ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία εφαρμογής Android για το σύστημα διαχείρισης βιβλιογραφικών αναφορών Zotero.

Του φοιτητή:

Ευάγγελου Παππά

Αρ. Μητρώου:2616

Επιβλέπων καθηγητής:

Ευστάθιος Αντωνίου

Θεσσαλονίκη 2015

ΠΡΟΛΟΓΟΣ

Η επιλογή του συγκεκριμένου θέματος προήλθε από τη θέληση για περαιτέρω γνώση στον τομέα των Android εφαρμογών. Οι συσκευές με ενσωματωμένο το λειτουργικό σύστημα Android συνεχώς αυξάνονται και εξελίσσονται. Σημαντικό ρόλο για την ανάληψη του συγκεκριμένου θέματος διαδραμάτισε και η υπάρχουσα τεκμηρίωση για το λειτουργικό Android στο διαδίκτυο.

ΠΕΡΙΛΗΨΗ

Το παρόν κείμενο χωρίζεται σε πέντε κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζεται η αρχιτεκτονική του Android και οι εκδόσεις που έχουν δημοσιευθεί μέχρι τώρα. Στο δεύτερο κεφάλαιο πραγματοποιείται ιστορική αναδρομή στις εκδόσεις του Android. Κατόπιν γίνεται εκτενής αναφορά στα εργαλεία ανάπτυξης που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής ZoteroApp και ο τρόπος εγκατάστασης τους. Το τέταρτο κεφάλαιο αναφέρεται στο σύστημα βιβλιογραφικών αναφορών Zotero. Συνοπτικά παρουσιάζονται οι δυνατότητες του και τα εργαλεία που παρέχει στου προγραμματιστές για την υλοποίηση επικοινωνίας με τους διακομιστές του. Παρακάτω, στο πέμπτο κεφάλαιο αρχικά παρουσιάζεται ο τρόπος δημιουργίας ενός Android Project και τα συστατικά του. Παρουσιάζεται ο τρόπος που δημιουργήθηκε η εφαρμογή ZoteroApp και τα βασικότερα συστατικά της.

ABSTRACT

This text is divided into five chapters. The first chapter presents the architecture of Android. The second chapter presents Android versions that have been published so far. Then an extensive reference to the development tools used to implement the application ZoteroApp and the possible methods of installation are given. The fourth chapter refers to the Zotero system for bibliographic references. It summarizes the features and tools provided to developers to implement communication with the servers. In the fifth chapter, we first explain how to setup an Android Project and its components. Moreover, we present how the application ZoteroApp was created and what was its main components. In the last chapter, although the application itself does not permanently store data received from the communication servers of zotero, we present what data storage options are.

Περιεχόμενα

Ευρετήριο εικόνων

ΕΙΣΑΓΩΓΗ

Τι είναι το Android

Το Android είναι ένα λειτουργικό σύστημα βασισμένο στον πυρήνα του λειτουργικού Linux, προορίζεται για έξυπνες συσκευές κινητής τηλεφωνίας (smartphones) και mobile computers, γνωστά ως tablets. Αναπτύχθηκε αρχικά από την Google και αργότερα από την Open Handset Alliance, η οποία είναι μια κοινοπραξία εταιριών λογισμικού (software) και υλικού (Hardware), όπου συμπεριλαμβάνεται και η Google. Η Java είναι η γλώσσα που χρησιμοποιείται για την συγγραφή του κώδκα. Η συσκευή ελέγχεται από βιβλιοθήκες λογισμικού, οι οποίες αναπτύχθηκαν από την Google. Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου του 2007. Η πρώτη συσκευή με Android λειτουργικό διαθέσιμη στο εμπόριο ήταν το Nexus One. Η συσκευή κατασκευάστηκε από την HTC Corporation και έγινε διαθέσιμη τον Ιανουάριο του 2010.

Λειτουργία του Android O.S.

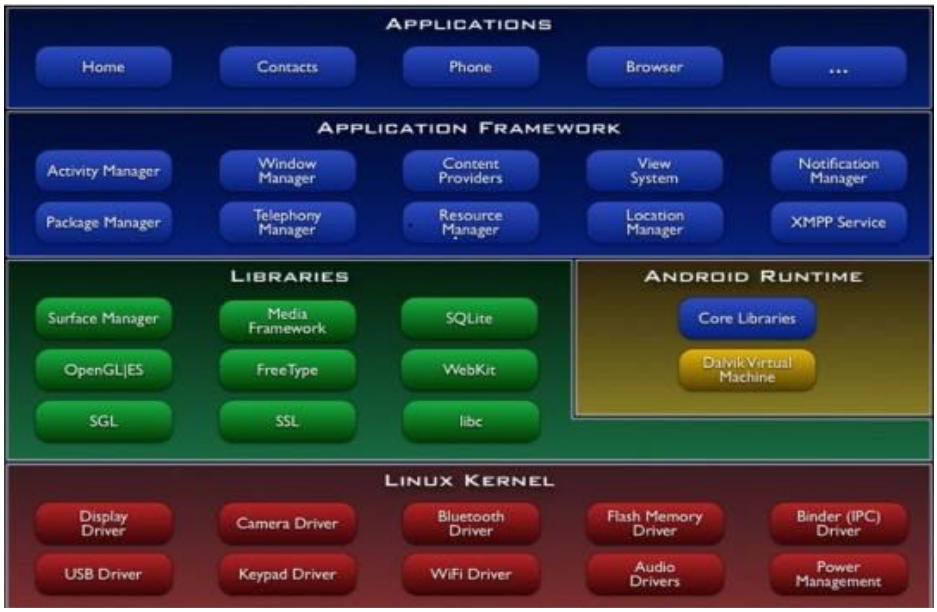
Όπως προαναφέρθηκε το Android χρησιμοποιεί την γλώσσα προγραμματισμού Java. Δηλαδή, ο προγραμματισμός Android εφαρμογών πραγματοποιείται χρησιμοποιώντας τα Java API που παρέχονται από την Google, τα οποία μεταγλωττίζονται σε αρχεία **.class**. Η κύρια διαφορά του Android είναι ότι δεν χρησιμοποιεί την εικονική μηχανή της Java (Java Virtual Machine) για την εκτέλεση αρχείων .class. Τα αρχεία .class μετασχηματίζονται σε αρχεία **.dex (Dalvik executables)** και **.jar**. Μετά τον μετασχηματισμό τους ομαδοποιούνται στο Android Package (APK) και είναι έτοιμα για διανομή και εγκατάσταση σε διάφορες συσκευές. Η παραπάνω διαδικασία είναι πολύ σημαντική, διότι τα αρχεία .dex που δημιουργούνται είναι πολύ συμπαγή και αποδοτικά καθώς προορίζονται για συσκευές με περιορισμένη μνήμη και μπαταρία.

