaylistLink(folderId) + "<br/>";
    }
    s += getZipLink(folderId) + "<br/>";
```

```
    return s;
  }

  private String getPlaylistLink(long folderId) {
    return "<a href=\"/playlist/" + folderId + "/playlist.m3u\">" +
           "MP3 Playlist</a>";
  }

  private String getZipLink(long folderId) {
    return "<a href=\"/zip/" + folderId + "/folder.zip\">" +
           "Zip of Entire Folder</a>";
  }

  private void addFileEntity(final Uri uri, HttpResponse response)
      throws IOException {
    if (mTransferStartedListener != null) {
      mTransferStartedListener.started(uri);
    }

    Cursor c = mContext.getContentResolver().query(uri, null, null,
null, null);
    c.moveToFirst();
    int nameIndex = c

.getColumnIndexOrThrow(FileSharingProvider.Files.Columns.DISPLAY_NAM
E);
    String name = c.getString(nameIndex);
    int dataIndex = c

.getColumnIndexOrThrow(FileSharingProvider.Files.Columns._DATA);
    Uri data = Uri.parse(c.getString(dataIndex));
```

```
    c = mContext.getContentResolver().query(data, null, null, null,
null);
    c.moveToFirst();
    int sizeIndex = c.getColumnIndexOrThrow(OpenableColumns.SIZE);
    int sizeBytes = c.getInt(sizeIndex);
    c.close();

    InputStream input =
mContext.getContentResolver().openInputStream(data);

    String contentType = "application/octet-stream";
    if (name.endsWith(".jpg")) {
      contentType = "image/jpg";
    }

    response.addHeader("Content-Type", contentType);
    response.addHeader("Content-Length", "" + sizeBytes);
    response.setEntity(new InputStreamEntity(input, sizeBytes));
  }

  private String folderToLink(String folderName, int folderId) {
    return "<a href=\"/folder/" + folderId + "\">" + folderName +
"</a>";
  }

  private String fileToLink(String fileName, int fileId) {
    return "<a href=\"/file/" + fileId + "/" + fileName + "\">" +
fileName
        + "</a>";
  }
```

```
}
```

FileShare\res\layout\action_progressbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ProgressBar
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/progressBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
</ProgressBar>
```

FileShare\res\layout\createfolder.xml

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >

<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/foldername_prefix"/>

<EditText android:id="@+id/foldername"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="@android:drawable/editbox_background" />

<LinearLayout
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">

<Button
android:id="@+id/ok_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/create_ok_button"/>

<Button android:id="@+id/cancel_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="10px"
android:text="@string/create_cancel_button"/>
/>

</LinearLayout>
</LinearLayout>

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

FileShare\res\layout\deletefolder.xml

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

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >

<LinearLayout android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="wrap_content">

<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/foldername_prefix"/>

<TextView android:id="@+id/foldername"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="10px"
/>

</LinearLayout>

<LinearLayout android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="wrap_content">

<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/foldersize_prefix"/>
```

```
<TextView android:id="@+id/foldersize"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="10px"
/>

</LinearLayout>

<LinearLayout android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="wrap_content">

<Button
android:id="@+id/ok_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/delete_ok_button"/>

<Button android:id="@+id/cancel_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="10px"
android:text="@string/delete_cancel_button"/>
/>

</LinearLayout>


</LinearLayout>
```

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



</LinearLayout>
 -->
```

FileShare\res\layout\help.xml

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

<TextView android:id="@+id/help"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/help_content"
/>

</LinearLayout>
```

FileShare\res\layout\main.xml

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_marginTop="20px"
    >



<Button
android:id="@+id/addfile"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/addfile_button"
android:layout_marginTop="20px"
>
</Button>

<Button
android:id="@+id/manage"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/manage_button"
>
</Button>

<Button
android:id="@+id/preferences"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/preferences_button"
>
</Button>

<Button
android:id="@+id/help"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/help_button"
>
</Button>

<ToggleButton
android:id="@+id/service"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textOn="@string/service_button_on"
android:textOff="@string/service_button_off"
>
</ToggleButton>

<LinearLayout android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20px"
android:layout_marginLeft="10px">

<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:text="@string/url_prefix"
/>

<TextView android:id="@+id/url"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="10px"
/>

</LinearLayout>

</LinearLayout>
```

FileShare\res\layout\password_dialog.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/layout_root"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:padding="10dp"
              >
    <EditText android:id="@+id/password"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              /><Button android:text="OK"
android:id="@+id/ok_button" android:layout_width="fill_parent"
android:layout_height="wrap_content"></Button>

</LinearLayout>
```

FileShare\res\layout\progress.xml

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >

<TextView
  android:id="@+id/sharefile_port"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
>
</TextView>

<ProgressDialog
android:id="@+id/sharefile_progress"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:title="Sharing a File"
>
</ProgressDialog>

</LinearLayout>
```

FileShare\res\menu\activity_main_actions.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <!-- Search / will display always -->
```

```
    <item android:id="@+id/action_search"
          android:icon="@drawable/ic_action_search"
          android:title="@string/action_search"
          android:showAsAction="always"
          android:actionViewClass="android.widget.SearchView"/>

    <!-- Refresh -->
    <item android:id="@+id/action_refresh"
          android:icon="@drawable/ic_refresh"
          android:title="@string/action_refresh"
          android:showAsAction="always" />

    <!-- Help -->
    <item android:id="@+id/action_help"
          android:icon="@drawable/ic_help"
          android:title="@string/action_help"
          android:showAsAction="ifRoom"/>

</menu>
```

FileShare\res\menu\file_share.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:showAsAction="never" />
        <!--  android:title="@string/action_settings" -->

</menu>
```

