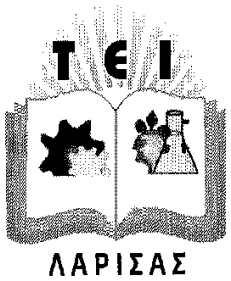


ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΛΑΡΙΣΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

" Ανάπτυξη mobile εφαρμογής διεπαφής χρήστη, για τις υπηρεσίες του Τμήματος Πληροφορικής & Τηλεπικοινωνιών, χρησιμοποιώντας την πλατφόρμα ανοιχτού κώδικα Android SDK. "

Σπουδαστές

Νίκος Παρδάλης Τ-2179

Θωμάς Καραδήμος Τ-2044

Επιβλέπων Καθηγητής

Χρήστος Σωμαράς

ΛΑΡΙΣΑ 2012

«Δηλώνουμε υπεύθυνα ότι το παρόν κείμενο αποτελεί προϊόν προσωπικής μελέτης και εργασίας και πως όλες οι πηγές που χρησιμοποιήθηκαν για τη συγγραφή της δηλώνονται σαφώς είτε στις παραπομπές είτε στη βιβλιογραφία. Γνωρίζουμε πως η λογοκλοπή αποτελεί σοβαρότατο παράπτωμα και είμαστε ενήμεροι για* την *επέλευση των νομίμων συνεπειών»

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Ονοματεπώνυμο, Υπογραφή

2. Ονοματεπώνυμο, Υπογραφή

3. Ονοματεπώνυμο, Υπογραφή

Περίληψη

Στην παρούσα πτυχιακή εργασία αναλύθηκαν και περιγράφονται τεχνικές ανάπτυξης εφαρμογών για smartphones και αναπτύχθηκε εφαρμογή με χρήση της πλατφόρμας Android SDK.

Στην εισαγωγή αναλύεται και παρουσιάζεται η ραγδαία ανάπτυξη της αγοράς των εφαρμογών για mobile πλατφόρμες και τις προοπτικές που ανοίγονται για τους νέους προγραμματιστές την σύγχρονη εποχή. Γίνεται αναφορά πάνω στις πλέον αναπτυσσόμενες πλατφόρμες και σύγκριση αυτών καθώς και των πλεονεκτημάτων και μειονεκτημάτων που παρουσιάζουν, δίνοντας έμφαση στην πλατφόρμα ανοιχτού κώδικα Android SDK.

Στην συνέχεια περιγράφεται η χρήση και ενσωμάτωση των εργαλείων στο σύστημα ανάπτυξης εφαρμογών του Android, και εμβαθύνουμε πάνω στις πιο χρήσιμες παραμέτρους τους χρησιμοποιώντας την αντικειμενοστραφής γλώσσα προγραμματισμού JAVA. Τέλος, χρησιμοποιώντας τα παραπάνω εργαλεία και τις εμπειρίες που αποκτήθηκαν από την μελέτη αυτών, αναπτύχθηκε μια εφαρμογή διεπαφής χρήστη πάνω στην πλατφόρμα της Google, Android SDK, και περιγράφονται οι δυσκολίες που συναντήθηκαν κατά την ανάπτυξη αυτής.

Στόχος της εφαρμογής είναι να ενοποιεί και να χρησιμοποιεί όσο το δυνατόν περισσότερες από τις υπηρεσίες/ιστότοπους του Τμήματος Πληροφορικής & Τηλεπικοινωνιών και να δίνει όλες τις διαθέσιμες πληροφορίες για αυτές στον χρήστη, όπως επίσης και να παρέχει γενικότερες πληροφορίες για το τμήμα (e-mails καθηγητών, χάρτης εργαστηρίων, κλπ).

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στο Λειτουργικό Σύστημα Android

1.1 Τι είναι το Android;

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα, βασισμένο στο Linux, για φορητές συσκευές όπως smartphones και tablets. Αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance η οποία είναι μια κοινοπραξία εταιριών λογισμικού, κατασκευής hardware και τηλεπικοινωνιών, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις φορητές συσκευές. Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

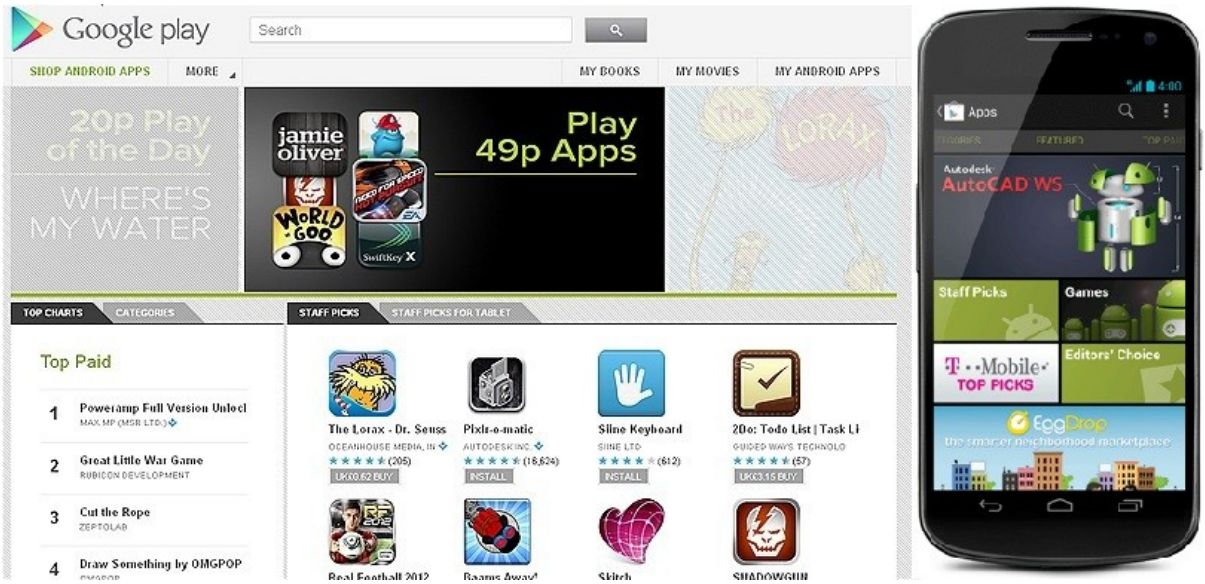
Τον Ιούλιο του 2005, η Google εξαγόρασε την Android Inc, μια μικρή εταιρεία με έδρα το Palo Alto στην California των ΗΠΑ. Εκείνη την εποχή ελάχιστα ήταν γνωστά για τις λειτουργίες της Android Inc, εκτός του ότι ανέπτυσσαν λογισμικό για κινητά τηλέφωνα. Αυτή ήταν η αρχή της φημολογίας περί σχεδίων της Google για να διεισδύσει στην αγορά κινητής τηλεφωνίας.

Στην Google, η ομάδα με επικεφαλής τον Andy Rubin ανέπτυξε μια κινητή πλατφόρμα που στηρίζεται στον πυρήνα του Linux, την οποία προώθησαν με την παροχή ενός ευέλικτου, αναβαθμίσιμου συστήματος. Έχει αναφερθεί ότι η Google είχε ήδη συγκεντρώσει μια σειρά από εταίρους hardware και software και επισήμανε στους παρόχους ότι ήταν ανοικτή σε διάφορους βαθμούς συνεργασίας εκ μέρους της. Έντυπα και ηλεκτρονικά μέσα ενημέρωσης σύντομα ανέφεραν φήμες ότι η Google ανέπτυσσε μια Google-branded συσκευή. Περισσότερες φήμες ακολούθησαν, αναφέροντας ότι η Google καθόριζε τις τεχνικές προδιαγραφές και έδειχνε πρωτότυπα στους κατασκευαστές κινητών τηλεφώνων και τους φορείς δικτύων. Τελικά η Google παρουσίασε το smartphone της Nexus One που χρησιμοποιεί το open source λειτουργικό σύστημα Android. Η συσκευή κατασκευάστηκε από την HTC , και έγινε διαθέσιμη στις 5 Ιανουαρίου 2010.

Εικόνα 1.1: Το Google Nexus One

1.2 Εφαρμογές Android

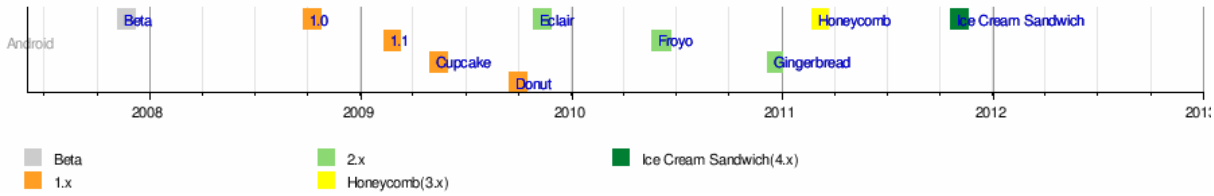
Το Android έχει μια μεγάλη κοινότητα προγραμματιστών που γράφουν εφαρμογές, οι οποίες επεκτείνουν τη λειτουργικότητα των συσκευών. Οι εφαρμογές γράφονται σε μια προσαρμοσμένη έκδοση της JAVA και μπορεί κάνεις να κατεβάσει από το online κατάστημα Google Play (πρώην Android Market) της Google όπως και από άλλα sites. Μέχρι τον Φεβρουάριο του 2012 περισσότερες από 450000 εφαρμογές ήταν διαθέσιμες για Android ενώ εκτιμάτε ότι ο αριθμός των downloads από το Android Market μέχρι το Δεκέμβριο του 2011 είχε υπερβεί τα 10 δισεκατομμύρια. Το Android είναι η πρώτη σε πωλήσεις παγκοσμίως πλατφόρμα για smartphones καθώς μέχρι το Φεβρουάριο του 2012 μετρούσε περισσότερες από 300 εκατομμύρια συσκευές σε χρήση.



Εικόνα 1.2: Το Google Play σε κανονική και mobile έκδοση.

1.3 Εξέλιξη του Android

Όπως αναφέραμε παραπάνω, το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα. Η εξέλιξη του λόγω της open source φύσης του είναι ραγδαία και αυτό αντικατοπτρίζεται στο γεγονός ότι οι 7 κύριες εκδόσεις του έχουν κυκλοφορήσει σε διάστημα 2.5 ετών, από τον Απρίλη του 2009 μέχρι τον Νοέμβριο του 2011 (Εικόνα 1.3).



Εικόνα 1.3: Χρονοδιάγραμμα Εκδόσεων του Android OS

Στην πληροφορική συνηθίζεται τα προϊόντα hardware και software να κυκλοφορούν εκτός από τον αριθμό έκδοσης τους, και με μία κωδική ονομασία. Η κωδική ονομασία μπορεί να είναι πχ ονόματα πόλεων (Windows Viena, Chicago), ονόματα ζώων (OSX Leopard, Lion), στην περίπτωση όμως του Android τα κώδικα ονόματα έρχονται στη μορφή επιδόρπιου!

1.3.1 Android 1.5 Cupcake

Η έκδοση “Cupcake”, βασισμένη στο Linux Kernel 2.6.27, παρουσιάστηκε στις 30 Απριλίου του 2009. Υποστηρίζει νέες λειτουργίες για την κάμερα τις συσκευής, όπως η καταγραφή και παρακολούθηση βίντεο από την λειτουργία της κάμερας και η άμεση μεταφόρτωση του βίντεο αλλά και των φωτογραφιών στο Youtube και το Picasa αντίστοιχα απευθείας από το τηλέφωνο. Έχει νέο έξυπνο πληκτρολόγιο με πρόβλεψη κειμένου. Υποστηρίζει πρότυπο Bluetooth A2DP και AVRCP ενώ έχει και την ικανότητα να συνδέεται αυτόματα σε μικροσυσκευές Bluetooth από μια συγκεκριμένη απόσταση. Ακόμα στην έκδοση αυτή έχει νέο γραφικό περιβάλλον με κινούμενες μεταβάσεις οθόνης.

Εικόνα 1.4: Το λογότυπο του Android 1.5 "Cupcake”

1.3.2 Android 1.6 Donut

Η έκδοση “Donut”, βασισμένη στο Linux Kernel 2.6.29, παρουσιάστηκε στις 15 Σεπτεμβρίου του 2009.Έχει ταχύτερη απόκριση σε σχέση με την προηγούμενη έκδοση. Υποστηρίζεται πλέον η επιλογή πολλαπλών αρχείων ταυτόχρονα, έχει ανανεωμένο γκάλερι και φωτογραφική μηχανή, καθώς και βελτιωμένο Android Market. Έχει ανανεωμένη φωνητική αναζήτηση, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενείς (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας κλήσης επαφών. Δυνατότητα αναζήτησης σελιδοδεικτών, ιστορικού, επαφών αλλά και στο διαδίκτυο από την αρχική οθόνη. Υποστήριξη για ανάλυση οθονών WVGA. Ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και με μηχανή μετατροπής κειμένου σε ομιλία (text-to-speech).

Εικόνα 1.5: Το λογότυπο του Android 1.6 "Donut"

1.3.3 Android 2.0/2.1 Eclair

Η έκδοση “Eclair”, βασισμένη και αυτή στον Linux Kernel 2.6.29, παρουσιάστηκε στις 26 Οκτωβρίου του 2009, ενώ τον Ιανουάριο του 2010 επανεκδόθηκε σε Android 2.1 Eclair (MR1).Σε αυτή την έκδοση υπάρχει ακόμα ταχύτερη απόκριση του υλικού σε σχέση με τις δυο προηγούμενες και πλέον υποστηρίζονται περισσότερες οθόνες και αναλύσεις. Υπάρχει νέος browser ο οποίος υποστηρίζει το πρότυπο HTML5, νέο User Interface, και βελτιωμένοι χάρτες Google (Google Maps 3.1.2). Έχει ενσωματωθεί η υποστήριξη φλας για την κάμερα η οποία έχει πλέον και ψηφιακό zoom. Επίσης έχει βελτιωθεί η κλάση MotionEvent ώστε να υπάρχει η δυνατότητα για γεγονότα πολλαπλής αφής (multitouch events). Υποστηρίζεται Bluetooth 2.1 και έχει βελτιωθεί και το πληκτρολόγιο.

Εικόνα 1.6: Το λογότυπο του Android 2.1 "Eclair"

1.3.4 Android 2.2 Froyo

Η έκδοση “Froyo”, βασισμένη στο Linux Kernel 2.6.32, παρουσιάστηκε στις 20 Μαΐου του 2010. Υπάρχουν βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση. Έχει ενσωματωθεί ο μηχανισμός JavaScript του Chrome V8 στον browser, υπάρχει πλέον Adobe Flash 10.1, ενώ υποστηρίζεται καλύτερα πλέον το Microsoft Exchange. Έχει γίνει ανανέωση του Android Market. Ο χρήστης μπορεί πλέον να ελέγχει αν θα γίνεται ή όχι κίνηση πακέτων δεδομένων από το δίκτυο κινητής τηλεφωνίας. Υπάρχει η δυνατότητα εγκατάστασης εφαρμογών στην κάρτα μνήμης και η μεταφορά τους εκεί από τη μνήμη του τηλεφώνου. Επίσης το τηλέφωνο πλέον μπορεί να μετατραπεί σε WiFi hotspot.

Εικόνα 1.7: Το λογότυπο του Android 2.2 "Froyo"

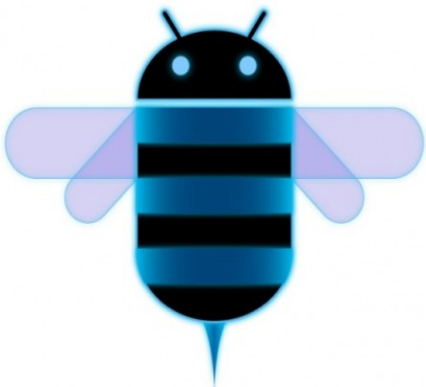
1.3.5 Android 2.3 Gingerbread

Η έκδοση “Gingerbread”, βασισμένη στο Linux Kernel 2.6.35.7, παρουσιάστηκε στις 6 Δεκεμβρίου του 2010, ενώ τον Φεβρουάριο του 2011 επανεκδόθηκε σε Android 2.3.3. Στην έκδοση αυτή υπάρχουν αλλαγές στο User Interface το οποίο έχει γίνει πιο απλό και ταχύ, ενώ υποστηρίζονται πλέον οθόνες μεγάλων μεγεθών και αναλύσεων. Υπάρχει πλέον το πρωτόκολλο SIP για κλήσεις μέσω VoIP, υποστηρίζεται ο τύπος βίντεο WebM/VP8 και ο κωδικοποιητής AAC, έχει βελτιωθεί ο ήχος καθώς και οι λειτουργίες απεικόνισης για την ανάπτυξη παιχνιδιών. Υπάρχει η δυνατότητα για Copy-Paste σε όλο το σύστημα και όχι μόνο στην ίδια εφαρμογή. Υποστηρίζεται το NFC (Near Field Communication) και η ύπαρξη πολλαπλών καμερών. Επίσης, έχει βελτιωθεί η ενεργειακή υποστήριξη και έχει γίνει μετάβαση από το σύστημα αρχείων YAFFS στο ext4 στις νέες συσκευές.

Εικόνα 1.8: Το λογότυπο του Android 2.3 "Gingerbread"

1.3.6 Android 3.0 Honeycomb

Η έκδοση “Honeycomb”, βασισμένη στο Linux Kernel 2.6.36, παρουσιάστηκε στις 9 Μαΐου του 2011,με την ιδιαιτερότητα ότι προοριζόταν αποκλειστικά για tablets. Οι αλλαγές που έγιναν στην έκδοση αυτή έχουν να κάνουν κυρίως με τη βελτίωση της υποστήριξης των tablets. Υπάρχει ένα νέο, εντελώς διαφορετικό, User Interface και υποστηρίζονται διπύρηνοι και τετραπύρηνοι επεξεργαστές. Ακόμα, εχει απλοποιηθεί το multitasking έτσι ώστε ο χρήστης να μπορεί με τη χρήση ενός πλήκτρου (recent apps) να περνάει από μια εφαρμογή σε άλλη. Υπάρχει η δυνατότητα για Video Chat μέσω της εφαρμογής Google Talk καθώς η ανάγνωση βιβλίων μέσω του Google eBooks. Επιπλέον, μπορούν να κρυπτογραφηθούν όλα τα δεδομένα χρήστη.



Εικόνα 1.9: Το λογότυπο του Android 3.0 "Honeycomb"

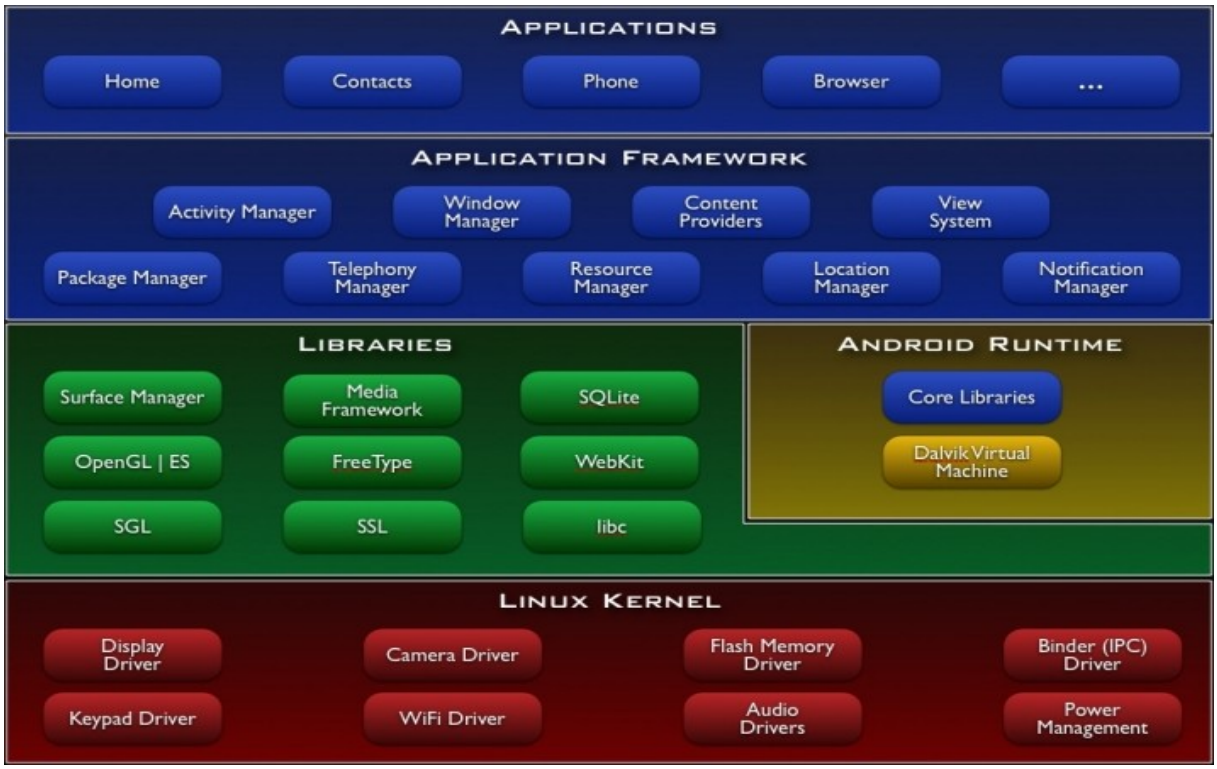
1.3.7 Android 4.0 Ice Cream Sandwich

Εικόνα 1.10: Το λογότυπο του Android 4.0 "Ice Cream Sandwich"

Η έκδοση “Ice Cream Sandwich”, βασισμένη στο Linux Kernel 3.0.1, παρουσιάστηκε στις 19 Οκτωβρίου του 2011. Για άλλη μια φορά έχει βελτιωθεί η ταχύτητα και η απόδοση του συστήματος. Πλέον στο User Interface, το οποίο είναι και παλι διαφορετικό, υπάρχουν εικονικά πλήκτρα τα οποία παίρνουν τη θέση των φυσικών ή αφής που υπήρχαν στις συσκευές. Βελτίωση της ασφάλεια του συστήματος με την προσθήκη αναγνώρισης προσώπου για να ξεκλειδώσει η συσκευή. Ο browser μπορεί να ανοίξει ταυτόχρονα μέχρι και 16 καρτέλες. Υπάρχει η δυνατότητα ο χρήστης να τερματίσει εφαρμογές οι οποίες τρέχουν στο background, ενώ μπορεί να θέσει και όρια στην κίνηση πακέτων δεδομένων. Η εφαρμογή Android Beam αξιοποιεί πλέον το NFC αφού επιτρέπει την αποστολή δεδομένων από τη συσκευή σε όσες βρίσκονται εντός μιας μικρής ακτίνας εμβέλειας. Ακόμα με την ύπαρξη του Wi-Fi Direct συσκευές μπορούν να συνδεθούν μεταξύ τους ασύρματα χωρίς την μεσολάβηση κάποιου access point. Τέλος, υποστηρίζεται η εγγραφή βίντεο σε 1080p.

1.4 Αρχιτεκτονική του Android

Το Android δεν είναι μόνο ένα λειτουργικό σύστημα. Είναι μια στοίβα λογισμικού η οποία αποτελείτε από το λειτουργικό σύστημα, τις υπηρεσίες διασύνδεσης με τις εφαρμογές (middleware) και τέλος από τις κύριες (core) εφαρμογές, μεταξύ αυτών, ενός email cient, μιας εφαρμογής διαχείρισης SMS, ενός ημερολογίου, ενός browser, εφαρμογή διαχείρισης επαφών, και άλλες οι οποίες έρχονται δεμένες με την υπόλοιπη στοιβάδα λογισμικού του Android. Στο επίσημο σχεδιάγραμμα που ακολουθεί (Εικόνα 1.11) θα δούμε οπτικά την αρχιτεκτονική αυτή.



Εικόνα 1.11: Η Αρχιτεκτονική του Android

Από ότι βλέπουμε λοιπόν η αρχιτεκτονική του λειτουργικού συστήματος αποτελείτε από 5 βασικά επίπεδα.

- Τον πυρήνα Linux (Linux Kernel)
- Τις εγγενείς και τις προηγμένες βιβλιοθήκες (Libraries)
- Την εικονική μηχανή Dalvik (Dalvik VM)
- Τον χρόνο εκτέλεσης (Android Runtime)
- Το πλαίσιο εφαρμογής (Application Framework)

1.4.1 Πυρήνας Linux (Linux Kernel)

Η βάση της στοίβας λογισμικού του Android είναι ο πυρήνας Linux. Ο τροποποιημένος πυρήνας του συστήματος βασίζεται στην έκδοση 2.6 (και στην έκδοση 3.0.1 για το Android 4.0) του Linux Kernel, η οποία υποστηρίζει όλες τις κύριες λειτουργίες του λειτουργικού συστήματος. Οι λειτουργίες αυτές αφορούν διαχείριση μνήμης, διαχείριση διεργασιών, λειτουργίες δικτύου, ασφάλεια του λειτουργικού, και ένα σύνολο οδηγών υλικού (hardware drivers). Οι οδηγοί αυτοί είναι υπεύθυνοι για την επικοινωνία του software με το hardware της συσκευής. Ενδεικτικά ο πυρήνας του Android περιέχει:

- Οδηγό προβολής οθόνης
- Οδηγό Wifi και Bluetooth
- Οδηγό κάμερας
- κλπ

Ο πυρήνας του Android μπορεί να βασίζεται στον πυρήνα του Linux, αλλά διαφέρει αρκετά από αυτόν. Ο λόγος είναι οι αλλαγές στην αρχιτεκτονική που έχει κάνει η Google για να είναι ελαφρύτερος και βελτιστοποιημένος για χρήση σε κινητές συσκευές. Αυτό σημαίνει ότι παρότι το Android είναι κατά βάση Linux, επί της ουσίας είναι αρκετά δύσκολο να τρέξουν εφαρμογές ή να χρησιμοποιηθούν βιβλιοθήκες από τη μία πλατφόρμα στην άλλη. Ο Linus Torvalds έχει αναφέρει ότι τελικά στο μέλλον το Android και το Linux θα μοιράζονται έναν κοινό πυρήνα, αλλά αυτό θα αργήσει 4-5 χρόνια ακόμα.

1.4.2 Βιβλιοθήκες

Στο δεύτερο επίπεδο της στοίβας έχουμε τις βιβλιοθήκες του Android. Αυτές ουσιαστικά αποτελούν τα APIs που είναι διαθέσιμα στους προγραμματιστές για την ανάπτυξη των εφαρμογών. Οι βιβλιοθήκες από μόνες τους δεν αποτελούν εφαρμογές αλλά ενσωματώνονται και χρησιμοποιούνται από τις εφαρμογές για τις διάφορες λειτουργίες που παρέχει η καθεμία από αυτές. Ουσιαστικά αποτελούν ένα από τα δομικά υλικά των εφαρμογών, και άρα είναι αναπόσπαστο κομμάτι τους. Οι δυνατότητες των βιβλιοθηκών του Android γίνονται εμφανείς στους προγραμματιστές στην στοίβα του πλαισίου εφαρμογής.

Το σύνολο σχεδόν των βιβλιοθηκών είναι γραμμένο σε C και C++, οι οποίες έχουν μεταγλωττιστεί για τη χρήση τους από το λειτουργικό. Μερικές από τις κύριες βιβλιοθήκες του Android είναι:

- **System C library** – μια ενσωμάτωση της standard βιβλιοθήκης συστήματος της C (libc) τροποποιημένη για κινητές συσκευές βασισμένες στο Linux.
- **Βιβλιοθήκες Πολυμέσων** – Υποστηρίζει αναπαραγωγή και εγγραφή πολλών δημοφιλών μέσων ήχου και εικόνας,όπως: MPEG4, H.264, MP3, AAC, AMR, JPG, και PNG
- **Surface Manager** – διαχειρίζεται την πρόσβαση στο υποσύστημα προβολής, και συνθέτει απρόσκοπτα δισδιάστατα και τρισδιάστατα επίπεδα γραφικών τα οποία προέρχονται από πολλαπλές εφαρμογές.
- **LibWebCore** – μια μοντέρνα μηχανή υποστήριξης πλοήγηση στο διαδίκτυο (browser engine) η οποία χρησιμοποιείτε και από τον ενσωματωμένο browser του Android αλλά και από τις WebViews που ενσωματώνονται στις εφαρμογές.
- **SGL** – η γνωστή μηχανή δισδιάστατων γραφικών
- **Βιβλιοθήκες 3D** – μια υλοποίηση βασισμένη στα APIs του OpenGL ES 1. Οι βιβλιοθήκες χρησιμοποιούν είτε τρισδιάστατη επιτάχυνση υλικού, όπου αυτή είναι διαθέσιμη, είτε μια υψηλά βελτιωμένη τρισδιάστατη επιτάχυνση λογισμικού σε περίπτωση που η πρώτη δεν είναι διαθέσιμη.
- **FreeType** – παρέχει ευκρίνεια γραφικών στα bitmaps και τις γραμματοσειρές των εφαρμογών του συστήματος.
- **SQLite** – μια πανίσχυρη και συνάμα πολύ ελαφριά σχεσιακή βάση δεδομένων

1.4.3 Η εικονική μηχανή Dalvik

Σχεδόν το σύνολο των APIs του Android βασίζονται στη γλώσσα προγραμματισμού Java. Στην Java ως γνωστόν υπάρχει η λεγόμενη Java Virtual Machine στην οποία εκτελείτε ο κώδικας bytecode των εφαρμογών. Στο Android υπάρχει κάτι παρόμοιο και δεν είναι άλλο από την εικονική μηχανή Dalvik.

Η Dalvik λοιπόν είναι η εικονική μηχανή μέσω της οποίας τρέχουν οι εφαρμογές του Android. Η κάθε εφαρμογή τρέχει μέσω τις δικής της εικονικής μηχανής στη δικιά της διεργασία και για αυτό το λόγο καμία εφαρμογή δεν έχει επαφή με την άλλη, ενώ εκτελούνται ταυτόχρονα. Η Dalvik δεν υποστηρίζει τον κώδικα bytecode, αντί αυτού οι κλάσεις της Java γίνονται compile σε αρχεία .dex ώστε να τρέξουν στην VM. Τα αρχεία dex ουσιαστικά αποτελούν συμπιεσμένα δεδομένα για εξοικονόμηση χώρου κατά την εκτέλεση.

Το Android είναι από τη φύση του multitasking λειτουργικό σύστημα και για αυτό επιτρέπει στις εφαρμογές του να τρέχουν σε πολλά νήματα ταυτόχρονα και να απασχολούν πολλές διαδικασίες εάν αυτό είναι αναγκαίο. Για να γίνει αυτό εφικτό η μηχανή Dalvik είναι σχεδιασμένη για να έχει ελάχιστο αντίκτυπο στη χρήση της μνήμης. Χάρη στον λιτό της σχεδιασμό, το σύστημα είναι σε θέση να τρέχει πολλές εικονικές μηχανές ταυτόχρονα.

1.4.4 Χρόνος Εκτέλεσης Εφαρμογής (Android Runtime)

Ο χρόνος εκτέλεσης των εφαρμογών του Android, βρίσκεται στο ίδιο επίπεδο με τις κύριες βιβλιοθήκες και την μηχανή Dalvik. Εδώ βρίσκουμε το κοινό σημείο επαφής μεταξύ των δυνατοτήτων που παρέχουν οι βιβλιοθήκες και του χρόνου εκτέλεσης της εικονικής μηχανής Dalvik τις λειτουργίες τις οποίας, περιγράψαμε παραπάνω.

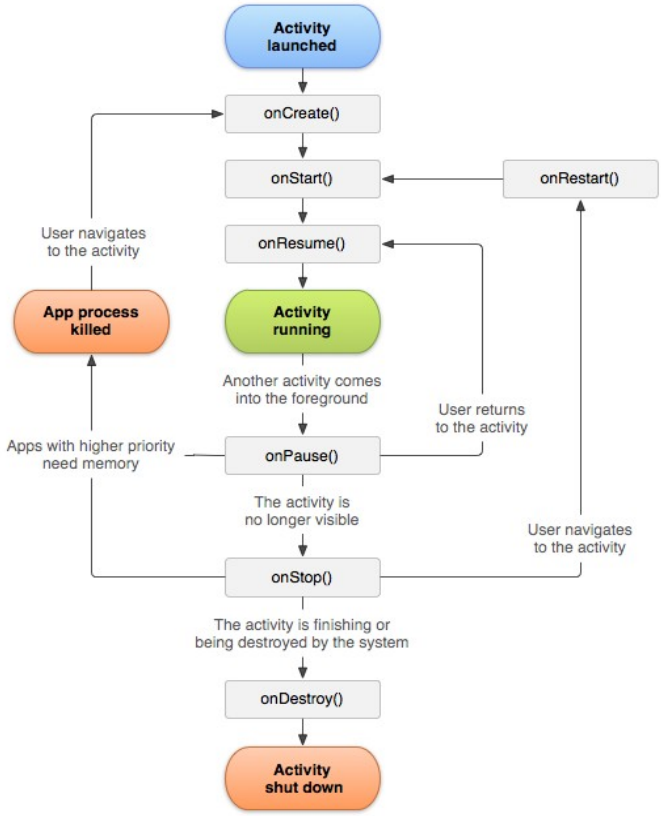
1.4.5 Πλαίσιο Εφαρμογής (Application Framework)

Το Android παρέχει στους developers μια ανοιχτού κώδικα πλατφόρμα ανάπτυξης και τη δυνατότητα να αναπτύξουν με αυτή ιδιαίτερα καινοτόμες και πλούσιες σε υλικό, εφαρμογές. Οι developers έχουν στην διάθεση τους τη δυνατότητα ελέγχου του υλικού της συσκευής και μέσω αυτής μπορούν να αποκτήσουν πρόσβαση σε υπηρεσίες εντοπισμού,

εκτέλεση διεργασιών παρασκηνίου, και πάρα πολλές ακόμη δυνατότητες οι οποίες βασίζονται στα APIs που είναι διαθέσιμα.

Στο επόμενο επίπεδο της αρχιτεκτονικής του Android λοιπόν, συναντάμε το πλαίσιο των εφαρμογών. Οι developers έχουν πρόσβαση σε όλα τα APIs μεταξύ αυτών και στα κύρια APIs που χρησιμοποιούν οι ενσωματωμένες εφαρμογές. Η δομή των εφαρμογών είναι τέτοια που ευνοείται η επαναχρησιμοποίηση δομικών συστατικών, και επίσης επιτρέπεται η χρήση των δυνατοτήτων τις μίας εφαρμογής από άλλες εφαρμογές, βέβαια κάτω από τις προδιαγραφές ασφάλειας του Android. Τα σημαντικότερα δομικά στοιχεία του πλαισίου εφαρμογών είναι:

- **Σύστημα προβολών (View System)** – αποτελεί ένα εκτενές σύνολο από αντικείμενα GUI τα οποία μπορούν να χρησιμοποιηθούν κατά το σχεδιασμό μιας εφαρμογής. Παραδείγματα προβολών είναι οι λίστες (listView), το πλέγμα (GridView), πεδία εισαγωγής κειμένου, κουμπιά, κλπ
- **Πάροχος Περιεχομένου (Content Provider) –** δίνει τη δυνατότητα στις εφαρμογές να μοιράζονται ή να ανταλλάσσουν δεδομένα μιας συγκεκριμένης μορφής η οποία ορίζεται από τον πάροχο. Παραδείγματα δεδομένων, είναι οι επαφές χρήστη και οι βάσεις δεδομένων των εφαρμογών.
- **Διαχειριστής Πόρων (Resource Manager) –** παρέχει πρόσβαση σε υλικό το οποίο δεν είναι σε μορφή κώδικα όπως πχ, εικόνες, αρχεία xml, πίνακες χαρακτήρων, κλπ
- **Διαχειριστής Ειδοποιήσεων (Notification Manager) –** δίνει στις εφαρμογές πρόσβαση στις υπηρεσίες ειδοποιήσεων χρήστη. Τέτοιες είναι οι ειδοποιήσεις στη notification bar, τα toast μηνύματα στο κάτω μέρος της οθόνης, η δόνηση του κινητού και η ενεργοποίηση της οθόνης, κλπ
- **Διαχειριστής Δραστηριοτήτων (Activity Manager) –** διαχειρίζεται τον κύκλο ζωής των δραστηριοτήτων και παρέχει δυνατότητα πλοήγησης από δραστηριότητα σε δραστηριότητα κρατώντας αποθηκευμένη στη μνήμη τη σειρά εκτέλεσης αυτών. Στο σχεδιάγραμμα (Εικόνα 1.12) φαίνεται λεπτομερώς ο κύκλος ζωής κάθε δραστηριότητας.



Εικόνα 1.12: Κύκλος ζωής μιας δραστηριότητας (Activity lifecycle)

1.5 Στο εσωτερικό μιας εφαρμογής του Android

Κάθε εφαρμογή αποτελείτε από ένα σύνολο αρχείων και φακέλων δομημένα σε μορφή project, τα οποία αφού γίνουν compiled μέσω του Android SDK μας δίνουν το αρχείο .apk. Το αρχείο αυτό αποτελεί την εφαρμογή και μπορούμε να εγκαταστήσουμε στις συσκευές μας.