ΚΕΦΑΛΑΙΟ 1

Αρχιτεκτονική του Android

Εισαγωγή

Το Android είναι σχεδιασμένο για να υποστηρίζει κυρίως κινητές συσκευές όπως κινητά τηλέφωνα (smartphones) και ταμπλέτες (tablets). Ως λειτουργικό σύστημα οφείλει να να παρέχει στον χρήστη την δυνατότητα να χρησιμοποιεί τους πόρους του συστήματος, όσο το δυνατό περισσότερο, μέσω μιας φιλικής προς τον χρήστη διεπαφής. Για την επίτευξη αυτού του στόχου το Android αποτελείται από μία στοίβα λογισμικού, με διακριτό και καθορισμένο ρόλο το κάθε τμήμα της στοίβας. Επιπλέον, το Android παρέχει ένα πακέτο ανάπτυξης λογισμικού (Software Development Kit) για την δημιουργία εφαρμογών.



Εικόνα: i

Συνοπτικά αναφέρουμε ότι στο κατώτερο επίπεδο βρίσκεται ο πυρήνας του Linux. Είναι γνωστό ότι συγκεκριμένος πυρήνας είναι συμβατός με πολλές αρχιτεκτονικές επεξεργαστών. Για αυτό το λόγο τα υπόλοιπα τμήματα χτίζονται πάνω του.
Το επίπεδο ακριβώς από πάνω του αποτελείται από τις βιβλιοθήκες του συστήματος. Μέσα σε αυτό το τμήμα συμπεριλαμβάνονται η βασική βιβλιοθήκη τη γλώσσας προγραμματισμού C, υλοποιήσεις για τις βιβλιοθήκες των γραφικών, των συστημάτων διαχείρισης βάσεων δεδομένων κ.α.
Στο επίπεδο χρόνου εκτέλεσης (Runtime) υπάρχει η εικονική μηχανή(Virtual machine) που διερμηνεύει τις εφαρμογές Android και των βασικών βιβλιοθηκών της Java που είναι απαραίτητες στην διαδικασία ανάπτυξης εφαρμογών.

Το Application Framework(επίπεδο των εφαρμογών) είναι το προτελευταίο επίπεδο, το οποίο προσφέρει στον προγραμματιστή μια πληθώρα δυνατοτήτων σχετιζόμενων με το λειτουργικό σύστημα και την ανάπτυξη εφαρμογών.

Στο υψηλότερο επίπεδο, αυτό των εφαρμογών(Application layer), το Android παρέχει κάποιες πρότυπες εφαρμογές όπως ο περιηγητής ιστού (browser), η εφαρμογή τηλεφώνου (phone dialer).

1.1 Πυρήνας Linux

Στο κατώτερο επίπεδο βρίσκεται ο πυρήνας Linux. Ο Linux kernel παρέχει στο Android το αφαιρετικό στρώμα ελέγχου του υλικού. Το Android χρησιμοποιεί τον πυρήνα Linux για την διαχείριση μνήμης, τη διαχείριση διεργασιών,τη δικτύωση και άλλες υπηρεσίες του λειτουργικού συστήματος. Πάνω σε αυτό το επίπεδο χτίζονται και τα υπόλοιπα επίπεδα του Android. Συγκεκριμένα, από την έκδοση 3.3 του πυρήνα του Linux γίνεται η συγχώνευση με τον κώδικα του Android. Πρακτικά σημαίνει ότι κάθε κατασκευαστής κινητών συσκευών μπορεί υλοποιώντας τους οδηγούς για τα περιφερειακά της συσκευής του να διαθέσει συσκευές με ενσωματωμένο σύστημα Android. Τα υπόλοιπα τμήματα της στοίβας μπορούν να μείνουν ως έχουν καθώς βασίζονται στο κατώτερο τμήμα της στοίβας.

1.2 Libraries

Στο αμέσως επόμενο επίπεδο βρίσκονται οι βιβλιοθήκες του συστήματος. Οι βιβλιοθήκες αυτές είναι γραμμένες σε γλώσσα προγραμματισμού C και C++. Είναι εφαρμογές που δεν μπορούν να κληθούν από μόνες τους, αλλά υπάρχουν για να καλούνται από προγράμματα υψηλότερου επιπέδου. Οι βιβλιοθήκες αυτές παρέχουν βασικές λειτουργικότητες όπως γραφικών και ήχου.

Δύο πολύ σημαντικές υλοποιήσεις είναι αυτές των OpenGL και SGL. Προσφέρουν την δυνατότητα στα παραπάνω επίπεδα της σχεδίασης τρισδιάστατων και δισδιάστατων γραφικών αντίστοιχα. Σε αυτό το επίπεδο υπάρχει η βιβλιοθήκη των Media που δίνει την δυνατότητα για την αναπαραγωγή, εγγραφή αρχείων ήχου και βίντεο.

Άξια αναφοράς είναι η βιβλιοθήκη που παρέχει μια μηχανή διαχείρισης συστήματος βάσεων δεδομένων, η SQLite. Οι εφαρμογές Android συνήθως χειρίζονται και αποθηκεύουν μεγάλο όγκο δεδομένων. Για αυτό είναι αναγκαία η ύπαρξη της μηχανής SQLite ως σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Δηλαδή η συλλογή των δεδομένων οργανωμένη σε σχετιζόμενους πίνακες που παρέχει μηχανισμό για ανάγνωση, εγγραφή ή τροποποίηση πάνω στα δεδομένα. Μια ακόμη βιβλιοθήκη που συναντάμε σε αυτό το επίπεδο είναι η Surface manager, δηλαδή ο διαχειριστής σχεδιαστικών επιφανειών. Πολύ σημαντική είναι η βιβλιοθήκη WebKit, για την απόδοση και προβολή ιστοσελίδων. Σκοπός της είναι να δίνει την δυνατότητα σε ένα browser να απεικονίζει τις ιστοσελίδες που επισκέπτεται ο χρήστης. Πρόσθετη λειτουργία είναι η απεικόνιση των γραμματοσειρών που αποτελούνται από bitmaps (raster fonts) αλλά και γραμματοσειρών που περιγράφονται μαθηματικά από καμπύλες bezier (vector fonts). Η συγκεκριμένη λειτουργία υποστηρίζεται από την βιβλιοθήκη FreeType. Ακόμη, το Android υποστηρίζει την χρήση του κρυπτογραφικού πρωτοκόλλου Secure Socket Layer μέσω της βιβλιοθήκης SSL . Τέλος, σε αυτό το επίπεδο υπάρχει η βασική

βιβλιοθήκη της C (libC) η οποία είναι υπεύθυνη για βασικές λειτουργίες του συστήματος όπως επεξεργασία και δημιουργία νημάτων, μαθηματικούς υπολογισμούς, εκχώρηση μνήμης.

1.3 Android Runtime

Όπως προαναφέρθηκε τα συστήματα Android προορίζονται για κινητές συσκευές με περιορισμένους πόρους. Τα προγράμματα Android είναι γραμμένα σε Java και ως τέτοια απαιτούν ένα διερμηνευτή για την εκτέλεση τους. Άρα για να ξεπεραστεί το εμπόδιο των περιορισμένων πόρων αναπτύχθηκε ειδικά για το Android ένας διερμηνευτής ο οποίος απαιτεί ένα ελάχιστο 64MB μνήμης.