FileShare\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">File Share</string>
    <string name="manage_button">Manage Shared Content</string>
    <string name="addfile_button">Add File to Shared Folder</string>
    <string name="choosefile_title">Find File to Share"</string>
    <string name="empty_directory">Empty</string>
    <string name="port">Port:9999</string>
    <string name="delete">Delete</string>
    <string name="foldername_prefix">Folder:</string>
    <string name="foldersize_prefix">Number of Files:</string>
    <string name="delete_ok_button">Delete</string>
    <string name="delete_cancel_button">Cancel</string>
    <string name="create">New</string>
    <string name="create_ok_button">Create</string>
    <string name="create_cancel_button">Cancel</string>
    <string name="service_button_on">Web Service On</string>
    <string name="service_button_off">Web Service Off</string>
    <string name="edit">Edit</string>
    <string name="ipaddress_prefix">IP Address:</string>
    <string name="port_prefix">Port:</string>
    <string name="url_prefix">URL:</string>
    <string name="unknown_ipaddress">Unknown</string>
    <string name="help_button">Help</string>
    <string name="share_all">Share All</string>
    <string name="error_title">Error</string>
    <string name="error_creating_folder">Problem Creating Shared
Folder</string>
    <string name="help_content"><b>How do I share files?</b>\nSelect
"Share a File" from the menu.
```

```
    Select a file to share, then select the shared folder.\n\n<b>How
do my friends access my shared files?</b>\nTell them to go to
http://youripaddress:9999
    \n\n<b>What is my IP Address?</b>\nYou\'ll have to find that out
by investigating your wireless settings.\n\n
    <b>How do I turn off/on the service?</b>\nThe sharing service is
automatically started when the File Share application is
    first started. The ability to turn on/off the service will be
available in a future release.</string>

<string name="share_url">Share URL</string>
<string name="shareurl_title">Share</string>
<string name="preferences_button">Preferences</string>

<string name="whats_new">* Toggle Web Service\n* Order files
alphabetically\n</string>
<string name="whats_new_title">What\'s New</string>
<string name="set_password_title">Set Password</string>

<string name="action_search">Search</string>
<string name="action_refresh">Refresh</string>
<string name="action_help">Help</string>

<!-- searchable.xml's strings -->
<string name="search_hint">Start searching!</string>
</resources>
```

FileShare\res\values-el\strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">File Share</string>
```

```
    <string name="manage_button">Διαχείριση κοινόχρηστου
περιεχομένου</string>
    <string name="addfile_button">Προσθήκη αρχείου στον κοινόχρηστο
φάκελο</string>
    <string name="choosefile_title">Εύρεση αρχείου προς
διαμοίραση</string>
    <string name="empty_directory">Κενό</string>
    <string name="port">Θύρα:9999</string>
    <string name="delete">Διαγραφή</string>
    <string name="foldername_prefix">Φάκελος:</string>
    <string name="foldersize_prefix">Αριθμός φακέλων:</string>
    <string name="delete_ok_button">Διαγραφή</string>
    <string name="delete_cancel_button">Ακύρωση</string>
    <string name="create">Νέο</string>
    <string name="create_ok_button">Δημιουργία</string>
    <string name="create_cancel_button">Ακύρωση</string>
    <string name="service_button_on">Υπηρεσία Web ενεργή</string>
    <string name="service_button_off">Υπηρεσία Web ανενεργή</string>
    <string name="edit">Επεξεργασία</string>
    <string name="ipaddress_prefix">IP Διεύθυνση:</string>
    <string name="port_prefix">Θύρα:</string>
    <string name="url_prefix">URL:</string>
    <string name="unknown_ipaddress">Άγνωστη διεύθυνση</string>
    <string name="help_button">Βοήθεια</string>
    <string name="share_all">Διαμοίραση όλων</string>
    <string name="error_title">Σφάλμα</string>
    <string name="error_creating_folder">Πρόβλημα κατά τη δημιουργία
κοινόχρηστου φακέλου</string>
    <string name="help_content"><b>Πώς μοιράζομαι
αρχεία;</b>\nΕπέλεξε "Διαμοίραση φακέλου" από το μενού.
    Επέλεξε ένα φάκελο προς διαμοίραση και επέλεξε το κοινόχρηστο
```

```
φάκελο.\n\n<b>Πώς μπορούν οι φίλοι μου να έχουν πρόσβαση στα κοινόχρηστα αρχεία μου;</b>\nΠες τους να εισάγουν τη διεύθυνση http://youripaddress:9999
    \n\n<b>Ποια είναι η IP διεύθυνσή μου;</b>\nΘα πρέπει να το βρεις από τις ασύρματες ρυθμίσεις σου.\n\n
    <b>Πώς ενεργοποιώ/απενεργοποιώ την υπηρεσία;</b>\nΗ υπηρεσία διαμοίρασης ενεργοποιείται αυτόματα μόλις ξεκινήσεις την εφαρμογή FileShare.
     </string>

<string name="share_url">URL για διαμοίραση</string>
<string name="shareurl_title">Μοιράσου</string>
<string name="preferences_button">Προτιμήσεις</string>

<string name="whats_new">* Ενεργοποίση / Απενεργοποίηση Υπηρεσίας Web\n* Οι φάκελοι εμφανίζονται αλφαβητικά\n</string>
<string name="whats_new_title">Τι νέο υπάρχει</string>
<string name="set_password_title">Ορισμός κωδικού πρόσβασης</string>

<string name="action_search">Εύρεση</string>
<string name="action_refresh">Ανανέωση</string>
<string name="action_help">Βοήθεια</string>
```

FileShare\res\xml\searchable.xml

```
<?xml version="1.0" encoding="utf-8"?>
<searchable
xmlns:android="http://schemas.android.com/apk/res/android"
    android:hint="@string/search_hint"
    android:label="@string/app_name" />
```