Ξεκινώντας, η κάθε εφαρμογή αποτελείτε όπως είπαμε από πολλά αρχεία δομημένα σε φακέλους. Όλες οι εφαρμογές πρέπει να έχουν ένα μοναδικό όνομα πακέτου (package name) το οποίο χρησιμοποιείτε από το λειτουργικό σύστημα για αναγνώριση της εφαρμογής. Μια εφαρμογή μπορεί να αποτελείτε από πολλά υποπακέτα, εφόσον αυτό είναι απαραίτητο λόγω της πολυπλοκότητας τις εφαρμογής, αλλά μόνο από ένα κύριο.

1.5.1 Το αρχείο AndroidManifest.xml

Κάθε project εφαρμογής περιέχει ένα αρχείο στο οποίο βρίσκονται καταχωρημένες οι σημαντικότερες πληροφορίες της εφαρμογής, και το αρχείο αυτό ονομάζεται AndroidManifest.xml. Πρόκειται όπως λέει και το όνομα του για ένα αρχείο xml μέσα στο οποίο ο προγραμματιστής καταχωρεί τις σημαντικότερες πληροφορίες της εφαρμογής για χρήση από το λειτουργικό σύστημα. Κάποιες από αυτές τις πληροφορίες είναι:

- Το όνομα του πακέτου της εφαρμογής
- Το κανονικό της όνομα που φαίνεται στον χρήστη
- Η έκδοση των APIs που χρησιμοποιούνται
- Ο αριθμός έκδοσης της εφαρμογής
- Οι άδειες χρήσης που ζητάει η εφαρμογή
- Όλες οι δραστηριότητες, πάροχοι περιεχομένου, υπηρεσίες, κλπ, που περιέχει και χρησιμοποιεί η εφαρμογή.

Όπως αντιλαμβανόμαστε πρόκειται για πολύ σημαντικό αρχείο και αποτελεί κύριο συστατικό κάθε εφαρμογής.

1.5.2 Οι φάκελοι src & res

Στον φάκελο src (εκ του source) περιέχονται τα αρχεία κλάσης τις Java όλων των Activities, Services, Content Providers, βοηθητικά αρχεία, κλπ. Ο φάκελος περιέχει το πακέτο ή τα πακέτα της εφαρμογής τα οποία περιέχουν τα αρχεία Java, και αποτελεί τον μοναδικό φάκελο στο project στον οποίο αποθηκεύονται τα αρχεία του κώδικα μας.

Ο φάκελος res (εκ του resources) περιέχει όλα τα αρχεία εικόνας, κειμένου, xml layout, κλπ τα οποία χρησιμοποιούνται από τις Activities που βρίσκονται στον φάκελο src. Φυσικά δεν βρίσκονται όλα τα αρχεία πόρων, σε έναν φάκελο, αλλά είναι χωρισμένα και ταξινομημένα σε υποφακέλους ανάλογα με το είδος τους. Συνηθισμένοι υποφάκελοι του κύριου φακέλου res, είναι ο φάκελος drawable ο οποίος περιέχει τα αρχεία εικόνας (.png, .jpg, .gif) τα οποία χρησιμοποιεί η εφαρμογή μας, ο φάκελος layout ο οποίος περιέχει όλα τα αρχεία xml τα οποία ορίζουν τα διάφορα layouts που υπάρχουν στην εφαρμογή, και τέλος ο φάκελος values στον οποίο αποθηκεύονται όλοι οι πόροι κειμένου που χρησιμοποιούνται στην εφαρμογή.

1.5.3 Οι υπόλοιποι φάκελοι του project

Ένα project αποτελείτε από περισσότερους από τους 3 βασικούς φακέλους, κάποιοι από τους οποίους μπορεί να θεωρηθούν και περιττοί αναλόγως την περίπτωση. Στο project λοιπόν περιλαμβάνονται και ο φάκελος με τα διαθέσιμα APIs αναλόγως την έκδοση που έχουμε επιλέξει να δουλέψουμε, ο φάκελος με τις διαθέσιμες βιβλιοθήκες που έχουμε εισάγει στο build path του project μας, και επίσης περιλαμβάνει και τις διαβαθμίσεις του φακέλου res, όπως είναι οι φάκελοι drawable-hdpi, drawable-mdpi, layout-port, menu, κλπ. Σε αυτούς περιλαμβάνονται τα ειδικά διαμορφωμένα αρχεία πόρων που έχουμε τοποθετήσει ώστε να είναι διαθέσιμα από το λειτουργικό σύστημα, αναλόγως την περίσταση.

1.5.4 Δομικά Μέρη μιας Εφαρμογής

Παραπάνω αναφέραμε ότι όλα τα δομικά μέρη της εφαρμογής πρέπει να αναφέρονται αναλυτικά στο αρχείο AndroidManifest.xml, πια είναι όμως αυτά τα δομικά μέρη και πια η λειτουργία του καθενός;

- **Δραστηριότητες (Activities)** – Πρόκειται ίσως για το κύριο δομικό στοιχείο μιας εφαρμογής. Δραστηριότητα είναι μια οθόνη διεπαφής χρήστη (GUI) και προβολής πληροφοριών. Κάθε εφαρμογή έχει τόσες Activities όσες και οι διαφορετικές οθόνες οι οποίες εμφανίζονται στον χρήστη. Όλες οι δραστηριότητες συνεργάζονται μεταξύ τους για να δώσουν στον χρήστη μια συνολική εμπειρία χρήσης της εφαρμογής.
- **Προθέσεις (Intents) –** Οι δραστηριότητες επικοινωνούν και εναλάσουν την λειτουργία τους μέσω των Intents. Ουσιαστικά τα Intents εξασφαλίζουν την μετάβαση από την μία δραστηριότητα σε μια άλλη και επίσης χρησιμοποιούνται για ανταλλαγή δεδομένων. Η ανταλλαγή δεδομένων, μπορεί να γίνει είτε μεταξύ των Activities μιας εφαρμογής, είτε από τη μία εφαρμογή στην άλλη. Παραδείγματος χάρη μπορούμε μέσω ενός Intent να εκκινήσουμε έναν browser ώστε να μας ανοίξει απευθείας ένα url το οποίο έχουμε παρέχει εμείς μέσω ενός Intent.
- **Υπηρεσίες (Services) –** Πρόκειται για λειτουργίες της εφαρμογής οι οποίες είναι σχεδιασμένες να τρέχουν στο παρασκήνιο και να επιστρέφουν αποτελέσματά ακόμη και όταν η εφαρμογή δεν είναι στο προσκήνιο. Πχ μια εφαρμογή media player μπορεί

μέσω μιας υπηρεσίας να συνεχίσει να παίζει μουσική ακόμη και όταν το κύριο παράθυρο της εφαρμογής δεν βρίσκεται στο προσκήνιο.

- **Πάροχος Περιεχόμενου (Content Providers) -** Η ανταλλαγή δεδομένων από μια εφαρμογή στην άλλη όπως είπαμε παραπάνω μπορεί να γίνει μέσω ενός Intent, ένας πάροχος περιεχομένου όμως έχει πιο σύνθετη λειτουργία. Οι content providers μιας εφαρμογής διαχειρίζονται συγκεκριμένα δεδομένα της εφαρμογής τα οποία έχει ορίσει ο προγραμματιστής κατά την κατασκευή του. Συνηθισμένα δεδομένα τα οποία μοιράζονται μέσω Content Providers, είναι οι βάσεις δεδομένων SQLite μιας εφαρμογής, και οι επαφές του χρήστη.
- **Δέκτες Μετάδοσης (Broadcast Receivers) –** Πρόκειται για ένα είδους υπηρεσία η οποία αντιλαμβάνεται κάποια γεγονότα του συστήματος και αναλαμβάνει να ενημερώσει το σύστημα η τις υπόλοιπες εφαρμογές. Ο σκοπός τους είναι διπλός καθότι μπορούν και να ενημερωθούν για κάποιο συμβάν από άλλες εφαρμογές, αλλά και να ειδοποιήσουν τις υπόλοιπες εφαρμογές και το σύστημα για κάποιο συμβάν που τις ενεργοποίησε. Δεν έχουν γραφικό περιβάλλον αλλά μπορούν να προβάλουν ειδοποίηση στον χρήστη μέσω της μπάρας ειδοποιήσεων. Συνήθως χρησιμοποιούνται ως διαμεσολαβητές μεταξύ των Activities και των Services μιας εφαρμογής.

1.6 Ασφάλεια στο Android

Τη στιγμή που μια εφαρμογή εγκαθίσταται στη συσκευή, λειτουργεί αποκλειστικά στο δικό της εικονική μηχανή η οποία αποτελεί και το πλαίσιο ασφαλείας (sandbox) της εφαρμογής. Το Android είναι ένα λειτουργικό σύστημα πολλών χρηστών στο οποίο:

- Η κάθε εφαρμογή αντιμετωπίζεται σαν διαφορετικός χρήστης
- Από προεπιλογή το σύστημα δίνει έναν μοναδικό αριθμό ID ο οποίος είναι άγνωστος στην εφαρμογή. Το σύστημα αναθέτει συγκεκριμένες άδειες χρήσης στα αρχεία της εφαρμογής, και μόνο η εφαρμογή με το σωστό ID μπορεί να έχει πρόσβαση σε αυτά.
- Κάθε εφαρμογή τρέχει στην δική της εικονική μηχανή (VM) απομονωμένη από τις υπόλοιπες εφαρμογές. Η κάθε VM εκκινείτε μόλις ζητηθεί από το σύστημα και κλείνει είτε επειδή δεν χρησιμοποιείτε πλέον, είτε επειδή το σύστημα θέλει να ελευθερώσει τους πόρους της μνήμης για χρήση από άλλη εφαρμογή.

Με αυτό τον τρόπο το Android χρησιμοποιεί την αρχή των ελαχίστων δικαιωμάτων. Η κάθε εφαρμογή έχει πρόσβαση μέσω του AndroidManifest μόνο σε όσους πόρους συστήματος χρειάζεται και κανέναν περισσότερο. Οι πόροι και τα δικαιώματα που απαιτούνται από μία εφαρμογή γίνονται γνωστά στον χρήστη τη στιγμή της εγκατάστασης της, και ο χρήστης μπορεί να επιλέξει να μην εγκαταστήσει μια εφαρμογή εφόσον δεν συμφωνεί να τις παρέχει πρόσβαση στους πόρους που ζητάει.

ΚΕΦΑΛΑΙΟ 2

Ανάπτυξη Εφαρμογών στο Android

2.1 Κύκλος Ανάπτυξης Εφαρμογής

Η ανάπτυξη εφαρμογών στο Android είναι μια σύνθετη και χρονοβόρα διαδικασία η οποία συνοψίζεται σε 4 βασικά στάδια, αλλά και αρκετά επί μέρους, τα οποία θα σχολιαστούν μεταξύ των βασικών.

2.1.1 Εγκατάσταση Λογισμικού

Στο πρώτο στάδιο της ανάπτυξης ο προγραμματιστής καλείτε να στήσει το περιβάλλον εργασίας στο οποίο θα γίνει ο σχεδιασμός, η ανάπτυξη, ο έλεγχος, και η λειτουργία των εφαρμογών. Μπορεί να επιλέξει όποιο περιβάλλον ανάπτυξης (IDE) τον εξυπηρετεί καλύτερα και να χρησιμοποιήσει όλα τα εργαλεία του Android SDK μηδενός εξαιρουμένου.

Στη συνέχεια θα πρέπει να δημιουργήσει έναν αριθμό από εικονικές συσκευές στην διαχείριση εικονικών συσκευών (AVD) για να δοκιμάσει την λειτουργία της εφαρμογής σε διαφορετικές πραγματικές συνθήκες λειτουργίας. Ιδανικά ο developer θα διαθέτει έναν αριθμό διαφορετικών φυσικών συσκευών ώστε να δοκιμάσει ο ίδιος πως συμπεριφέρεται η εφαρμογή του σε κάθε περίπτωση, όμως αυτή η πρακτική μπορεί να αποδειχθεί πολυδάπανη και χρονοβόρα. Εδώ αναλαμβάνουν δράση η ευελιξία των AVDs, για τις οποίες θα γράψουμε περισσότερα παρακάτω.

Εικόνα 2.1: Πρώτο βήμα - Εγκατάσταση Λογισμικού

2.1.2 Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής

Πρόκειται αν μη τι άλλο για τη πιο χρονοβόρα και πολύπλοκη διαδικασία. Σε αυτό το στάδιο ο προγραμματιστής πρέπει να αποφασίσει για τις δυνατότητες και το περιεχόμενο που θα περιλαμβάνει η εφαρμογή, να εντοπίσει ποιες από αυτές τις δυνατότητες είναι εφικτές και ποιες θέλουν παραπάνω έρευνα για να προστεθούν στο μέλλον, να σχεδιάσει το layout με γνώμονα την λειτουργικότητα και να αποφύγει υπερβολές στο σχεδιασμό, και τέλος να δέσει αρμονικά τον κώδικα με το layout για να φέρει το τελικό αποτέλεσμα.

Η διαδικασία ξεκινάει με ένα νέο Project το οποίο θα περιέχει τον πηγαίο κώδικα, τις εικόνες, τα κείμενα και γενικά ότι χρειάζεται η εφαρμογή για να τρέξει ως οφείλει. Στο project του ο developer θα πρέπει να φροντίσει ώστε το υλικό του να είναι τακτοποιημένο και ο κώδικας του ευανάγνωστος ώστε να ακολουθήσει η διαδικασία του Debugging.

Εικόνα 2.2: Δεύτερο βήμα - Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής

2.1.3 Αποσφαλμάτωση (Debugging) και Δοκιμαστική Φάση Εφαρμογής

Η διαδικασία του debugging είναι εξίσου κρίσιμη και μερικές φορές και εξίσου χρονοβόρα με την διαδικασία ανάπτυξης του πηγαίου κώδικα της εφαρμογής. Αποτελείτε από αρκετά επί μέρους στάδια τα οποία αναλύονται παρακάτω.

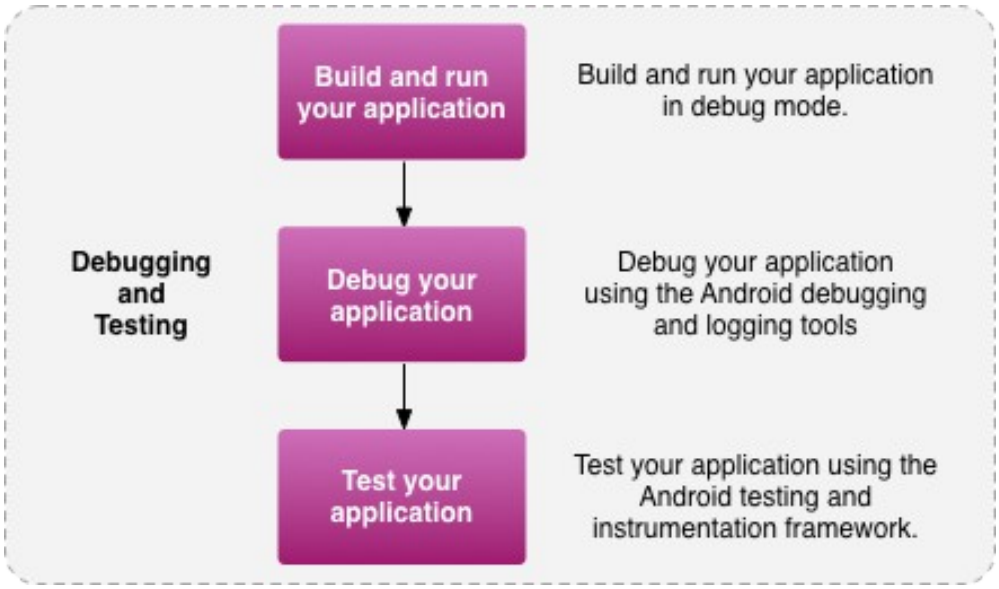
Το πρώτο στάδιο αφορά το αρχικό χτίσιμο της εφαρμογής και η λειτουργία αυτής σε debug mode. Για να γίνει το compile της εφαρμογής φυσικά τα περισσότερα περιβάλλοντα ανάπτυξης (IDE) προϋποθέτουν ότι ο κώδικας δεν έχει κανένα συντακτικό λάθος, αλλιώς ειδοποιούν τον χρήστη να τα διορθώσει. Αφού γίνει το compile η εφαρμογή μπορεί να δοκιμαστεί είτε σε εικονική συσκευή μέσω του AVD Manager, είτε απευθείας σε φυσική συσκευή μέσω ADB push εντολής. Για το ADB θα μιλήσουμε εκτενώς παρακάτω.

Στο δεύτερο στάδιο ο προγραμματιστής καλείτε να αντιμετωπίσει τα λειτουργικά και αισθητικά προβλήματα της εφαρμογής του, πρώτα εντοπίζοντας τα στην λειτουργία της συσκευής και μετά διορθώνοντας τα κομμάτια του κώδικα που δημιουργούν τα σφάλματα. Το κύριο εργαλείο που κάνει αυτή τη διαδικασία εφικτή είναι το “LogCat” το οποίο μας επιστρέφει το stack trace του κώδικα στο σημείο εκείνο που συνέβη το σφάλμα. Υπάρχουν φυσικά και άλλα εργαλεία τα οποία θα αναλυθούν εκτενώς παρακάτω.

Στο τρίτο στάδιο ο προγραμματιστής αφού έχει τελειώσει την αποσφαλμάτωση (debugging) επιστρέφει στο βήμα ένα, δηλαδή στο compile και τη δοκιμή της εφαρμογής σε εικονική η φυσική συσκευή ώστε να διαπιστώσει τα αποτελέσματα του 2 βήματος, της αποσφαλμάτωσης.

Ένα προαιρετικό στάδιο είναι η “Δημόσια δοκιμαστική φάση” της εφαρμογής. Σε αυτή τη φάση εθελοντές προσφέρονται να δοκιμάσουν τις λειτουργίες της εφαρμογής στις συσκευές τους και να αναφέρουν προβλήματα, παρατηρήσεις, προτάσεις και άλλα σχόλια που μπορεί προκύψουν από τη χρήση της εφαρμογής.

Φυσικά η διαδικασία του debugging είναι σαν ένα βρόγχος (loop) που επαναλαμβάνεται συνέχεια μέχρι να εντοπιστούν και να διορθωθούν όλα τα σφάλματα της εφαρμογής, και για αυτό το λόγο μπορεί να αποδειχθεί εξαιρετικά χρονοβόρα.



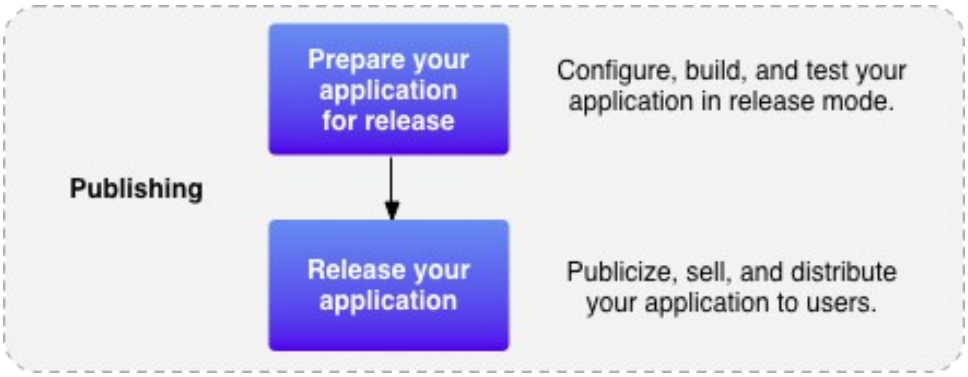
Εικόνα 2.3: Τρίτο βήμα - Δοκιμή και Debugging της εφαρμογής

2.1.4 Τελική έκδοση και δημοσίευση της εφαρμογής στο κοινό

Στο τέταρτο και τελευταίο στάδιο της ανάπτυξης ο προγραμματιστής έχει να κάνει μερικές τελευταίες κινήσεις. Πρώτον πρέπει να έχει διορθώσει όλα τα σφάλματα που προέκυψαν από την διαδικασία αποσφαλμάτωσης, να κάνει τις τελευταίες ρυθμίσεις και tweaks της εφαρμογής, και να κάνει το τελικό compile της εφαρμογής σε κανονική λειτουργία αυτή τη φορά και όχι debug.

Στη συνέχεια ακολουθεί η διάθεση της εφαρμογής με το μέσο της επιλογής του developer. Μπορεί να την διαθέσει στο Google Play, αφού πρώτα κάνει λογαριασμό developer, ή να την διαθέσει σε κάποιο εναλλακτικό market όπως το marketplace της Amazon. Μπορεί κατά τη δημοσίευση σε οποιοδήποτε να ορίσει τιμή πώλησης ή να διαθέσει την εφαρμογή δωρεάν. Τη στιγμή που γράφονται αυτές οι γραμμές δεν είναι εφικτό ακόμη για τους Έλληνες Developers να διαθέσουν εφαρμογές επί πληρωμή, με την Google να μην έχει ανακοινώσει ακόμη πότε αυτό θα γίνει εφικτό.

Επίσης άλλο ένα μέσο διάθεσης μπορεί να είναι η προσωπική η εταιρική ιστοσελίδα του δημιουργού. Το μειονέκτημα φυσικά σε αυτή τη περίπτωση είναι η έλλειψη ελέγχου για updates της εφαρμογής από έναν αυτόματο μηχανισμό ελέγχου και λήψης όπως είναι τα διάφορα marketplaces.



Εικόνα 2.4: Τέταρτο βήμα - Δημοσίευση της Εφαρμογής

2.2 Android SDK

Το Android SDK (Software Developers Kit) αποτελεί μια συλλογή εργαλείων και βιβλιοθηκών που καθιστούν εφικτή την ανάπτυξη εφαρμογών στο Android. Τι στιγμή που γράφονται αυτές οι γραμμές, το SDK έχει φτάσει στην έκδοση r19 η οποία υποστηρίζει το Android 4.0.3. Το λογισμικό ανάπτυξης λοιπόν περιλαμβάνει μια μεγάλη λίστα με εργαλεία ανάπτυξης. Σε αυτά περιλαμβάνονται:

- Εργαλεία Debugging των εφαρμογών
- Βιβλιοθήκες
- Εξομοιωτής συσκευών (Android Virtual Machines)
- Documentation
- Δείγματα Κώδικα
- Tutorials

Το SDK υποστηρίζει πολλά δημοφιλή λειτουργικά συστήματα συμπεριλαμβανομένων όλων των σύγχρονων διανομών Linux, το MAC OS X 10.4.9 και μεταγενέστερα, και τα Windows XP και τις μεταγενέστερες εκδόσεις.

Το λογισμικό ανάπτυξης αποτελείτε από πακέτα τα οποία βρίσκονται αποθηκευμένα σε ένα επίσημο repo της Google, και ο προγραμματιστής μπορεί να κατεβάσει πέραν των βασικών πακέτων, και άλλα τα οποία υποστηρίζουν παλαιότερες εκδόσεις του Android, ή άλλες συσκευές εκτός κινητών συσκευών (πχ Google TV Addon).

Όσον αφορά την υποστήριξη παλαιότερων εκδόσεων του Android, το SDK κάνει εφικτή την υποστήριξη σε αυτές δίνοντας στον προγραμματιστή την δυνατότητα να στοχεύσει αυτός σε πια APIs θα απευθύνεται η εφαρμογή του. Αυτό είναι αναγκαίο λόγω του ότι πολλοί χρήστες έχουν παλαιότερες λειτουργικές συσκευές οι οποίες κυκλοφορήσαν με παλαιότερες εκδόσεις του Android (πχ 1.6 ή 2.1), και ο κατασκευαστής της συσκευής δεν έχει ή δεν πρόκειται να βγάλει αναβάθμιση για την συσκευή τους. Το πρόβλημα αυτό είναι γνωστό σαν διάσπαση του Android (Android Fragmentation) και θα αναλυθεί εκτενώς παρακάτω.

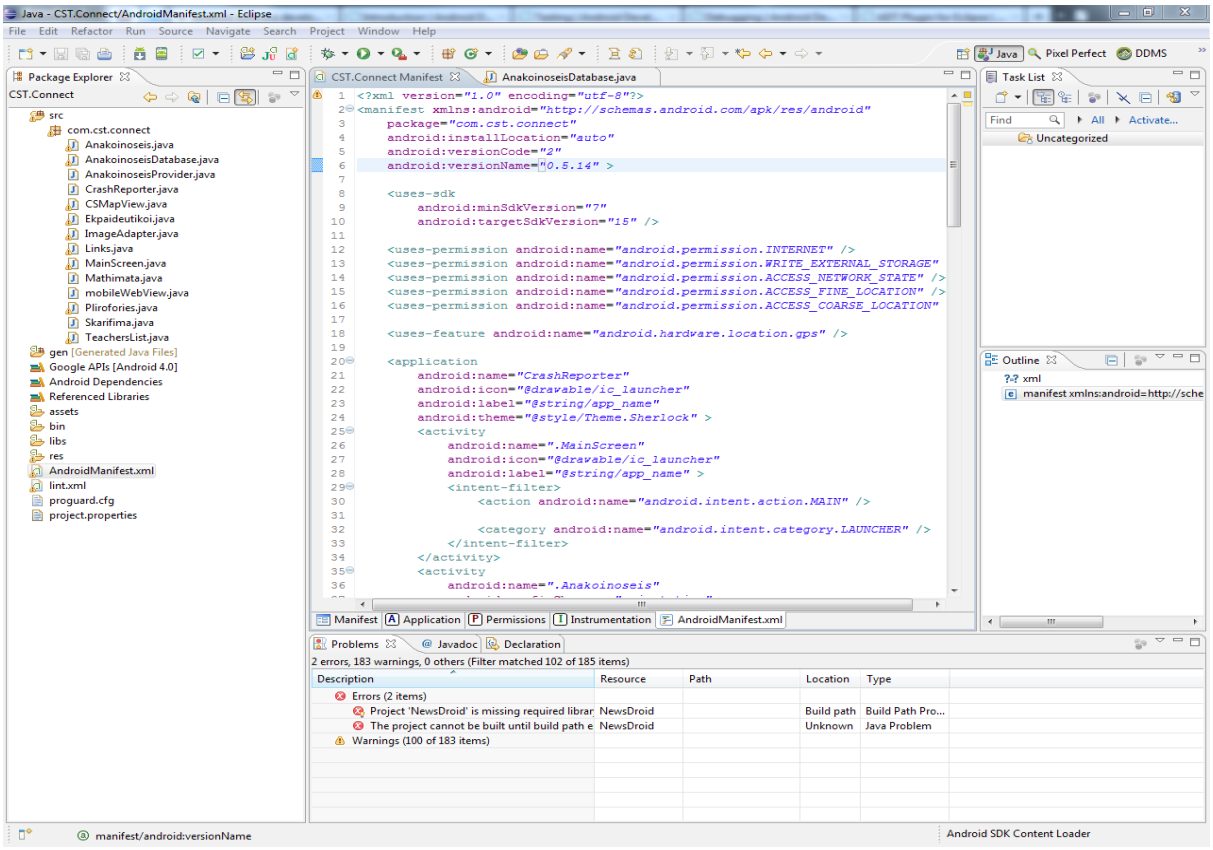
2.3 Χρήση του Eclipse IDE μαζί με το ADT (Android Development Tools)

Ο προγραμματισμός στο Android βασίζεται στην γλώσσα Java και ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει έναν οποιονδήποτε text editor για να γράψει κώδικα για να επεξεργαστεί τα αρχεία *.Java και *.XML και μετέπειτα να τα κάνει compile μέσω γραμμής εντολών χρησιμοποιώντας το JDK (Java Development Kit). Ο συγκεκριμένος τρόπος ανάπτυξης δεν είναι ιδιαίτερα φιλικός στον χρήστη γιαυτό συνίσταται η χρήση ενός IDE (Integrated Development Environment) που να υποστηρίζει Java, όπως το Eclipse ή το Netbeans.

Η Google υποστηρίζει επίσημα το Eclipse και έχει αναπτύξει ειδικά για αυτό το ADT plugin, το οποίο παρέχει σύνδεση με το Android SDK με όλες τις δυνατότητες που περιλαμβάνει αυτό. Επίσης το plugin παρέχει σύνδεση με τον AVD Manager, για διαχείριση και εκκίνηση από το GUI του, εικονικών συσκευών Android για δοκιμές και debugging των εφαρμογών.

Φυσικά όπως είπαμε και παραπάνω, ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει τον Text Editor ή IDE της επιλογής του για τη δημιουργία του κώδικα και μετέπειτα να χρησιμοποιήσει τα εργαλεία JDK και Apache Ant μέσω γραμμής εντολών για να κάνει compile την εφαρμογή του ώστε να την τεστάρει με όλες τις δυνατότητες που το παρέχει το Android SDK.

Η επιλογή ενός IDE που κάνει όλη την πολύπλοκη δουλεία για μας είναι προφανής λοιπόν. Επίσης τα περισσότερα παραδείγματα και άρθρα για το Android στηρίζονται στο γεγονός ότι η πλειονότητα των developers χρησιμοποιεί το Eclipse μαζί με το ADT plugin οπότε ξεκινάνε με αυτό σαν δεδομένο.



Εικόνα 2.5: Προεπιλεγμένη διάταξη του Eclipse IDE

2.4 Προκλήσεις ανάπτυξης εφαρμογών στο Android

Η ανάπτυξη εφαρμογών στο προγραμματιστικό περιβάλλον του Android είναι μια αρκετά απαιτητική διαδικασία, διότι απαιτεί την υποστήριξη εκατοντάδων συσκευών, την υλοποίηση και υποστήριξη ενός λειτουργικού περιβάλλοντος διεπαφής χρήστη, και άλλα πολλά τα οποία θα αναλύσουμε παρακάτω.

2.4.1 Android Design Guidelines

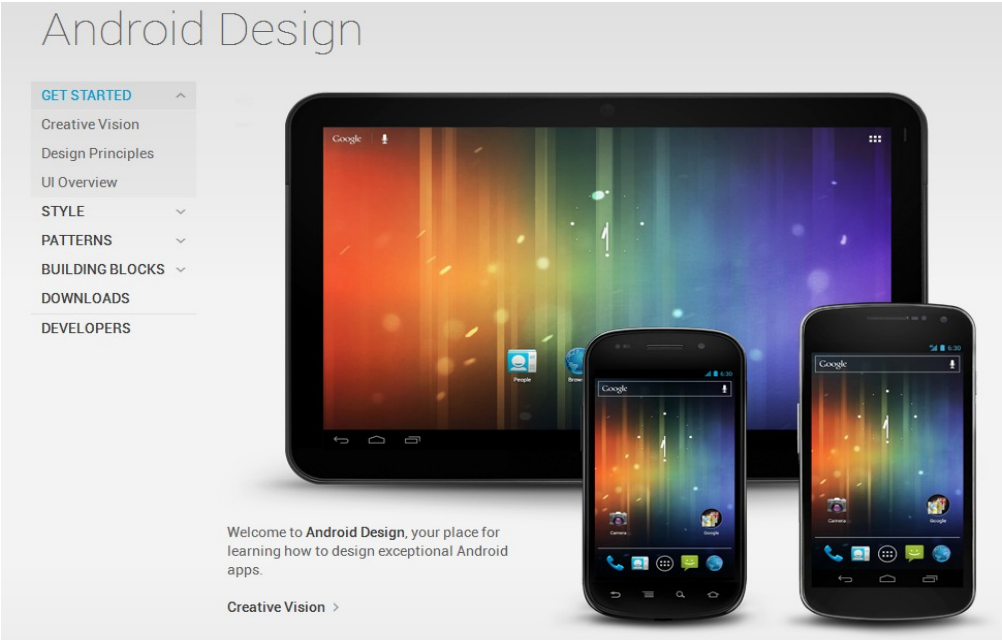
Η υλοποίηση μιας νέας εφαρμογής στο Android αλλά και στα υπόλοιπα λειτουργικά συστήματα, ξεκινάει από τις λειτουργικές απαιτήσεις, δηλαδή τις δυνατότητες και λειτουργίες που θα υποστηρίζει η εφαρμογή, και συνεχίζει με τον σχεδιασμό του UI layout που θα δίνει πρόσβαση στον χρήστη στις παραπάνω λειτουργίες. Ο σχεδιασμός λοιπόν έχει

μεγαλύτερη σημασία από τις ίδιες τις λειτουργίες τις εφαρμογής μιας και είναι το σημείο πρόσβασης προς αυτές!

Μια κακοσχεδιασμένη εφαρμογή η οποία κρύβει τις λειτουργίες τις πίσω από πολλά κουμπιά και μενού ενδέχεται να μπερδέψει τον χρήστη και ίσως και να τον αποτρέψει από το να την χρησιμοποιήσει. Αυτό φυσικά δεν είναι επιθυμητό γιαυτό και υπάρχουν κάποια ενδεικτικά επίσημα Guidelines (οδηγίες) τα οποία υποδεικνύουν στους developers τις ιδεατές και μη συμπεριφορές κατά τον σχεδιασμό της εφαρμογής τους.

Στα πρώτα χρόνια του Android ο σχεδιασμός τον εφαρμογών ήταν είτε καθαρή μεταφορά (port) από άλλο OS (πχ iOS, Symbian) και συνήθως δεν ακολουθούσε καμία κοινή γραμμή σχεδιασμού με τις υπόλοιπες εφαρμογές κάτι που πολλές φορές μπέρδευε τον χρήστη που είχε συνηθίσει το κοινό layout πολλών εφαρμογών των υπολοίπων mobile OSes. Αυτό η Google προσπάθησε να το αλλάξει με την έλευση του Android 4.0 (Ice Cream Sandwich) όποτε και δημοσίευσε στο Ίντερνετ τη σελίδα Android Design για να καθοδηγήσει τους developers σε μία κοινή γραμμή ανάπτυξης εφαρμογών ώστε να πετύχει αύξηση λειτουργικότητας.

Η αύξηση λειτουργικότητας επιτυγχάνεται λόγω του ότι ο χρήστης θα έχει γνωστά σημεία επαφής σε κάθε εφαρμογή οπότε δεν θα χρειάζεται να ψάχνει εκ νέου πως να επιστρέψει στην αρχική οθόνη ή που βρίσκεται το μενού των επιλογών, κλπ.



Εικόνα 2.6: Αρχική Σελίδα Android Design

Στο site υπάρχει πληθώρα παραδειγμάτων “καλού σχεδιασμού” τα οποία αφορούν την χρήση της Action Bar, την χρήση των swappable tabs, την δυνατότητα δηλαδή να αλλάζουμε οθόνες σέρνοντας αριστερά η δεξιά το δάχτυλό μας στην οθόνη της συσκευής, κλπ. Επίσης υπάρχει και η ενότητα “Downloads” στην οποία υπάρχει ένα πλήρες πακέτο εικονιδίων τα οποία μπορούμε να χρησιμοποιήσουμε στα μενού των εφαρμογών μας η όπου αλλού θέλουμε!

Οι οδηγίες φυσικά είναι ενδεικτικές και όχι αναγκαστικές. Αν κάποιος developer θέλει να “παρεκτραπεί” και να δώσει κάποιο εντελώς διαφορετικό interface το οποίο όμως θα εξυπηρετεί καλύτερα τους δικούς του σκοπούς, μπορεί να το κάνει ελεύθερα. Αυτή άλλωστε είναι και η ομορφιά του Android!

2.4.2 Υποστήριξη πολλαπλών συσκευών

Το λειτουργικό σύστημα Android τρέχει σε μια πληθώρα συσκευών οι οποίες μπορεί να έχουν πολύ διαφορετικές προδιαγραφές η μία από την άλλη. Η διαφοροποίηση των συσκευών εντοπίζεται:

- Στις πολλές εκδόσεις του Android που υπάρχουν. Αυτή τι στιγμή που γράφονται αυτές ο γραμμές υπάρχουν 9 διαθέσιμες εκδόσεις (1.5-4) με κυρίαρχη έκδοση την 2.3 Gingerbread.
- Στην μεγάλη ποικιλία hardware που κυκλοφορεί στην αγορά. Υπάρχουν συσκευές με επεξεργαστή στα 600Mhz και 256MB διαθέσιμης μνήμης RAM και υπάρχουν και συσκευές με επεξεργαστή 4 πυρήνων (Quad Core) στα 1.5Mhz και 1GB μνήμης RAM. Εκτός από τις διαφορές σε επίπεδο microchip η κύρια διαφορά μεταξύ των συσκευών εντοπίζεται στην μεγάλη ποικιλία διαστάσεων οθόνης και πυκνότητας pixel

Ο developer λοιπόν για να κάνει την εφαρμογή του προσβάσιμη σε όσο τον δυνατόν περισσότερες συσκευές χρηστών, πρέπει να λάβει σοβαρά υπόψιν του τις 2 παραπάνω παραμέτρους και να σχεδιάσει την εφαρμογή του έτσι ώστε αυτή να υποστηρίζει την πλειονότητα των συσκευών. Βέβαια αυτό σημαίνει συνεχή προσαρμογή της εφαρμογής στις νέες συνθήκες που μπορεί να προκύψουν, και χρήση των νέων δυνατοτήτων που ενδεχομένως θα παρέχει μια νέα έκδοση του λειτουργικού, χωρίς να επηρεάζεται η υποστήριξη στις παλαιότερες συσκευές. Αυτό είναι μεγάλο ζήτημα το οποίο θα αναλυθεί εκτενώς παρακάτω.