Το όνομα του διερμηνευτή είναι Dalvik. Κάθε εφαρμογή Android εκτελείται σε ένα δικό της στιγμιότυπο του Dalvik, απομονώνοντας την από τις άλλες εφαρμογές. Με αυτό τον τρόπο διασφαλίζει ευστάθεια και ασφάλεια στο λειτουργικό σύστημα. Το Dalvik κατά την εκτέλεση του παράγει αρχεία bytecodes με την κατάληξη .dex, τα οποία είναι πολύ πιο συμπαγή από τα γνωστά αρχεία java bytecodes με αποτέλεσμα να απαιτούν λιγότερη μνήμη για την εγκατάσταση τους. Τα αρχεία εγκατάστασης έχουν την κατάληξη .apk όπου περιέχουν τα αρχεία .dex με τον bytecode προς εκτέλεση. Προκειμένου τα .dex αρχεία να είναι μικρά, τυχόν δεδομένα που εμφανίζονται πολλές φορές στις κλάσεις μιας εφαρμογής αποθηκεύονται μόνο μία φορά και όπου χρειάζεται δημιουργούνται αναφορές προς αυτά.

Επιπλέον το Android Runtime έχει ένα σημαντικό χαρακτηριστικό το JIT (Just In Time) μεταφραστή. Στόχος του —JIT είναι να μεταφράσει τον κώδικα byte σε κώδικα μηχανής, ώστε να αυξηθεί η ταχύτητα εκτέλεσης της εφαρμογής. Υπάρχουν δύο ειδών μεταφραστές, οι βασισμένοι σε μεθόδους και οι βασισμένοι σε ακολουθίες εκτέλεσης. Οι πρώτοι μεταφράζουν ολόκληρες μεθόδους πριν την εκτέλεσή τους σε γλώσσα μηχανής ενώ οι δεύτεροι ένα σύνολο εντολών που ανήκουν σε ένα κοινό δρόμο εκτέλεσης. Για την σωστή διαχείριση της μνήμης το Android Runtime παρέχει ένα garbage collector. Το δεύτερο συστατικό αυτού του

επιπέδου είναι είναι η βιβλιοθήκες πυρήνα JAVA (core libraries). Περιέχει τα πακέτα java.* , javax.* τα οποία παρέχουν κοινές δομές δεδομένων, μηχανισμούς συγχρονισμού και εισόδου εξόδου αρχείων. Παρέχει τα πακέτα Android.* που είναι ειδικά την διαχείριση του κύκλου ζωής των εφαρμογών Android. Τα πακέτα org.* τα οποία παρέχουν διάφορες λειτουργίες ιστού ή διαδικτύου και τα πακέτα junit.* που υποστηρίζουν τον δομικό έλεγχο των εφαρμογών.

1.4 Application Framework

Πάνω από το επίπεδο των βιβλιοθηκών βρίσκεται το επίπεδο Application Framework. Είναι το επίπεδο που ενδιαφέρει περισσότερο τον προγραμματιστή, καθώς παρέχει API (Application Program Interface) τα οποία παρέχουν όλα τα δομικά στοιχεία για την ανάπτυξη Android εφαρμογών. Λειτουργίες που μπορούν να ενσωματωθούν είναι η πρόσβαση στο διαδίκτυο, πρόσβαση στους αποθηκευτικούς χώρους της συσκευής και η δημιουργία διεπαφών χρήστη. Αυτό το πλαίσιο είναι προ-εγκατεστημένο στο Android και όλες οι βιβλιοθήκες του είναι γραμμένες σε γλώσσα Java.

Παρακάτω αναφέρονται τα σημαντικότερα δομικά στοιχεία του πλαισίου ως εξής:

- **Views Systems** (Σύστημα όψεων): Τα στιγμιότυπα τις κλάσης **View** μας δίνουν την δυνατότητα να συνθέσουμε μία διεπιφάνεια χρήστη. Δηλαδή όλα τα ορατά στοιχεία στην οθόνη (π.χ. Κουμπιά) αποτελούν στιγμιότυπα αυτής της κλάσης. Η κλάση View παρέχει ένα σύνολο μεθόδων που εκτελούνται μόλις ανιχνευθεί κάποιο γεγονός στην διεπιφάνεια χρήστη, με αυτό τον τρόπο επιτυγχάνεται η αλληλεπίδραση χρήστη με την διεπιφάνεια (π.χ. Πάτημα κουμπιού).
- **Package Manager** (Διαχειριστής πακέτων): Είναι ουσιαστικά μια βάση δεδομένων που παρακολουθεί όλες τις εφαρμογές που είναι εγκατεστημένες στην συσκευή.

- **Window Manager** (Διαχειριστής παραθύρων): Διαχειρίζεται τα παράθυρα που παρουσιάζονται οι εφαρμογές.
- **Resource Manager** (Διαχειριστής πόρων): Είναι υπεύθυνος για την διαχείριση των μη μεταγλωττίσιμων πόρων, όπως συμβολοσειρές (String), γραφικά στοιχεία και αρχεία με φύλλα διάταξης (layouts).
- **Notification Manager** (Διαχειριστής ειδοποιήσεων): Γεγονότα που συμβαίνουν με σκοπό να ενημερώσουν το χρήστη χωρίς να διακόψουν την εργασία του. Είναι μηνύματα το οποία εμφανίζονται στην μπάρα ειδοποιήσεων (notification bar).
- **Location Manager** (Διαχειριστής τοποθεσίας): Δίνει πληροφορίες για την τρέχουσα θέση του τηλεφώνου.
- **Content Provider** (Παροχέας περιεχομένου): Είναι ουσιαστικά βάσεις δεδομένων που επιτρέπουν την αποθήκευση και διαμοιρασμό δομημένων πληροφοριών. Χαρακτηριστικό παράδειγμα είναι οι επαφές που είναι αποθηκευμένες στη συσκευή. Οι συγκεκριμένες πληροφορίες είναι αποθηκευμένες σε ένα Content Provider και είναι διαθέσιμες για ανάγνωση και τροποποίηση σε πολλές εφαρμογές όπως το τηλέφωνο, εφαρμογή MMS, facebook κ.α.
- **Activity Manager** (Διαχειριστής δραστηριοτήτων): Μια Activity (Δραστηριότητα) ουσιαστικά αναπαριστά μια οθόνη με την περιεχόμενη διεπιφάνεια χρήστη. Ο διαχειριστής δραστηριοτήτων είναι υπεύθυνος για τον έλεγχο του κύκλου ζωής των δραστηριοτήτων (Εικόνα: xiii), παρέχει στον χρήστη την δυνατότητα πλοήγησης σε προηγούμενες οθόνες. Για παράδειγμα μια εφαρμογή διαχείρισης e-mail, η οποία μπορεί να παρέχει μία δραστηριότητα για τα εισερχόμενα μηνύματα, άλλη μία για σύνταξη νέου μηνύματος και άλλη μία για τα απεσταλμένο. Παρόλο που όλες αυτές οι δραστηρίοτητες δουλεύουν μαζί για να επιτύχουν μία συνεκτική εμπειρία χρήστη, είναι ανεξάρτητες η μία από την άλλη.

1.5 Application

Τέλος το επίπεδο των εφαρμογών, σε αυτό το επίπεδο βρίσκονται εγκατεστημένες οι εφαρμογές του Android (π.χ. Εφαρμογή αποστολής SMS). Όλες οι δυνατότητες που έχουν προαναφερθεί, όλα τα API του Application Framework είναι διαθέσιμα στον προγραμματιστή για να τα συνδυάσει και να δημιουργήσει εφαρμογές για το περιβάλλον Android.

Σύνοψη

Στο πρώτο κεφάλαιο παρουσιάστηκε η αρχιτεκτονική του Android. Παρουσιάστηκε η στοίβα λογισμικού και τα επίπεδα και αναφέρθηκαν τα ποιο σημαντικά συστατικά τους. Στο παρακάτω κεφάλαιο γίνεται αναφορά στην ιστορική αναδρομή του Android.

ΚΕΦΑΛΑΙΟ 2

Ιστορική Αναδρομή

Εισαγωγή

Το λειτουργικό Android ξεκινά από τον Νοέμβριο του 2007, με την πρώτη δοκιμαστική του έκδοση (beta). Η πρώτη εμπορική έκδοση κυκλοφόρησε τον Σεπτέμβριο του 2009. Από τότε το Android έχει περάσει από πολλές εκδόσεις που η κάθε μια βελτιώνει την προηγούμενη και διορθώνει ατέλειες της. Στην ενότητα που ακολουθεί θα γίνει αναφορά στις εκδόσεις του και στις διαφορές που υπάρχου μεταξύ τους.

2.1 Android 1.0

Ήταν η πρώτη έκδοση του Android. Η πρώτη συσκευή που κυκλοφόρησε με αυτό το λειτουργικό σύστημα ήταν της HTC το μοντέλο Dream. Συμπεριελάμβανε μεταξύ άλλων Android Web Browser, την εφαρμογή Android Market, την εφαρμογή youtube και Notification Bar, δηλαδή την μπάρα ειδοποιήσεων του Android με την δυνατότητα ρύθμισης ήχου, δόνησης και του LED ενημέρωσης.

2.2 Android 1.1

Η επόμενη έκδοση ήταν η 1.1 και κυκλοφόρησε τον Φεβρουάριο του 2009, αρχικά μόνο στη συσκευή HTC Dream. Η νέα αυτή έκδοση διόρθωσε σφάλματα της προηγούμενης και εισήγαγε κάποια νέα χαρακτηριστικά.

2.3 Android 1.5 Cupcake

Η έκδοση “Cupcake”, βασισμένη στο LinuxKernel 2.6.27, παρουσιάστηκε στις 30 Απριλίου του 2009. Υποστηρίζει νέες λειτουργίες για την κάμερα της 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. Μετά από αυτή την έκδοση όλες οι επόμενες έχουν ονόματα από γλυκά.

2.4 Android 1.6 Donut

Η έκδοση “Donut”, βασισμένη στο LinuxKernel 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).

2.5 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 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. 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.

2.6 Android 2.2 Froyo

Η έκδοση “Froyo”, βασισμένη στο Linux Kernel 2.6.32, παρουσιάστηκε στις 20 Μαΐου του 2010. Αποτελεί μια από τις πιο δημοφιλής εκδόσεις. 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 AdobeFlash 10.1, ενώ υποστηρίζεται καλύτερα πλέον τοMicrosoft Exchange. Έχει γίνει ανανέωση του AndroidMarket. 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. Επιπλέον το τηλέφωνο πλέον μπορεί να μετατραπεί σε WiFi hotspot.

2.7 Android 2.3 Gingerbread

Η έκδοση “Gingerbread”, βασισμένη στο Linux Kernel 2.6.35.7, παρουσιάστηκε στις 6 Δεκεμβρίου του2010, ενώ τον Φεβρουάριο του 2011 επανεκδόθηκε σε Android 2.3.3. Στην έκδοση αυτή υπάρχουν αλλαγές στη διεπιφάνεια χρήστη (UI) 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 FieldCommunication) και η ύπαρξη πολλαπλών καμερών. Βελτιωμένη είναι η ενεργειακή υποστήριξη και έχει γίνει μετάβαση από το σύστημα αρχείων YAFFS στο ext4 στις νέες συσκευές.

2.8 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. Έχει 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.

2.9 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. Επιπλέον, 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.

2.10 Android 4.1

Η έκδοση Android 4.1 ανακοινώθηκε από την Google στις 27 Ιουνίου του 2012. Η έκδοση αυτή είναι βασισμένη στον πυρήνα του Linux 3.1.10 και κύριος

σκοπός ήταν η βελτίωση της διεπαφής χρήστη. Βελτιώσεις που έγιναν περιλαμβάνουν την σημαντική βελτίωση των ηχητικών επιδόσεων, της μπάρας ενημερώσεων, την προσβασιμότητα των λειτουργιών και άλλα. Το SDK για την ανάπτυξη εφαρμογών για το Android Jelly Bean είναι ήδη διαθέσιμο σε οποιονδήποτε επιθυμεί να αναπτύξει εφαρμογές για τις τελευταίας τεχνολογίας κινητές συσκευές και ολοένα περισσότεροι κατασκευαστές το υιοθετούν για τις κινητές συσκευές τους.

2.11 Android 4.4 KitKat

Είναι η δέκατη κύρια έκδοση του Android. Ανακοινώθηκε το Σεπτέμβρη του 2013. Αυτή η έκδοση διανεμήθηκε μαζί το Nexus 5 και προορίζεται για μεγαλύτερο εύρος συσκευών από την προηγούμενες εκδόσεις Android. Το ελάχιστο μέγεθος της μνήμης RAM 340MB από 521MB στις προηγούμενες εκδόσεις.

2.12 Android 5.0 Lollipop

Η έκδοση 5.0 παρουσιάστηκε τον Ιούνιο του 2014, το λειτουργικό σύστημα θα είναι διαθέσιμο το Νοέμβριο του 2014 για συσκευές που τρέχουν διανομές του Android. Από τις πιο πολλά υποσχόμενες αλλαγές είναι ο επανασχεδιασμός της διασύνδεσης χρήστη με τη χρήση της φιλοσοφίας σχεδιασμού “material design”. Άλλες αλλαγές περιλαμβάνουν βελτιώσεις στο σύστημα ειδοποιήσεων. Σημαντικές αλλαγές έγιναν και εσωτερικά στην πλατφόρμα, με το Android Runtime να αντικαθιστά επίσημα το Dalvik για βελτιωμένη απόδοση εφαρμογών. Οι αλλαγές έγιναν με σκοπό να βέλτιστη χρήση της μπαταρίας.

Σύνοψη

Στο παραπάνω κεφάλαιο παρουσιάστηκε η ιστορική αναδρομή του λειτουργικού Android, δηλαδή τις εκδόσεις που έχουν κυκλοφορήσει μέχρι σήμερα. Παρακάτω, θα αναλυθούν τα εργαλεία ανάπτυξης που χρησιμοποιήθηκαν για την ολοκλήρωση της εφαρμογής.