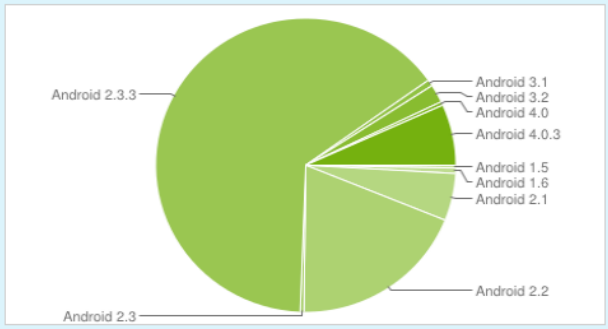
2.4.2.1 Υποστήριξη παλαιότερων εκδόσεων του Android

Όπως γράψαμε παραπάνω αλλά και στο εισαγωγικό κομμάτι της εργασίας, αυτή τη στιγμή υπάρχουν 9 διαθέσιμες εκδόσεις του Android με πιο πρόσφατη την έκδοση 4.0.3 με κωδική ονομασία “Ice Cream Sandwich” ή “ICS” και κυρίαρχη έκδοση την 2.3.3 “Gingerbread”.

Αυτή η συνεχής εξέλιξη της πλατφόρμας αποτελεί πλεονέκτημα αλλά και πρόκληση για τον προγραμματιστή ο οποίος θα πρέπει να ακολουθεί τις εξελίξεις και να χρησιμοποιεί τις νέες δυνατότητες που του προσφέρει η κάθε έκδοση, χωρίς να παραγκωνίζει την υποστήριξη στις παλαιότερες εκδόσεις του Android. Αυτό είναι ένα σημαντικό πρόβλημα καθότι κάποια νέα features δεν υποστηρίζονται στα παλιότερα APIs και άρα καθιστούν αδύνατη τη χρήση τους σε κάποια παλαιότερη έκδοση του Android. Η Google έχει προσπαθήσει να λύσει αυτό το πρόβλημα βγάζοντας μαζί με κάθε νέα έκδοση του λειτουργικού της, και μιας “βιβλιοθήκης συμβατότητας” η οποία αναλαμβάνει να κάνει διαθέσιμα τα νέα εργαλεία στα παλιότερα APIs.

Σε γενικές γραμμές η συγκεκριμένη λύση λειτουργεί αρκετά ικανοποιητικά αλλά δυστυχώς δεν μεταφέρονται πάντα όλες οι νέες δυνατότητες στις παλαιότερες εκδόσεις του λειτουργικού συστήματος με αυτό τον τρόπο.

Τρανό παράδειγμα αυτού είναι η υποστήριξη της ActionBar η οποία προστέθηκε στην έκδοση 3.0 του Android και πλέον ανήκει στις βασικές οδηγίες σχεδιασμού εφαρμογών (android design guidelines). Η βιβλιοθήκη υποστήριξης την κάνει διαθέσιμη στις παλαιότερες εκδόσεις, αλλά με σαφής περιορισμούς στην χρήση της! Ευτυχώς το συγκεκριμένο feature το υποστηρίζουν άψογα 2 βιβλιοθήκες που έχουν αναπτυχθεί από προγραμματιστές του Android, και έχει καλυφτεί το κενό που άφησε η Google.



Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.3%
Android 1.6	Donut	4	0.6%
Android 2.1	Eclair	7	5.2%
Android 2.2	Froyo	8	19.1%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.4%
Android 2.3.3 - Android 2.3.7		10	64.6%
Android 3.1	Honeycomb	12	0.7%
Android 3.2		13	2%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0.4%
Android 4.0.3 - Android 4.0.4		15	6.7%

Data collected during a 14-day period ending on June 1, 2012

Εικόνα 2.7: Στατιστικά στοιχεία ενεργών συσκευών Android – Ιούνιος 2012

Όπως βλέπουμε από τα παραπάνω στατιστικά στοιχεία που ενημερώνονται αυτόματα κάθε 2 εβδομάδες από την Google, τη μερίδα του λέοντος κατέχουν οι εκδόσεις 2.3 με 65% και η 2.2 με ποσοστό 19,1%. Οι νεότερη έκδοση του Android, 4.0 κατέχει μόλις το 7,1% παρότι κυκλοφόρησε τον Οκτώβριο του 2011, δηλαδή 7 μήνες από τη στιγμή που γράφονται αυτές οι γραμμές. Αυτό δυστυχώς πρόκειται για το φαινόμενο διάσπασης (fragmentation) του Android το οποίο θα αναλυθεί εκτενέστερα παρακάτω.

Γενικά μια καλή προγραμματιστική συμπεριφορά σύμφωνα με την κοινότητα, είναι η εκάστοτε εφαρμογή μας να υποστηρίζει τουλάχιστον το 90% των ενεργών συσκευών όσον αφορά την έκδοση Android που φοράνε. Αυτό από ότι βλέπουμε από το παραπάνω γράφημα είναι αρκετά εύκολο καθώς υποστηρίζοντας μόνο τις εκδόσεις 2.2 και 2.3 αυτή τη στιγμή, έχουμε υποστηρίξει το ~85% των ενεργών συσκευών! Παρόλα αυτά έχουμε όμως αποκλείσει έναν σημαντικό αριθμό χρηστών οι οποίοι χρησιμοποιούν παλαιότερες (1.6, 2.1) αλλά και νεότερες (4.0) συσκευές.

Οι developers πρέπει να βλέπουν και να αξιολογούν το μερίδιο των συσκευών με τέτοιο τρόπο ώστε να υποστηρίζουν όσο το δυνατό περισσότερες συσκευές χρηστών χωρίς όμως να υποβαθμίζουν την ποιότητα και λειτουργικότητα της συσκευής τους. Πρόκειται για μια λεπτή ισορροπία που επιτυγχάνεται μετά από αρκετή προσπάθεια από μέρος των developers.

Η υποστήριξη των διαφορετικών εκδόσεων ορίζεται στο αρχείο AndroidManifest.xml και εφόσον έχει καθοριστεί ένα κατώτατο στοχευμένο API, η εφαρμογή μας δεν μπορεί να

εγκατασταθεί σε συσκευή που φοράει παλαιότερη έκδοση από αυτή που υποστηρίζει το API.

Καλή πρακτική σχεδιασμού λοιπόν είναι να στοχεύσουμε ένα αρκετά χαμηλό API το οποίο όμως δεν θα μας αναγκάσει να κάνουνε συμβιβασμούς στις λειτουργίες της εφαρμογής, και στο τέλος να κάνουμε compile την τελική έκδοση με τη νεότερη έκδοση του SDK ώστε να εξασφαλίσουμε υποστήριξη και στις νεότερες συσκευές.

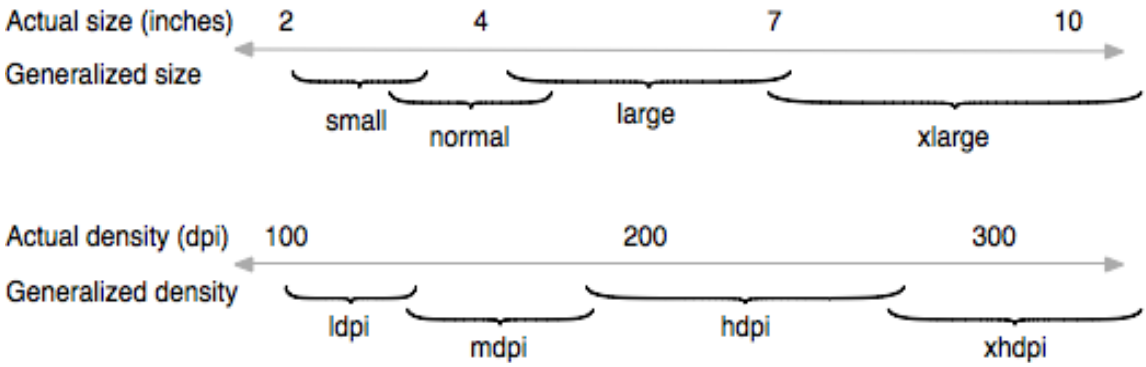
2.4.2.2 Υποστήριξη πολλαπλών διαστάσεων οθόνης και πυκνότητας pixel

Η ταυτόχρονη υποστήριξη των πολλών εκδόσεων του Android είναι η πρώτη πρόκληση του προγραμματιστή του. Η δεύτερη μεγάλη πρόκληση είναι η ταυτόχρονη υποστήριξη των πολλαπλών διαστάσεων οθόνης που διαθέτουν οι εκατοντάδες συσκευές που κυκλοφορούν στην αγορά, και η διαφορετική πυκνότητα pixel που διαθέτει η κάθε μία από αυτές.

Όπως γράψαμε και παραπάνω το γραφικό περιβάλλον μιας εφαρμογής είναι ίσως σημαντικότερο και από τις δυνατότητες που παρέχει, καθότι ένα κακοσχεδιασμένο layout μπορεί να κάνει την εφαρμογή δύσχρηστη ή ακόμη και άχρηστη! Άρα είναι πολύ σημαντικό για τον προγραμματιστή να λάβει υπόψιν του την πληθώρα αναλύσεων και διαφορετικών διαστάσεων οθόνης που διαθέτουν οι συσκευές.

Όσον αφορά το εύρος των συσκευών μπορεί να φτάσει από τις 2.4' (μικρά chatphones) έως τις 13' (μεγάλα tablets), και οι ανάλυση αυτών των συσκευών ξεκινάει από τα 240x320 pixels (QVGA) και φτάνει μέχρι τα 1920x1080 (Full HD) για τα νεότερα tablets! Δημοφιλείς αναλύσεις είναι η η 240x320(QVGA), η 240x400 (WQVGA), η 320x480 (HVGA), και φυσικά η πιο δημοφιλής όλων η 480x800 (WVGA). Νεότερες και μεγαλύτερης οθόνης συσκευής υποστηρίζουν και μεγαλύτερες αναλύσεις όπως η 540x960 (qHD) και 720x1280(WXGA).

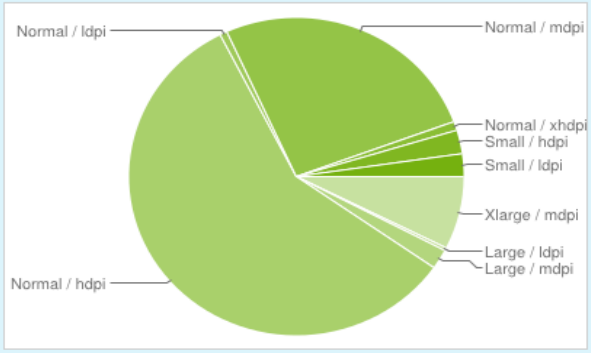
Όπως βλέπουμε υπάρχει μια πληθώρα αναλύσεων για να υποστηρίξει ο προγραμματιστής η κάθε μία με περισσότερο η λιγότερο διαθέσιμο χώρο στην οθόνη. Το android για πρακτικούς λόγους έχει χωρίσει τις διαφορετικές αναλύσεις οθονών σε τέσσερις κατηγορίες οθονών οι οποίες συσχετίζονται άμεσα με τέσσερις κατηγορίες πυκνότητες pixel ανά ίντσα.



Εικόνα 2.8: Διαχωρισμός Αναλύσεων και Πυκνότητας σε Κατηγορίες.

Όπως βλέπουμε από το παραπάνω σχεδιάγραμμα, το Android χωρίζει το μέγεθος τις οθόνης σε 4 επιμέρους κατηγορίες αναλόγως το μέγεθος της σε ίντσες, και οι αναλύσεις χωρίζονται επίσης σε 4 επιμέρους κατηγορίες DPI (Dots Per Inch). Αυτό γίνεται ώστε να διευκολύνει όσο το δυνατόν περισσότερο τους προγραμματιστές, να βελτιώσουν την εμφάνιση των εφαρμογών τους με όσο το δυνατόν λιγότερο κόπο.

Όπως και στην περίπτωση των πολλών εκδόσεων του Android που έχουν πρόσβαση στο μάρκετ, η Google παρέχει στους developers την κατανομή μεγέθους οθόνης προς DPI για να βοηθήσει τους προγραμματιστές να σχεδιάσουν τις εφαρμογές τους αποδοτικότερα.



	ldpi	mdpi	hdpi	xhdpi
small	2.3%		2.4%	
normal	0.7%	26.2%	57.8%	0.9%
large	0.3%	2%		
xlarge		7.4%		

Data collected during a 7-day period ending on May 1, 2012

Από ότι βλέπουμε από το παραπάνω σχεδιάγραμμα, τα πράγματα εν τέλη δεν είναι και τόσο τραγικά για τον developer! Όπως και στις εκδόσεις των API 2 μεγέθη κατέχουν τη μερίδα του λέοντος, και τα 2 μάλιστα στις ίδιες αναλύσεις (normal) αλλά σε διαφορετικό

μέγεθος DPI. Σε γενικές γραμμές οι πιο συνηθισμένες διατάξεις που πρέπει να δοκιμάσει ο developer την εφαρμογή του συνοψίζονται στον παρακάτω πίνακα.

	ldpi	**mdpi**	**hdpi**
Small	240x320 pixels 120 dpi		240x400 pixels 240 dpi
Normal		480x800 pixels 160 dpi	480x800 pixels 240 dpi

Πίνακας 2.1: Συνηθισμένες Αναλύσεις/Κατανομή pixels

Φυσικά δεν πρέπει να αγνοηθούν οι υπόλοιπες διατάξεις μεγέθους οθόνης προς dpi, αλλά όπως και στην περίπτωση της υποστήριξης των API, αυτό είναι μια λεπτή ισορροπία η οποία είναι στο χέρι του προγραμματιστή να την κρατήσει. Η πλατφόρμα του Android φυσικά εκτός από το να μας δημιουργεί πολύπλοκες καταστάσεις προς διαχείριση, μας δίνει επιλογές και δυνατότητες ώστε να αντεπεξέλθουμε στο απαιτητικό έργο της ανάπτυξης εφαρμογής στο περιβάλλον του. Οι δυνατότητες συνοψίζονται ως εξής:

- Ο developer μπορεί να επιλέξει ποιες οθόνες θα υποστηρίζει η εφαρμογή του με τον ίδιο τρόπο που επιλέγει πια API θα υποστηρίζονται, δηλαδή μέσω του AndroidManifest.xml. Αυτό σημαίνει ότι η υποστήριξη διαφορετικών οθονών είναι καθαρά επιλογή του προγραμματιστή, ο οποίος αν κρίνει αναγκαίο μπορεί να αποκλείσει την λειτουργία της εφαρμογής στις οθόνες που δεν επιθυμεί να υποστηρίξει.
- Στην συνηθέστερη περίπτωση που ο προγραμματιστής θέλει να υποστηρίξει επιπλέον αναλύσεις και διαστάσεις οθόνης πέρα από τις συνηθισμένες, το SDK του δίνει τη δυνατότητα παρέχοντας του δύο εξαιρετικές δυνατότητες:
    - Η πρώτη είναι η δυνατότητα χρήσης διαφορετικών layout ανά διαφορετικό μέγεθος οθόνης. Αυτό σημαίνει ότι ο developer δεν χρειάζεται να συμβιβαστεί σε ένα layout xml ώστε να καλύψει όλες τις οθόνες. Μπορεί να χρησιμοποιήσει όσα θέλει, για να καλυφθούν όσο το δυνατόν περισσότερα μεγέθη και αναλύσεις οθόνης. Δεν πρέπει να ξεχνάμε άλλωστε ότι οι μεγαλύτερες οθόνες παρέχουν και περισσότερο χώρο στον developer για να προβάλει επιπλέον υλικό, με διαφορετικό ίσως τρόπο από ότι θα το προέβαλε σε μια μικρότερη οθόνη

- Η δεύτερη δυνατότητα που παρέχει είναι αυτή της χρήσης πολλών γραφικών, διαφορετικών διαστάσεων, ώστε να εξυπηρετούνται σωστά όλες οι αναλύσεις οθόνης. Η υλοποίηση αυτής της δυνατότητας είναι εξαιρετικά απλή. Ο developer δημιουργεί φακέλους στο project του, με το όνομα της διάταξης που θέλει να παρέχει γραφικά η layout (πχ drawable – large – hdpi) ή χρησιμοποιεί τους υπάρχοντες φακέλους, και αποθηκεύει στον καθένα το ίδιο γραφικό (*.png, *jpg, ή *.gif) αλλά στην ανάλυση που επιθυμεί να προβληθεί αυτό στην εκάστοτε διαφορετική διάταξη μεγέθους οθόνης προς DPI.

2.5 Δοκιμή και Αποσφαλμάτωση (Debugging) της Εφαρμογής.

Πριν να εκδώσει την εφαρμογή του στο κοινό, ο προγραμματιστής του Android, αλλά και οποιασδήποτε άλλης πλατφόρμας, οφείλει να δοκιμάσει ενδελεχώς, κάθε λειτουργική και αισθητική παράμετρο της εφαρμογής ώστε να εξασφαλίσει ότι ο χρήστης της θα έχει όσο το δυνατόν καλύτερη εμπειρία χρήσης! Καλύτερη εμπειρία χρήσης φυσικά οδηγεί σε καλές εντυπώσεις, και περισσότερους χρήστες με την πάροδο του χρόνου. Μια κακή εμπειρία χρήσης, οδηγεί σε δυσαρεστημένους χρήστες και άρα αποτυχία.

Ο αυτοσκοπός της ανάπτυξης εφαρμογών είναι η ικανοποίηση των αναγκών όσο το δυνατόν μεγαλύτερου αριθμού χρηστών! Αυτό φυσικά δεν επιτυγχάνεται αμέσως. Πρέπει να ακολουθηθεί μια εκτενής περίοδος δοκιμών για να διαπιστωθεί ότι η εφαρμογή θα αποδίδει όπως οφείλει σε όλες τις καθημερινές της χρήσεις, και αφού περάσει επιτυχώς το στάδιο των δοκιμών, ακολουθεί η ώρα της δημοσίευσης.

Πως γίνεται λοιπόν η διαδικασία της αποσφαλμάτωσης; Μιας εφαρμογή μπορεί να αποτελείτε από μερικές απλές κλάσεις των 50 γραμμών κώδικα, ή μπορεί να αποτελείτε από δεκάδες κλάσεις χωρισμένες σε πακέτα, αποτελούμενες από δεκάδες χιλιάδες γραμμές κώδικα η καθεμία! Η έρευνα για σφάλματά στην δεύτερη περίπτωση, έχει πολύ χειρότερη αναλογία πιθανοτήτων, από την έρευνα για “ψύλλους στα άχυρα”.

Η ανάγκη για εργαλεία ελέγχου, καλύπτεται στο έπακρο με μια σειρά εφαρμογών τα οποία δένοντας αρμονικά με το περιβάλλον ανάπτυξης μας (IDE) μας λύνουν τα χέρια, στο δύσκολο έργο του εντοπισμού σφαλμάτων. Τα εργαλεία αυτά αφορούν τον έλεγχο της εφαρμογής σε εικονικές μηχανές (Android Virtual Devices), καταγραφή σφαλμάτων (logcat), εργαλεία ελέγχου μνήμης και άλλων λειτουργιών της συσκευής (DDMS), κλπ.

2.5.1 Android Debug Bridge (ADB)

Για να λειτουργήσουν σωστά τα εργαλεία που αναφέραμε παραπάνω, χρειάζεται κάποιο είδος προγράμματος client-server που να συνδέει τον υπολογιστή με τις συσκευές μας, εικονικές και μη. Τον ρόλο αυτό αναλαμβάνει το Android Debug Bridge (ADB). Πρόκειται για ένα εργαλείο γραμμής εντολών που έρχεται μαζί με το Android SDK και το οποίο αποτελείτε από 3 μέρη:

- Έναν client ο οποίος τρέχει στον υπολογιστή που έχουμε στήσει το SDK. Μπορούμε είτε να τον εκκινήσουμε χειροκίνητα είτε να χρησιμοποιήσουμε κάποιο εργαλείο το οποίο ξεκινάει αυτόματα δικό του client, όπως το DDMS ή το ADT Plugin.
- Έναν server ο οποίος τρέχει σαν υπηρεσία παρασκηνίου στον υπολογιστή που βρίσκεται το SDK, όπως και ο client. Ο server εξασφαλίζει την επικοινωνία μεταξύ του client και του εργαλείου "δαίμονα" (daemon) που τρέχει στη συσκευή.
- Ο "δαίμονας" (daemon) που τρέχει σαν διεργασία παρασκηνίου στην εικονική η πραγματική συσκευή που χρησιμοποιείτε για εξομοίωση.

Όταν ξεκινάει το ADB, ο client ελέγχει αν υπάρχει κάποια υπάρχουσα διεργασία του server που να εκτελείτε ήδη, αλλιώς δημιουργεί μια νέα. Μετά δημιουργεί μια τοπική TCP σύνδεση στην θύρα 5037 και είναι έτοιμος να δεχτεί εντολές. Μετά ελέγχει το εύρος θυρών TCP από 5555 μέχρι 5585, στο οποίο επικοινωνούν οι συσκευές εξομοίωσης, και ελέγχει αν υπάρχουν διαθέσιμες και πόσες είναι αυτές. Αφού εντοπίσει κάποια συσκευή ελέγχει αν σε αυτή τη συσκευή τρέχει ο "δαίμονας", και αν ναι δημιουργείτε σύνδεση adb με την συσκευή.

Αφότου πραγματοποιηθεί σύνδεση μεταξύ συσκευής και client μπορούμε να χρησιμοποιήσουμε όλες τις δυνατότητες που μας παρέχει το ADB για να ασκήσουμε πλήρη έλεγχο της συσκευής μας. Οι εντολές μέσω γραμμής εντολών δίνονται ως εξής:

adb [-d|-e|-s <serialNumber>] <command>

όπου -d είναι η απευθείας εντολή εάν υπάρχει μόνο μία συνδεδεμένη συσκευή, -e η απευθείας εντολή εάν υπάρχει μόνο μία συνδεδεμένη **εικονική** συσκευή, -s <serialNumber> η απευθείας εντολή στην συσκευή που έχει το <serialNumber>, και <command> είναι η εντολή που θέλουμε να εκτελεστεί. Παράδειγμα χρήσης:

adb -s emulator-5556 install helloWorld.apk

Στο παραπάνω παράδειγμα γίνεται εγκατάστασης της εφαρμογής με όνομα "helloWorld.apk" στην συσκευή εξομοίωσης που ακούει στη θύρα 5556.

2.5.2 Εικονικές Συσκευές Android (Android Virtual Devices – AVD)

Όπως γράψαμε και παραπάνω, ο developer πρέπει πριν να εκδώσει την εφαρμογή του να την δοκιμάσει σε ένα αριθμό συσκευών για να εξασφαλίσει την ομαλή λειτουργία της σε όλες τις συνθήκες. Φυσικά το κόστος των συσκευών είναι αρκετά μεγάλο για να αποθαρρύνει τον προγραμματιστή να έχει στην κατοχή του 10-20 συσκευές για να ελέγξει σε όλες τις λειτουργίες και την εμφάνιση της εφαρμογής του. Το πρόβλημα αυτό έρχεται να λύσει η ύπαρξη των εικονικών συσκευών του Android.

Πρόκειται για μια συσκευή εξομοίωσης η οποία μας επιτρέπει να εξομοιώσουμε την λειτουργία και συμπεριφορά μιας κανονικής συσκευής, ορίζοντας τις επιλογές υλικού και λογισμικού που θέλουμε στον εξομοιωτή του Android. Με αυτό τον τρόπο ο developer μπορεί να ελέγξει την εφαρμογή του σε μια σειρά από πραγματικά σενάρια λειτουργίας και να πάρει γρήγορα και άμεσα feedback για τη λειτουργία της εφαρμογής του. Μια εικονική συσκευή αποτελείτε από:

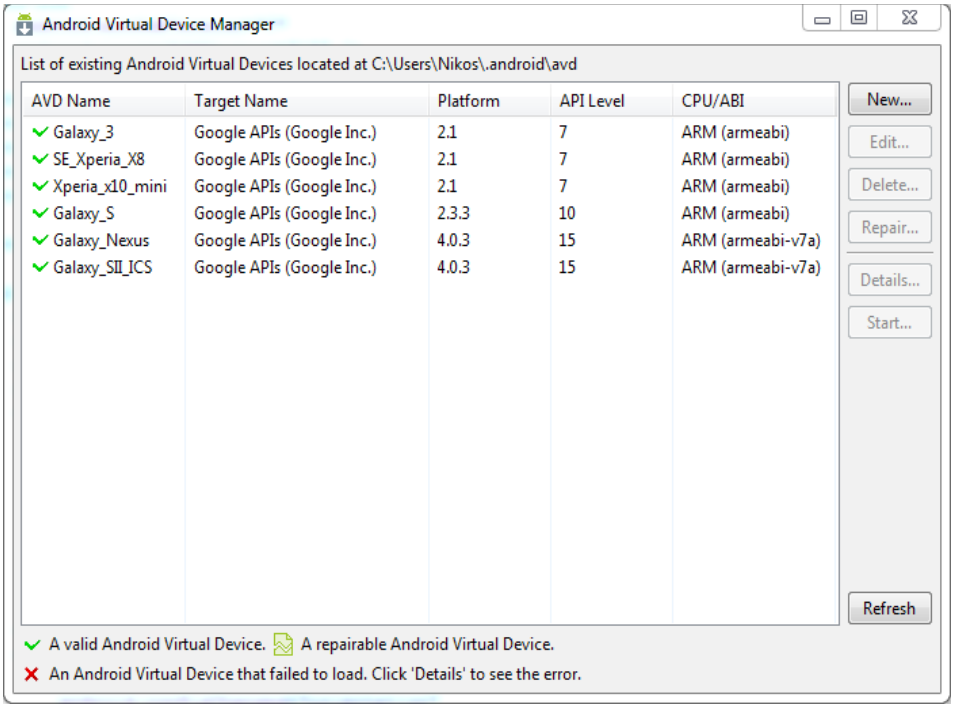
- **Το προφίλ του υλικού:** Σε αυτό προσδιορίζονται οι ιδιότητες και τα χαρακτηριστικά της εικονικής συσκευής. Μπορούμε παραδείγματος χάρη να ορίσουμε την ανάλυση της οθόνης και την πυκνότητα σε pixel (dpi), το μέγεθος της μνήμης RAM, αν η συσκευή θα έχει κάμερα, υποστήριξη GPS, κλπ.
- **Την έκδοση του Android:** Επιλογή της έκδοσης της πλατφόρμας του Android που θέλουμε να εξομοιώσει η εικονική συσκευή. Μπορούμε επίσης να επιλέξουμε και μεταξύ ειδικών εκδόσεων της πλατφόρμας, μεταξύ των οποίων τις Google TV, και άλλων.
- **Έξτρα χώρος αποθήκευσης:** Εδώ αποθηκεύονται όλα τα δεδομένα της εφαρμογής, και επίσης μπορούμε να ορίσουμε μια εικονική κάρτα μνήμης ώστε να επεκτείνουμε τον αποθηκευτικό χώρο, όπως θα κάναμε και σε μια πραγματική συσκευή.

2.5.2.1 Δημιουργία διαφορετικών εικονικών συσκευών

Η δημιουργία εικονικών μηχανών είναι μια ιδιαίτερα εύκολη και γρήγορη διαδικασία. Η διαχείριση αυτών των συσκευών γίνεται από το γραφικό περιβάλλον της εφαρμογής AVD Manager, η οποία έρχεται μαζί με το SDK και στην περίπτωση που χρησιμοποιούμε το Eclipse μαζί με το ADT plugin, αυτή ενσωματώνεται στο γραφικό περιβάλλον του Eclipse.

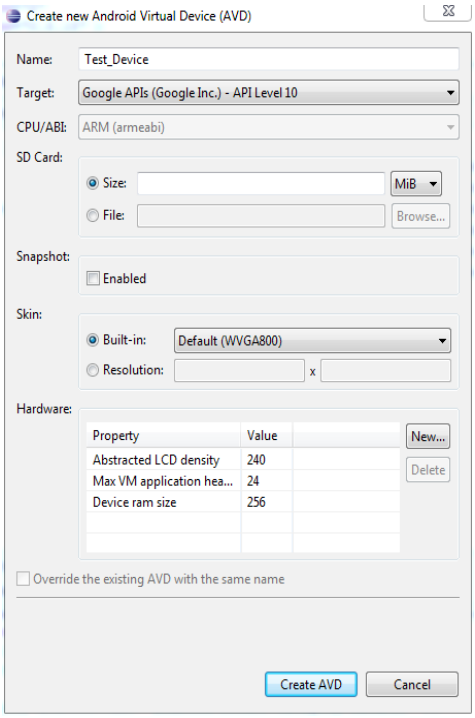
Το γραφικό περιβάλλον του AVD Manager είναι πολύ λιτό. Αποτελείτε από μια λίστα με τις εικονικές συσκευές που έχουμε δημιουργήσει και στη δεξιά πλευρά υπάρχουν 7 κουμπιά διαχείρισης των συσκευών μας.

Παρακάτω (Εικόνα 2.10) εμφανίζεται η λίστα με τις εικονικές συσκευές που δοκιμάστηκε η εφαρμογή **CST Connect**, κατά τη διάρκεια της δοκιμαστικής της φάσης. Φυσικά το προφίλ υλικού δεν είναι ακριβώς αυτό που διαθέτουν οι συσκευές που κατονομάζονται στην λίστα, αλλά οι διαστάσεις οθόνης και η έκδοση API, είναι ακριβής.



Εικόνα 2.10: AVD Manager

Πατώντας το κουμπί **New** εμφανίζεται ένα νέο παράθυρο δημιουργίας εικονικής συσκευής (Εικόνα 2.11), στο οποίο μπορούμε να ορίσουμε το όνομα, την έκδοση του Android που θέλουμε να τρέξει η συσκευή, την ανάλυση οθόνης, και τα υπόλοιπα χαρακτηριστικά του hardware που επιθυμούμε. Αφού πατήσουμε το κουμπί **create AVD**, η συσκευή μας είναι έτοιμη προς χρήση και προστίθεται στη λίστα μαζί με τις υπόλοιπες εικονικές συσκευές.



Εικόνα 2.11 - Παράθυρο δημιουργίας νέας Εικονικής Μηχανής

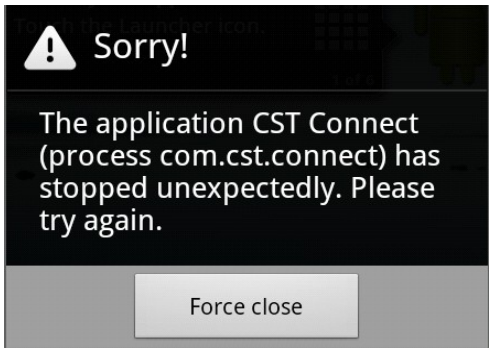
2.5.3 Εργαλείο καταγραφής συμβάντων – LogCat

Το Android διαθέτει ένα μηχανισμό καταγραφής συμβάντων, σκοπός του οποίου είναι η συλλογή και προβολή των αρχείων αποσφαλμάτωσης του συστήματος. Τα δεδομένα των διάφορων εφαρμογών αλλά και του λειτουργικού συστήματος συγκεντρώνονται σε μια σειρά από buffers, τους οποίους μετά μπορούμε να προβάλουμε και να φιλτράρουμε με την εντολή "logcat".

Στον προγραμματισμό υπάρχουν οι λεγόμενες "εξαιρέσεις" (exceptions), καταστάσεις δηλαδή που προκύπτουν όταν κάτι δεν πάει καλά, και αυτό έχει σαν αποτέλεσμα την διακοπή λειτουργίας του προγράμματος σε περίπτωση που δεν έχουμε φροντίσει να "χειριστούμε" την εξαίρεση. Συνηθισμένο παράδειγμα εξαίρεσης λειτουργίας του Android είναι η "NullPointerException", η οποία μας εμφανίζεται όταν προσπαθούμε να προσπελάσουμε κάποια μεταβλητή η αντικείμενο που έχει μηδενική (Null) τιμή. Προγραμματιστικά βέβαια υπάρχει η δυνατότητα να βάλουμε δικλείδα ασφαλείας σε μερικά επίφοβα σημεία του κώδικα

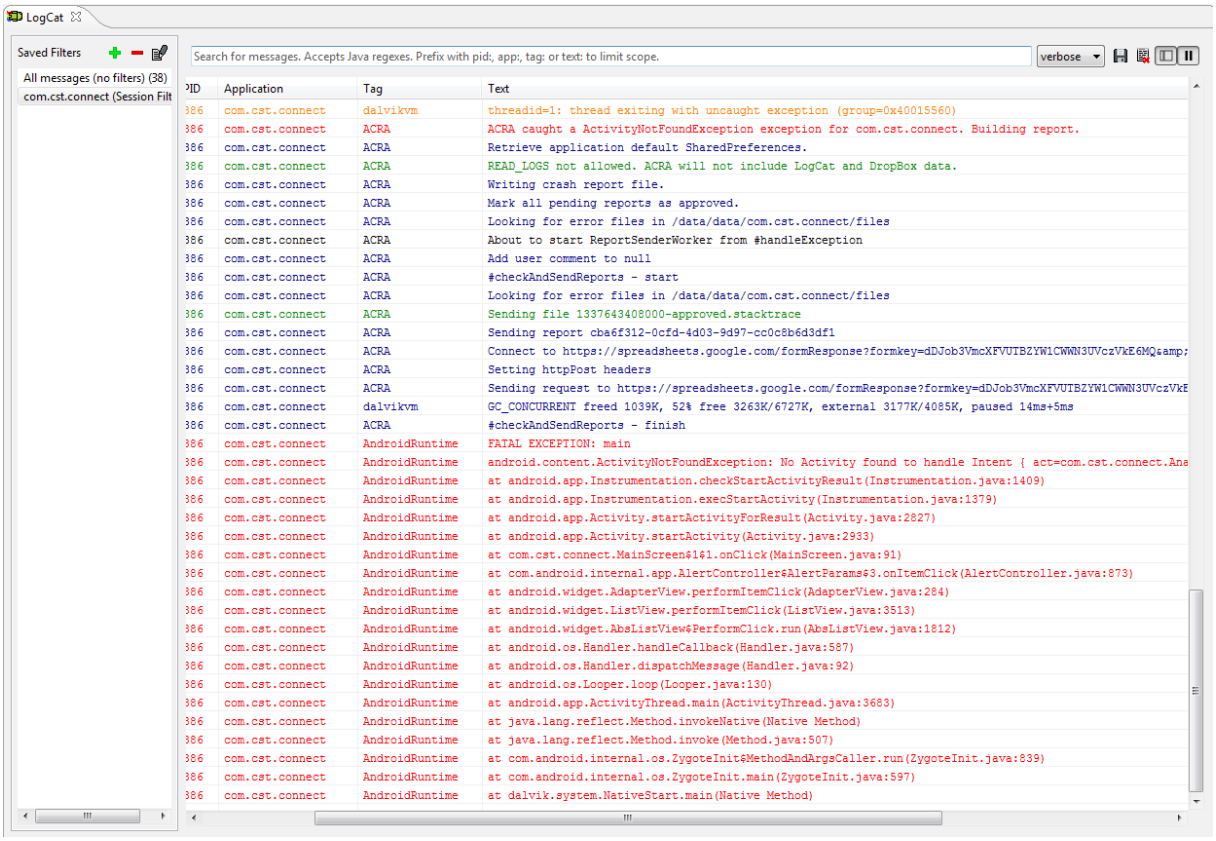
μας, και να σταματήσουμε την απότομη διακοπή λειτουργίας της εφαρμογής, εμφανίζοντας έναντι μόνο το μήνυμα σφάλματος στα logs του συστήματος.

Τα σφάλματα λειτουργίας μιας εφαρμογής στο Android, που προκύπτουν από εξαιρέσεις λειτουργίας, συνήθως προκαλούν τον άμεσο τερματισμό της εφαρμογής προβάλλοντας ένα παράθυρο με το όνομα της εφαρμογής που τερματίστηκε, και ένα απλό μήνυμα σφάλματος (Εικόνα 2.12), δίνοντας μας την "επιλογή" να πατήσουμε "Force Close".



Εικόνα 2.12: Παράθυρο Force Close

Το LogCat λοιπόν είναι μια εντολή γραμμής εντολών η οποία μπορεί να χρησιμοποιηθεί μέσω του ADB για να δούμε τα debug logs της συσκευής που δουλεύουμε, και άρα ως συνεπακόλουθο, της εφαρμογής που αναπτύσσουμε ώστε να εντοπίσουμε τις πηγές των σφαλμάτων, οι οποίες συνήθως αν όχι πάντα, είναι exceptions στον κώδικα μας. Το eclipse ενσωματώνει μία GUI έκδοση του LogCat (Εικόνα 2.13) για αποτελεσματικότερη αποσφαλμάτωση του κώδικα μας.



Εικόνα 2.13: Η γραφική απεικόνιση του Logcat όπως εμφανίζεται στο Eclipse IDE

Στην παραπάνω εικόνα βλέπουμε την τυπική διάταξη του logcat τη στιγμή μάλιστα που έχει συμβεί μια εξαίρεση λειτουργίας στην εφαρμογή μας και η οποία εμφανίζεται με κόκκινα χαρακτηριστικά γράμματα ξεκινώντας με τη φράση “Fatal Exception”. Οι κόκκινες γραμμές που προβάλλονται είναι όλες εξαιρετικά σημαντικές, κυρίως όμως μας ενδιαφέρουν αυτές που κάνουν λόγο για τη φύση της εξαίρεσης, και για τη γραμμή του κώδικα αλλά και την κλάση στην οποία συνέβη το σφάλμα! Στην συγκεκριμένη περίπτωση είχαμε μια “ActivityNotFoundException” η οποία συνέβη στην κλάση “MainScreen” στη γραμμή 91.