ΚΕΦΑΛΑΙΟ 3

Εργαλεία Ανάπτυξης.

Εισαγωγή

Όπως έχουμε δει παραπάνω για την δημιουργία εφαρμογών Android χρησιμοποιείται η γλώσσα προγραμματισμού Java. Ο κάθε προγραμματιστής εφαρμογών Android μπορεί να έχει εγκατεστημένο για την διαδικασία ανάπτυξης οποιοδήποτε από τα τρία πιο δημοφιλή λειτουργικά συστήματα, δηλαδή, Windows, Linux ή Mac OS. Σε αυτό το κεφάλαιο θα γίνει παρουσίαση των εργαλείων που χρησιμοποιήθηκαν για της ανάγκες ανάπτυξης της εφαρμογής.

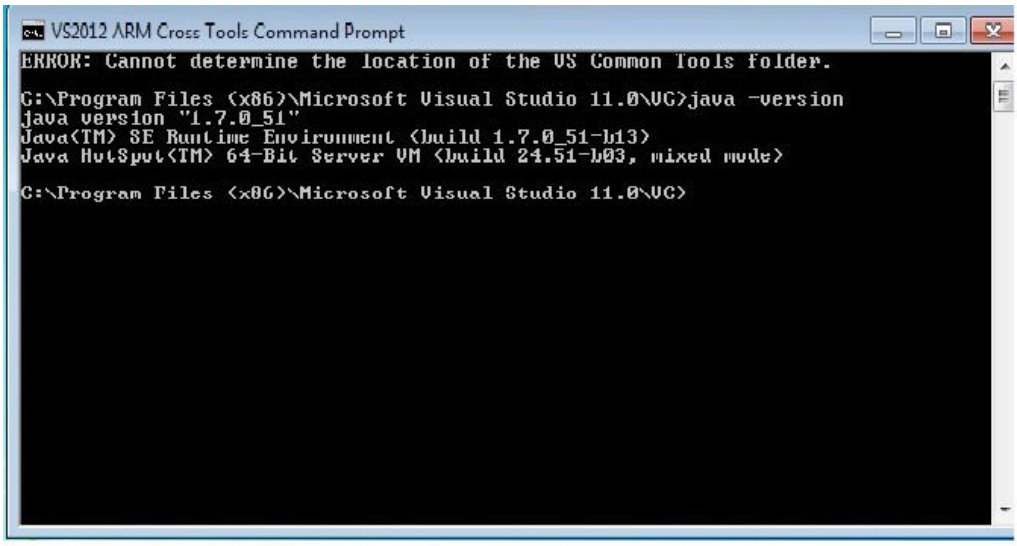
Τα εργαλεία είναι τα εξής:

- Java Development Kit
- Eclipse IDE
- Android SDK
- Android Developer Tool (Eclipse plug – in)

Το λειτουργικό σύστημα στο οποίο έγινε η εγκατάσταση των παραπάνω εργαλείων, για τις ανάγκες της συγκεκριμένης πτυχιακής εργασίας είναι τα Windows 7 στην 64-bit έκδοση τους. Τα εργαλεία που αναφέρονται διατίθενται δωρεάν και η εγκατάσταση τους είναι σχετικά εύκολη.

3.1 Εγκατάσταση Java Development Kit.

Για την ανάπτυξη και δοκιμή οποιασδήποτε Java εφαρμογής είναι απαραίτητη η εγκατάσταση του JDK που παρέχεται από τον ιστότοπο της Oracle. Το JDK είναι υποσύνολο του Java SDK το οποίο είναι απαραίτητο για την ανάπτυξη και εκτέλεση προγραμμάτων java. Όσο αφορά το Android, το JDK είναι αρκετό. Η έκδοση που χρησιμοποιήθηκε στην παρούσα πτυχιακή εργασία είναι η JDK 7. Για να γίνει έλεγχος εάν εγκαταστάθηκε σωστά το JDK ή ποια έκδοση του είναι εγκατεστημένη, αρκεί να πληκτρολογήσουμε την εντολή **java -version** από την γραμμή εντολών. (Εικόνα: ii).



Εικόνα ii

3.2 Εγκατάσταση Eclipse IDE

Η εγκατάσταση του Eclipse IDE (Intergrated Development Enviroment) είναι μία πολύ απλή υπόθεση. Από τον ιστότοπο www.eclipse.org/downloads/ κατεβάζουμε το φάκελο με τα αρχεία του προγράμματος σε όποια θέση επιθυμούμε στο δίσκο. Κατόπιν, “τρέχουμε” το αρχείο εφαρμογής Eclipse, το πρόγραμμα είναι πλήρως λειτουργικό.

Το Eclipse παρέχει δυνατότητες οργάνωσης όλου του κώδικα του project. Ενσωματώνει λειτουργία εικονικών εξυπηρετητών για τον έλεγχο

ορθής λειτουργίας web εφαρμογών. Με την χρήση του ADT plug -in, που θα δούμε παρακάτω, το καθιστά κατάλληλο για την ανάπτυξη Android εφαρμογών.

3.3 Εγκατάσταση Android SDK

Το Android SDK –παρέχει βιβλιοθήκες και εργαλεία ανάπτυξης απαραίτητα για την δημιουργία, τον έλεγχο, την εκτέλεση και αποσφαλμάτωση εφαρμογών Android. Το SDK διατίθεται στον ιστότοπο

https://developer.android.com/sdk/installing/index.html.

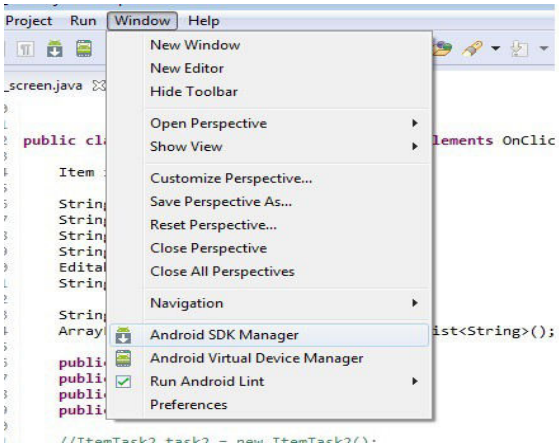
Στο συγκεκριμένο ιστότοπο δίνεται η δυνατότητα να κατεβάσουμε ολόκληρο το πακέτο μαζί με το Eclipse IDE με ενσωματωμένο το ADT και το Android SDK tools. Εάν έχουμε προ εγκατεστημένη κάποια έκδοση του Eclipse δίνεται η δυνατότητα να κατεβάσουμε την standalone version του Android SDK tools. Στην πρώτη περίπτωση η εγκατάσταση είναι πολλή απλή. Το πρώτο βήμα είναι αποσυμπιέσουμε τον φάκελο (adt-bundle-windows-x86_64-20140702) με τα αρχεία που λάβαμε σε όποια θέση επιθυμούμε στον σκληρό δίσκο. Κατόπιν, μέσα στον αποσυμπιεσμένο φάκελο των αρχείων (adt-bundle-windows-x86_64-20140702) εντοπίζουμε το αρχείο εφαρμογής Eclipse και το “τρέχουμε”. Πλέον το Eclipse IDE είναι έτοιμο με ενσωματωμένο το ADT plug-in και το Android SDK tools.

Στη δεύτερη περίπτωση η διαδικασία είναι λίγο πιο μεγάλη αλλά παραμένει σχετικά απλή. Το πακέτο που έχουμε κατεβάσει είναι ένα εκτελέσιμο αρχείο το οποίο πυροδοτεί τη διαδικασία εγκατάστασης, δηλαδή εκκινεί ένα πρόγραμμα εγκατάστασης. Αρχικά, το πρόγραμμα εγκαταστάτης κάνει έλεγχο του Η/Υ εάν υπάρχει ήδη εγκατεστημένο το JDK, εάν όχι τον εγκαθιστά αυτόματα από το διαδίκτυο. Κατόπιν το πρόγραμμα εγκαταστάτης αποθηκεύει αυτόματα το Android SDK tools στον σκληρό δίσκο (ή επιλέγει ο χρήστης όπου επιθυμεί). Μόλις η εγκατάσταση ολοκληρωθεί τότε αυτόματα ανοίγει το πρόγραμμα Android SDK

manager.

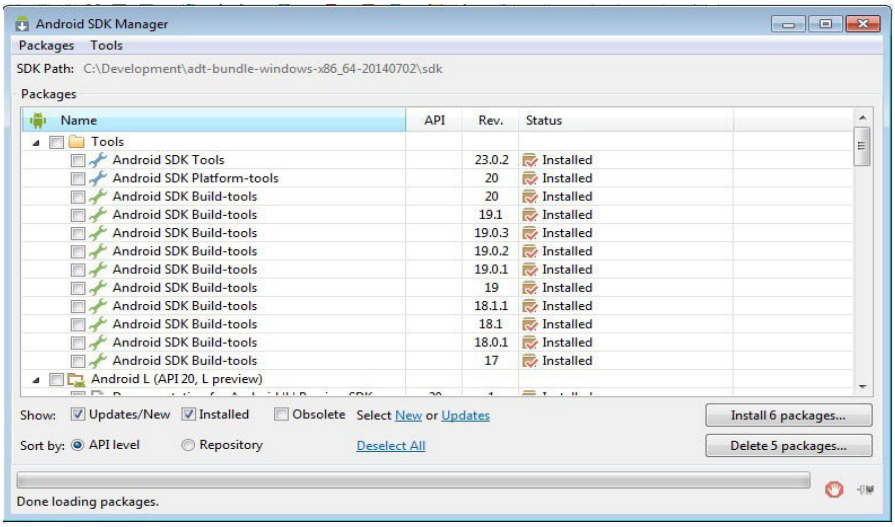
Για να είμαστε σε θέση να αρχίσουμε την δημιουργία εφαρμογών και στις δύο περιπτώσεις πρέπει πρώτα να εγκαταστήσουμε κάποια πακέτα βιβλιοθηκών τα οποία δεν είναι προ εγκατεστημένα με τον Android SDK manager. Η διαδικασία έχει ως εξής:

Ανοίγουμε το Android SDK manager μέσα από το μενού το Eclipse (Εικόνα iii)



Εικόνα: iii

Κατόπιν επιλέγουμε τα εξής: Android SDK tools, Android SDK Platform-tools, Android SDK Build tools και τέλος επιλέγουμε τον φάκελο με την έκδοση του Android που θα εγκαταστήσουμε. Στην συγκεκριμένη πτυχιακή εργασία επιλέχθηκε η έκδοση Android 4.4.2 (API 19). Παρακάτω παρατίθεται εικόνα (Εικόνα iv) του μενού Android SDK manager.

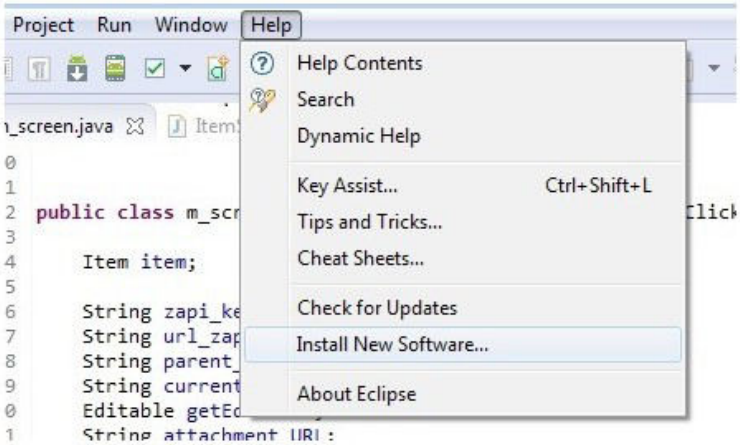


Εικόνα: iv

3.4 Εγκατάσταση του ADT

Όπως αναφέρθηκε παραπάνω υπάρχει η δυνατότητα να εγκατασταθεί όλο το πακέτο μαζί δηλαδή, το Android SDK, με το Eclipse IDE και το ADT plug-in. Σε διαφορετική περίπτωση μπορεί να εγκατασταθεί μέσα από το μενού του Eclipse. Η διαδικασία έχει ως εξής:

Από την μπάρα του μενού επιλέγουμε Help--> Install New Software.. (Εικόνα v)

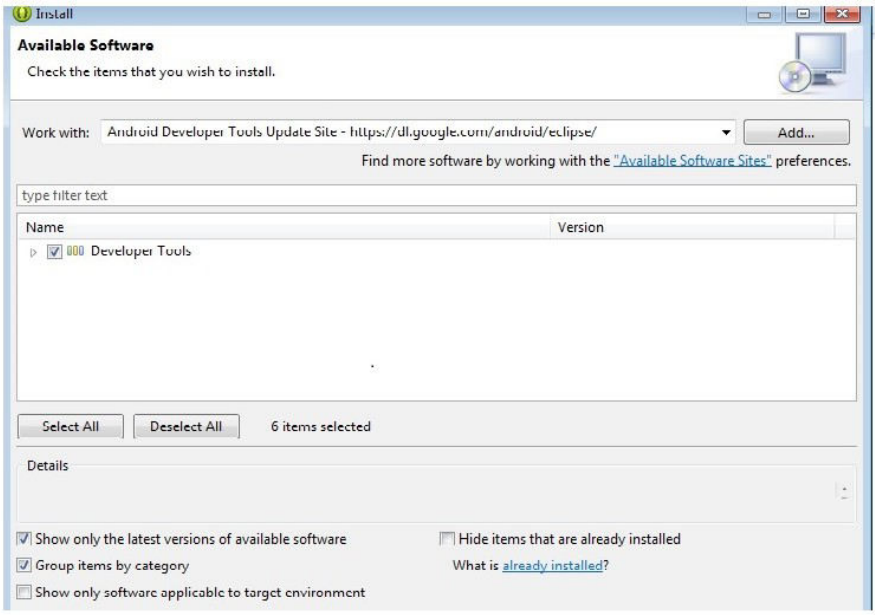


Εικόνα: v

Κατόπιν εμφανίζεται ένα μενού διαλόγου όπου στο πλαίσιο κειμένου, Work with, επιλέγουμε

Android Developer Tools Update Site - http://dl-ssl.google.com/android/eclipse/

και μετά επιλέγουμε το Developer Tools



Εικόνα: vi

Κατόπιν πατάμε Next και αφού συμφωνήσουμε με την άδεια χρήσης πατάμε Finish,ώστε να εκκινήσει η διαδικασία εγκατάστασης. Μόλις τελειώσει η διαδικασία κάνουμε επανεκκίνηση του Eclipse. Πλέον, είμαστε σε θέση να ξεκινήσουμε την διαδικασία ανάπτυξης εφαρμογών Android.