Με το παραπάνω απλό παράδειγμα μπορούμε να αντιληφθούμε την χρησιμότητα ύπαρξης του logcat. Άμεσα μας έκανε γνωστό το ακριβές σημείο του κώδικα μας που προκάλεσε το σφάλμα και μάλιστα λόγω τις φύσης της εξαίρεσης, μπορούμε να το διορθώσουμε σε χρόνο μηδέν. Η συγκεκριμένη εξαίρεση προκύπτει όταν μέσω ενός Intent κάνουμε μετάβαση από την μία Activity σε μία άλλη, και είτε συνήθως λόγω ορθογραφικού σφάλματος είτε πχ λόγω παράλειψης ενσωμάτωσης της Activity στο αρχείο AndroidManifest, η Activity που ορίζουμε στο Intent, δεν είναι διαθέσιμη.

Βέβαια υπάρχουν πολλές άλλες εξαιρέσεις (πάνω από 200 διαφορετικές) οι οποίες προκύπτουν σε διάφορες ανύποπτες στιγμές και χάρη στο logcat μπορούμε αρχικά να τις εντοπίσουμε και μετά να τις αντιμετωπίσουμε κάνοντας τις απαραίτητες διορθώσεις/αλλαγές στον κώδικα μας.

Φυσικά υπάρχουν και τα σφάλματα στον κώδικα τα οποία δεν προκαλούν αναγκαστικό κλείσιμο της εφαρμογής, αλλά παρόλα αυτά συμβάλουν στην μη σωστή λειτουργία της. Αυτού του είδους τα σφάλματα φυσικά δεν τα πιάνει ο compiler, ούτε εμφανίζονται με την μορφή που εμφανίζονται τα σφάλματα που προκύπτουν από εξαιρέσεις. Το αποτέλεσμα συνήθως αυτών των σφαλμάτων είναι μια κενή λίστα, μια λάθος τοποθετημένη εικόνα, κάποιο λάθος κείμενο, κλπ.

Πως αντιμετωπίζουμε λοιπόν ένα σφάλμα το οποίο δεν εμφανίζει κάποιο stack trace με ακριβές σημείο κώδικα προς διόρθωση, όπως συμβαίνει στην περίπτωση των εξαιρέσεων; Για το το σκοπό μπορούμε να χρησιμοποιήσουμε το σύστημα καταγραφής του Android, για να πάρουμε τις πληροφορίες που θέλουμε. Η κλάση Log υπάρχει για αυτό ακριβώς το σκοπό. Περιλαμβάνει έναν αριθμό από διαφορετικές μεθόδους αναλόγως με τον τύπο του σφάλματος που ψάχνουμε. Η πιο συνηθισμένη μέθοδος είναι η Log.d(), όπου το d συμβολίζει τη λέξη Debug.

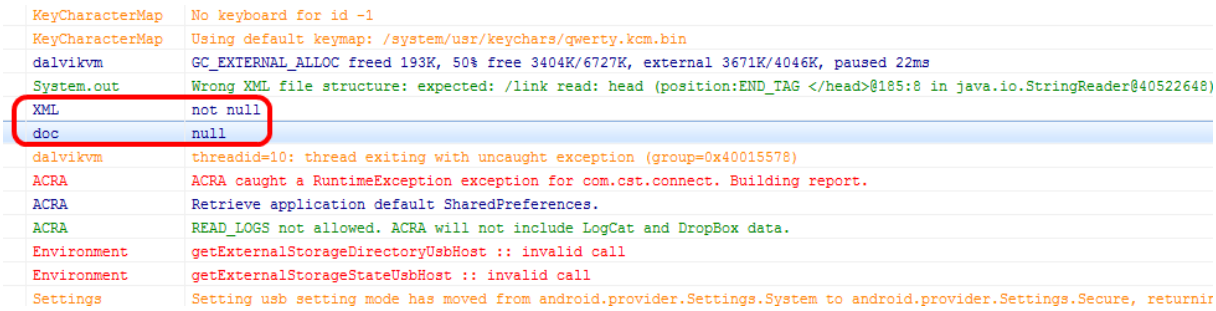
Χρησιμοποιώντας την μέθοδο Log.d(String, String) μπορούμε να εμφανίσουμε στα logs της εφαρμογής πολλές πληροφορίες αποσφαλμάτωσης. Στην περίπτωση πχ μιας κενής λίστας αντικειμένων, μπορούμε να τοποθετήσουμε την μέθοδο με παραμέτρους:

```
Log.d("Debug", text);
```

Όπου “Debug” είναι η λέξη που θα εμφανιστεί δίπλα από το επιθυμητό Log, και text είναι η μεταβλητή τύπου String την τιμή τις οποίας θέλουμε να εμφανίσουμε στο Logcat για να δούμε τι περιέχει. Μια άλλη χρήση της μεθόδου είναι να εντοπίσουμε αν μια μεταβλητή που δεν είναι String, έχει τιμή η είναι κενή και μας επιστρέφει Null. Ακολουθεί παράδειγμα:

```
if (XML==null){
      Log.d(“XML”, “null”);
}else{
    Log.d(“XML”, “not null”);
}if (doc==null){
    Log.d(“doc”, “null”);
}else{
    Log.d(“doc”, “not null”);}
```

Στις παραπάνω γραμμές κώδικα, μέσω μια απλής if else, ελέγχουμε αν οι μεταβλητες “XML” και “doc” είναι κενές και ο έλεγχος μας επιστρέφει “null” ή “not null”. Σε κάθε περίπτωση λέμε στην Log.d να μας εκτυπώσει στο Logcat: το αντίστοιχο αποτέλεσμα (Εικόνα 2.14).



Εικόνα 2.14: Χρησιμοποιώντας την μέθοδο Log.d()

Αυτό βέβαια δεν μας επιστρέφει τις πραγματικές τιμές που μπορεί να έχει η μεταβλητές που ελέγξαμε, σε περίπτωση που δεν είναι Null, αλλά έτσι μπορούμε να εντοπίσουμε την υπαίτια για κάποιο NullPointerException που μπορεί να παίρνουμε από την εφαρμογή μας.

2.5.4 Dalvik Debug Monitor Server (DDMS)

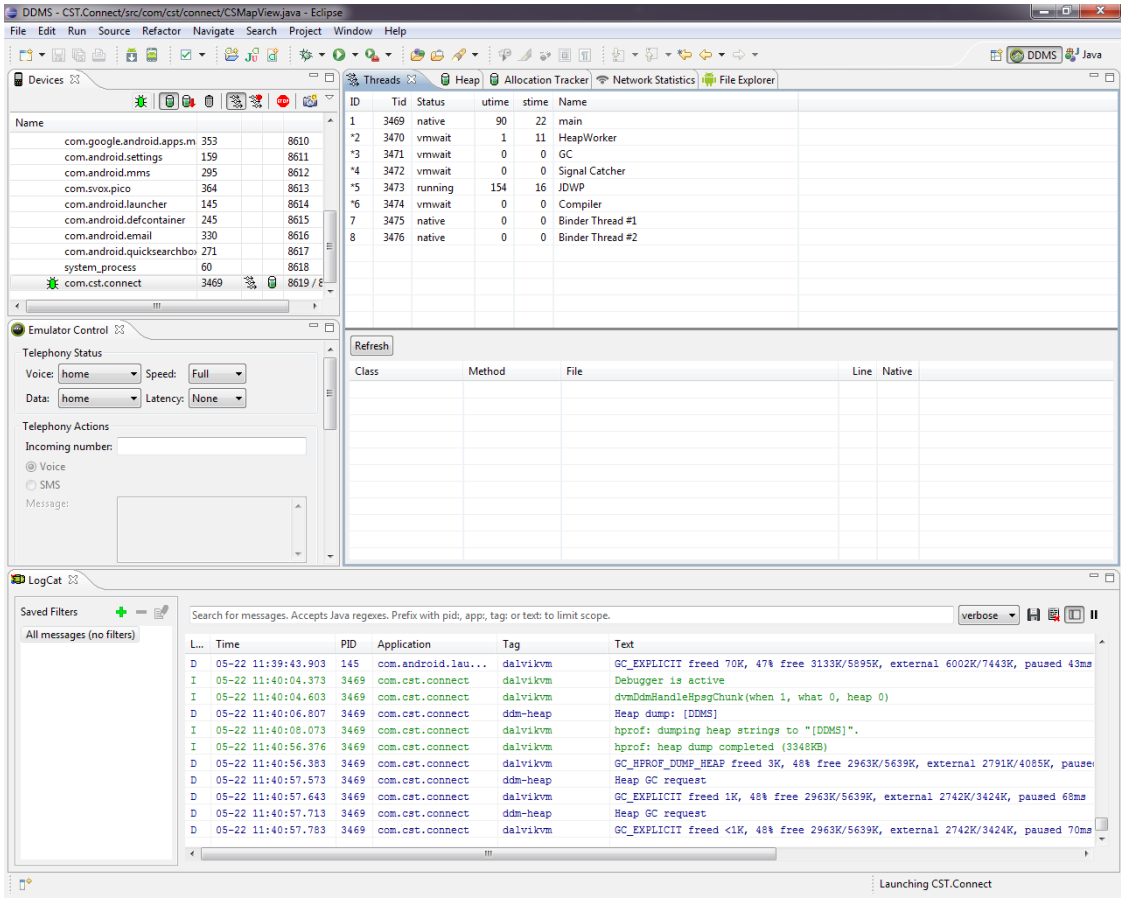
Το Android συνοδεύεται μεταξύ των άλλων, από ένα εργαλείο αποσφαλμάτωσης το οποίο ονομάζεται Dalvik Debug Monitor Server (DDMS), και το οποίο παρέχει:

- Υπηρεσίες προώθησης θυρών (port forwarding)
- Λήψη εικόνας τις επιφάνειας εργασίας τις συσκευής
- Πληροφορίες για τις διεργασίες και τα νήματα (threads) τις συσκευής
- Το εργαλείο logcat
- Πληροφορίες δικτύου και εισερχομένων κλήσεων
- Δημιουργία ψευδών SMS και πληροφοριών τοποθεσίας,
- και άλλα πολλά

Το DDMS είναι ενσωματωμένο στο Eclipse και επίσης συμπεριλαμβάνεται στο Android SDK. Δουλεύει κανονικά είτε είναι συνδεδεμένο σε εικονική συσκευή μέσω του emulator είτε είναι συνδεδεμένο σε κανονική φυσική συσκευή. Από προεπιλογή αν είναι συνδεδεμένες δύο συσκευές ταυτόχρονα και η μία είναι εικονική, τότε αυτό θα επιλέξει σαν προεπιλεγμένη την εικονική. Σε προηγούμενο κεφάλαιο μιλώντας για την ασφάλεια στο Android, αναφέραμε ότι κάθε εφαρμογή τρέχει στην δική της εικονική μηχανή (VM). Κάθε μηχανή έχει και μία μοναδική θύρα επικοινωνίας στην οποία μπορεί να συνδεθεί κάποιο εργαλείο αποσφαλμάτωσης.

Κατά την εκκίνηση του το DDMS συνδέεται στο ADB. Αφού συνδεθεί κάποια συσκευή στο υπολογιστή μέσω ADB, αυτόματα δημιουργείτε μια υπηρεσία παρακολούθησης μεταξύ του DDMS και του ADB η οποία ειδοποιεί το DDMS πότε ξεκινάει και πότε σταματάει η λειτουργία μιας εικονικής μηχανής. Όταν η VM βρίσκεται σε λειτουργία, το DDMS παίρνει το process ID της VM και μέσω του ADB δημιουργεί μια σύνδεση με τον debugger της εικονικής μηχανής, μέσω του adb daemon.

Το DDMS αναθέτει μια μοναδική θύρα επικοινωνίας σε κάθε εικονική μηχανή στην συσκευή η στις συσκευές τις οποίες είναι συνδεδεμένο. Η ανάθεση ξεκινάει από την θύρα 8600 για την πρώτη VM και συνεχίζει για όσες εικονικές μηχανές, τρέχουν ταυτόχρονα στην συσκευή μας. Ισχύει ότι η κάθε VM έχει μία θύρα επικοινωνίας για debugging, αλλά το DDMS μπορεί να ακούσει σε πολλές θύρες ταυτόχρονα ώστε να λάβει δεδομένα από παντού.



Εικόνα 2.15: Παράθυρο DDMS

Στην παραπάνω εικόνα (Εικόνα 2.15) βλέπουμε ένα τυπικό παράθυρο του DDMS όπως αυτό εμφανίζεται στο Eclipse. Αριστερά φαίνονται οι συνδεδεμένες συσκευές (μία εικονική μέσω emulator) και από κάτω υπάρχει η λίστα με τις διαθέσιμες εικονικές μηχανές που τρέχουν στην συσκευή. Έχοντας επιλέξει την VM της εφαρμογής com.cst.connect μας έχουν γίνει διαθέσιμα το logcat output στο κάτω μέρος της οθόνης και στο δεξιό κομμάτι τις οθόνης οι υπόλοιπες πληροφορίες που συλλέγει ο debugger από την εφαρμογή.

Το DDMS είναι ίσως το χρησιμότερο από όλα τα debugging tools μιας και ενσωματώνει από τα πιο απλά (πχ logcat) μέχρι εξειδικευμένα εργαλεία debugging του Android το network statistics, και απευθύνεται κυρίως σε πιο έμπειρους developers οι οποίοι μπορούν να αξιολογήσουν αποτελεσματικότερα τις ενδείξεις που επιστρέφει το εργαλείο.

2.5.5 Application Crash Reporter for Android (ACRA)

Η συλλογή των debugging logs λοιπόν, είναι μια πολύ απλή διαδικασία η οποία πραγματοποιείται πολύ γρήγορα εφόσον συνδέσουμε την συσκευή μας μέσω του Android Debug Bridge. Τι γίνεται όμως στην περίπτωση που θέλουμε να συλλέξουμε τα logs μιας συσκευής χρήστη η οποία εμφάνισε κάποιο σφάλμα; Όπως αντιλαμβανόμαστε δεν είναι εφικτό να του ζητήσουμε να συνδέσει τη συσκευή του μέσω ADB και αφού τα πάρει να μας τα στείλει με email! Και στην περίπτωση που τυχαίναμε σε περίπτωση συνεργάσιμου και γνώστη χρήστη, πόσο εύκολο θα ήταν να του ζητάγαμε να μας στέλνει συνεχώς το stack trace του logcat ώστε να εντοπίσουμε το σφάλμα και να το διορθώσουμε;

Οι μηχανικοί του Android έχουν μεριμνήσει για αυτό και γιαυτό έχουν ενσωματώσει στις εφαρμογές που είναι συνδεδεμένες με το marketplace (έκδοση Android 2.2 και πάνω) έναν μηχανισμό απομακρυσμένης συλλογής logs. Τα stack traces των logs γίνονται προσβάσιμα μέσω της developer console του developer στο Google Play, και μέσω αυτών ο προγραμματιστής μπορεί να εντοπίσει το σφάλμα στον κώδικα του και να φροντίσει να το διορθώσει το συντομότερο δυνατό.

Τι γίνεται όμως στην περίπτωση που κάποιος developer δεν έχει κάνει ακόμη διαθέσιμη την εφαρμογή του στο κοινό, μέσω του Google Play; Η πιο συνηθέστερα, πως γίνεται να συλλέξει ο developers τα logs των χρηστών στην beta testing φάση της εφαρμογής του; Όπως είπαμε παραπάνω είναι εξαιρετικά ενοχλητικό και μη βολικό από την πλευρά μας να ζητάμε συνεχώς από τον χρήστη να μας τα στέλνει με email, και δυστυχώς το Android δεν έχει μεριμνήσει για αυτή την περίπτωση.

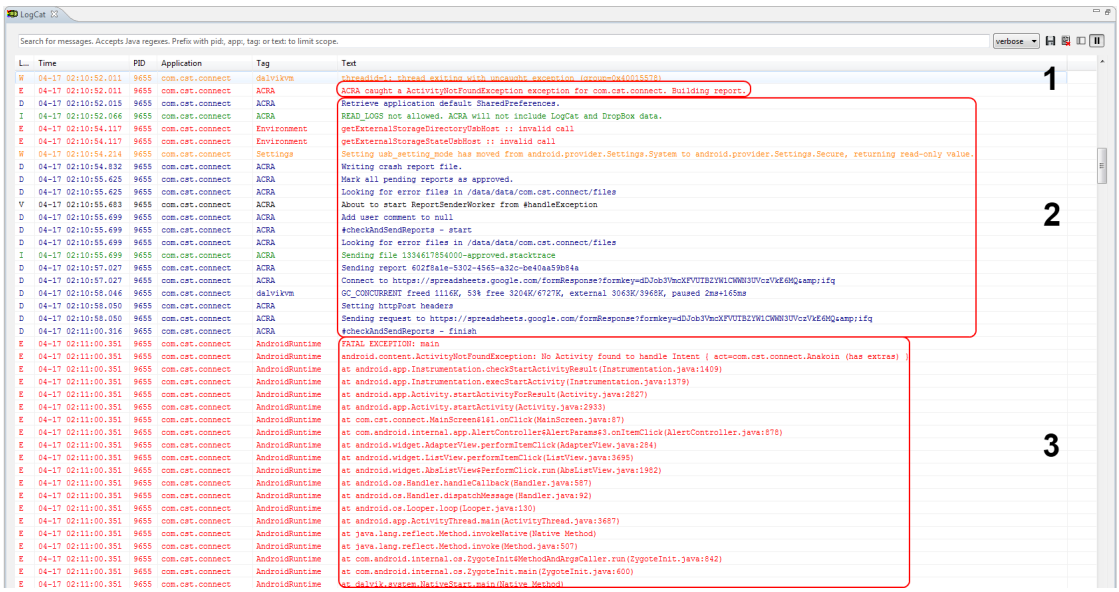
Το κενό λοιπόν αυτό έρχονται να καλύψουν κάποιες βιβλιοθήκες οι οποίες έχουν αναπτυχθεί με τα εργαλεία του Android SDK και σκοπός τους είναι ακριβώς αυτός: η συλλογή και αποστολή των stack traces από το κινητό του χρήστη σε εμάς για περεταίρω ανάλυση! Ένα παράδειγμα τέτοια βιβλιοθήκης η οποία χρησιμοποιείτε από πολλές δημοφιλείς εφαρμογές και χρησιμοποιήθηκε κατά κόρον και στο beta testing της εφαρμογής CST Connect, είναι η βιβλιοθήκη ACRA.

Η ACRA (αρχικά για “Application Crash Report for Android”) είναι μια βιβλιοθήκη του Android η οποία αυτόματα συλλέγει τα δεδομένα των αναφορών σφάλματος και τα μας τα αποστέλλει σε μορφή εγγράφου GoogleDoc. Η βιβλιοθήκη είναι open source (Apache License 2.0) και φιλοξενείτε στην σελίδα code.google.com/p/acra/.

Τα χαρακτηριστικά της βιβλιοθήκης είναι τα εξής:

- Δυνατότητα παραμετροποίησης ειδοποιήσεων στην συσκευή του χρήστη (silent report, ειδοποίηση toast, ειδοποίηση στην Notification Bar + παράθυρο διαλόγου.
- Είναι δυνατόν να χρησιμοποιηθεί σε όλες τις εκδόσεις του Android και όχι μόνο από την 2.2 και πάνω όπως η επίσημη βιβλιοθήκη!
- Λεπτομερείς πληροφορίες για την συσκευή στην οποία συνέβη το σφάλμα λειτουργίας, πολύ περισσότερες από αυτές που δίνει η επίσημη βιβλιοθήκη!
- Μπορούμε να προσθέσουμε και τις δικές μας μεταβλητές περιεχομένου στις αναφορές που στέλνει η βιβλιοθήκη.
- Είναι εφικτή η αποστολή αναφορών χωρίς να έχει συμβεί κάποιο σφάλμα λειτουργίας της εφαρμογής!
- Δουλεύει σε όλες τις εφαρμογές είτε αυτές έχουν κυκλοφορήσει μέσω του Google Play, είτε βρίσκονται σε φάση beta testing οπότε η βιβλιοθήκη της Google δεν είναι διαθέσιμη.
- Μπορούμε να επιλέξουμε εκτός από κάποιο έγγραφο GoogleDoc η ACRA να μας στέλνει απευθείας τις αναφορές σε κάποιον http server, είτε να μας τις στέλνει απευθείας μέσω email.
- Τα reports που έχουμε ρυθμίσει να αποθηκεύονται σε κάποιο έγγραφο GoogleDoc, μπορεί να γίνει προσβάσιμο άμεσα από μια ομάδα developers ώστε να αντιμετωπισθεί γρηγορότερα το σφάλμα.

Όπως βλέπουμε η ACRA έχει πολλές εξαιρετικές δυνατότητες και πραγματικά λύνει τα χέρια του developer όσον αφορά το κομμάτι του remote debugging. Η πιο συνηθισμένη χρήση της βιβλιοθήκης είναι τα silent reports ώστε ο χρήστης πέρα από το παράθυρο του Force Close (Εικόνα 2.12) δεν παρατηρεί κάτι άλλο. Μετά το silent report ενημερώνεται άμεσα το έγγραφο GoogleDoc που έχουμε δημιουργήσει και συνδέσει με τη βιβλιοθήκη και μετέπειτα μπορούμε να επιλέξουμε να ενημερωνόμαστε άμεσα με email κάθε φορά που γίνεται αλλαγή στο έγγραφό μας. Με λίγα λόγια ένα σύστημα άμεσου bug reporting! Φυσικά μπορεί να χρησιμοποιηθεί ταυτόχρονα με την επίσημη βιβλιοθήκη της Google, αλλά μάλλον η δεύτερη είναι περιττή.



Εικόνα 2.16: Η λειτουργία της ACRA όπως φαίνεται στο LogCat

Στην παραπάνω εικόνα βλέπουμε τι συμβαίνει στην συσκευή του χρήστη όταν πραγματοποιείται κάποιο κρίσιμο σφάλμα λειτουργίας. Καταρχήν (βήμα 1) η ACRA "πιάνει" την εξαίρεση (exception) που προέκυψε και ξεκινάει άμεσα να χτίζει την αναφορά της. Στο δεύτερο βήμα ακολουθεί η διαδικασία συλλογής δεδομένων από όλες τις πηγές που έχουμε ορίσει και μετά η ACRA αποστέλλει την αναφορά στο έγγραφό Google Doc που έχουμε ορίσει. Στο τρίτο και τελευταίο βήμα η ACRA αφήνει το σύστημα να συνεχίσει τη λειτουργία του και να εμφανίσει στα logs το stack trace του σφάλματος που προέκυψε, μαζί με το παράθυρο Force Close που εμφανίζεται στον χρήστη.

Αυτή η διαδικασία είναι εντελώς αόρατη στον χρήστη και από ότι φαίνεται στην εικόνα παραπάνω (Εικόνα 2.16) καθυστερεί ελάχιστα το σύστημα μιας και διαρκεί μόλις 8-10ms!

2.6 Κατακερματισμός του Android

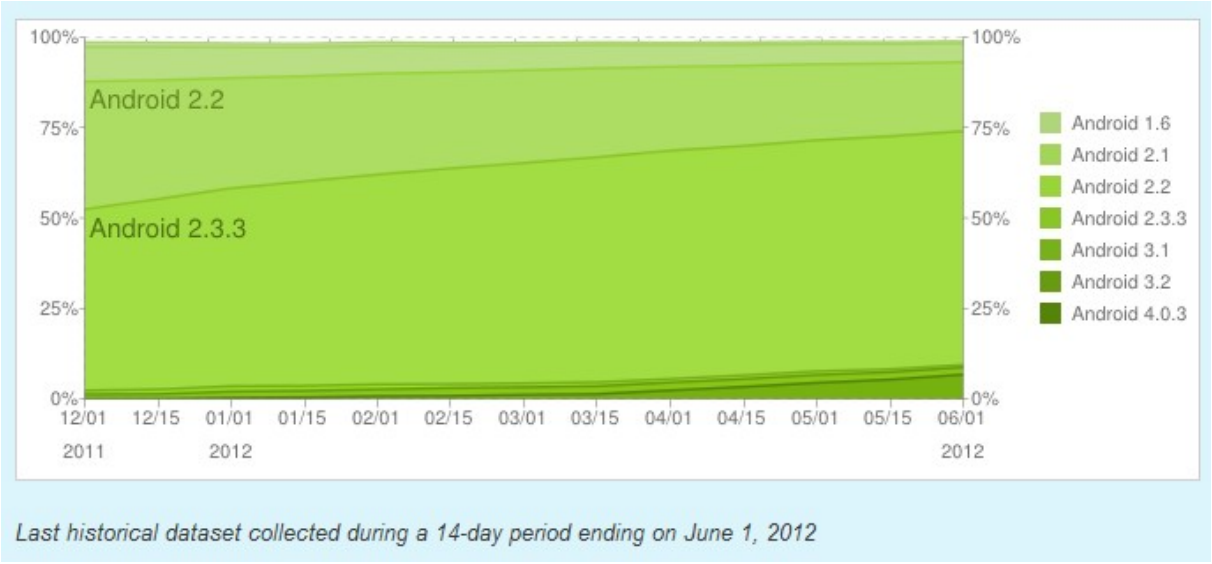
Στις προηγούμενες σελίδες έγινε λεπτομερή αναφορά σε όλες τις προκλήσεις που έχουν να αντιμετωπίσουν οι developers του Android, και επίσης έγινε λεπτομερής ανάλυση τις διαδικασίας ανάπτυξης εφαρμογών αλλά και των διαθέσιμων εργαλείων ανάπτυξης, δοκιμής και αποσφαλμάτωσης που παρέχει το Android και η κοινότητα του. Σε πολλά σημεία έγινε αναφορά στο πρόβλημα κατακερματισμού του Android. Τι είναι αυτό όμως;

Όπως γνωρίζουμε το Android ανήκει και αναπτύσσεται από την Google η οποία ανά τακτά διαστήματα χρονικά διαστήματα κάνει διαθέσιμη την νέα έκδοση με τη μορφή πηγαίου κώδικα στην κοινότητα. Ο κώδικας είναι open source φυσικά και οποιοσδήποτε μπορεί να τον κατεβάσει τοπικά και είτε να τον χρησιμοποιήσει για να φτιάξει μια έκδοση του λειτουργικού για λειτουργία σε κάποια συσκευή, είτε να συνεισφέρει στην περεταίρω ανάπτυξη του. Αυτή είναι και η δύναμη του λειτουργικού συστήματος! Η open source φύση του έχει προσελκύσει δεκάδες χιλιάδες ταλαντούχους developers ανά τον πλανήτη όπως και μεγάλες εταιρίες τηλεπικοινωνιών, οι οποίες επενδύουν εκατομμύρια δολάρια για ανάπτυξη συσκευών με λειτουργικό σύστημα Android!

Το Android λοιπόν χάρη στη δυναμική του έχει κατακτήσει ένα μεγάλο μερίδιο της αγοράς και είναι διαθέσιμο σε δεκάδες συσκευές κινητής τηλεφωνίας. Συνήθως οι κατασκευαστές κυκλοφορούν τις συσκευές τους είτε με την νεότερη έκδοση του λειτουργικού συστήματος, είτε με την αμέσως προηγούμενη μαζί με την υπόσχεση της αναβάθμισης στην νεότερη όταν αυτή έχει δοκιμαστεί επαρκώς από την εταιρία. Δυστυχώς όμως το θέμα των αναβαθμίσεων είναι αρκετά “ευαίσθητο”, διότι οι εταιρίες με στόχο την ευκολία και το κέρδος δεν κυκλοφορούν επαρκείς αναβαθμίσεις. Αντί αυτού προτιμούν να διαθέτουν τις νέες εκδόσεις λειτουργικού στις νεότερες συσκευές τους, με σκοπό να τις προτιμήσουν οι χρήστες, αφήνοντας τις παλιές συσκευές με την υπόσχεση τις αναβάθμισης.

Το κενό των εταιριών έρχεται να καλύψει η τεράστια κοινότητα developers του Android οι οποίοι όντας κάτοχοι “παραμελημένων“ συσκευών φροντίζουν να παρέχουν οι ίδιοι τις πολυπόθητες αναβαθμίσεις στα κινητά τους και άρα και συνήθως και των υπόλοιπων χρηστών! Αυτές οι εκδόσεις του Android είναι γνωστές ως “Custom Roms” και φυσικά υποστηρίζονται αποκλειστικά από τους developers τους και όχι από τις κατασκευάστριες εταιρίες των κινητών.

Παρά την τεράστια υποστήριξη της κοινότητας ο κατακερματισμός στο Android παραμένει υψηλός και εκτός και αν αλλάξει κάτι στον τρόπο των αναβαθμίσεων, τότε ίσως στο μέλλον να διογκωθεί ακόμη περισσότερο. Στο παρακάτω γράφημα φαίνεται καλύτερα το πρόβλημα.



Εικόνα 2.17: Χρονικό πλαίσιο υιοθέτησης των εκδόσεων του Android

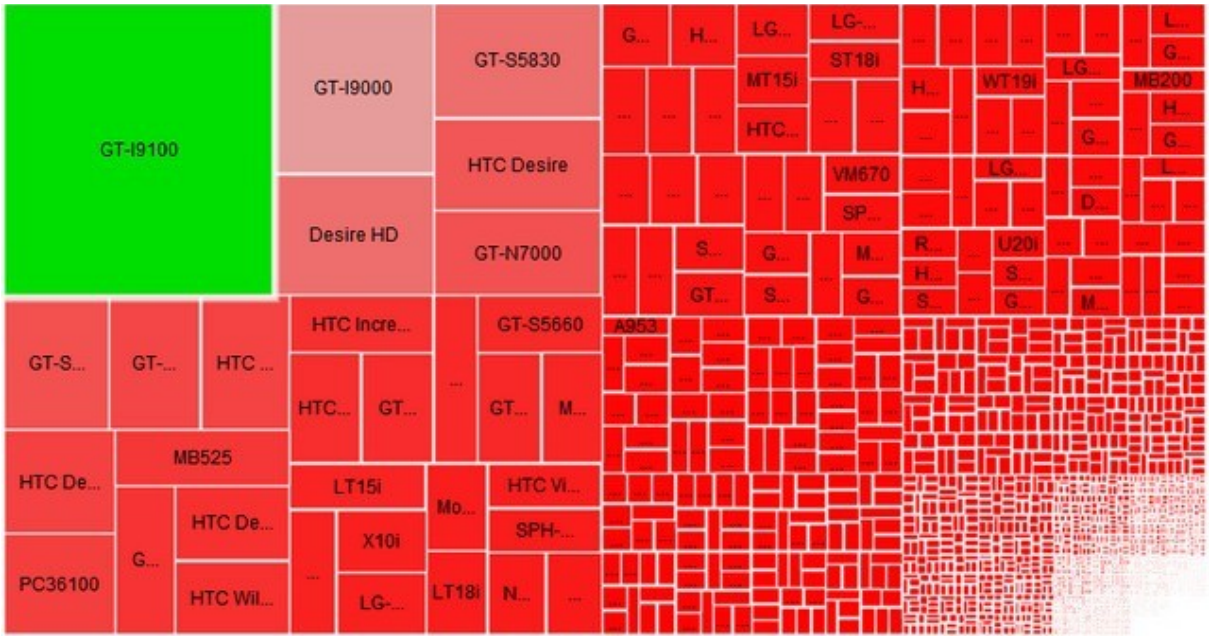
Όπως βλέπουμε το μοτίβο υιοθέτησης των νέων εκδόσεων του Android είναι αρκετά αργό, και αυτό οφείλετε κυρίως στην αδιαφορία των κατασκευαστών να παρέχουν στις παλαιότερες συσκευές υποστήριξη. Οι περισσότερες συσκευές υποστηρίζουν τις νεότερες εκδόσεις του Android χάρη στη συνεισφορά της κοινότητας, αλλά οι περισσότεροι χρήστες δεν έχουν τις απαραίτητες γνώσεις η εμπειρία για να χρησιμοποιήσουν κάποια από αυτές τις custom εκδόσεις.

Φυσικά όλες αυτές οι συσκευές δεν γίνεται να αγνοηθούν από τους προγραμματιστές εφαρμογών του Android γιατί αποτελούν μέρος τις πίτας και άρα πιθανοί χρήστες των εφαρμογών τους. Παραπάνω αναφερθήκαμε λεπτομερώς στις προκλήσεις προγραμματισμού σε διαφορετικά API και τα προβλήματα συμβατότητας που προκαλούν στον προγραμματιστή αυτά. Είναι συχνό φαινόμενο πολλές εταιρίες να επιλέγουν πρώτα το iOS για να αναπτύξουν εφαρμογές, με το Android να ακολουθεί δεύτερο και ο κύριο λόγος φυσικά είναι η υποστήριξη των εκατοντάδων συσκευών που τρέχουν τις διάφορες εκδόσεις του λειτουργικού συστήματος.

Μεγάλες εταιρίες ανάπτυξης mobile εφαρμογών μάλιστα αποσύρουν την υποστήριξη τους στο Android γιατί το θεωρούν ασύμφορο! Παράδειγμα αυτής της περίπτωσης είναι το παιχνίδι “Battleheart” της εταιρίας Mika Mobile που είναι διαθέσιμο για Android και iOS. Κάποιος μπορεί εύκολα αν αντιληφθεί ότι εφόσον μια μεγάλη εταιρία με πολλούς developers βρίσκει ασύμφορη για ανάπτυξη, την πλατφόρμα του Android, ο απλός developer που προσπαθεί να βγάλει ένα μικρό εισόδημα από εκεί, βρίσκεται σε ακόμη δυσκολότερη θέση.

2.6.1 Στατιστικά κατακερματισμού του Android από την εφαρμογή OpenSignalMaps

Για να γίνει πιο αντιληπτό το μέγεθος του προβλήματος, θα αναφερθούμε σε μια πρόσφατη δημοσίευση blog του κατασκευαστή της εφαρμογής OpenSignalMaps. Οι developers της συγκεκριμένης εφαρμογής συνέλεξαν στατιστικά στοιχεία όλων των συσκευών που χρησιμοποίησαν τις υπηρεσίες τους για διάστημα 6 μηνών και αποφάσισαν να μοιραστούν τα αποτελέσματα με τη μορφή γραφικών παραστάσεων.



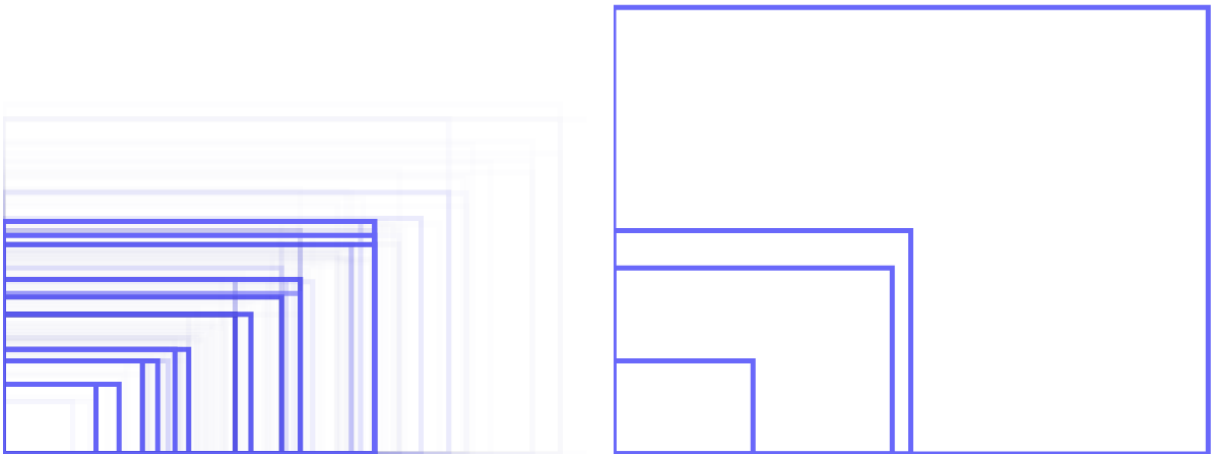
Εικόνα 2.18: Γραφική απεικόνιση συσκευών που χρησιμοποίησαν το OpenSignalMaps

Το παραπάνω εντυπωσιακό σχεδιάγραμμα είναι η γραφική αναπαράσταση της κατανομής του συνόλου των διαφορετικών συσκευών οι οποίες χρησιμοποίησαν την εφαρμογή σε διάστημα 6 μηνών. Στο σχεδιάγραμμα (εικόνα 2.18) απεικονίζονται περίπου 4000 διαφορετικές συσκευές, με κυρίαρχη το Samsung Galaxy S II! Φυσικά ο πραγματικός αριθμός συσκευών είναι αρκετά μικρότερος, αλλά εμφανίζονται πολύ περισσότερες λόγω της χρήσης custom roms οι οποίες μερικές φορές αλλάζουν την έκδοση build της συσκευής. Αυτό εξηγεί γιατί περίπου 1400 από τις παραπάνω συσκευές εμφανίζονται μόνο μία φορά στα στατιστικά χρήσης της εφαρμογής.

Οι μετρήσεις της εταιρίας φυσικά δεν σταμάτησαν στον αριθμό των συσκευών οι οποίες χρησιμοποίησαν την εφαρμογή, αλλά και στο λειτουργικό σύστημα (API Level) που υποστήριζαν αυτές. Τα 2 γραφήματα που ανάρτησε η εταιρία έχουν απόσταση ενός έτους το ένα από το άλλο και μας δείχνουν περίπου την ίδια κατανομή που είδαμε και στο σχεδιάγραμμα του Android Market. Κυρίαρχη έκδοση πέρσι ήταν η 2.3 ενώ φέτος κυρίαρχη έκδοση είναι η 2.3.3, στην οποία έχουν διορθωθεί κάποια bugs σε σχέση με την 2.3 και ουσιαστικά έχει ελάχιστες διαφορές στα APIs της. Επίσης πέρσι οι 2 πιο δημοφιλείς εκδόσεις του Android, καταλάμβαναν το 90% της αγοράς ενώ φέτος με την έλευση του Android 4.0 αυτό το ποσοστό έχει πέσει γύρω στο 75%.

Εικόνα 2.19: Κατανομή των APIs (Απρίλιος 2011 - Απρίλιος 2012)

Τέλος η εταιρία δημοσίευσε και τα στατιστικά στοιχεία των διαφορετικών αναλύσεων των συσκευών που χρησιμοποίησαν την εφαρμογή στο διάστημα των 6 μηνών και κατόπιν την σύγκριναν με το εύρος οθονών των συσκευών που χρησιμοποιούν iOS. Στην περίπτωση του Android μάλιστα ελαχιστοποίησε την κατανομή των διαστάσεων, τονίζοντας τις 13 πιο δημοφιλείς. Σε σχέση με τις μόλις 4 διαφορετικές αναλύσεις των συσκευών του iOS εύκολα αντιλαμβανόμαστε την πρόκληση υποστήριξης όλων αυτών των διαφορετικών αναλύσεων οθόνης ταυτόχρονα.



Εικόνα 2.20: Σύγκριση αναλύσεων Android vs iOS

ΚΕΦΑΛΑΙΟ 3

Ανάλυση Απαιτήσεων Εφαρμογής και Σχεδιασμός Layout

3.1 Απαιτήσεις Σχεδιασμού Εφαρμογής

Ο σκοπός της εργασίας είναι η ανάπτυξη μιας εφαρμογής η οποία θα δίνει πρόσβαση σε ανακοινώσεις, νέα και πληροφορίες για το τμήμα Πληροφορικής και Τηλεπικοινωνιών του ΤΕΙ Λάρισας. Ο σχεδιασμός της εφαρμογής λοιπόν θα πρέπει να είναι τέτοιος ώστε ο χρήστης αμέσως να εντοπίζει που βρίσκεται το αντικείμενο που τον ενδιαφέρει να ενημερωθεί η να χρησιμοποιήσει, και μέσω αυτού να έχει πρόσβαση στα υπόλοιπα παρεμφερή αντικείμενα πληροφοριών του τμήματος.

Η εφαρμογή θα παρέχει πρόσβαση στις ανακοινώσεις του τμήματος, τα δημόσια νέα, τις ανακοινώσεις ημερολογίου και φυσικά τις ανακοινώσεις των καθηγητών, μέσω των RSS Feeds που είναι διαθέσιμα στην ιστοσελίδα του τμήματος. Θα χρειαστούμε λοιπόν έναν XML Parser για την λήψη των Feeds και μια Activity για να προβάλουμε την λίστα των ανακοινώσεων στον χρήστη. Φυσικά εφόσον θα παίρνουμε πολλά διαφορετικά feeds ίδιου τύπου όμως, θα χρησιμοποιήσουμε την ίδια Activity για να προβάλουμε κάθε φορά το απαιτούμενο feed.

Για διευκόλυνση του χρήστη θα δώσουμε πρόσβαση σε όλες τις ανακοινώσεις μέσω ενός κουμπιού και μετά μέσω ενός παράθυρου διαλόγου ο χρήστης θα επιλέγει σε ποιες ανακοινώσεις θέλει να έχει πρόσβαση. Οι ανακοινώσεις κατά την λήψη τους θα αποθηκεύονται σε μια βάση δεδομένων SQLite, ειδικά διαμορφωμένη για να κρατάει τα 4 βασικά πεδία που περιέχει η κάθε ανακοίνωση.

Στη συνέχεια ο χρήστης μέσω της εφαρμογής θα πρέπει να έχει πρόσβαση σε βασικές πληροφορίες για τα μαθήματα του τμήματος, όπως αυτά ορίζονται στον οδηγό σπουδών, και από εκεί και μετά θα πρέπει να έχει πρόσβαση στο πρόγραμμα μαθημάτων, τον ολοκληρωμένο οδηγό σπουδών, αλλά φυσικά και στις βαθμολογίες του που βρίσκονται καταχωρημένες στον Διόνυσο. Όλα αυτά θα είναι προσβάσιμα από ένα κουμπί στο κεντρικό μενού και μετέπειτα ο χρήστης θα μπορεί να επιλέξει να έχει πρόσβαση στις πληροφορίες που χρειάζεται.

Αρχικά λοιπόν θα χρειαστούμε μια λίστα για να κρατήσει τα μαθήματα του κάθε εξαμήνου, και στη συνέχεια μέσω ενός μηχανισμού εναλλαγής προβολής, ο χρήστης θα μπορεί να δει και τα μαθήματα των υπόλοιπων εξαμήνων τα οποία θα χρησιμοποιούν την ίδια λίστα για προβολή. Το πρόγραμμα σπουδών και το πρόγραμμα μαθημάτων θα γίνονται διαθέσιμα προς λήψη από την ιστοσελίδα του τμήματος, και εφόσον ληφθούν μία φορά μετά θα είναι διαθέσιμα στον χρήστη για να τα ανοίξει μέσω ενός κατάλληλου προγράμματος office η pdf reader. Τέλος οι βαθμολογίες των μαθημάτων του χρήστη θα προβάλλονται μέσω μιας WebView η οποία θα χρησιμοποιηθεί αργότερα και σε άλλο σημείο της εφαρμογής.

Οι υπόλοιπες πληροφορίες που θα έχει πρόσβαση ο χρήστης εκτός των μαθημάτων, αφορούν:

- το εκπαιδευτικό προσωπικό του τμήματος
- πληροφορίες για τη γραμματεία
- πληροφορίες για το ενιαίο μητρώο
- τηλέφωνα επικοινωνίας
- πληροφορίες γενικά για το τμήμα

και οι οποίες θα εκχωρηθούν σε ένα νέο κουμπί το οποίο με τη σειρά του μέσω ενός διαλόγου, θα δίνει πρόσβαση σε αυτές. Εφόσον ο τύπος των πληροφοριών είναι παρόμοιος με αυτόν των μαθημάτων, θα χρησιμοποιήσουμε τους ίδιους μηχανισμού προβολής για να εμφανίσουμε στον χρηστη το επιθυμητό αποτέλεσμα μέσω ενός γνώριμου περιβάλλοντος.

Το τμήμα εκτός της κεντρικής του ιστοσελίδας www.cs.teilar.gr, χρησιμοποιεί ένα αριθμό βοηθητικών ιστοσελίδων όπως το e-class, ο dionysos, κλπ. Ο χρήστης θα μπορεί μέσω της εφαρμογής να βρει μαζεμένες σε μία λίστα όλες τις βοηθητικές ιστοσελίδες του τμήματος, και να τις ανοίξει μέσω του browser του κινητού του. Κάποιες από τις βοηθητικές ιστοσελίδες θα τις προβάλουμε απευθείας μέσω της εφαρμογής, χρησιμοποιώντας την WebView που έχουμε ήδη χρησιμοποιήσει για την προβολή των βαθμολογιών.

Τέλος η εφαρμογή θα ενσωματώνει μία προβολή χαρτών της Google, πάνω στην οποία θα βρίσκονται σημειωμένα με pin τα σημαντικότερα σημεία στον χώρο του ΤΕΙ τα οποία ενδιαφέρουν τον φοιτητή του τμήματος Πληροφορικής, ώστε να έχει ένα γρήγορο σημείο αναφοράς. Στην ίδια Activity θα ενσωματωθούν μέσω μενού:

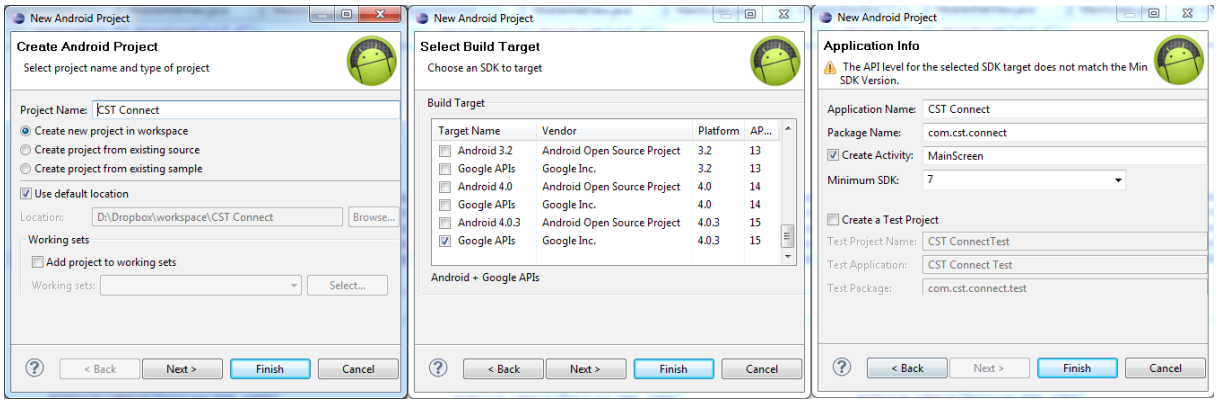
- λειτουργία πλοήγησης στο χώρο του ΤΕΙ μέσω των υπηρεσιών εντοπισμού που παρέχει το API του Android
- η κάτοψη (σκαρίφημα) του κεντρικού κτηρίου του Τμήματος Πληροφορικής.

3.2 Δημιουργία νέου Android Project στο Eclipse

Αφού έχουμε ορίσει τις απαιτούμενες ανάγκες τις εφαρμογής και έχουμε στο μυαλό μας τον βασικό κορμό, θα προχωρήσουμε στην υλοποίηση της. Όλα ξεκινάνε από τη δημιουργία ενός νέου Android project στο Eclipse (Εικόνα 3.1). Το όνομα που δηλώνουμε κατά τη δημιουργία του project δεν είναι το όνομα πακέτου της εφαρμογής, αλλά είναι το όνομα του φακέλου στον οποίο θα αποθηκευτεί το project, δηλαδή το όνομα του project.

Στο επόμενο βήμα επιλέγουμε την έκδοση και το είδος των APIs που θα χρησιμοποιήσουμε για να χτίσουμε την εφαρμογή μας. Για καλύτερη συμβατότητα επιλέγουμε την έκδοση 4.0 των Google APIs για να έχουμε πρόσβαση και στο API των Google maps.

Στο τρίτο και τελευταίο βήμα πρέπει να συμπληρώσουμε αρκετά σημαντικά πεδία. Στο πρώτο πεδίο εισάγουμε το όνομα της εφαρμογής το οποίο θα χρησιμοποιείτε όπου γίνεται αναφορά της εφαρμογής μας (ρυθμίσεις εφαρμογών, Google Play Store, κλπ). Στη συνέχεια εισάγουμε το όνομα του κεντρικού πακέτου της εφαρμογής μας. Το πεδίο του ονόματος δεν μας ενδιαφέρει αν είναι μοναδικό ή όχι καθότι είναι δυνατόν να κυκλοφορούν αρκετές εφαρμογές με το ίδιο όνομα. Στην περίπτωση του πεδίου του πακέτου όμως, πρέπει να επιλέξουμε ένα μοναδικό όνομα, το οποίο θα χρησιμοποιεί για την αρχειοθέτηση της εφαρμογής μας στις συσκευές αλλά και στο Play Store.



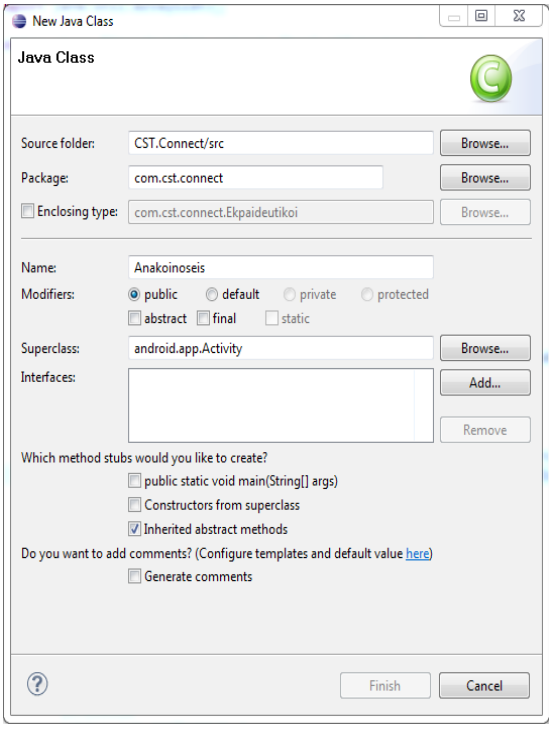
Εικόνα 3.1: Τα 3 στάδια δημιουργίας ενός νέου project

Στη συνέχεια επιλέγουμε να δημιουργηθεί αυτόματα η πρώτη Activity που θα εμφανίζεται μόλις εκκινεί η εφαρμογή μας και τις δίνουμε το όνομα που επιθυμούμε. Τέλος επιλέγουμε την χαμηλότερη έκδοση του Android που θα υποστηρίζει η εφαρμογή μας. Στην περίπτωση μας έχουμε θέσει ως κατώτερη έκδοση την 2.1 Eclair (έκδοση API – 7). Λόγω προβλημάτων συμβατότητας επιλέξαμε να αφήσουμε τις παλαιότερες εκδόσεις (1.5 & 1.6) χωρίς υποστήριξη μιας και το ποσοστό τους στην αγορά καταλαμβάνει μόλις το 1%.

Τα περισσότερα χαρακτηριστικά της εφαρμογής που δηλώσαμε παραπάνω, θα χρησιμοποιηθούν για να δημιουργηθεί αυτόματα το αρχείο AndroidManifext.xml. Αυτό σημαίνει ότι αργότερα μπορούμε να τροποποιήσουμε κάποιο από τα παραπάνω πεδία που δηλώσαμε, όπως πχ το όνομα της εφαρμογής, εφόσον αυτό κριθεί απαραίτητο.

3.3 Δημιουργία των Activities της εφαρμογής

Αφού έχουμε ξεκαθαρίσει τις ανάγκες τις εφαρμογής μας ήρθε η ώρα να δηλώσουμε τις απαραίτητες οθόνες διεπαφής χρήση, δηλαδή τις Activities. Έχουμε ήδη δημιουργήσει αυτόματα την Activity “MainScreen” η οποία θα εμφανίζεται στον χρήστη αμέσως μόλις ανοίξει την εφαρμογή, και θα τη χρησιμοποιήσουμε ως κεντρικό σημείο επαφής με τις υπόλοιπες λειτουργίες τις εφαρμογής. Για να δημιουργήσουμε μια νέα Activity στο project μας, πηγαίνουμε στη διαδρομή **File/New/Class** στο Eclipse, και βλέπουμε το παράθυρο δημιουργίας νέας κλάσης. Στη συνέχεια δηλώνουμε το όνομα της εφαρμογής και στο πεδίο “Superclass” εισάγουμε την κλάση android.app.Activity έτσι ώστε η δική μας κλάση να ενσωματώσει όλα τα χαρακτηριστικά της superclass Activity.



Εικόνα 3.2: Δημιουργία της Activity - "Anakoinoseis"

Τελειώνοντας με τις υπόλοιπες ρυθμίσεις της νέα μας Activity πατάμε το κουμπί finish για να προστεθεί η κλάση “Anakoinoseis” στον φάκελο src του project. Με τον ίδιο τρόπο θα δημιουργήσουμε και τις υπόλοιπες Activities της εφαρμογής μας.

Η activity “Anakoinoseis” θα χρησιμοποιηθεί για να για να γίνουν όλες οι απαραίτητες λειτουργίας λήψης, αποθήκευσης και εμφάνισης των ανακοινώσεων του τμήματος στους χρήστες. Συνολικά η εφαρμογή διαθέτει 10 κλάσεις που ανήκουν στην κατηγορία των Activities για την διεπαφή με τον χρήστη:

- **Anakoinoseis –** Η κεντρική οθόνη προβολής τις λίστας των ανακοινώσεων.
- **CSMapView –** Οθόνη προβολής του χάρτη του ΤΕΙ με τα σημεία ενδιαφέροντος.
- **Ekpaideutikoi –** Οθόνη προβολής πληροφοριών για το Εκπαιδευτικό Προσωπικό του τμήματος (Μόνιμους και Συνεργάτες).
- **Links –** Λίστα προβολής συνδέσμων των ιστοσελίδων του τμήματος.
- **MainScreen** – Κεντρική οθόνη της εφαρμογής η οποία παρέχει πρόσβαση στις υπόλοιπες Activities.
- **Mathimata -** Λίστες προβολής των μαθημάτων του τμήματος όπως είναι αυτά κατανεμημένα ανά εξάμηνο σπουδών.
- **MobileWebView –** Ενσωματωμένος Web Browser ο οποίος θα χρησιμοποιηθεί σε μερικές περιπτώσεις για προβολή περιεχομένου Web μέσω της εφαρμογής.
- **Plirofories –** Λίστες που περιλαμβάνουν τις πληροφορίες τμήματος, τις γραμματείας, του ενιαίου μητρώου, και τα τηλέφωνα επικοινωνίας.
- **Skarifima –** Εικόνα κάτοψης του κεντρικού κτηρίου του τμήματος με δυνατότητα Zoom in/out
- **TeachersList –** Ενιαία λίστα καθηγητών, για γρήγορη πρόσβαση στις ανακοινώσεις του καθενός.

3.4 Δήλωση των Activities στο AndroidManifest.xml

Όλες οι Activities της εφαρμογής πρέπει να είναι δηλωμένες στο Αρχείο AndroidManifest.xml, στο οποίο δηλώνονται και οι υπόλοιπες ιδιότητες τους όπως το “intent-filter” η παράμετρος του οποίου χρησιμοποιείτε όταν θέλουμε να μεταβούμε από τη μία δραστηριότητα στην άλλη μέσω Intent. Αυτή τι στιγμή υπάρχει μόνο μία Activity δηλωμένη στο αρχείο xml και αυτή δεν είναι άλλη από την “MainScreen” η οποία δηλώθηκε στην αρχή του project μας.

```
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainScreen"
            android:icon="@drawable/ic_launcher"
            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>
```

Όπως βλέπουμε, μεταξύ των tags "application" υπάρχει μία και μοναδική activity δηλωμένη με τα αντίστοιχα tags, τα οποία θα χρησιμοποιήσουμε για να προσθέσουμε τις υπόλοιπες Activities, ξεκινώντας από την "Anakoinoseis":

```
<activity
      android:name=".Anakoinoseis"
      android:label="@string/app_name" >
      <intent-filter>
            <action android:name="com.cst.connect.Anakoinoseis" />
            <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
</activity>
```

Στο πρώτο πεδίο "name" δηλώνουμε το όνομα της Activity ακριβώς όπως και στο αρχείο κλάσης που δημιουργήσαμε παραπάνω. Στο δεύτερο πεδίο "name" δηλώνουμε το όνομα του intent-filter το οποίο αποτελείτε από το όνομα πακέτου και το όνομα της κλάσης. Στο τρίτο πεδίο "name" του tag "category" αλλάζουμε το τελευταίο κομμάτι και αντί για "*LAUNCHER*" γράφουμε "*DEFAULT*". Η Activity η οποία έχει την πρώτη παράμετρο στην δήλωση της σημαίνει ότι είναι η πρώτη Activity που θα εμφανίζεται στον χρήστη κατά την εκκίνηση της εφαρμογής, ενώ την παράμετρο "*DEFAULT*" παίρνουν όλες οι υπόλοιπες Activities του project μας. Παρομοίως θα δηλώσουμε και τις υπόλοιπες 8 δραστηριότητες που αναφέραμε παραπάνω και θα τις εισάγουμε ανάμεσα στα "application" tags του AndroidManifest.

3.5 Δημιουργία layout της κεντρικής οθόνης

Αφού ολοκληρώσαμε το στήσιμο του project, τη δημιουργία των Activities αλλά και τη δήλωση αυτών στο αρχείο AndroidManifest.xml, συνέχεια έχει ο σχεδιασμός των βασικών layouts που θα χρησιμοποιηθούν στην εφαρμογή, μαζί με τα βοηθητικά layouts που θα ενσωματώνει το κάθε κύριο layout εφόσον αυτά κριθούν απαραίτητα. Θα ξεκινήσουμε φυσικά από την κεντρική Activity "MainScreen". Στην αρχή θα ενσωματώσουμε την βιβλιοθήκη "ActionBarSherlock" η οποία θα μας δώσει πρόσβαση σε μια custom ActionBar, στη συνέχεια θα σχεδιάσουμε το υπόλοιπο γραφικό περιβάλλον της "MainScreen"

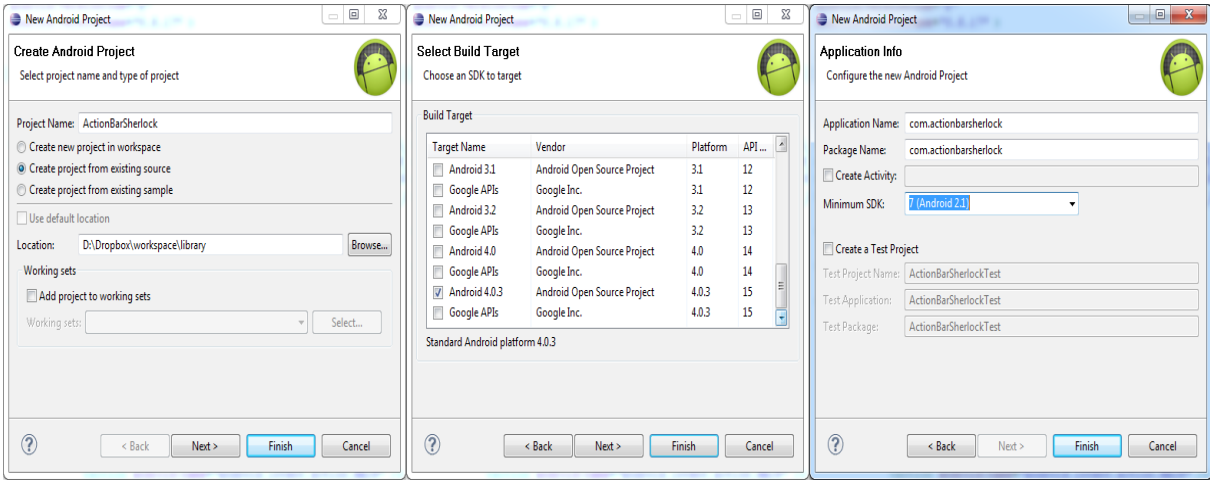
3.5.1 Χρήση της βιβλιοθήκης ActionBarSherlock

Όπως αναφέραμε στο κεφάλαιο 2, το Android συνιστά στους developers να ακολουθήσουν κάποιους κανόνες "καλού σχεδιασμού εφαρμογών". Μεταξύ αυτών των κανόνων συνιστάται και η χρήση της Action Bar για πλοήγηση στα μενού των εφαρμογών, αλλά και εύκολη πρόσβαση στα μενού και σε άλλες επιλογές της κάθε Activity. Η action bar που έχει ενσωματωμένη το Android, περιέχεται στο API 10 και πάνω, πράγμα που σημαίνει ότι οι παλαιότερες εκδόσεις του Android δεν μπορούν να την χρησιμοποιήσουν. Αντί αυτής λοιπόν θα κάνουμε χρήση μιας βιβλιοθήκης η οποία θα μας δώσει πρόσβαση σε μια custom υλοποίηση της Action Bar η οποία ακούει στο όνομα "ActionBarSherlock".

Πρόκειται για μια βιβλιοθήκη η οποία διανέμεται κάτω από την άδεια Apache 2.0 και ο κάθε developer μπορεί να την ενσωματώσει στην εφαρμογή του και με αυτό τον τρόπο να ενσωματώσει μια πλήρη υλοποίηση της Action Bar ακόμη και σε παλαιότερες εκδόσεις του Android. Ο developer της βιβλιοθήκης, Jake Wharton, διατηρεί την ιστοσελίδα http://actionbarsherlock.com/, στην οποία βρίσκεται η τελευταία έκδοση της βιβλιοθήκης, μαζί με οδηγίες χρήσης και παραδείγματα ενσωμάτωσης.

Η βιβλιοθήκη ενσωματώνεται στην εφαρμογή μας μέσω της δημιουργίας νέου project στο workspace που δουλεύουμε, και ενσωμάτωση αυτού στο κεντρικό μας project ως βιβλιοθήκη. Η μόνη διαφορά από τη δημιουργία ενός νέου project είναι ότι κάνουμε χρήση υπάρχων κώδικα και άρα το νέο library project δημιουργείτε με βάση τον πηγαίο κώδικα ενός φακέλου που δηλώνουμε και περιέχει και το απαιτούμενο αρχείο AndroidManifest.xml. Στη συνέχεια αφού δημιουργήσουμε το project της βιβλιοθήκης(Εικόνα 3.3) πηγαίνουμε στις

ιδιότητες του δικού μας project και εκεί προσθέτουμε στην λίστα με τις βιβλιοθήκες που χρησιμοποιούμε, την ActionBarSherlock.



Εικόνα 3.3: Τα 3 στάδια δημιουργίας ενός library project χρησιμοποιώντας υπάρχων κώδικα.

Στη συνέχεια για να το ενσωματώσουμε την action bar σε όλες μας τις Activities, πηγαίνουμε στο AndroidManifest.xml και κάτω από την παράμετρο "android-name" στο κεντρικό tag "activity", προσθέτουμε την παράμετρο:

```
android:theme="@style/Theme.Sherlock"
```

από αυτό το σημείο και στο εξής όλες οι δραστηριότητες που έχουμε και όσες θα χρησιμοποιήσουμε θα μπορούν να κάνουν χρήση της ActionBarSherlock, η οποία αντικαθιστά και την υπάρχουσα μπάρα με το όνομα της εφαρμογής.

Για να γίνει χρήση από της νέας μπάρας από τις Activities πρέπει να κάνουμε μια μικρή αλλαγή και στις 10 κλάσεις τύπου Activity που χρησιμοποιεί η εφαρμογή μας. Θα αλλάξουμε την SuperClass την οποία ανήκει η κάθε Activity, από "Activity" σε "SherlockActivity" οπότε θα γίνει:

```
public class Anakoinoseis extends SherlockActivity
```

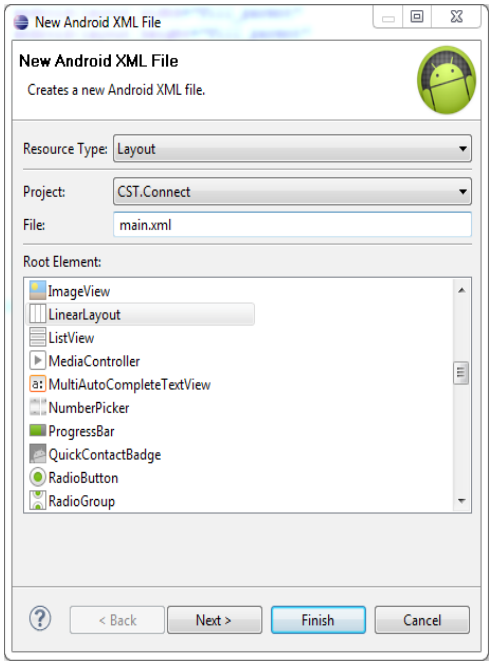
και θα ενσωματώσουμε και την αντίστοιχη κλάση από την βιβλιοθήκη που εισάγαμε:

```
import com.actionbarsherlock.app.SherlockActivity;
```

3.5.2 Δημιουργία του GridView layout

Αφού ολοκληρώσαμε την αρχική ενσωμάτωση της action bar ακολουθεί ο σχεδιασμός της κεντρικής οθόνης. Θέλουμε η εφαρμογή μας να δίνει πρόσβαση στις υπόλοιπες δυνατότητες μέσω μερικών κουμπιών τα οποία θα είναι οργανωμένα σε πλέγμα. Η χρήση της GridView για την υλοποίηση αυτή κρίθηκε μονόδρομος. Μέσα στην GridView θα χρησιμοποιήσουμε 6 αντικείμενα ως "κουμπιά" τα οποία θα συνδέουν την κεντρική Activity με τις υπόλοιπες.

Δημιουργούμε λοιπόν ένα νέο Layout μέσω του Eclipse, πηγαίνοντας στο **File/New/Android XML File.** Εκεί επιλέγουμε τον τύπο του xml (Layout) στην περίπτωση μας, το όνομα του (main.xml) και τον τύπο του root element (LinearLayout) και πατάμε Finish για να δημιουργηθεί το νέο μας layout. Επιλέξαμε το LinearLayout και όχι το GridView σαν root γιατί είναι πιο εύκολο στην διαχείριση και ενδεικνύεται να χρησιμοποιείτε ως root στα περισσότερα layouts. Μέσα στον κεντρικό layout εμείς θα προσθέσουμε την GridView μαζί με τις παραμέτρους της Για να γίνει αυτό εφικτό θα προσθέσουμε στο xml μας τον παρακάτω κώδικα:



Εικόνα 3.4: Δημιουργία νέου αρχείου Layout

```
<GridView

  android:id="@+id/gridview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:gravity="center"
  android:listSelector="#00000000"
  android:numColumns="2"
          >
</GridView>
```

Τα tags "GridView" περιγράφουν τον τύπο του layout που προσθέτουμε, η παράμετροι "`layout_width`" & "`layout_height`" ορίζουν τον χώρο που θα χρησιμοποιεί η GridView για προβολή, με την παράμετρο *`"fill_parent"`* να υποδηλώνει ότι θα χρησιμοποιηθεί όλος ο διαθέσιμος χώρος που παρέχει το layout γονέας, δηλαδή το

LinearLayout. Η παράμετρος "id" χρησιμοποιείτε ως σημείο αναφοράς του layout από τον κώδικα μας. Η "gravity" δηλώνει την τοποθέτηση του αντικειμένου του κάθε κελιού μέσα στο layout. Η "listSelector" αλλάζει το χρώμα επιλογής του κάθε αντικειμένου, από το προεπιλεγμένο πορτοκαλί σε "κανένα χρώμα". Τέλος η παράμετρος "numColumns" δηλώνει τον αριθμό των στηλών που επιθυμούμε να έχει η GridView.

3.5.3 Δημιουργία xml αντικειμένου της GridView

Σειρά παίρνουν η δημιουργία των αντικειμένων τα οποία θα χρησιμοποιεί η GridView για να γεμίσει τις λίστες τις. Ξεκινάμε δημιουργούμε ένα νέο layout όπως κάναμε παραπάνω, και το ονομάζουμε "gridview_item". Θα χρησιμοποιήσουμε και πάλι ένα LinearLayout ως βασικό, και μέσα σε αυτό θα συμπεριλάβουμε ένα ImageView layout για την εικόνα που εμφανίζεται, και ένα TextView layout για το κείμενο κάτω από την κάθε εικόνα. Το τελικό αποτέλεσμα θα είναι κάπως έτσι:

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

    <ImageView
        android:id="@+id/grid_item_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:padding="7dp"
        android:scaleType="centerInside" />

    <TextView
        android:id="@+id/grid_item_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal|bottom"
        android:textStyle="bold" />
</LinearLayout>
```

3.5.4 Χρήση διαφορετικού layout ανά orientation

Μια παράμετρος που πρέπει να λάβουμε σοβαρά υπόψιν είναι το πως θα εμφανίζεται η GridView σε κάθε προσανατολισμό της συσκευής. Υπάρχουν 2 προσανατολισμοί ο "portrait" ο οποίος είναι ο κάθετος προσανατολισμός της συσκευής, και ο "landscape" ο οποίος είναι ο οριζόντιος. Σε κάθε προσανατολισμό έχουμε διαφορετικό χώρο στη διάθεση μας και είναι στο χέρι μας το πως θα τον εκμεταλλευτούμε! Επί της παρούσης θα φροντίσουμε να εμφανίζεται σωστά το υπάρχων layout και στις δύο κατευθύνσεις. Δυστυχώς αν αφήσουμε την παράμετρο "`numColumns`" να πάρει την τιμή "`auto`" έχουμε σαν αποτέλεσμα σε portrait να έχουμε 2 στήλες, όπως επιθυμούμε, αλλά και στο landscape παρά τον άπλετο χώρο η ρύθμιση auto εμφανίζει 2 στήλες αντί για 3! Για να λύσουμε αυτό το απλό πρόβλημα, θα χρησιμοποιήσουμε την δυνατότητα του Android για χρήση διαφορετικών αρχείων layout ανά περίσταση.

Αρχικά θα δημιουργήσουμε 2 νέους υποφακέλους στον φάκελο "res" του project μας ο ένας με το όνομα "layout-port" και ο άλλος με το όνομα "layout-res". Μέσα στον κάθε φάκελο θα τοποθετήσουμε από μία έκδοση του αρχείου layout "main.xml". Η μόνη τους διαφορά θα είναι στην τιμή της παραμέτρου "`numColumns`" στην πρώτη περίπτωση της οποία η τιμή θα είναι "2" και στην δεύτερη περίπτωση της landscape orientation, η τιμή θα είναι "3".

3.6 Δημιουργία της λίστας Ανακοινώσεων

Συνεχίζουμε την δημιουργία των layouts με ένα layout λίστας το οποίο θα χρησιμοποιηθεί αρχικά στην Activity των ανακοινώσεων και μετά θα το ξαναχρησιμοποιήσουμε σε άλλες 2 Activities τύπου λίστας. Δημιουργούμε ένα νέο LinearLayout όπως παραπάνω το οποίο ονομάζουμε "list.xml", και στη συνέχεια προσθέτουμε μια κενή ListView συνοδευόμενη από μια κενή TextView.

```
<ListView
        android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
       />
```

```
<TextView
    android:id="@android:id/empty"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:visibility="gone" />
```

Η TextView υπάρχει για να "γεμίσει" με τα αντικείμενα λίστας που θα δημιουργήσουμε τώρα αμέσως για τις ανακοινώσεις.

3.6.1 Δημιουργία αντικειμένου λίστας

Στη λίστα των ανακοινώσεων θέλουμε να εμφανίζεται μόνο ο τίτλος και η ημερομηνία της ανακοίνωσης, οπότε χρειαζόμαστε 2 TextViews την μία κάτω από την άλλη. Δημιουργούμε λοιπόν ένα νέο layout με το όνομα "rss_list_item.xml" και μέσα στην LinearLayout προσθέτουμε τα εξής:

```
<TextView
        android:id="@+id/title"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:textSize="20sp"
        android:textStyle="bold"/>

    <TextView
        android:id="@+id/pubDate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:textSize="14sp" />
```

Δίνουμε στην κάθε TextView το επιθυμητό id αναφοράς ώστε αργότερα να τις τροφοδοτήσουμε με κείμενο. Πλέον στην παράμετρο "`layout_height`" δίνουμε τιμή "*`wrap_content`*" που σημαίνει ότι θέλουμε να καταλαμβάνει μόνο όσο χώρο χρειάζεται το κείμενο και καθόλου παραπάνω. Την χρησιμοποιούμε λοιπόν για να αποφύγουμε κάποιες παραμορφώσεις στην εμφάνιση της λίστας. Στη συνέχεια θέτουμε την παράμετρο "`padding`" να έχει ως τιμή "*`2dp`*". Τα dp είναι συντομογραφία για τον όρο density independent pixels και σημαίνει ότι το το κενό το οποίο ορίζουμε με το padding θα φαίνεται το ίδιο σε όλες τις

οθόνες ανεξαρτήτως της διάστασης τους.

Οι παράμετροι "textSize" και "textStyle" ορίζουν το μέγεθος και το στυλ τις γραμματοσειράς. Στις TextViews καλό είναι να χρησιμοποιήται η μονάδα "sp" αντί της "dp" η οποία είναι συντομογραφία του όρου scale independent pixels. Λειτουργεί ακριβώς όπως και η dp αλλά επηρεάζεται από τις ρυθμίσεις γραμματοσειράς της κάθε συσκευής.

3.6.2 Δημιουργία παράθυρου διαλόγου ανακοίνωσης

Αφού τελειώσαμε με την δημιουργία και το "γέμισμα της λίστας" όσον αφορά τα layouts, ήρθε η ώρα να δημιουργήσουμε τα layouts του παράθυρου διαλόγου της ανακοίνωσης. Δημιουργούμε 2 νέα αρχεία layout, το πρώτο το ονομάζουμε "custom_dialog.xml" και το δεύτερο "custom_title.xml". Όπως φαίνεται και από τα ονόματα των δύο αρχείων, το ένα θα χρησιμοποιηθεί για να προβάλουμε το κυρίως σώμα του παράθυρου διαλόγου με το κείμενο της ανακοίνωσης και το δεύτερο θα περιέχει τον τίτλο της ανακοίνωσης στον οποίο όμως θα συνδέσουμε το link της ανακοίνωσης ώστε ο χρήστης κάνοντας κλικ πάνω του, μπορεί να δει την ανακοίνωση όπως αυτή αναρτήθηκε στο site του τμήματος.

Το πρώτο xml λοιπόν θα περιέχει μια πολύ απλή TextView μόνο με τις παραμέτρους **"layout_height"**, και **"id"**, ενώ στο δεύτερο xml θα έχουμε πάλι μια TextView, αλλά αυτή τη φορά θα χρησιμοποιήσουμε και την παράμετρο "clickable" στην οποία θα δώσουμε τιμή "true". Η παράμετρος αυτή ενεργοποιεί τα συμβάντα click στην τρέχουσα View.

3.6.3 Ενσωμάτωση Menu επιλογών στην action bar

Πλέον έχουμε δημιουργήσει όλα τα layouts τα οποία θα χρησιμοποιηθούν στην οθόνη των ανακοινώσεων, μας μένει μόνο να προσθέσουμε ένα κουμπί ώστε ο χρήστης να μπορεί να κάνει refresh την λίστα για να δει αν υπάρχουν νέες ανακοινώσεις. Το νέο αρχείο xml που θα χρησιμοποιήσουμε για αυτό το σκοπό θα είναι τύπου Menu αντί για Layout. Αφού το δημιουργήσουμε και το ονομάσουμε "menu_refresh", θα βάλουμε ένα και μοναδικό αντικείμενο μέσα.