Σύνοψη

Στο παραπάνω κεφάλαιο έγινε παρουσίαση του τρόπου εγκατάστασης εργαλείων τα οποία είναι απαραίτητα για την δημιουργία εφαρμογών Android. Στο επόμενο κεφάλαιο θα γίνει αναφορά στο Zotero και τα προσφερόμενα Web APIs προς του προγραμματιστές εφαρμογών.

ΚΕΦΑΛΑΙΟ 4

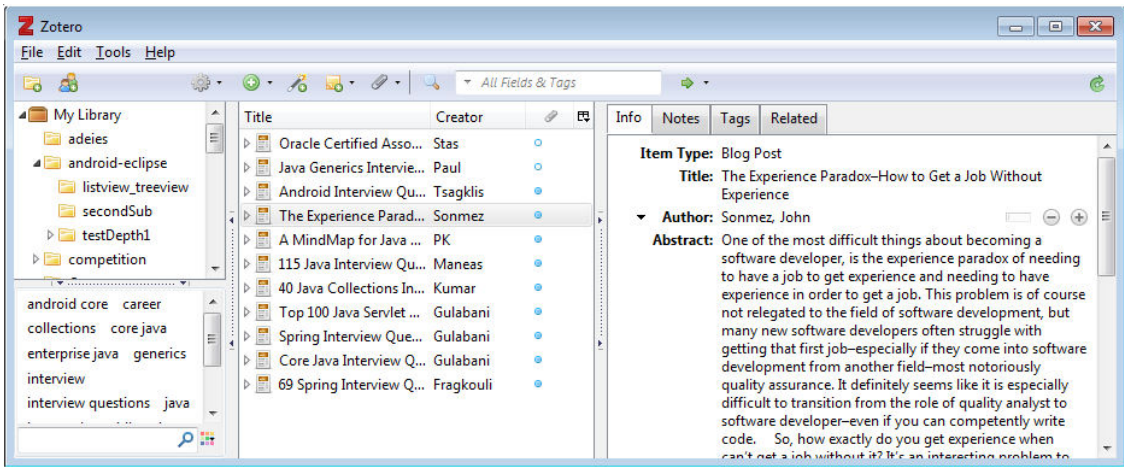
Zotero

Εισαγωγή

Στο παρακάτω κεφάλαιο θα γίνει παρουσίαση του συστήματος διαχείρισης βιβλιογραφίας Zotero. Θα αναφερθούν συνοπτικά η δημιουργία του και η εξέλιξή του. Γίνεται αναφορά στις δυνατότητές του αλλά και στα διαθέσιμα API (Application Programming Interface) που παρέχονται από τον κατασκευαστή, για την υλοποίηση επικοινωνίας με τους διακομιστές του Zotero.

4.1 Τι είναι το Zotero

Το Zotero (zoh – TAIR -oh) είναι ένα ελεύθερο, εύκολο στη χρήση εργαλείο το οποίο βοηθά στην συλλογή, οργάνωση και διαμοιρασμό βιβλιογραφικών πόρων https://www.zotero.org/ .



Εικόνα: vii

Δημιουργήθηκε από το Roy Rosenzweig Center for History and New Media στο Manson University. Ξεκίνησε αρχικά ως πρόσθετο για το πρόγραμμα περιήγησης Mozilla Firefox. Κατόπιν, εξελίχθηκε ως αυτόνομη εφαρμογή με συμβατότητα για τα τρία δημοφιλέστερα λειτουργικά συστήματα (Windows, Linux,Mac OS). Βασικό του χαρακτηριστικό και κύριο πλεονέκτημά του είναι ότι συλλέγει με μεγάλη ευκολία πληθώρα βιβλιογραφικών αναφορών από το διαδίκτυο, μέσω προσθέτων που είναι διαθέσιμα, για τα πιο δημοφιλή προγράμματα περιήγησης.

Το Zotero ανιχνεύει αυτόματα το περιεχόμενο από το πρόγραμμα περιήγησης και μας επιτρέπει να το προσθέσουμε με ένα απλό κλικ στην προσωπική μας βιβλιοθήκη. Όλη η έρευνα είναι προσβάσιμη από ένα ενιαίο φιλικό προς χρήστη περιβάλλον, το οποίο παρέχει δυνατότητα αναζήτησης. Μπορούμε να προσθέσουμε αρχεία PDF, εικόνες, αρχεία ήχου και βίντεο, στιγμιότυπα εικόνων από ιστοσελίδες και άλλα πολλά. Το Zotero αυτόματα δημιουργεί ευρετήριο από το περιεχόμενο κείμενου της βιβλιοθήκης του χρήστη, το οποίο κάνει την αναζήτηση σε όλη την βιβλιοθήκη πολύ εύκολη.

Η οργάνωση τον στοιχείων (items) της έρευνας της βιβλιοθήκης γίνεται σε σε συλλογές (collections). Ο χρήστης μπορεί να δημιουργήσει συλλογές και υποσυλλογές όπως ακριβώς επιθυμεί. Κάθε στοιχείο μπορεί να αποθηκεύεται σε οποιοδήποτε αριθμό από συλλογές ή υποσυλλογές. Στο σύνολό τους τα παραπάνω απαρτίζουν την βιβλιοθήκη του χρήστη. Το Zotero χρησιμοποιεί δεδομένα της βιβλιοθήκης ή της βάσης δεδομένων για να δημιουργήσει ετικέτες (tags). Δίνεται η δυνατότητα και στον χρήστη να δημιουργήσει τις δικές του ετικέτες ορίζοντας τις δικές του λέξεις κλειδί για κάθε στοιχείο της βιβλιογραφίας.

Βασικό χαρακτηριστικό του Zotero είναι ο συγχρονισμός της τοπικής βάσης δεδομένων του με μια αντίστοιχη βάση δεδομένων που φιλοξενείται στους διακομιστές του κατασκευαστή. Το Zotero αυτόματα συγχρονίζει τα δεδομένα, ώστε να είναι πάντα ενημερωμένα ανεξάρτητα την συσκευή που χρησιμοποιείται από τον χρήστη. Σε περίπτωση που γίνει εγκατάσταση της εφαρμογής Zotero σε νέο ηλεκτρονικό υπολογιστή, με το πάτημα ενός κουμπιού γίνεται μεταφορά αντιγράφου ολόκληρης της βιβλιοθήκης από τον διακομιστή δικτύου. Επιπλέον,

εάν δεν υπάρχει εγκατεστημένη η εφαρμογή στη συσκευή, τότε ο χρήστης μπορεί να έχει πρόσβαση στα δεδομένα και την έρευνα του πολύ εύκολα από οποιοδήποτε πρόγραμμα περιήγησης ιστού (wed browser).