```
<item
  android:id="@+id/refresh"
  android:icon="@drawable/ic_refresh_inverse"
  android:title="Refresh feed"
  android:showAsAction="ifRoom"/>
```

Η παράμετρος "icon" χρησιμοποιείτε για να ορίσουμε το γραφικό που θέλουμε να εμφανίζεται, στην περίπτωση μας ένα τυπικό refresh. Η παράμετρος "showAsAction" χρησιμοποιείτε για να ενσωματωθεί το αντικείμενο μας στην Action Bar.

3.7 Δημιουργία layout Χάρτη

Στη δημιουργία του layout της Activity του χάρτη θα λειτουργήσουμε λίγο διαφορετικά σε σχέση με τα υπόλοιπα layouts. Δημιουργούμε κανονικά ένα νέο layout με όνομα "mapview.xml" και αφού δημιουργηθεί, διαγράφουμε την LinearLayout που περιέχει και αντί αυτής εισάγουμε τον παρακάτω κώδικα:

```
<com.google.android.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="@string/map_api_key"
    android:clickable="true"
    />
```

Αρχικά χρησιμοποιούμε το custom tag "com.google.android.maps.MapView" το οποίο κάνει χρήση της κλάσης συστήματος MapView για να προβάλει μια τέτοια, και εκτός από τις ήδη γνωστές παραμέτρους εισάγουμε και τη παράμετρο "apiKey". Για να χρησιμοποιήσουμε το API των χαρτών πρέπει να ζητήσουμε από την Google ένα κλειδί το οποίο προκύπτει από το MD5 ενός αρχείου στον υπολογιστή μας. Η διαδικασία αυτή γίνεται για να μην κορεστεί από χρήστες το API των Google Maps, αλλά παρόλα αυτά είναι απλή και γρήγορη διαδικασία. Το κλειδί είναι ένα μεγάλο αλφαριθμητική το οποίο έχουμε εισάγει σαν πόρο String με το όνομα "map_api_key". Παρακάτω θα δούμε περισσότερα για την εισαγωγή resources στο project μας.

3.7.1 Δήλωση χρήσης της βιβλιοθήκης στο AndroidManifest

Για να κάνουμε χρήση του API των χαρτών, εξασφαλίσαμε ένα κλειδί από την Google, επιλέξαμε την έκδοση των APIs που περιέχουν τους χάρτες στην αρχή της δημιουργίας του project μας αλλά μένει κάτι ακόμη. Να δηλώσουμε την χρήση της βιβλιοθήκης στο αρχείο AndroidManifest.xml. Αυτό γίνεται πολύ απλά εισάγοντας την παρακάτω γραμμή κώδικα:

```
<uses-library android:name="com.google.android.maps" />
```

3.7.2 Χρήση μενού για προσθήκη κουμπιών στην Action Bar

Για να ολοκληρώσουμε την Activity των χαρτών, μένει να εισάγουμε τα αντικείμενα Menu τα οποία θα εμφανίζονται στην Action Bar και θα μας γίνουν πρόσβαση το πρώτο στην Activity "Skarifima" και το δεύτερο στην δυνατότητα λήψης οδηγιών πλοήγησης στο τμήμα. Δημιουργούμε λοιπόν νέο xml τύπου Menu, το ονομάζουμε "menu_maps.xml" και εισάγουμε 2 items όπως κάναμε και στην περίπτωση του menu_refresh.

Επειδή όμως η Activity "Skarifima" δεν λειτουργεί σε APIs μικρότερα της έκδοσης 8 (Android 2.2) θα κάνουμε ένα δεύτερο αρχείο layout το οποίο ονομάζουμε "menu_maps_eclair.xml" και το οποίο θα περιέχει μόνο ένα αντικείμενο Menu, αυτό της πλοήγησης. Μέσα στον κώδικα της Activity θα γίνεται έλεγχος του API της συσκευής που τρέχει την εφαρμογή και θα χρησιμοποιεί το κατάλληλο αρχείο xml.

3.8 Δημιουργία WebView layout για χρήση προβολής ιστοσελίδων

Κατά το σχεδιασμό της εφαρμογής προβλέψαμε την χρήση μιας webView Activity η οποία θα χρησιμοποιείτε κάθε φορά που θέλουμε να προβάλουμε περιεχόμενο web μέσω της εφαρμογής μας. Δημιουργούμε νέο layout λοιπόν με το όνομα "webview.xml" και αντί για LinearLayout ως root, επιλέγουμε απευθείας το WebView. Δεν θέτουμε καμία άλλη παράμετρο εκτός από το "id" της view.

```
<WebView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

</WebView>
```

3.9 Δημιουργία και χρήση κοινών layout για παρόμοιες Activities

Έχουμε ήδη δημιουργήσει μερικά layouts τα οποία θα χρησιμοποιηθούν σε παραπάνω από μία Activities, αλλά η συγκεκριμένη περίπτωση είναι λίγο διαφορετική καθότι θα πρέπει να εισάγουμε μια βιβλιοθήκη για να υποστηρίξουμε το layout αυτό. Η βιβλιοθήκη ονομάζεται "ViewPagerIndicator" και μας επιτρέπει να εναλλάσσουμε προβολές (Views) σε μία Activity μέσω swipe κίνησης αριστερά και δεξιά. Η βιβλιοθήκη μας δίνει τη δυνατότητα να εισάγουμε custom τίτλους πάνω από κάθε προβολή, και κρατάει στο δεξιό και στο αριστερό της μέρος τα ονόματα των διπλανών προβολών.

Στη συνέχεια θα χρησιμοποιήσουμε αντικείμενα λίστας όπως κάναμε και παραπάνω στις ανακοινώσεις, για να γεμίσουμε με περιεχόμενο τις view μας. Θα χρειαστούμε συνολικά 3 αντικείμενα λίστας, μιας και τόσες διαφορετικές λίστες θα προβάλουμε στην ViewPager. Επίσης θα χρειαστούμε και 4 custom layouts τα οποία θα χρησιμοποιηθούν για την προβολή των πληροφοριών.

3.9.1 Χρήση της βιβλιοθήκης ViewPagerIndicator

Ξεκινώντας τη διαδικασία θα πρέπει να ενσωματώσουμε την βιβλιοθήκη "ViewPagerIndicator" που εντελώς συμπτωματικά έχει αναπτυχθεί από τον ίδιο άνθρωπο που ανέπτυξε και την "ActionBarSherlock", τον Jake Wharton! Η ενσωμάτωση της βιβλιοθήκης λοιπόν είναι ακριβώς η ίδια διαδικασία με την ενσωμάτωση της action bar, οπότε δεν θα την επαναλάβουμε εδώ πέρα.

Στη συνέχεια θα κατασκευάσουμε το layout που θα χρησιμοποιηθεί από τις 3 Activities για να προβάλουν την ViewPager. Νέο αρχείο layout λοιπόν με όνομα "viewpager_layout" και στη συνέχεια εισάγουμε τον παρακάτω κώδικα:

```
<com.viewpagerindicator.TitlePageIndicator
        android:id="@+id/indicator"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
```

Και τα δύο αντικείμενα του xml μας είναι αποτελούν custom Tags. Το πρώτο αναφέρεται στην κλάση "TitlePageIndicator" την οποία ενσωματώσαμε μαζί με την βιβλιοθήκη, και το δεύτερο αναφέρεται στην κλάση "ViewPager" η οποία περιέχεται στο πακέτο με τις βιβλιοθήκες συμβατότητας του Android. Όπως και με τις ListView, το δεύτερο αντικείμενο θα χρησιμοποιηθεί για να προβάλουμε τα custom layouts που θα σχεδιάσουμε παρακάτω.

3.9.2 Δημιουργία αντικειμένου για την λίστα μαθημάτων

Όπως κάναμε και στις προηγούμενες περιπτώσεις αντικειμένου λίστας βλέπουμε τις ανάγκες του αντικειμένου που θέλουμε να προβάλουμε και στη συνέχεια με αυτό στο μυαλό μας προχωράμε στο σχεδιασμό του. Το layout του αντικειμένου λίστας των μαθημάτων είναι λίγο πολύπλοκο λόγω των διαφορετικών αντικειμένων που θέλουμε να προβάλουμε σε αυτό. Σε αυτή τη περίπτωση αντικειμένου, θέλουμε να εμφανίσουμε μια μικρή εικόνα στο αριστερό μέρος του αντικειμένου και δίπλα από αυτήν θα εμφανίσουμε δύο σειρές με τρεις διαφορετικές TextViews η κάθε μία. Ακολουθεί ο κώδικας και ο σχολιασμός του.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="false"
    android:focusable="false"
    android:focusableInTouchMode="false"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/mathima_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingRight="1dp" >
    </ImageView>
```

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

    <TextView
        android:id="@+id/row_title"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:paddingLeft="1dp"
        android:textSize="15sp"
        android:textStyle="bold" />

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

        <TextView
            android:id="@+id/row_omada"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="10sp"
            android:textStyle="normal" />

        <TextView
            android:id="@+id/row_typos"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="10sp"
            android:textStyle="normal" />

        <TextView
            android:id="@+id/row_dm"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="10sp"
            android:textStyle="normal" />
    </LinearLayout>

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

        <TextView
            android:id="@+id/row_theoreia"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="10sp"
            android:textStyle="normal" />

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

```
                android:layout_width="wrap_content"
                android:layout_height="fill_parent"
                android:paddingLeft="1dp"
                android:textSize="10sp"
                android:textStyle="normal" />

            <TextView
                android:id="@+id/row_ergastirio"
                android:layout_width="wrap_content"
                android:layout_height="fill_parent"
                android:paddingLeft="1dp"
                android:textSize="10sp"
                android:textStyle="normal" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

Όπως βλέπουμε πρόκειται για ένα ιδιαίτερα πολύπλοκο αντικείμενο το οποίο όμως επειδή δημιουργήθηκε βήμα βήμα, δεν αποτέλεσε εμπόδιο. Χρησιμοποιήσαμε καταρχήν μια LinearLayout σαν ρίζα, στην οποία θέσαμε στην παράμετρο "`orientation`" την τιμή "*`horizontal`*" ώστε τα αντικείμενα μέσα της να στοιχίζονται οριζόντια αντί για κάθετα. Στη συνέχεια βάλαμε μέσα μια ImageView και δίπλα από αυτήν άλλη μια LinearLayout η οποία όμως έχει default προσανατολισμό "*`vertical`*". Αυτό σημαίνει ότι τα αντικείμενα που θα τοποθετηθούν μέσα θα έχουν κάθετη διάταξη δίπλα από τη εικόνα. Εμείς όμως θέλουμε να εμφανίζονται τρεις TextViews σε κάθε σειρά, δηλαδή οριζόντια. Δημιουργούμε λοιπόν άλλες δύο LinearLayouts την μία κάτω από την άλλη οι οποίες όμως θα έχουν "`orientation` =*`"horizontal"`*" και από τρεις TextViews η κάθε μία, οι οποίες θα εμφανίζονται κάθετα όπως επιθυμούμε.

3.9.3 Δημιουργία αντικειμένου για την λίστα των καθηγητών

Το αντικείμενο που θα χρησιμοποιήσουμε για την λίστα των μόνιμων καθηγητών είναι παρόμοιο με το παραπάνω, αλλά λίγο πιο απλό καθότι θα εμφανίσουμε μια εικόνα στο αριστερό μέρος και δίπλα της θα έχουμε τρεις κάθετες TextViews οι οποίες θα περιέχονται σε μια LinearLayout.

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

    <ImageView
        android:id="@+id/monimoi_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingRight="1dp" >
    </ImageView>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/teacher_name"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="16sp"
            android:textStyle="normal"
            android:textColor="#FD6600" />
        <TextView
            android:id="@+id/teacher_qualification"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="12sp"
            android:textStyle="normal" />
        <TextView
            android:id="@+id/teacher_title"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:paddingLeft="1dp"
            android:textSize="12sp"
            android:textStyle="normal"
            android:textColor="#125F15" />
    </LinearLayout>
</LinearLayout>
```

Το αντικείμενο λίστας των συνεργατών καθηγητών θα αποτελείτε μόνο από μία TextView, ουσιαστικά η πιο απλή μορφή λίστας, και γιαυτό θα το παραλείψουμε.

3.9.4 Δημιουργία layouts των πληροφοριών

Παραπάνω δημιουργήσαμε αντικείμενα λίστας τα οποία θα χρησιμοποιήσουμε στο επόμενο κεφάλαιο για να γεμίσουμε με κείμενο και εικονίδια τις λίστες μας. Η εφαρμογή όμως στην Activity “Plirofories” θα εμφανίζει 4 layouts τα οποία θα είναι απλό διαμορφωμένο κείμενο και όχι λίστες. Για να πετύχουμε το επιθυμητό αποτέλεσμα, θα δημιουργήσουμε ολόκληρο το κείμενο TextView προς TextView χρησιμοποιώντας κενές Views ενδιάμεσα για να δημιουργήσουμε κενά. Αφού ολοκληρώσουμε την υλοποίηση του layout το κάνουμε “inflate” στην οθόνη που θέλουμε να εμφανίζεται. Ο κώδικας της κενής View είναι ο εξής:

```
<View
        android:layout_width="fill_parent"
        android:layout_height="15dip"
        android:background="#000000" />
```

3.10 Προσθήκη πόρων συστήματος

Πλέον έχουμε ολοκληρώσει την κατασκευή του σκελετού της εφαρμογής και μας μένει η τελική υλοποίηση ενσωματώνοντας τον απαραίτητο κώδικα στις Activities αλλά και στις βοηθητικές κλάσεις που θα δημιουργήσουμε αργότερα. Αυτό που μας μένει να κάνουμε πριν να προχωρήσουμε εκεί, είναι να τροφοδοτήσουμε τους φακέλους drawable και values με τα απαραίτητα resources εικόνας και κειμένου. Τα αρχεία που χρειαζόμαστε λοιπόν θα ληφθούν ως επί το πλείστον από την ιστοσελίδα του τμήματος, αλλά και από τα βοηθητικά sites.

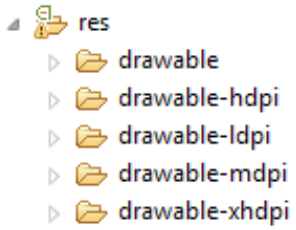
3.10.1 Επεξεργασία και προσθήκη κατάλληλων Drawables

Αρχικά θα πρέπει να τοποθετήσουμε το εικονίδιο της εφαρμογής μας. Το Android ζητάει η κάθε ανάλυση να έχει το δικό της εικονίδιο, αλλιώς στην θέση του εμφανίζει το προεπιλεγμένο εικονίδιο με το πράσινο ρομποτάκι. Επειδή δεν το θέλουμε αυτό θα κάνουμε resize το εικονίδιο της εφαρμογής μας για να τοποθετήσουμε την σωστή διάσταση εικονιδίου εφαρμογής στον σωστό φάκελο. Για τις διαστάσεις ισχύουν οι τιμές του παρακάτω πίνακα (Εικόνα 3.5).

	ldpi (120 dpi) (Low density screen)	mdpi (160 dpi) (Medium density screen)	hdpi (240 dpi) (High density screen)	xhdpi (320 dpi) (Extra-high density screen)
Launcher Icon Size	36 x 36 px	48 x 48 px	72 x 72 px	96 x 96 px

Εικόνα 3.5: Απαιτούμενη ανάλυση εικονιδίου εφαρμογής ανά ανάλυση οθόνης

Αφού ολοκληρώσαμε τη δημιουργία του εικονιδίου της εφαρμογής μας θα συνεχίσουμε με τα εικονίδια της κεντρικής μας Activity. Θα χρειαστούμε λοιπόν 6 εικονίδια τα οποία όμως θέλουμε να εμφανίζονται σωστά σε όλες τις διαστάσεις και για αυτό θα χρησιμοποιήσουμε τη δυνατότητα πολλαπλών resources που μας δίνει το Android. Θα δημιουργήσουμε λοιπόν 2 εκδόσεις των εικόνων μας, η πρώτη στα 80x80 pixels και η δεύτερη έκδοση στα 140x140 pixels. Η πρώτη έκδοση θα τοποθετηθεί στον φάκελο "drawable – mdpi" και η δεύτερη στον φάκελο "drawable-hdpi". Έτσι εξασφαλίζουμε ότι σε καμία περίπτωση οθόνης δεν θα έχουμε φαινόμενα cropping των εικονιδίων μας.



Εικόνα 3.6: Οι φάκελοι των drawables

Τα υπόλοιπα εικονίδια της εφαρμογής που θα χρειαστούμε για τα αντικείμενα λίστας, θα έχουν διαστάσεις 80x80 και θα τοποθετηθούν όλα στον κεντρικό φάκελο drawable. Ο λόγος που δεν ακολουθούμε την ίδια διαδικασία resize και διαμοιρασμού εικόνων στους αντίστοιχους φακέλους είναι γιατί τοποθετώντας τις εικόνες μας στον φάκελο drawable αφήνουμε στο Android να κάνει το απαραίτητο resize των εικόνων, όπου αυτό κρίνει ότι χρειάζεται. Για την κεντρική οθόνη επιλέχτηκε η λύση των πολλαπλών resources λόγω μη ικανοποιητικού resizing από το λειτουργικό στην πρώτη περίπτωση.

3.10.2 Εισαγωγή String Resources

Για το τέλος αφήσαμε την εισαγωγή των String Resources τα οποία αποθηκεύονται στον φάκελο Values. Οι πόροι αυτοί μπορεί να αποτελούνται από αλφαριθμητικά (Strings), πίνακες αλφαριθμητικών, ακέραιους, πίνακες ακεραίων, λίστες με κωδικούς χρωμάτων για εύκολη αναφορά κλπ. Στην περίπτωση μας θα χρησιμοποιήσουμε αρκετούς πίνακες για να γεμίσουμε με κείμενο τις λίστες που δημιουργήσαμε παραπάνω. Δημιουργούμε λοιπόν ένα νέο αρχείο xml τύπου "Values" και το ονομάζουμε "examino_A_arrays.xml". Μέσα σε αυτό το xml θα ενσωματώσουμε συνολικά 8 πίνακες οι οποίοι θα αποτελέσουν τον κορμό. Ένας

πίνακας με αλφαριθμητικά λοιπόν δηλώνεται ως εξής:

```
<string-array name="Examino_A_title">
        <item>Μαθηματική Ανάλυση 1 [130]</item>
        <item>Θεωρία Πιθανοτήτων &amp;amp; Στατιστική [131]</item>
        <item>Γραμμική Άλγεβρα [132]</item>
        <item>Ηλεκτρονικά &amp;amp; Τηλεπικοινωνίες [133]</item>
        <item>Προγραμματισμός Ι [134]</item>
        <item>Ξένη Γλώσσα Ι [941]</item>
    </string-array>
```

Με τον ίδιο τρόπο θα περάσουμε σε πίνακες όλες τις απαραίτητες τιμές για το γέμισμα του αντικειμένου λίστας του πρώτου εξαμήνου. Παρόμοια διαδικασία θα ακολουθήσει και στα υπόλοιπα 7 xmls (όσα και τα υπόλοιπα εξάμηνα). Στη συνέχεια θα δημιουργήσουμε τα xmls για τις λίστες των καθηγητών, όπου εκεί θα εισάγουμε πίνακες αλφαριθμητικών με τις απαραίτητες πληροφορίες όπως όνομα, email, κλπ.

Κεφάλαιο 4

Υλοποίηση της Εφαρμογής "CST Connect"

4.1 Υλοποίηση της κεντρικής οθόνης

Στο κεφάλαιο αυτό θα ασχοληθούμε με την υλοποίηση του κώδικα ο οποίος χρησιμοποιώντας τα υπάρχοντα layout και resources, θα συμπληρώσει το παζλ της εφαρμογής μας. Θα ξεκινήσουμε φυσικά από την υλοποίηση του κώδικα της Activity MainScreen η οποία αποτελεί τον κορμό της εφαρμογής. Αρχικά θα φτιάξουμε μια νέα κλάση για την προσθήκη των εικόνων και του κειμένου σε έναν custom adapter τον οποίο θα χρησιμοποιήσουμε στην MainScreen για να εμφανίσουμε τις εικόνες. Στη συνέχεια με τη χρήση Listeners και switch ελέγχων θα συνδέσουμε τα αντικείμενα μας με τις Activities που αντιστοιχούν, και τέλος θα προσθέσουμε μια μέθοδο για έλεγχο της κατάστασης δικτύου της συσκευής του χρήστη.

4.1.1 Δημιουργία κλάσης ImageAdapter

Δημιουργούμε λοιπόν μέσω του Eclipse μία νέα κλάση την οποία ονομάζουμε "ImageAdapter" και στις παραμέτρους της SuperClass δίνουμε την τιμή "android.widget.BaseAdapter", πατάμε finish και προχωράμε στην υλοποίηση της κλάσης. Ξεκινάμε λοιπόν με την δήλωση μιας μεταβλητής τύπου Context που θα την ονομάσουμε "mContext". Η δήλωση της μεταβλητής Context είναι αρκετά συνηθισμένη και γίνεται για να αναφερόμαστε στο τρέχον περιβάλλον της κάθε κλάσης. Η δήλωση γίνεται ως εξής:

```
private Context mContext;
```

Δίνουμε την παράμετρο "private" για να ορίσουμε ότι η μεταβλητή θα είναι ορατή μόνο από την συγκεκριμένη κλάση και όχι από τις υπόλοιπες. Στη συνέχεια θα δηλώσουμε έναν πίνακα ακεραίων η τιμές των οποίων θα αντιστοιχούν στις 6 εικόνες που θα εισάγουμε στην κεντρική μας οθόνη.

```
private Integer[] mThumbIds = { R.drawable.rss_icon, R.drawable.map_icon,
                R.drawable.book_icon, R.drawable.info_icon,
                R.drawable.link_icon,R.drawable.emails_icon };
```

και ακολουθεί η δήλωση του πίνακα αλφαριθμητικών ο οποίος θα περιέχει τις λέξεις η οποίες θα τοποθετηθούν κάτω από τις εικόνες.

```
private String[] mThumbIds_Strings = { "Ανακοινώσεις", "Χάρτης",
            "Μαθήματα", "Πληροφορίες", "Χρήσιμα Links", "Mobile Sites" };
```

Εφόσον χρησιμοποιούμε μια super class, πρέπει υποχρεωτικά να ενσωματώσουμε και τις βασικές τις μεθόδους της, ακόμη και αν δεν τις αξιοποιήσουμε. Εμείς θα χρησιμοποιήσουμε τις πρώτες 2 μεθόδους "**`ImageAdapter(Context c)`**" και "`getCount()`" τις οποίες θα καλέσουμε αργότερα από την "MainScreen".

```
public ImageAdapter(Context c) {

      this.mContext = c;
}
public int getCount() {
      return mThumbIds.length;
}
public Object getItem(int position) {
      return null;
}
public long getItemId(int position) {
      return 0;
}
```

Σειρά έχει η υλοποίηση της κύριας μεθόδου της κλάσης, μέσα στην οποία θα δηλώνουμε μια View με όνομα "gridView" και μέσω αυτής θα κάνουμε inflate, θα χρησιμοποιήσουμε δηλαδή, το layout "gridview_item" που δημιουργήσαμε παραπάνω, και το οποίο αποτελεί το αντικείμενο τις GridView με την οποία θα ασχοληθούμε στην Activity "MainScreen".

```
public View getView(int position, View convertView, ViewGroup parent) {
 View gridView;
 if (convertView == null) {
     LayoutInflater inflater = (LayoutInflater) mContext
     .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
     gridView = new View(mContext);
      gridView = inflater.inflate(R.layout.gridview_item,null);
} else {
      gridView = convertView;
}
      TextView textView = (TextView)
      gridView.findViewById(R.id.grid_item_text);
            textView.setText(mThumbIds_Strings[position]);
      ImageView imageView = (ImageView)
            gridView.findViewById(R.id.grid_item_image);
      imageView.setImageResource(mThumbIds[position]);
      return gridView;}
```

Η διαδικασία ξεκινάει με έναν έλεγχο της convertView και αν είναι κενή (null) τότε εμφανίζεται το layout. Αν δεν είναι κενή, τότε προχωράμε κανονικά στην δήλωση μιας TextView και μιας ImageView. Αυτές με τι σειρά τους καλούν το "*id*" του αντικειμένου που θέλουν να αντιστοιχήσουν και μετά γίνεται η αντιστοίχηση των πινάκων που δημιουργήσαμε παραπάνω, στα νέα μας GridView αντικείμενα. Τώρα μένει να τα προσθέσουμε μέσω ενός adapter στην "MainScreen".

4.1.2 Χρήση της ImageAdapter στην Activity Mainscreen

Ξεκινώντας την υλοποίηση της Activity "MainScreen", θα δηλώσουμε το layout το οποίο θέλουμε να χρησιμοποιεί η activity και δεν είναι άλλο από το "main.xml" στην αρχή της βασικής μεθόδου "onCreate" θα προσθέσουμε την εξής γραμμή:

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

Τώρα μας μένει να δηλώσουμε την Gridview και να συνδέσουμε τον ImageAdapter που έχουμε ήδη δημιουργήσει:

```
GridView gridview = (GridView) findViewById(R.id.gridview);
gridview.setAdapter(new ImageAdapter(this));
```

Η GridView είναι έτοιμη και πλέον τα εικονίδια εμφανίζονται κανονικά στην κεντρική μας οθόνη! Το πρώτο βήμα ολοκληρώθηκε αλλά έχουμε αρκετά ακόμη για την ολοκλήρωση της εφαρμογής. Καταρχήν πρέπει να συνδέσουμε τα εικονίδια με τις Activities που δημιουργήσαμε.

4.1.3 Σύνδεση των εικόνων τις GridView με Activities

Η σύνδεση των υπόλοιπων Activities με τα εικονίδια τις κεντρικής οθόνης θα γίνει με τη χρήση Intents. Πρώτα όμως θα πρέπει να ενσωματώσουμε έναν listener στην GridView ο οποίο θα αντιλαμβάνεται σε πια θέση του πλέγματος έχει γίνει κλικ και μέσω μιας switch θα ενεργοποιείται το ανάλογο κομμάτι κώδικα για την εκκίνηση των Activities. Η χρήση του listener γίνεται ως εξής:

```
gridview.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v,
                                    int position, long id) {
            switch (position)
}
```

Χρησιμοποιώντας λοιπόν την μέθοδο "**setOnItemClickListener**" θα "ακούμε" για γεγονότα κλικ στα εικονίδια του πλέγμα μας, και με την μέθοδο "`onItemClick`" θα περνάμε την μεταβλητή "`position`" μέσα στην switch η οποία με τη σειρά της θα εκτελεί τον αντίστοιχο κώδικα.

4.1.4 Δημιουργία παράθυρου AlertDialog και χρήση των Intents

Αφού δημιουργήσαμε τον μηχανισμό σύνδεσης της κεντρικής οθόνης με τις υπόλοιπες, συνεχίζουμε με την προσθήκη των παραθύρων διαλόγου τα οποία θα χρησιμοποιούν μερικά εικονίδια για να συνδέονται εν τέλει με πάνω από μία Activities, η θα περνάνε διαφορετικές παραμέτρους στην ίδια Activity μέσω Intents.

Ας τα πάρουμε με τη σειρά λοιπόν και ας ξεκινήσουμε με την περιγραφή λειτουργίας του εικονιδίου "Ανακοινώσεις". Το εικονίδιο θα συνδέεται με 2 Activities, την "Anakoinoseis" και την "TeachersList". Η πρώτη θα δέχεται τα rss feeds που θέλουμε να εμφανίσουμε και θα τα προβάλει σε λίστα, ενώ η δεύτερη είναι μια στατική λίστα με τα ονόματα και τα emails όλων των εκπαιδευτικών του τμήματος, τα οποία με κλικ παρέχουν στην Activity "Anakoinoseis" το απαραίτητο rss feed ώστε να γίνει το parsing και η εμφάνιση.

Θα χρειαστούμε λοιπόν ένα παράθυρο διαλόγου το οποίο να δίνει στον χρήστη τη δυνατότητα να επιλέξει πιο rss θέλει να δει, και μέσω Intent θα περνάει στην Activity "Anakoinoseis" το κατάλληλο rss. Αρχικά θα δηλώσουμε 2 μεταβλητές πριν την μέθοδο onCreate της "MainScreen" οι οποίες θα χρησιμοποιηθούν από όλα τα παράθυρα διαλόγου που θα φτιάξουμε. Οι μεταβλητές τύπου Intent θα χρησιμοποιηθούν για την εκκίνηση των επιθυμητών Activities.

```
Context context = this;
Intent intent, mobile;
AlertDialog.Builder builder = null;
AlertDialog alertDialog = null;
```

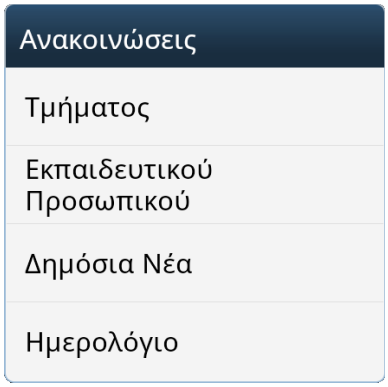
Το παράθυρο διαλόγου θα κατασκευαστεί μέσω ενός builder, και θα εμφανιστεί στον χρήστη μέσω της μεθόδου "show()" της `AlertDialog`. Η μεταβλητή context θα χρησιμοποιηθεί για να αναφερθούμε στην τρέχουσα οθόνη προβολής, όπως ακριβώς κάναμε και με την ImageAdapter παραπάνω. Στη συνέχεια θα δημιουργήσουμε έναν πίνακα λέξεων οι οποίες θα προβληθούν στην λίστα με τις επιλογές, και θα θέσουμε και έναν τίτλο στο παράθυρο μας. Μετά μέσω ενός listener θα πιάνουμε τα κλικ τους χρήστη και από εκεί μέσω Intent θα ξεκινάμε την επιθυμητή Activity.

```
CharSequence[] items = {"Τμήματος","Εκπαιδευτικού Προσωπικού",
                        "Δημόσια Νέα","Ημερολόγιο" };
AlertDialog.Builder builder = new AlertDialog.Builder(context);
builder.setTitle("Ανακοινώσεις");
builder.setItems(items,new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int item) {
}
```

Μέσα στις αγκύλες θα γίνεται έλεγχος με if της μεταβλητής "`item`" και αντίστοιχα θα ξεκινάει η επιθυμητή Activity με τις σωστές παραμέτρους. Η λέξη "Τμήματος" με τιμή item==0 θα εκτελεί τον παρακάτω κώδικα:

```
intent = new Intent("com.cst.connect.Anakoinoseis");
intent.putExtra("RSS","http://www.cs.teilar.gr/CS/rss.jsp");
intent.putExtra("title","Ανακοινώσεις Τμήματος");
startActivity(intent);
```

Το παραπάνω κομμάτι κώδικα δίνει τιμή στην μεταβλητή που δηλώσαμε παραπάνω και τις λέει να ξεκινήσει την Activity της οποίας το intent-filter έχουμε βάλει μέσα στα εισαγωγικά. Μετά μέσω της μεθόδου putExtra εισάγουμε 2 Strings στο σώμα του Intent για να περάσουν με τη σειρά τους στην επόμενη activity μέσω της μεθόδου εκκίνησης: "`startActivity(intent);`" Η διαδικασία λήψης των extras από το Intent θα περιγραφεί παρακάτω στην υλοποίηση της Activity "Anakoinoseis".



Εικόνα 4.1: Παράθυρο διαλόγου Ανακοινώσεων

Με τον ίδιο κώδικα αλλά αλλαγμένες μεταβλητές στις μεθόδους "putExtra" θα εκκινήσουμε την "Anakoinoseis" και στις υπόλοιπες περιπτώσεις εκτός από την 2 περίπτωση επιλογής στην οποία θα εκκινήσουμε την Activity "TeachersList". Η τελική εμφάνιση του παράθυρου διαλόγου φαίνεται στην εικόνα 4.1

4.1.5 Υλοποίηση κώδικα μενού και υπόλοιπων αντικειμένων

Ολοκληρώνοντας την υλοποίηση της Activity MainScreen θα δημιουργήσουμε άλλες 3 μεθόδους, την "`onCreateOptionsMenu(Menu menu)`", την "`onOptionsItemSelected (MenuItem item)`" και την "`isOnline()`" οι οποίες θα τοποθετηθούν κάτω από την "`onCreate`". Η πρώτη θα δημιουργεί το μενού των επιλογών, η δεύτερη θα δημιουργεί ένα παράθυρο διαλόγου με το πάτημα του μενού επιλογών, και η τρίτη θα ελέγχει την κατάσταση του δικτύου της συσκευής, και σε περίπτωση μη διαθέσιμης σύνδεσης θα εμφανίζει ένα σχετικό μήνυμα. Ας ξεκινήσουμε με την πρώτη:

```
public boolean onCreateOptionsMenu(Menu menu) {
      super.onCreateOptionsMenu(menu);
      getSupportMenuInflater().inflate(R.menu.menu_about, menu);
      return true;
}
```

Δημιουργούμε λοιπόν ένα αντικείμενο μενού και του αναθέτουμε να χρησιμοποιήσει το layout "*menu_about*". Στην επόμενη μέθοδο θα δημιουργήσουμε το παράθυρο διαλόγου το οποίο εμφανίζει κάποιες πληροφορίες μέσω ενός παράθυρου διαλόγου το οποίο θα χρησιμοποιεί επίσης τις μεταβλητές "builder" και "alertdialog" που δηλώσαμε παραπάνω.

```
public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case R.id.about:

            LayoutInflater inflater = (LayoutInflater) context
                              .getSystemService(LAYOUT_INFLATER_SERVICE);
            View layout = inflater.inflate(R.layout.custom_dialog,
                        (ViewGroup) findViewById(android.R.id.list),false);
            builder = new AlertDialog.Builder(context);
            builder.setView(layout);
```

Σε αυτό το σημείο θα χρησιμοποιήσουμε ένα κομμάτι κώδικα το οποίο θα βρίσκει την τρέχουσα έκδοση της εφαρμογής μας και θα την προσθέτει σε μια μεταβλητή τύπου String, ώστε να την εμφανίσουμε στην επικεφαλίδα του παραθύρου.

```
String versionName = null;

versionName = getPackageManager().getPackageInfo(
            getPackageName(), 0).versionName;
builder.setTitle("CST Connect v" + versionName);
builder.setIcon(android.R.drawable.ic_menu_info_details);
TextView text = (TextView) layout.findViewById(R.id.dialog_text);
```

Μετά χρησιμοποιώντας την μέθοδο "setText" θα εισάγουμε το κείμενο της επιλογής μας στο παράθυρο διαλόγου. Μετά θα χρησιμοποιήσουμε τις 2 διαθέσιμες μεθόδους δημιουργίας κουμπιών που διαθέτει ο builder και με την χρήση 2 listeners θα ανταποκριθούμε αναλόγως την επιλογή του χρήστη.

```
builder.setNeutralButton("Feedback",new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int id) {
      Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
      emailIntent.setType("html/text");
      emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,new String[]
      { "nikos.pard@gmail.com" });
      emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
      "CST Connect Feedback");
      startActivity(Intent.createChooser(emailIntent,"Αποστολή Email
      με..."));
      }
});

builder.setPositiveButton("Κλείσιμο",new DialogInterface.OnClickListener(){
      public void onClick(DialogInterface dialog, int id) {
            alertDialog.dismiss();
      }
});
```

Στην πρώτη περίπτωση δημιουργήσαμε ένα κουμπί feedback το οποίο μέσω ενός Intent στέλνει τα στοιχεία για αποστολή ενός email μέσω όποιας εφαρμογής προτιμάει ο χρήστης, και το δεύτερο κουμπί κλείνει το παράθυρο διαλόγου. Φυσικά δεν πρέπει να ξεχάσουμε να δημιουργήσουμε και να εμφανίσουμε το παράθυρο μας!

```
            alertDialog = builder.create();

            alertDialog.show();
```

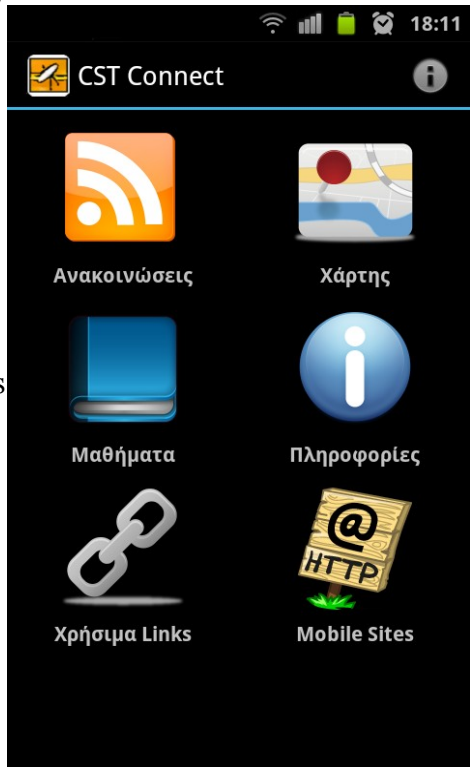
Η τελευταία μεθοδος όπως είπαμε θα ελέγχει την κατάσταση του δικτύου και θα επιστρέφει "**true**" ή "**false**" αντίστοιχα.

```
	private Boolean isOnline() {
		ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
		NetworkInfo ni = cm.getActiveNetworkInfo();
		if (ni != null && ni.isConnected())
			return true;
		return false;
	}
```

Χρησιμοποιώντας έναν έλεγχο με if else στα σημεία που επιθυμούμε, μπορούμε να ελέγχουμε αν η συσκευή του χρήστη έχει Ίντερνετ, και αν όχι θα του εμφανίζουμε ένα μήνυμα ειδοποίησης "toast" το οποίο θα τον ενημερώνει σχετικά και θα τον συμβουλεύει να ενεργοποιήσει μια σύνδεση δεδομένων και να δοκιμάσει αργότερα. Αυτό γίνεται εφικτό με τον παρακάτω κώδικα:

```
Toast.makeText(MainScreen.this,R.string.DataFailureWarning,
Toast.LENGTH_LONG).show();
```

Τελειώσαμε λοιπόν με την υλοποίηση της Activity "MainScreen" και το αποτέλεσμα είναι αυτό της εικόνας 4.2. Σκοπός της κεντρικής οθόνης κάθε εφαρμογής είναι να δίνει στον χρήστη να καταλάβει που βρίσκεται η κάθε πληροφορία που χρειάζεται. Πιστεύουμε ότι έγινε η καλύτερη δυνατή κατανομή των Activities στα εικονίδια του κεντρικού μας μενού. Το εικονίδιο των ανακοινώσεων δίνει πρόσβαση σε όλα τα διαθέσιμα feeds του τμήματος, το κουμπί των μαθημάτων εμφανίζει παράθυρο διαλόγου μέσω του οποίου μπορούμε να επιλέξουμε μεταξύ τις εμφάνισης των μαθημάτων και τις εμφάνισής των βαθμολογιών, το κουμπί πληροφορίες εμφανίζει παράθυρο διαλόγου μέσω του οποίου μπορούμε να επιλέξουμε μεταξύ των πληροφοριών των καθηγητών και του τμήματος, και τα υπόλοιπα εικονίδια έχουν προφανή χρήση και αποτέλεσμα.



Εικόνα 4.2: Αρχική οθόνη εφαρμογής "CST Connect"

4.2 Υλοποίηση των Ανακοινώσεων

Συνεχίζουμε την υλοποίηση της εφαρμογής μας με την Activity "Anakoinoseis", η οποία εκτός από πολύ σημαντική είναι και ιδιαίτερα σύνθετη στη λειτουργία της. Θα αναλύσουμε λοιπόν βήμα βήμα τη λειτουργία της και θα σχολιάσουμε εκτενώς τα δομικά της συστατικά. Η κύκλος λειτουργίας της Activity είναι ο εξής:

a) Η Activity λαμβάνει μέσω Intent το RSS που πρέπει να εμφανίσει
b) Στη συνέχεια γίνεται το parsing του RSS
c) Οι ανακοινώσεις ενημερώνουν την Βάση Δεδομένων
d) Γίνεται προβολή των ανακοινώσεων σε λίστα
e) Με κλικ πάνω σε μια ανακοίνωση ανοίγει παράθυρο διαλόγου με το κείμενο της ανακοίνωσης
f) Εφόσον υπάρχει διαθέσιμο συνημμένο αρχείο εμφανίζεται κουμπί για λήψη, ή για άνοιγμα του αρχείου εφόσον έχει κατέβει πρωτύτερα.
g) Σε περίπτωση λήψης νέου αρχείου εμφανίζεται ειδοποίηση στην μπάρα ειδοποιήσεων με το όνομα του αρχείο, και με κλικ πάνω στην ειδοποίηση ανοίγει το αρχείο με το κατάλληλο πρόγραμμα

4.2.1 Δημιουργία και χρήση του DOM parser

Μόλις ξεκινήσει η Activity, αμέσως τραβάμε το rss μέσω του Intent που την ξεκίνησε και το χρησιμοποιούμε για να δημιουργήσουμε μια νέα μεταβλητή τύπου "`HttpPost`" η οποία θα χρησιμοποιεί από την κλάση του parser:

```
Intent sender = getIntent();
String rss = sender.getExtras().getString("RSS");

httpPost = new HttpPost(rss);
```

Πλέον μπορούμε να χρησιμοποιήσουμε την μεταβλητή "***`httpPost`***" για να κάνουμε το parsing του rss μέσω της κλάσης "`ParseXMLmethods`" την οποία θα ενσωματώσουμε για ευκολία μέσα στην Activity "Anakoinoseis".Η κλάση θα αποτελείτε από 4 μεθόδους την "`getXML`" η οποία θα μας επιστρέφει το feed, την "*`XMLfromString`*" η οποία θα δέχεται σαν παράμετρο το feed και θα το σώζει σε ένα αρχείο τύπου Document, και τις μεθόδους

“getValue” και “getElementValue” οι οποίες θα χρησιμοποιηθούν για να ανακτήσουμε τα δεδομένα απο το RSS Feed. Ξεκινάμε λοιπόν με την “getXML()”:

```
public static String getXML() {

String line = null;
try {
      DefaultHttpClient httpClient = new DefaultHttpClient();
      HttpResponse httpResponse = httpClient.execute(httpPost);
      HttpEntity httpEntity = httpResponse.getEntity();
      line = EntityUtils.toString(httpEntity);
} catch (Exception e) {
      line = "Internet Connection Error >> " + e.getMessage();
}
      return line;
}
```

Πρόκειται για μια στάνταρ μέθοδο δημιουργίας σύνδεσης http και λήψης περιεχομένων πηγαίου κώδικα. Στη συνέχεια θα υλοποιήσουμε την μέθοδο “XMLfromString” η οποία επεξεργάζεται το feed που πήραμε μέσω της παραπάνω μεθόδου και επιστρέφει το υλικό που του παρέχουμε σε μορφή Nodes.

```
public final static Document XMLfromString(String xml) {
      Document doc = null;
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
      DocumentBuilder db = dbf.newDocumentBuilder();
      InputSource is = new InputSource();
      is.setCharacterStream(new StringReader(xml));
      doc = db.parse(is);
      return doc;
}
```

Δημιουργούμε μια μεταβλητή τύπου DocumentBuilder η οποία με τη σειρά της χρησιμοποιώντας τη μέθοδο “parse” μετατρέπει το feed σε Nodes τα οποία μετά εκχωρούνται στην μεταβλητή doc, η οποία και επιστρέφεται. Ακολουθεί η “getElementValue”.

```
public static String getElementValue(Node elem) {
      Node kid;
      if (elem != null) {
            if (elem.hasChildNodes()) {
                  for (kid = elem.getFirstChild(); kid != null;) {
                        return kid.getNodeValue();
                  }
             }
      }
      return ""
}
```

Η μέθοδος λαμβάνει σαν παράμετρο ένα αντικείμενο τύπου Node και αφού ελέγξει αν είναι κενό η όχι, επιστρέφει τα αντικείμενα "παιδιά" του. Την παραπάνω μέθοδο την δημιουργήσαμε για να χρησιμοποιηθεί από την τελευταία μέθοδο "`getValue`":

```
public static String getValue(Element item, String str) {
      NodeList n = item.getElementsByTagName(str);
      return ParseXMLmethods.getElementValue(n.item(0));    }
```

Αυτή είναι η κύρια μέθοδος που θα χρησιμοποιηθεί για το parsing των τμημάτων του xml. Παίρνει σαν παράμετρο το αντικείμενο από το οποίο θα χρησιμοποιήσει τους Nodes, και σαν δεύτερη παράμετρο, το όνομα του Node που θέλουμε να επιστρέψει. Πλέον είμαστε έτοιμο για να τραβήξουμε τα RSS που θέλουμε.

4.2.2 Λήψη RSS Feed μέσω AsyncTack

Αφού ολοκληρώσαμε την κατασκευή του parser μας, ήρθε η ώρα να τον χρησιμοποιήσουμε. Η διαδικασία λήψης των feeds θα γίνει μέσω της κλάσης "GetDataTask" η οποία χρησιμοποιεί σαν Super Class την "`AsyncTask<Void, Void, Integer>`". Η AsyncTask ουσιαστικά χρησιμοποιείτε όταν θέλουμε να κάνουμε μια διαδικασία να πραγματοποιηθεί στο background του UI, και στο τέλος να εμφανίσει τα επιθυμητά αποτελέσματα. Η μέθοδος στην οποία θα γίνει το parsing είναι η "`doInBackground`":

```
protected Integer doInBackground(Void... params) {
String xml = ParseXMLmethods.getXML();
Document doc = ParseXMLmethods.XMLfromString(xml);
      if (doc != null) {
            NodeList children = doc.getElementsByTagName("item");
            ZERO_FLAG = false;
      if (children.getLength() == 0) {
            ZERO_FLAG = true;
            publishProgress();
      }
      for (int i = 0; i < children.getLength(); i++) {
      HashMap<String, String> map = new HashMap<String, String>();
      Element e = (Element) children.item(i);
      map.put("title", ParseXMLmethods.getValue(e, "title"));
      map.put("pubDate", ParseXMLmethods.getValue(e, "pubDate"));
      map.put("link", ParseXMLmethods.getValue(e, "link"));
      map.put("description",ParseXMLmethods.getValue(e, "description"));
      localist.add(map);
}
```

Χρησιμοποιώντας τις μεθόδους του DOM parser ανακτάμε και εισάγουμε σε μία λίστα τύπου πίνακα (Arraylist<HashMap>), την localist, όλες τις τιμές του RSS Feed που μας ενδιαφέρουν ώστε να τις εμφανίσουμε αργότερα στην ListView μας. Επίσης στην αρχή κάνουμε και έναν έλεγχο αν υπάρχουν διαθέσιμα αντικείμενα "παιδιά" στο Node, και εμφανίζουμε ένα μήνυμα ότι δεν υπάρχουν διαθέσιμες ανακοινώσεις στο σενάριο που δεν υπάρχουν διαθέσιμα αντικείμενα.

4.2.3 Δημιουργία Βάσης Δεδομένων για αποθήκευση των ανακοινώσεων

Η εφαρμογή θα κάνει χρήση μιας βάσης δεδομένων SQLite για τοπική αποθήκευση των ανακοινώσεων που θα τραβάμε από τα RSS Feeds. Θα σχεδιάσουμε λοιπόν έναν πίνακα στον οποίο θα εισάγουμε όλες τις ανακοινώσεις. Για να μπορούμε να βρίσκουμε αυτές που θέλουμε βάση RSS, θα βάλουμε και μία στήλη "RSS" στην οποία θα αποθηκεύεται το url του feed έτσι ώστε με μια εντολή query θα παίρνουμε πίσω τις επιθυμητές ανακοινώσεις. Δημιουργούμε μια νέα κλάση την οποία θα ονομάσουμε "FeedsDbAdapter" και ξεκινάμε την υλοποίηση της ΒΔ, με τις δηλώσεις της ΒΔ και του πίνακα.

```
private static final String DEBUG_TAG = "RSSDatabase";
private static final int DB_VERSION = 1;
private static final String DB_NAME = "rss_data";

public static String TABLE = "list";
public static final String ID = "_id";
public static final String RSS = "_rss";
public static final String TITLE = "_title";
public static final String PUBDATE = "_pubdate";
public static final String DESCRIPTION = "_description";
public static final String LINK = "_link";
```

Αφού ολοκληρώσαμε την δήλωση της βάσης και του πίνακα συνεχίζουμε με την υλοποίηση τους. Κάνουμε μια νέα κλάση με όνομα "DatabaseHelper" η οποία θα έχει σαν SuperClass την "SQLiteOpenHelper". Μέσα στην μέθοδο "onCreate" θα γίνει η δημιουργία του πίνακα.

```
public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE " + TABLE + " ("+ ID + " integer PRIMARY KEY
AUTOINCREMENT, " + RSS + " text NOT NULL, " + TITLE + " text NOT NULL, " +
PUBDATE + " text NOT NULL, " + DESCRIPTION + " text NOT NULL, " + LINK + "
text NOT NULL);"); }
```

Μένει να δημιουργήσουμε και την μέθοδο “onUpgrade” για να ολοκληρωθεί η βοηθητική κλάση DatabaseHandler.

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS " + TABLE);
      onCreate(db);}
```

Συνεχίζοντας την δημιουργία της βάσης μας, θα δημιουργήσουμε μια μέθοδο η οποία θα ανοίγει την ΒΔ για εισαγωγή δεδομένων, μία μέθοδο που θα την κλείνει, μία που θα εισάγει νέες ανακοινώσεις, μία που θα ανανεώνει τις ήδη υπάρχουσες και μία που θα εμφανίζει τις επιλεγμένες ανακοινώσεις.

```
private DatabaseHelper mDbHelper;
private SQLiteDatabase mDb;

public FeedsDbAdapter open() throws SQLException {
      mDbHelper = new DatabaseHelper(mCtx);
      mDb = mDbHelper.getWritableDatabase();
      return this;
}

public void close() {
      mDbHelper.close();
}
```

Πλέον μπορούμε να χρησιμοποιήσουμε τις παραπάνω μεθόδους για να ανοίξουμε και να κλείσουμε την ΒΔ μας. Θα ολοκληρώσουμε την ΒΔ με την δημιουργία των μεθόδων δημιουργίας, ενημέρωσης, και προβολής των ανακοινώσεων.

```
public long createAnakoinosi(String rss, String title, String pubdate,
            String description, String link) {

            ContentValues values = new ContentValues();
            values.put(RSS, rss);
            values.put(TITLE, title);
            values.put(PUBDATE, pubdate);
            values.put(DESCRIPTION, description);
            values.put(LINK, link);

            return mDb.insert(TABLE, null, values);
      }
```

```
public boolean updateAnakoinosi(long rowId, String rss, String title,
            String pubdate, String description, String link) {

            ContentValues args = new ContentValues();
            args.put(RSS, rss);
            args.put(TITLE, title);
            args.put(PUBDATE, pubdate);
            args.put(DESCRIPTION, description);
            args.put(LINK, link);

            return mDb.update(TABLE, args, ID + "=" + rowId, null) > 0;}

public Cursor fetchAllAnakoinoseis() {

      return mDb.query(TABLE, new String[] { ID, RSS,  TITLE, PUBDATE,
                  DESCRIPTION,LINK }, null, null, null, null, null);    }
```

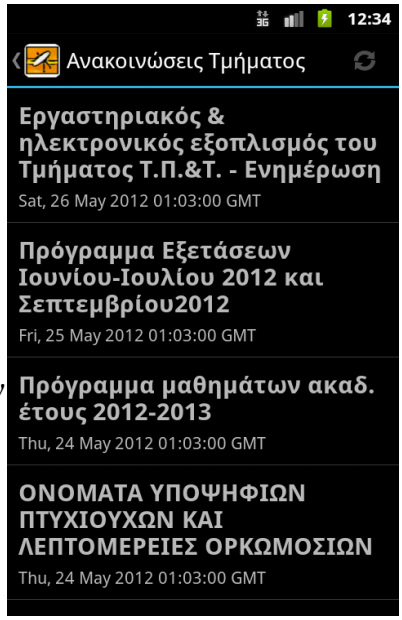
Η Βάση Δεδομένων μας είναι έτοιμη προς χρήση. Δεν θα ενσωματώσουμε μέθοδο διαγραφής ανακοινώσεων, γιατί αυτό δεν προβλέπεται από την λειτουργία της εφαρμογής.

4.2.4 Εμφάνιση των Ανακοινώσεων στην ListView

Αφού ολοκληρώσαμε την λήψη των ανακοινώσεων σε προηγούμενο βήμα, μένει να εισάγουμε την λίστα που δημιουργήσαμε στην ListView της Activity “Anakoinoseis”. Θα το πραγματοποιήσουμε εύκολα με τη χρήση ενός SimpleAdapter.

```
SimpleAdapter adapter = new SimpleAdapter(this,
 localist, R.layout.rss_list_item,
new String[] {"title","pubDate"},
new int[] {R.id.title, R.id.pubDate });
final ListView lv = (ListView)
findViewById(android.R.id.list);
lv.setAdapter(adapter);
```

Στις παραμέτρους της SimpleAdapter, θα ορίσουμε σαν προτεινόμενο layout το αντικείμενο λίστας που είχαμε δημιουργήσει για να εισάγουμε τον τίτλο και την ημερομηνία της κάθε ανακοίνωσης. Στη συνέχεια δημιουργούμε μια ListView και χρησιμοποιούμε τον adapter για να εμφανίσουμε τις ανακοινώσεις. Το τελικό αποτέλεσμα φαίνεται στην εικόνα 4.3



Εικόνα 4.3: Λίστα Ανακοινώσεων Τμήματος

4.2.5 Δημιουργία παράθυρου διαλόγου για προβολή ανακοίνωσης

Αφού εμφανίσουμε την λίστα μας πρέπει να δημιουργήσουμε ένα παράθυρο διαλόγου μέσα στο οποίο να προβάλεται το κείμενο της ανακοίνωσης. Ξεκινάμε με ένα listener στην ListView ο οποίος θα ενεργοποιεί το παράθυρο διαλόγου.

```
lv.setOnItemClickListener(new OnItemClickListener() {
public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {}
```

θα χρησιμοποιήσουμε μια μεταβλητή τύπου HashMap για να πάρουμε το κείμενο της ανακοίνωσης από τη localist που εισάγαμε στην ListView.

```
final HashMap<String, String> o =
      (HashMap<String, String>)lv.getItemAtPosition(position);
String message = o.get("description").replaceAll(" ", "");
```

Παίρνουμε λοιπόν το αντικείμενο “description” από την λίστα και χρησιμοποιώντας την μέθοδο “replace all” κάποιους από τους περιττούς χαρακτήρες οι οποίοι πέρασαν στο κείμενο μέσω του DOM parser. Στη συνέχεια θα χρησιμοποιήσουμε τα 2 custom layouts που έχουμε δημιουργήσει για να εμφανίσουμε το παράθυρο διαλόγου.

```
LayoutInflater inflater = (LayoutInflater)
      mContext.getSystemService(LAYOUT_INFLATER_SERVICE);
      View layout = inflater.inflate(R.layout.custom_dialog,
            (ViewGroup) findViewById(android.R.id.list), false);
      View title_layout = inflater.inflate(R.layout.custom_title,null);
```

Θα τοποθετήσουμε στην TextView του κάθε layout, το κείμενο που θέλουμε να εμφανίζεται.

```
      TextView title = (TextView)
      title_layout.findViewById(R.id.dialog_title);
      title.setText(Html.fromHtml
      ("<a href=" + o.get("link") + ">"+ o.get("title") + "</a>"));

      TextView text = (TextView) layout.findViewById(R.id.dialog_text);
      text.setText(XMLcleansing(message, rss));
      Linkify.addLinks(text, Linkify.ALL);
```

Κάνουμε χρήση της μεθόδου XMLCleansing για να αφαιρέσουμε τους χαρακτήρες μορφοποίησης από το κείμενο των ανακοινώσεων. Η μέθοδος αυτή δεν υπάρχει ακόμη, θα την δημιουργήσουμε όμως σε επόμενο βήμα. Συνέχεια παίρνει η δημιουργία του παράθυρου διαλόγου.

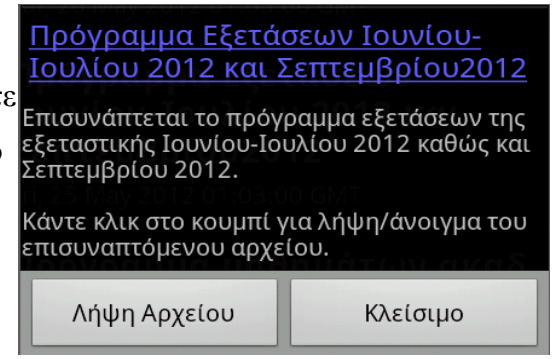
```
builder = new AlertDialog.Builder(mContext);

builder.setView(layout);
builder.setCustomTitle(title_layout);
```

Τα κουμπιά που εμφανίζονται στο παράθυρο διαλόγου είναι συνδεδεμένα με listeners για να ανταποκρίνονται στις ανάγκες μας. Δημιουργούμε λοιπόν ένα κουμπί για το κλείσιμο του παραθύρου και για το δεύτερο κουμπί κάνουμε έναν έλεγχο if else μέσω του οποίου θα ελέγχουμε αν το αρχείο πρέπει να ληφθεί οπότε το κουμπί θα γράφει "λήψη αρχείου", αλλιώς θα γράφει "Άνοιγμα Αρχείου" και θα ανοίγει με κλικ το αρχείο εφόσον αυτό είναι διαθέσιμο.

```
builder.setNeutralButton("Κλείσιμο",new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int id) {
            alertDialog.dismiss();
      }
});

if (flag == true) {
      File download = new File("/sdcard/CST Connect Downloads/"+ filename);
if (download.exists()) {
      FILE_EXISTS = true;
} else {
      FILE_EXISTS = false;
}
if (!FILE_EXISTS) {

builder.setPositiveButton("Λήψη Αρχείου",
      new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog,int id) {
      if (isOnline()) {
            startDownload();
      } else {
      Toast.makeText(Anakoinoseis.this,R.string.DataFailureWarning,
      Toast.LENGTH_LONG).show();
      }
}

private void startDownload() {
            new DownloadFileAsync().execute(url);
}
});
```

```
} else if (FILE_EXISTS) {
      builder.setPositiveButton("Άνοιγμα Αρχείου",
      new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int id) {
                  openFile();
            }
      });
}
```

Κάναμε χρήση 2 μεθόδων που δεν έχουμε δημιουργήσει ακόμη, η μία είναι η "execute" της κλάσης DownloadFileAsync η οποία χρησιμοποιείτε για να κατεβάσουμε το συνημμένο αρχείο και να το αποθηκεύσουμε, και η "openFile()" η οποία μας βοηθάει να ανοίξουμε το αρχείο με το κατάλληλο πρόγραμμα. Η τελική μορφή του παράθυρου διαλόγου μαζί με τον τίτλο, το κείμενο και τα κουμπιά φαίνεται στη εικόνα 4.4



Εικόνα 4.4: Παράθυρο Διαλόγου Ανακοίνωσης

4.2.6 Χρήση μεθόδου για καθαρισμό των ανακοινώσεων

Οι ανακοινώσεις των Εκπαιδευτικών αρκετές φορές περιέχουν διάφορα "σκουπίδια" τα οποία προέρχονται από την μορφοποίηση του Word η άλλων προγραμμάτων. Στον κώδικα html που προβάλει η σελίδα του τμήματος, η διαμόρφωση φαίνεται κανονικά, στην λήψη του κειμένου μέσω του DOM parser όμως περνάνε όλοι οι χαρακτήρες διαμόρφωσης και αυτό κάνει όσες ανακοινώσεις περιέχουν τέτοιους χαρακτήρες μορφοποίησης, αρκετά δυσανάγνωστες. Η λύση στο πρόβλημα είναι ο καθαρισμός του κειμένου των ανακοινώσεων με την χρήση Regular Expressions.

Πρόκειται για εκφράσεις οι οποίες είναι ρυθμισμένες να ταιριάζουν σε μοτίβα μορφοποίησης τα οποία συναντάμε στο κείμενο που θέλουμε να διορθώσουμε. Αφού ταιριάξει το μοτίβο με ένα κομμάτι του κειμένου, επιλέγουμε να το σβήσουμε και να καθαρίσουμε το συγκεκριμένο κομμάτι κειμένου. Ακολουθεί παράδειγμα με τον καθαρισμό των ανακοινώσεων του τμήματος.

```
if (rss.equals("http://www.cs.teilar.gr/CS/rss.jsp")) {
```

```
if (message.contains("white\">")) {
      String pattern = "(?i)(?s)(<span style)(.+?)(white\">)";
      message = message.replaceAll(pattern, "");
      flag = false;
}
```

Συνολικά χρησιμοποιήσαμε πάνω από 20 regular expressions για να καθαρίσουμε όλες τις ανακοινώσεις. Εάν οι καθηγητές ανεβάζαν καθαρό το κείμενο χωρίς καμία μορφοποίηση δεν θα υπήρχε αυτό το πρόβλημα, παρόλα αυτά εμείς έπρεπε να λύσουμε το πρόβλημα και το κάναμε. Επίσης στην μέθοδο γίνεται έλεγχος για ύπαρξη συνημμένου αρχείου και προσθήκη του ονόματος του και του link λήψης του σε δύο μεταβλητής τύπου String.

```
if (message.contains("<BR />")) {

      int start = message.indexOf("HREF='") + 6;
      int start_file = message.indexOf("News/") + 5;
      int end = message.indexOf("' title");
      filename = message.substring(start_file, end);
      url = message.substring(start, end);

      String pattern = "(?i)(<BR />)(.+?)(</A>)";
      message = message .replaceAll(pattern,
"\n\nΚάντε κλικ στο κουμπί για λήψη/άνοιγμα του επισυναπτόμενου αρχείου.");
      flag = true;
} else {
      flag = false;
}
```

Η μεταβλητή flag χρησιμοποιείτε για τον έλεγχο εμφάνισης του αντίστοιχου κουμπιού στο παράθυρο διαλόγου της ανακοίνωσης.

4.2.7 Δημιουργία μεθόδου για λήψη συνημμένου αρχείου

Σε αυτό το κομμάτι τις διαδικασίας θα φτιάξουμε μία κλάση με το όνομα DownloadFileAsync η οποία θα έχει σαν SuperClass την "`AsyncTask<String, String, String>`". Αυτό το κάνουμε και πάλι, όπως και στην περίπτωση της λήψης των ανακοινώσεων, γιατί θέλουμε να μην απασχολήσουμε το κύριο UI thread μας με την λήψη του αρχείου, αλλά να το κάνουμε στο background. Θα ξεκινήσουμε με την δημιουργία της σύνδεσης και το άνοιγμα του φακέλου που θα αποθηκεύονται τα αρχεία. Σε περίπτωση που ο φάκελος δεν υπάρχει, θα τον δημιουργήσουμε εκ νέου.

```
URL url = new URL(aurl[0]);
```

```
URLConnection connection = url.openConnection();
      connection.connect();
      int lenghtOfFile = connection.getContentLength();
      File downloadsDirectory = new File("/sdcard/CST Connect Downloads/");
      if (!downloadsDirectory.exists()) {
      downloadsDirectory.mkdir();
}
```

Στη συνέχεια ανοίγουμε το link του url και δημιουργούμε το αρχείο χρησιμοποιώντας την μεταβλητή filename που πήραμε από την μέθοδο καθαρισμού του κειμένου

```
InputStream input = new BufferedInputStream(url.openStream());
OutputStream output = new FileOutputStream(
      "/sdcard/CST Connect Downloads/" + filename);
```

Με τον παρακάτω κώδικα θα δημιουργήσουμε ένα παράθυρο διαλόγου το οποίο θα δείχνει την πρόοδο του αρχείου την ώρα που αυτό θα κατεβαίνει. Τέλος θα κλείσουμε το αρχείο που δημιουργήσαμε.

```
      byte data[] = new byte[1024];
      long total = 0;
      while ((count = input.read(data)) != -1) {
            total += count;
            publishProgress("" + (int) ((total * 100) / lenghtOfFile));
            output.write(data, 0, count);
      }
      output.flush();
      output.close();
      input.close();
```

4.2.8 Εμφάνιση ειδοποίησης στην Notification Bar και άνοιγμα αρχείου

Το αρχείο αφού κατέβει θα εμφανίζει μια ειδοποίηση στην notification bar. Από εκεί ο χρήστης κάνοντας κλικ μπορεί να το ανοίξει χρησιμοποιώντας το ανάλογο πρόγραμμα το οποίο επιλέγεται βάση ελέγχου. Ξεκινάμε λοιπόν δημιουργώντας μια νέα ειδοποίηση, της αναθέτουμε εικονίδιο, και τις δίνουμε το κείμενο που θα εμφανίσει μόλις δημιουργηθεί.

```
      String ns = Context.NOTIFICATION_SERVICE;
```

```
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(ns);
int icon = R.drawable.notification_icon;
CharSequence tickerText = "Λήψη αρχείου ολοκληρώθηκε";
long when = System.currentTimeMillis();
Notification notification = new Notification(icon, tickerText, when);
```

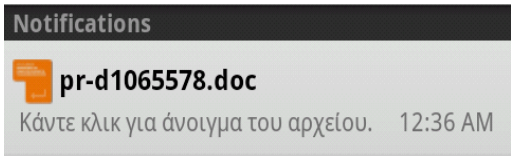
Σειρά παίρνουν οι ρυθμίσεις της ειδοποίησης μαζί με το όνομα του αρχείου που θα εμφανίζει και το κείμενο δίπλα από αυτό.

```
notification.defaults |= Notification.DEFAULT_SOUND;
notification.flags |= Notification.FLAG_AUTO_CANCEL;
CharSequence contentTitle = filename;
CharSequence contentText = "Κάντε κλικ για άνοιγμα του αρχείου.";
```

Στη συνέχεια θα δημιουργήσουμε τον απαραίτητο κώδικα για άνοιγμα του αρχείου μέσω Intent. Θα γίνεται ανίχνευση του τύπου του και αυτόματη επιλογή του σωστού ώστε να προταθούν στον χρήστης τα σωστά προγράμματα ανοίγματος.

```
Intent notificationIntent = new Intent();
notificationIntent.setAction(android.content.Intent.ACTION_VIEW);
File file = new File("/sdcard/CST Connect Downloads/" + filename)
MimeTypeMap mime = MimeTypeMap.getSingleton();
String ext = file.getName().substring(file.getName().indexOf(".")+1);
String type = mime.getMimeTypeFromExtension(ext);
notificationIntent.setDataAndType(Uri.fromFile(file), type);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
            notificationIntent, 0);
notification.setLatestEventInfo(this, contentTitle, contentText,
            contentIntent);
mNotificationManager.notify(LIST_UPDATE_NOTIFICATION, notification);
```

Η ειδοποίηση μας πλέον είναι έτοιμη και θα εμφανίζεται όπως στην εικόνα 4.5



Εικόνα 4.5: Ειδοποίηση Λήψης Αρχείου

Τέλος τα δημιουργήσουμε την μέθοδο

openFile() με την οποία θα ανοίγουμε το αρχείο όποτε αυτό ζητηθεί.

```
public void openFile() {
 Intent fileIntent = new Intent();
fileIntent.setAction(android.content.Intent.ACTION_VIEW);
 File file = new File("/sdcard/CST Connect Downloads/" + filename);
MimeTypeMap mime = MimeTypeMap.getSingleton();
 String ext = file.getName().substring(file.getName().indexOf(".")+1);
String type = mime.getMimeTypeFromExtension(ext);
fileIntent.setDataAndType(Uri.fromFile(file), type);
startActivity(fileIntent);
}
```

4.2.9 Δήλωση αδειών χρήστη στο AndroidManistest

Η υλοποίηση της Activity έχει ολοκληρωθεί και πλέον έχουμε έναν πλήρη μηχανισμό λήψης, αποθήκευσης και προβολής των ανακοινώσεων του τμήματος. Για να λειτουργήσουν όλα τα αντικείμενα που δημιουργήσαμε όμως, χρειάζονται κάποιες άδειες χρήσης του συστήματος, οι οποίες θα μας δώσουν την δυνατότητα να:

- έχουμε πρόσβαση στο Ίντερνετ
- να ελέγχουμε την κατάσταση του δικτύου
- να αποθηκεύσουμε αρχεία στην κάρτα SD της συσκευής

Θα δηλώσουμε λοιπόν τις τρεις παραπάνω άδειες στο αρχείο AndroidManifest.xml

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

Αυτές οι τρεις άδειες θα ζητηθούν από το σύστημα, και ο χρήστης θα ενημερωθεί για αυτές κατά την εγκατάσταση της εφαρμογής.

4.3 Χρήση των Google maps για την υλοποίηση του χάρτη

Συνεχίζουμε την υλοποίηση των Activities με την "CSMapView" η οποία θα μας εμφανίζει τον χάρτη του τμήματος με τα σημεία ενδιαφέροντος. Στην activity εκτός από την προβολή του χάρτη θα χρησιμοποιήσουμε μεθόδους εντοπισμού θέσης της συσκευής του χρήστη έτσι ώστε να παρέχουμε οδηγίες πλοήγησης στον χώρο του TEI χρησιμοποιώντας το API των Google Maps. Πρώτα όμως τα πρέπει να λάβουμε ένα κλειδί από την Google για την χρήση του API.

4.3.1 Λήψη κλειδιού από την Google για χρήση του API

Για να δουλέψει η MapView που χρησιμοποιήσαμε στο layout του χάρτη, πρέπει να τις παρέχουμε ένα κλειδί χρήσης του API των χαρτών. Το κλειδί μας το παρέχει η Google μέσω της σελίδας https://developers.google.com/android/maps-api-signup η οποία μας ζητάει να αποδεχτούμε τους όρους χρήσης του API, και να παρέχουμε το md5 fingerprint του αρχείου keytool το οποίο βρίσκεται στον φάκελο που έχουμε εγκαταστήσει το JAVA SDK στο λειτουργικό μας σύστημα. Αφού εντοπίσουμε το αρχείο, χρησιμοποιούμε την εντολή "keytool -list -keystore .keystore" στην γραμμή εντολών, για να πάρουμε το md5 fingerprint του αρχείου. Στη συνέχεια πηγαίνουμε στην ιστοσελίδα του παραπάνω συνδέσμου, και παρέχοντας το στο σχετικό πεδίο, μας παρέχεται το κλειδί του API.

Το κλειδί είναι ένα αλφαριθμητικό 40 χαρακτήρων το οποίο είναι καλό να αποθηκεύσουμε σε ένα String resource και να του δώσουμε ένα όνομα αναφοράς, πχ "maps_api_key" ώστε να το έχουμε διαθέσιμο προς χρήση στο layout των χαρτών. Αν το κλειδί μας είναι λανθασμένο τότε οι χάρτες δεν θα εμφανίζονται στην MapView

4.3.2 Ενημέρωση του AndroidManifest

Για να χρησιμοποιήσουμε την βιβλιοθήκη των Google Maps, πρέπει να δηλώσουμε την χρήση της στο αρχείο AndroidManifest.xml, γιατί δεν αποτελεί μέρος του κεντρικού API. Εισάγουμε λοιπόν την παρακάτω γραμμή.

```
<uses-library android:name="com.google.android.maps" />
```

Επίσης θα χρειαστούμε άδεια χρήσης του GPS και του εντοπισμού βάση ασυρμάτων δικτύων, για την μέθοδο εντοπισμού θέσης της εφαρμογής μας.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-feature android:name="android.hardware.location.gps" />
```

4.3.3 Χρήση της βιβλιοθήκης MapViewBalloons

Έχοντας σημειώσει στον χάρτη μας όλα τα σημεία ενδιαφέροντος, θα προσθέσουμε μια βιβλιοθήκη η οποία μας δίνει την δυνατότητα να εμφανίσουμε ένα μπαλονάκι με το όνομα του σημείου ενδιαφέροντος μόλις ο χρήστης κάνει "tap" με το δάχτυλο του σε αυτό. Θα χρησιμοποιήσουμε την βιβλιοθήκη "MapView Ballons" η οποία έχει αναπτυχθεί από τον Jeff Gilfelt και είναι διαθέσιμη για χρήση στο github, υπό την Apache License 2.0.

Η ενσωμάτωση της βιβλιοθήκης γίνεται με τον ίδιο τρόπο που ενσωματώσαμε και τις δύο προηγούμενες βιβλιοθήκες (ActionBarSherlock & ViewPagerIndicator) δημιουργώντας ένα νέο project βάση του υπάρχοντα κώδικα της βιβλιοθήκης, και εισάγοντας την βιβλιοθήκη στην λίστα με τις βιβλιοθήκες αναφοράς του project μας.

4.3.4 Υλοποίηση και εισαγωγή σημείων ενδιαφέροντος στον χάρτη

Είμαστε έτοιμοι λοιπόν να χρησιμοποιήσουμε το API των χαρτών. Θα ξεκινήσουμε με μια μικρή τροποποίηση στην SuperClass της Activity μας. Οι Activities που χρησιμοποιούν MapView στο layout τους πρέπει να είναι τύπου "MapActivity". Στην περίπτωση μας επειδή θέλουμε να χρησιμοποιήσουμε και την ActionBar μαζί με τον χάρτη η MapActivity θα γίνει "SherlockMapActivity".

```
public class CSMapView extends SherlockMapActivity
```

Στη συνέχεια θα χρησιμοποιήσουμε το layout της MapView, και θα ρυθμίσουμε τον

χάρτη μας να χρησιμοποιεί την απεικόνιση δορυφόρου με ενεργοποιημένα τα χειριστήρια zoom.

```
mapView = (MapView) findViewById(R.id.mapview);

mapView.setBuiltInZoomControls(true);
mapView.setSatellite(true);
```

Μέσω του mapcontroller και ρυθμίζοντας το αρχικό zoom, θα εστιάσουμε τον χάρτη μας ακριβώς πάνω από τον χώρο του ΤΕΙ.

```
mapController = mapView.getController();

mapController.setCenter(new GeoPoint((int) (39.6275f * 1e6),
                  (int) (22.3821f * 1e6)));
mapController.setZoom(18);
```

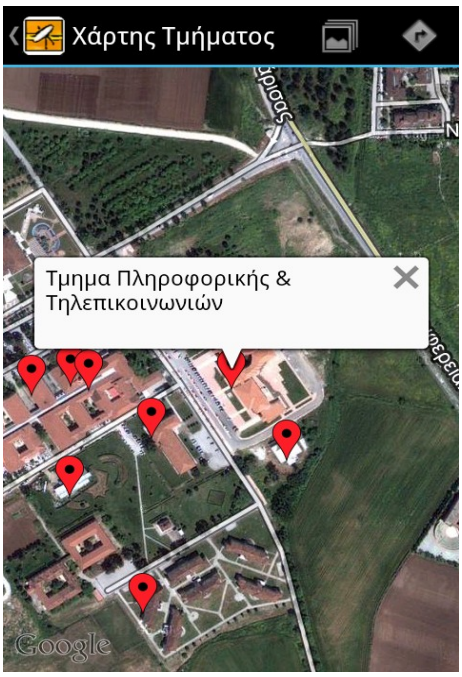
Τα σημεία ενδιαφέροντος δεν είναι δυνατόν να τοποθετηθούν απευθείας πάνω στον χάρτη, γιαυτό θα χρησιμοποιήσουμε μια κλάση τύπου overlay, που θα τα εμφανίζει.

```
CSOverlay csoverlay = new CSOverlay();

List<Overlay> overlayList = mapView.getOverlays();

overlayList.add(csoverlay);
```

Τέλος θα δηλώσουμε όλα τα αντικείμενα με τις συντεταγμένες τους στον χάρτη, ώστε να τα εμφανίσουμε στον χάρτη χρησιμοποιώντας ένα γραφικό το οποίο έχουμε εισάγει στον φάκελο drawables.

```
GeoPoint CST = new GeoPoint((int) (39.627604f * 1e6),

                  (int) (22.383723f * 1e6));
OverlayItem overlayItem_CST = new OverlayItem(CST,
            "Τμημα Πληροφορικής & Τηλεπικοινωνιών", "");


mapOverlays = mapView.getOverlays();
drawable = this.getResources().getDrawable(R.drawable.map_pin_mini);
itemizedOverlay = new MapItems(drawable, mapView);
itemizedOverlay.addOverlay(overlayItem_CST);
```

Επαναλαμβάνουμε τον παραπάνω κώδικα για όλα τα σημεία ενδιαφέροντος, δηλώνοντας κάθε φορά και νέα μεταβλητή τύπου “GeoPoint” και νέα τύπου “OverlayItem” με τις αντίστοιχες συντεταγμένες. Η τελική μορφή της Activity μας φαίνεται στην εικόνα 4.6



Εικόνα 4.6: Προβολή Χάρτη Τμήματος

4.3.5 Χρήση εντοπισμού θέσης για πλοήγηση

Η τελευταία δυνατότητα που θα προσθέσουμε στην Activity "CSMapView", είναι ο εντοπισμός της θέσης του χρήστη και η δυνατότητα λήψης οδηγιών πλοήγησης, εφόσον ο εντοπισμός θέσης είναι επιτυχής. Θα ξεκινήσουμε αλλάζοντας λίγο την δήλωση της κλάσης μας.

```
public class CSMapView extends SherlockMapActivity implements
LocationListener
```

Πλέον η Activity μας θα μπορεί να ανταποκρίνεται στην αλλαγή θέσης της συσκευής. Συνεχίζουμε με την δημιουργία των τριών μεθόδων η οποίες θα μας επιτρέψουν να εντοπίσουμε τη θέση της συσκευής. Ξεκινάμε με την μέθοδο "getBestProvider" η οποία θα ελέγξει ποιες μέθοδοι είναι διαθέσιμες και αναλόγως θα θέσει τα κριτήρια εντοπισμού. Εαν είναι ενεργό το GPS, ή ο εντοπισμός μέσω δικτύων WiFi, θα επιλέξει το GPS, αλλιώς όποιο από τα δύο είναι διαθέσιμα. Σε περίπτωση που κανέναν δεν είναι διαθέσιμο θα παραπέμψουμε τον χρήστη να ενεργοποιήσει τις μεθόδους εντοπισμού της συσκευής του.

```
public String getBestProvider() {
```

```
        String bestProvider = null;
        locationManager = (LocationManager)
              getSystemService(Context.LOCATION_SERVICE);
        gps_enabled = locationManager
                          .isProviderEnabled(LocationManager.GPS_PROVIDER);
        network_enabled = locationManager
                    .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        Criteria criteria = new Criteria();
        criteria.setPowerRequirement(Criteria.POWER_MEDIUM);
        if (!network_enabled && !gps_enabled) {
              Toast.makeText(getApplicationContext(),
        "Δεν έχει ενεργοποιηθεί καμία υπηρεσία εντοπισμού.",
              Toast.LENGTH_SHORT).show();

        } else if (network_enabled && !gps_enabled) {
              criteria.setAccuracy(Criteria.ACCURACY_COARSE);
              bestProvider = locationManager.getBestProvider(criteria, true);
        } else if ((gps_enabled && !network_enabled)
                    || (network_enabled && gps_enabled)) {
              criteria.setAccuracy(Criteria.ACCURACY_FINE);
              bestProvider = locationManager.getBestProvider(criteria, true);
              }

              return bestProvider;
        }
```

Η επόμενη μέθοδος είναι η "`setCurrentLocation`" η οποία όταν καλείτε ενημερώνει την μεταβλητή τρέχουσας θέσης.

```
public void setCurrentLocation(Location location) {
      int currLatitude = (int) (location.getLatitude() * 1E6);
      int currLongitude = (int) (location.getLongitude() * 1E6);
      currentPoint = new GeoPoint(currLatitude, currLongitude);
      currentLocation = new Location("");
      currentLocation.setLatitude(currentPoint.getLatitudeE6() / 1e6);
      currentLocation.setLongitude(currentPoint.getLongitudeE6() / 1e6);
}
```

Τέλος η "`getLastLocation()`" χρησιμοποιώντας τις δύο παραπάνω μεθόδους, ενημερώνει τη θέση της συσκευής ώστε να είναι εφικτή η παροχή οδηγιών πλοήγησης.

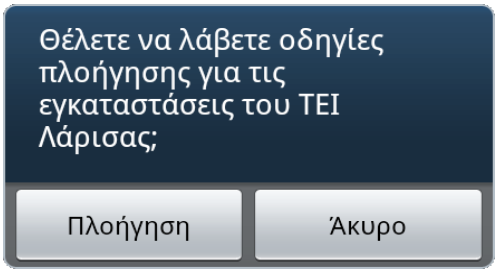
```
public void getLastLocation() {
      provider = getBestProvider();
      currentLocation = locationManager.getLastKnownLocation(provider);
      }
}
```

4.3.6 Υλοποίηση και χρήση της μεθόδου navigate()

Χρησιμοποιώντας ως παράμετρο εισαγωγής την μεταβλητή "currentLocation" θα δημιουργήσουμε την μέθοδο "navigate" η οποία όταν καλείτε θα μας δίνει τη δυνατότητα να λάβουμε οδηγίες πλοήγησης από την τρέχουσα τοποθεσίας μας, στον χώρο του ΤΕΙ.

```
public void navigate(GeoPoint currentPoint) {
        String current;
        Intent intent;

        String CST = "39.627604,22.383723";
        current = String.valueOf(currentPoint);
        current = current.substring(0, 2) + "." + current.substring(2, 11)
                        + "." + current.substring(11, 17);

        intent = new Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://maps.google.com/maps?saddr=
                    "+current +" &daddr=" + CST));
        startActivity(intent);
}
```

Το παράθυρο διαλόγου που θα εμφανίζεται όταν ο χρήστης ζητάει οδηγίες πλοήγησης, απεικονίζεται στην εικόνα 4.7



Εικόνα 4.7: Παράθυρο διαλόγου λήψης οδηγιών πλοήγησης

4.4 Υλοποίηση παρόμοιων Activities

Στην εφαρμογή κάνουμε χρήση των ίδιων layout σε αρκετές Activities. Σε αυτή τη φάση υλοποίησης της εφαρμογής θα δημιουργήσουμε κώδικα ο οποίος θα επαναληφθεί αρκετές φορές σε διάφορες Activities για την λήψη του τελικού αποτελέσματος. Οι παρόμοιες Activities που θα υλοποιήσουμε είναι η "Mathimata", η "Plirofories" και η "Ekpaideutikoi", οι οποίες ενσωματώνουν από μια ViewPager για εναλλαγή των view με κίνηση swipe του δαχτύλου μας στην οθόνη, και προβολή σύνθετων αντικειμένων λίστας τα οποία δημιουργήσαμε στην φάση του σχεδιασμού των layouts.

4.4.1 Υλοποίηση της ViewPager

Ξεκινάμε δηλώνοντας την κλάση "ViewPagerAdapter" η οποία εχει σαν SuperClass την "PagerAdapter" και ενσωματώνει την κλάση "TitleProvider"

```
class ViewPagerAdapter extends PagerAdapter implements TitleProvider {
```

Η κλάση αυτή χρησιμοποιεί έναν πίνακα τύπου String ο οποίος περιέχει τα ονόματα τίτλων που θα εμφανίζονται στις καρτέλες τις ViewPager. Στη συνέχεια με τη χρήση μεθόδων τα εμφανίζει και δημιουργεί από κάτω τους το αντικείμενο που θα κρατάει την κάθε λίστα ή άλλη προβολή που θέλουμε να εμφανίσουμε στην οθόνη.

```
private final String[] titles = new String[] { "Εξάμηνο Α",
      "Εξάμηνο Β", "Εξάμηνο Γ", "Εξάμηνο Δ", "Εξάμηνο Ε",
      "Εξάμηνο ΣΤ", "Εξάμηνο Ζ", "Εξάμηνο Η" };
private final Context context;

public ViewPagerAdapter(Context context) {
       this.context = context;
      }

@Override
public String getTitle(int position) {
      return titles[position];
}

@Override
public int getCount() {
      return titles.length;
}

@Override
public Object instantiateItem(View pager, int position) {
ListView v = new ListView(context);
((ViewPager) pager).addView(v, 0);
return v;
```

Ακριβώς την ίδια υλοποίηση κλάσης θα χρησιμοποιήσουμε και στις άλλες δύο Activities οι οποίες φυσικά θα χρησιμοποιούν άλλο πίνακα με αποθηκευμένα ονόματα τίτλων.

4.4.2 Ενσωμάτωση των πόρων συστήματος στις λίστες

Αφού δημιουργήσαμε τις συρόμενες προβολές μας, μένει να προσθέσουμε τα αρχεία κειμένου και εικόνων, που είχαμε εισάγει σε προηγούμενο κεφάλαιο στους φακέλους "values" & "resources", σε λίστες οι οποίες θα εμφανίζονται μέσα στις συρόμενες προβολές τις ViewPager. Θα δείξουμε την προσθήκη υλικού στην Activity "Mathimata", ώστε να φανεί καλύτερα η πολυπλοκότητα του κάθε αντικειμένου λίστας που θα δημιουργήσουμε.

Αρχικά θα ανακτήσουμε τους πόρους που έχουμε εισάγει στο σύστημα και θα τους προσθέσουμε σε τοπικούς πίνακες τύπου String.

```
ArrayList<HashMap<String, Object>> examino_A = new
ArrayList<HashMap<String, Object>>();
      String[] examino_A_title = getResources().getStringArray(
                              R.array.Examino_A_title);
      String[] examino_A_omada = getResources().getStringArray(
                              R.array.Examino_A_omada);
      String[] examino_A_typos = getResources().getStringArray(
                              R.array.Examino_A_typos);
      String[] examino_A_dm = getResources().getStringArray(
                              R.array.Examino_A_dm);
      String[] examino_A_theoreia = getResources().getStringArray(
                              R.array.Examino_A_theoreia);
      String[] examino_A_frontistirio = getResources().getStringArray(
                              R.array.Examino_A_frontistirio);
      String[] examino_A_ergastirio = getResources().getStringArray(
                              R.array.Examino_A_ergastirio);
```

Ακολουθεί η εισαγωγή των πινάκων, σε ένα αντικείμενο τύπου List το οποίο αργότερα θα προσθέσουμε στην ListView μέσα στην ViewPagerAdapter

```
for (int i = 0; i < examino_A_title.length; i++) {
      HashMap<String, Object> m = new HashMap<String, Object>();
      m.put("Τίτλος", examino_A_title[i]);
      m.put("Ομάδα", "Ομάδα: " + examino_A_omada[i]);
      m.put("Τύπος", " // Τύπος: " + examino_A_typos[i]);
      m.put("ΔΜ", " // ΔΜ:" + examino_A_dm[i]);
      m.put("Θεωρεία", "Θεωρία: " + examino_A_theoreia[i]);
      m.put("Φροντιστήριο", " // Φροντιστήριο: "
                                    + examino_A_frontistirio[i]);
      m.put("Εργαστήριο", " // Εργαστήριο: "
                                    + examino_A_ergastirio[i]);

      if (examino_A_typos[i].equals("Υποχρεωτικό")) {
      m.put("Εικόνα",getResources().getDrawable(R.drawable.red_book));
```

```
        } else if (examino_A_typos[i].equals("Επιλογής Υποχρεωτικό")) {
        m.put("Εικόνα",getResources().getDrawable(R.drawable.blue_book));

        } else if (examino_A_typos[i].equals("Προαιρετικό")) {
        m.put("Εικόνα",getResources().getDrawable(R.drawable.green_book));
        }
        examino_A.add(m);
        }
return (List<HashMap<String, Object>>) examino_A;
```

Εισάγουμε με τη χρήση μίας for όλα τα αντικείμενα σε θέσης του χάρτη πίνακα της List, και κάνουμε έναν έλεγχο το αποτέλεσμα του οποίου ορίζει το ποιο drawable, θα προστεθεί σε κάθε αντικείμενο λίστας. Κόκκινο βιβλίο για τα υποχρεωτικά μαθήματα, μπλε για τα επιλογής και πράσινο για τα προαιρετικά. Την ίδια διαδικασία θα επαναλάβουμε για όλα τα εξάμηνα, και αντίστοιχα στις υπόλοιπες Activities για την προσθήκη των υπόλοιπον πόρων στα αντικείμενα λίστας.

4.4.3 Εισαγωγή λίστας στις ViewPager

Για να εμφανίσουμε τα αντικείμενα τύπου List θα κάνουμε χρήση μιας SimpleAdapter η οποία θα αντιστοιχεί τα αντικείμενα τις List στα Ids των TextViews του layout αντικειμένου λίστας που θα δηλώσουμε.

```
SimpleAdapter examino_A_adapter = new SimpleAdapter(context,
        examino_A_group_list(), R.layout.mathimata_list_item,
        new String[] { "Τίτλος", "Ομάδα", "Τύπος", "ΔΜ",
                        "Θεωρεία", "Φροντιστήριο", "Εργαστήριο" },
        new int[] { R.id.row_title, R.id.row_omada,
              R.id.row_typos, R.id.row_dm, R.id.row_theoreia,
              R.id.row_frontistirio, R.id.row_ergastirio })
```

Χρησιμοποιώντας μια Switch στην μέθοδο "`instantiateItem`" της κλάσης "`ViewPagerAdapter`" θα ενσωματώσουμε όσες λίστες χρειαζόμαστε σε κάθε περίπτωση για να εμφανίσουμε όλο το υλικό μας. Αφού ολοκληρώσουμε την διαδικασία για όλες τις Activities και τα αντικείμενα λίστας τους, το τελικό αποτέλεσμα θα είναι όπως τις εικόνας 4.8

Εικόνα 4.8: Οι τρεις υλοποιήσεις χρήσης της ViewPager

4.5 Υλοποίηση WebView Activity για προβολή ιστοσελίδων

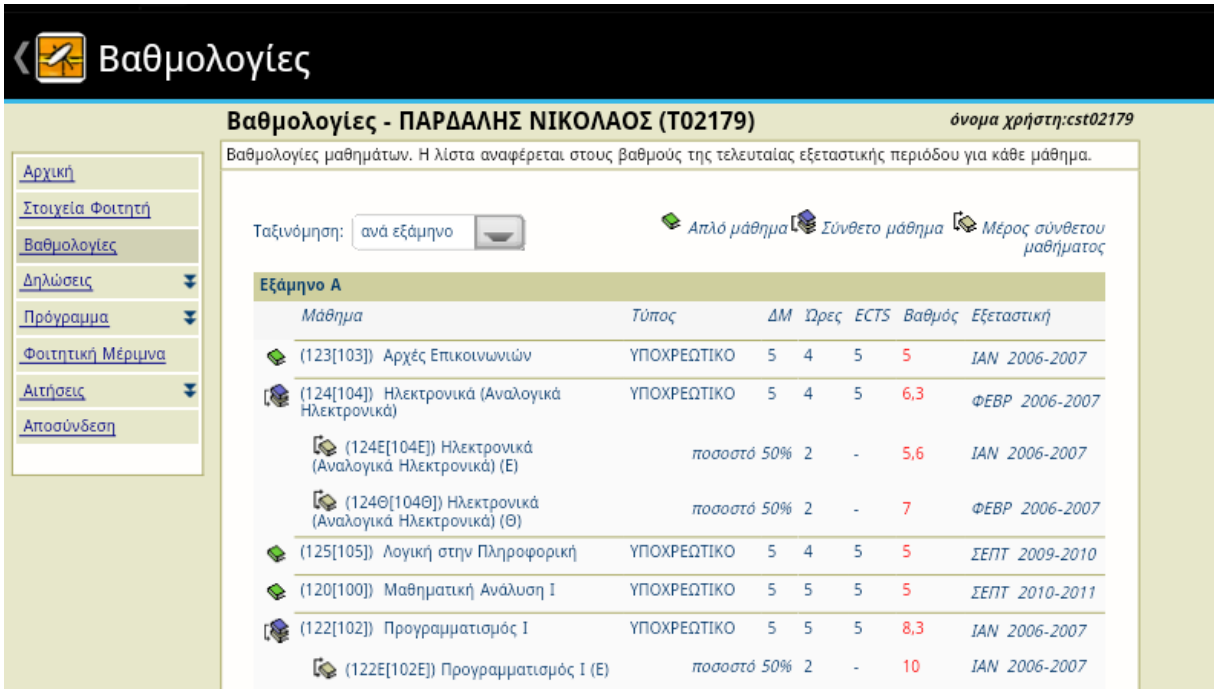
Θα ολοκληρώσουμε την υλοποίηση του κορμού της εφαρμογής με την Activity "MobileWebView" η οποία θα χρησιμοποιηθεί για να εμφανίσουμε περιεχόμενο web μέσω της εφαρμογής μας. Αρχικά θα χρησιμοποιήσουμε το layout που έχουμε δημιουργήσει για την εμφάνιση της WebView, και θα τις δώσουμε την δυνατότητα να υποστηρίζει την JavaScript, και θα προσθέσουμε τη δυνατότητα zoom μέσω controls.

```
webview = (WebView) findViewById(R.id.webview);

webview.getSettings().setJavaScriptEnabled(true);
webview.getSettings().setBuiltInZoomControls(true);
```

Στη συνέχεια θα λάβουμε το url μέσω Intent άλλης Activity και θα το προβάλουμε μέσω της WebView.

```
Intent sender = getIntent();

String url = sender.getExtras().getString("Link");
webview.loadUrl(url);
```

Πλέον μπορούμε να προβάλουμε Web υλικό χωρίς να ανοίξουμε κάποιον browser, αλλά μέσα από την εφαρμογή μας. Στην παρακάτω εικόνα βλέπουμε πως θα προβάλετε η οθόνη των βαθμολογιών μέσω της εφαρμογής.



Εικόνα 4.9: Οθόνη προβολής βαθμολογιών

Κεφάλαιο 5

Αποσφαλμάτωση συμπεράσματα, και μελλοντική εξέλιξη της εφαρμογής

5.1. Ενσωμάτωση βιβλιοθήκης ACRA

Το κύριο μέρος της υλοποίησης της εφαρμογής ολοκληρώθηκε και είμαστε ένα βήμα πριν να ξεκινήσουμε την δοκιμαστική φάση της εφαρμογής στην οποία θα ζητήσουμε από μερικούς εθελοντές να δοκιμάσουν την εφαρμογή, ώστε να συλλέξουμε σφάλματα και τις εμπειρίες χρήσης τους. Πρώτα όμως πρέπει να υλοποιήσουμε τον μηχανισμό συλλογής απομακρυσμένων σφαλμάτων, που δεν είναι άλλος από την βιβλιοθήκη ACRA. Την πρώτη γνωριμία μας με την βιβλιοθήκη την είχαμε κάνει στο κεφάλαιο 2.4.5, τώρα ήρθε η ώρα να την ενσωματώσουμε στην εφαρμογή μας. Θα ξεκινήσουμε με την λήψη της από την σελίδα http://code.google.com/p/acra.

Η ενσωμάτωση της βιβλιοθήκης ACRA είναι πιο απλή υπόθεση σε σχέση με τις προηγούμενες βιβλιοθήκες καθότι περιέχεται σε ένα αρχείο .jar το οποίο απλά θα τοποθετήσουμε στο Project μας και θα το προσθέσουμε στο build path. Δεν χρειάζεται να δημιουργήσουμε νέο project και να το χρησιμοποιήσουμε σαν βιβλιοθήκη, όπως κάναμε στις τρεις προηγούμενες περιπτώσεις. Τα επόμενα βήματα χρήσης της βιβλιοθήκης, είναι η δημιουργία του εγγράφου GoogleDoc, η υλοποίηση της κλάσης, και η ενημέρωση του AndroidManifest.

5.1.1. Δημιουργία του αρχείου CrashReports για λήψη των αναφορών

Για να συλλέξουμε τα δεδομένα του logcat από τις συσκευές που θα τρέχει η ACRA, θα χρησιμοποιήσουμε ένα έγγραφο τύπου GoogleDoc το οποίο φυσικά θα είναι αποθηκευμένο στο cloud της Google και θα δέχεται τις αναφορές σφάλματος. Ξεκινάμε εισάγοντας το αρχείο “CrashReports-template.csv” το οποίο περιέχεται στο zip που κατεβάσαμε προηγουμένως, και έχει όλες τις προεπιλεγμένες στήλες που θα γίνεται η αποθήκευση των δεδομένων αποσφαλμάτωσης. Το ονομάζουμε “CST Connect - CrashReports” και στη συνέχεια πηγαίνουμε στη διαδρομή “Tools / Form / Create a form”

του μενού των Google Docs και δημιουργούμε μια νέα κενή φόρμα.

Στην φόρμα που δημιουργήσαμε κάτω δεξιά θα μας εμφανιστεί η τιμή της μεταβλητής "formkey" την οποία θα αντιγράψουμε και θα το εισάγουμε στην εφαρμογή μας ώστε η ACRA να ξέρει σε πια φόρμα θα αποστέλλει τις αναφορές σφάλματος. Αφού ολοκληρώσαμε και την δημιουργία του αρχείου συλλογής σφαλμάτων, θα ενεργοποιήσουμε από το μενού την υπηρεσία ειδοποιήσεων με mail της Google, και θα επιλέξουμε να ενημερωνόμαστε κάθε φορά που θα υποβάλετε μια νέα αναφορά σφάλματος στην φόρμα που δημιουργήσαμε. Αυτό θα μας επιτρέπει να ανταποκρινόμαστε στις ανάγκες τις αποσφαλμάτωσης σε πραγματικό χρόνο, και χωρίς την παραμικρή ενόχληση στην πλευρά του χρήστη!

5.1.2. Δημιουργία της κλάσης CrashReporter

Για να χρησιμοποιήσουμε την βιβλιοθήκη από την εφαρμογη μας, θα πρέπει να υλοποιήσουμε μια κλάση η οποία θα ξεκινάει τη λειτουργία συλλογής και αποστολής αναφορών σφάλματος, όταν αυτά προκύψουν. Θα δημιουργήσουμε μια νέα κλάση την "CrashReporter" η οποία θα έχει σαν SuperClass την κλάση "Application".

```
Public class CrashReporter extends Application
```

Πάνω από την κλάση σε ένα annotation, θα δηλώσουμε την τιμή της μεταβλητής formkey την οποία πήραμε προηγουμένως από την διαδικασία δημιουργίας νέας φόρμας.

```
@ReportsCrashes(formKey = "dDJob3VmcXFVUTBZYW1CWWN3UVczVkE6MQ")
```

Μένει να υλοποιήσουμε και τον υπόλοιπό κώδικα της κλάσης μας, ο οποίος θα ξεκινά την λειτουργία της βιβλιοθήκης όποτε αυτό χρειαστεί.

```
public void onCreate() {
    // The following line triggers the initialization of ACRA
    ACRA.init(this);
    super.onCreate();
}
```

5.1.3. Ενημέρωση AndroidManifest.xml

Το τελευταίο βήμα ενσωμάτωσης της βιβλιοθήκης απαιτεί την προσθήκη μιας γραμμής κώδικα στο αρχείο AndroidManifest.xml, η οποία θα κάνει αναφορά στην νέα κλάση που δημιουργήσαμε. Κάτω από τα "application" tags του AndroidManifest θα προσθέσουμε την παράμετρο "android:name=":

```
<application
    android:name="CrashReporter"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.Sherlock" >
```

5.2. Δοκιμαστική περίοδος της Εφαρμογής (Beta Testing)

Στη δοκιμαστική φάση της συσκευής ζητήσαμε εθελοντές συνάδελφους, με κινητά Android, να δοκιμάσουν την έκδοση v0.5.12 της εφαρμογής και να μας μεταφέρουν τις εμπειρίες χρήσης τους. Συνολικά ανταποκρίθηκαν 6 συμφοιτητές μας τους οποίους ευχαριστούμε θερμά διότι τα συμπεράσματα που βγάλαμε μας βοήθησαν να διορθώσουμε κάποια Bugs της εφαρμογής τα οποία μας είχαν ξεφύγει στην φάση της υλοποίησης.

Συνολικά διορθώθηκαν 4 σημαντικά bugs τα οποία είχαν να κάνουν με τους χάρτες, τις ανακοινώσεις και την εμφάνιση των εικονιδίων στις μικρές οθόνες. Τα περισσότερα μικροπροβλήματα εμφανίστηκαν σε συσκευή μικρής οθόνης η οποία μάλιστα είχε την έκδοση 2.1 του Android. Τα ίδια προβλήματα δεν εμφανίστηκαν καθόλου σε συσκευές με μεγαλύτερη οθόνη και νεότερες εκδώσεις του Android. Αυτό αποδεικνύει περίτρανα το πρόβλημα του fragmentation του λειτουργικού συστήματος και έπρεπε εν τέλει να καταβάλουμε σημαντική προσπάθεια για να γίνει η εφαρμογή συμβατή με τις παλαιότερες συσκευές

Η συσκευές που χρησιμοποιήθηκαν για τη φάση του Beta Testing εμφανίζονται στον παρακάτω πίνακα. Η δοκιμαστική φάση της εφαρμογής κρίθηκε αρκετά επιτυχημένη διότι οι συσκευές των συναδέλφων μας, κάλυπταν ένα μεγάλο μέρος των συνηθέστερων διατάξεων του Android, και έτσι μπορέσαμε να απομονώσουμε όλα τα σημαντικά Bugs σε σύντομο χρονικό διάστημα.

Κατασκευαστής Συσκευής	Όνομα Συσκευής	Αριθμός μοντέλου	Έκδοση Android	Ανάλυση Οθόνης	Ανάλυση οθόνης/ Αναλογία DPI
Samsung	Galaxy S	GT-I9000	2.3.7	800x400	WVGA800-hdpi
ZTE	Blade	Blade	2.3.4	800x400	WVGA800-hdpi
Samsung	Galasy S2	GT-I9100	4.0.3	800x400	WVGA800-hdpi
Huawei	IDEOS X5	U8800	2.3.7	800x400	WVGA800-hdpi
LG	GT540 Optimus	GT540	2.3.7	320x480	HVGA-mdpi
LG	Optimus One	P500	2.3.5	320x480	HVGA-mdpi
Sony Ericsson	Xperia X8	e15	2.1	320x480	HVGA-mdpi
Sony Ericsson	X10 mini pro	U20i	2.1	240x320	QVGA-ldpi

Πίνακας 5.1: Λίστα συσκευών που συμμετείχαν στην δοκιμαστική φάση

5.3. Μελλοντική εξέλιξη της εφαρμογής CST Connect

Όπως ισχύει για όλα τα λογισμικά, έτσι και για την εφαρμογή CST Connect η ανάπτυξη της δεν σταματά σε αυτό το σημείο, αλλά θα συνεχιστεί και στο μέλλον μιας και οι προοπτικές πρόσθεση νέων δυνατοτήτων και λειτουργιών είναι πολλές! Αρχικός στόχος μας είναι να εξελίξουμε το σύστημα λήψης των ανακοινώσεων για να προστεθεί η δυνατότητα αυτόματης λήψης νέων ανακοινώσεων στην συσκευή, και ενημέρωση του χρήστη μέσω της notification bar. Υπάρχει σκέψη να προστεθεί λίστα προσθήκης “αγαπημένων” ώστε ο χρήστης μπορεί να προσθέσει όσες ανακοινώσεις θέλει να παρακολουθεί, και να λαμβάνει ειδοποιήσεις μόνο για αυτές!

Κατά το τελικό κομμάτι της εργασίας, έγινε συζήτηση για ενσωμάτωση των υπηρεσιών τις ιστοσελίδας cronos.teilar.gr στην εφαρμογή CST Connect, αλλά λόγω λήξης της προθεσμίας υποβολής της πτυχιακής εργασίας δεν έγινε εφικτή η μελέτη των αναγκών της ενσωμάτωσης. Η ιστοσελίδα Cronos συγκεντρώνει όλες τις πληροφορίες από τις υπηρεσίες που είναι εγγεγραμμένος ο χρήστης όπως ανακοινώσεις από το e-class, βαθμολογίες από το dionysos, κλπ. Σαν εξέλιξη της εφαρμογής είναι πιθανό να παρέχουμε πρόσβαση στον Cronos χρησιμοποιώντας τα στοιχεία σύνδεσης του χρήστη και κάνοντας λήψη και προβολή σε mobile μορφή, των πληροφοριών χρήστη που συγκεντρώνει η ιστοσελίδα. Τέλος, ο κώδικας της εφαρμογής “CST Connect” θα ανέβει στο GIT της linux team όπου ο καθένας θα μπορεί να έχει πρόσβαση στον πηγαίο κώδικα της.

5.4. Συμπεράσματα

Η ανάπτυξη εφαρμογής στο Android είναι μια ιδιαίτερα απαιτητική εργασία η οποία απαιτεί σημαντικές γνώσεις Java και ταλέντο στον προγραμματισμό, πάνω από όλα όμως χρειάζεται το μεράκι του προγραμματιστή να δημιουργήσει κάτι πραγματικά χρήσιμο.

Η υλοποίηση της εφαρμογής απαίτησε να αφιερώσουμε αρκετό χρόνο στην εκμάθηση της γλώσσας Java διότι προερχόμενοι από τον κύκλο σπουδών "Δικτύων και Τηλεπικοινωνιών" δεν την είχαμε διδαχτεί ποτέ. Για να ξεκινήσουμε το project λοιπόν ασχοληθήκαμε ατομικά με την εκμάθηση των βασικών συστατικών της JAVA και μετά από 2 μήνες μελέτης ξεκινήσαμε την υλοποίηση της εφαρμογής. Συνολικά η υλοποίηση πήρε 4-5 μήνες μαζί με την διαδικασία του Debugging και την φάση του Beta Testing.

Συναντήσαμε αρκετές δυσκολίες καθότι αρκετές από τις δομές τις εφαρμογής ήταν αρκετά απαιτητικές για αρχάριους προγραμματιστές, όπως η λήψη και προβολή των αρχείων xml των RSS, και η υλοποίηση της Βάσης Δεδομένων. Μέσω στενής συνεργασίας και ομαδικής δουλείας, όμως καταφέραμε να φέρουμε εις πέρας την εργασία και να αποκτήσουμε στην πορεία πολλά εφόδια τα οποία αδιαμφισβήτητα θα αξιοποιήσουμε στο μέλλον.

Βιβλιογραφία

- J. Friesen, 2010, “Learn Java for Android Development”, Apress
- W.M. Lee, 2011, “Beginning Android Application Development”, Wrox
- M. Murphy, 2011, "Android Programming Tutorials, 3rd Edition”, CommonsWare
- J. Morris, 2011, “Android User Interface Development Beginner's Guide”, Pakt Publishing
- C. Hasenan, 2008, “Android Essentials”, Firstpress
- J. Steele, 2010, “The Android Developer's Cookbook”, Addison & Wesley
- R. Meier, 2010, “Professional Android 2 Application Development”, Wrox

- Ανάλυση της αρχιτεκτονικής του Android
  http://developer.android.com/guide/basics/what-is-android.html
- Ανάλυση των βασικών συστατικών μιας εφαρμογής του Android, και ο συσχετισμός τους με την αρχιτεκτονική του συστήματος
  http://developer.android.com/guide/basics/what-is-android.html
- Πληροφορίες σχετικά με τη δομή και τη χρήση του του αρχείου AndroidManifest.xml
  http://developer.android.com/guide/topics/manifest/manifest-intro.html
- Οι φάσεις του σχεδιασμού τις εφαρμογής περιλαμβάνονται στην παρακάτω πηγή
  http://developer.android.com/guide/developing/index.html
- Οδηγίες εγκατάστασης του Android SDK
  http://developer.android.com/sdk/installing.html
- Η επίσημη ιστοσελίδα του Eclipse IDE, το οποίο χρησιμοποιήθηκε κατά την υλοποίηση του project και είναι και προτεινόμενο από την Google.
  http://www.eclipse.org/
- Οδηγίες εγκατάστασης του ADT plugin
  http://developer.android.com/sdk/eclipse-adt.html#installing
- Η ιστοσελίδα “Design Guidelines”, η οποία περιέχει πληροφορίες, παραδείγματα, και οδηγίες ώστε να χρησιμοποιηθούν από τους developers για τον σχεδιασμό των εφαρμογών τους.
  http://developer.android.com/design/index.html

- Ο καταμερισμός των εκδόσεων του Android όπως αυτά καταγράφονται από την πρόσβαση των συσκευών στο Google Play Store.
  http://developer.android.com/resources/dashboard/platform-versions.html
- Ο καταμερισμός των διαστάσεων οθόνης προς την πυκνότητα pixel των συσκευών που χρησιμοποιούν το Play Store
  http://developer.android.com/resources/dashboard/screens.html
- Οδηγίες υποστήριξης πολλαπλών αναλύσεων στις εφαρμογές
  http://developer.android.com/guide/practices/screens_support.html
- Ανάλυση και οδηγίες χρήσης του Android Debug Bridge
  http://developer.android.com/guide/developing/tools/adb.html
- Σχετικά με την διαχείριση εικονικών συσκευών
  http://developer.android.com/guide/developing/devices/index.html
- Πληροφορίες σχετικά με το εργαλείο LogCat
  http://developer.android.com/guide/developing/tools/logcat.html
- Η βιβλιοθήκη ανοιχτού κώδικα ACRA η οποία χρησιμοποιήθηκε στο project για να παρέχει απομακρυσμένη αναφορά σφαλμάτων.
  http://code.google.com/p/acra/
- Έρευνα της δημιουργού εταιρίας της εφαρμογής "OpenSignalMaps" σχετικά με το fragmentation του Android, από την οποία χρησιμοποιήθηκαν δεδομένα και εικόνες.
  http://opensignalmaps.com/reports/fragmentation.php
- Steven J. Vaughan-Nichols, August 18, 2011, "Linus Torvalds on Android, the Linux fork", Zdnet.com
  http://www.zdnet.com/blog/open-source/linus-torvalds-on-android-the-linux-fork/9426
- Casey Johnston, March 12 2012, "Instagram comes to Google Play as big developer leaves", arstechina.com
  http://arstechnica.com/gadgets/2012/03/instagram-comes-to-google-play-as-big-developer-leaves/
- Η κοινότητα προγραμματιστών "Stack Overflow" παρέχει ένα μέρος για ερωτήσεις και απαντήσεις στα περισσότερα προβλήματα που μπορεί να συναντήσει ένας developer του Android, και η συνεισφορά της ήταν ανεκτίμητη.
  www.stackoverflow.com