Το Zotero παρέχει την δυνατότητα για την δημιουργία ομάδας (group). Κάθε ομάδα μπορεί να μοιραστεί την βιβλιοθήκη της. Οι ομάδες μπορούν να είναι δημόσιες (public) ή ιδιωτικές.

4.2 Zotero Web API

Ο server του Zotero επιτρέπει την πρόσβαση στις βιβλιοθήκες των χρηστών μέσω τον Web API που παρέχει. Το ZoteroApp είναι η εφαρμογή που αναπτύχθηκε ως client. Για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν αιτήματα τα οποία παρέχουν μόνο “read only” πρόσβαση στις online βιβλιοθήκες του Zotero.

Το URL, βάση, για την πραγματοποίηση των API αιτημάτων είναι το:

https://api.zotero.org

Όλα τα αιτήματα πρέπει να είναι **HTTPS**.

Υπάρχουν διάφορες εκδόσεις του API που είναι διαθέσιμες. Η εφαρμογή για την παρούσα πτυχιακή εργασία βασίστηκε κυρίως στην 2η έκδοση. Την στιγμή της συγγραφής του συγκεκριμένου κειμένου η έκδοση 3 έχει γίνει η κύρια. Η επιλογή της έκδοσης μπορεί να γίνει με δύο τρόπους

- Δηλώνοντας την έκδοση ως HTTP header
- Δηλώνοντας την παράμετρο v στο ερώτημα προς τον εξυπηρετητή (v=3).

Πιστοποίηση του χρήστη δεν απαιτείται για δημόσιες βιβλιοθήκες. Η πρόσβαση σε μη δημόσιες βιβλιοθήκες επιτυγχάνεται με την χρήση του πρωτοκόλλου Oauth ή με την προτροπή προς τον χρήστη να δημιουργήσει το δικό του API key για να μπορεί να το χρησιμοποιήσει με αυτές τις υπηρεσίες. Ένα API key μπορεί να συμπεριλαμβάνεται στα ερωτήματα, όπως αναφέραμε και

παραπάνω με την έκδοση με δύο τρόπους, δηλώνοντας ως HTTP header, ή να συμπεριλαμβάνεται ως παράμετρος στο URL ερώτημα.

Τα URL για δεδομένα σε κάποια βιβλιοθήκη του Zotero πρέπει να περιέχουν την παράμετρο **/users/USERID/**. Η παράμετρος users καθορίζει ότι το αίτημα είναι προς μια συκριμένη βιβλιοθήκη χρήστη. Όπου USERID είναι ένας αριθμός ταυτότητα, μοναδικός για κάθε χρήστη. Το USERID λαμβάνεται αυτόματα κατά την εγγραφή του χρήστη στο Zotero. Το Zotero δίνει τη δυνατότητα και για πρόσβαση σε βιβλιοθήκες που ανήκουν σε ομάδα χρηστών. Η αντίστοιχη παράμετρος είναι **/groups/GROUPID/.**

Κατ' επέκταση εάν θέσουμε τη παράμετρο /**items**/ θα έχουμε ένα ολοκληρωμένο αίτημα προς μία βιβλιοθήκη του Zotero. Αντίστοιχες παράμετροι με την σημασία τους παρατίθεται παρακάτω.

- **/items/**: Το σετ με όλα τα στοιχεία της βιβλιοθήκης.
- **/items/top/**: Το σετ με όλα τα στοιχεία κορυφή της βιβλιοθήκης.
- **/items/trash/:**Το σετ με τα στοιχεία στον κάδο.
- **/items/<itemkey>/**: Ένα συγκεκριμένο στοιχείο της βιβλιοθήκης.
- **/items/<itemkey>/children**: Το σετ με τα στοιχεία παιδιά κάτω από ένα συγκεκριμένο στοιχείο.
- **/items/<itemkey>/tag**: Το σετ με τις ετικέτες που σχετίζονται με το συγκεκριμένο στοιχείο.
- **/collections/**: Το σετ με τις συλλογές που οργανώνονται τα στοιχεία.
- **/collections/top**: Το σετ με τις συλλογές κορυφή της βιβλιοθήκης.
- **/collections/<collectionkey>/**: Μία συγκεκριμένη συλλογή.

- **/collections/<collectionkey>/collections/**: Το σετ των υπό – συλλογών μέσα σε μία συγκεκριμένη βιβλιοθήκη.
- **/collections/<collectionkey>/items/**:Το σετ των στοιχείων μέσα σε μία συγκεκριμένη συλλογή.

4.3 Μορφοποίηση αποτελέσματος.

Atom feeds

Αρχικά, πρέπει να γίνει αναφορά στην μορφοποίηση Atom. Το **Atom Syndication Format** είναι μια μορφή της γλώσσας XML που χρησιμοποιείται για web feeds. Τα web feeds επιτρέπουν στο λογισμικό να κάνουν έλεγχο για ενημερώσεις σε μια ιστοσελίδα.Τα web feed παράγονται από ένα εξειδικευμένο πρόγραμμα του ιδιοκτήτη της σελίδας. Κατόπιν τα feed μπορούν να μεταμορφωθούν από διάφορα προγράμματα.

Τα feed περιέχουν entries, τα οποία μπορεί να είναι επικεφαλίδες, κείμενα, περιλήψεις ή και σύνδεσμοι, μαζί με διάφορα μεταδεδομένα.

Η μορφοποίηση Atom αναπτύχθηκε ως εναλλακτική της μορφοποίησης RSS. Με τα Atom feed έγινε προσπάθεια ώστε να υπάρχει συνεχής καινοτομία και ξεπεραστούν εμπόδια συμβατότητας

Το παραγόμενο αποτέλεσμα με τα δεδομένα είναι ένα αρχείο XML (Atom feed). Για την καλύτερη επεξεργασία των αποτελεσμάτων υπάρχει η δυνατότητα αλλαγής της μορφοποίησης του αποτελέσματος (π.χ. Json, bib). Στη συγκεκριμένη πτυχιακή εργασία επιλέχθηκε η αναπαράσταση των αποκρίσεων σε μορφή Atom και JSON. Η μορφή δεδομένων JSON είναι εύκολη για ανάγνωση και γράψιμο για τον άνθρωπο και εύκολο για ανάλυση και δημιουργία από μηχανές. Εκτενέστερη αναφορά για την μορφή δεδομένων JSON γίνεται στο Κεφάλαιο 4. Μπορούμε να επιτύχουμε μία απόκριση json με την παράμετρο **/?format=json.** Σημειώνεται ότι στην έκδοση 3 του zotero η default μορφοποίηση του αποτελέσματος είναι JSON.

Για παράδειγμα το παρακάτω URL επιστρέφει όλα τα στοιχεία της βιβλιοθήκης σε μορφή JSON .

https://api.zotero.org/users/1737513/items/?format=json

Υπάρχει διαθέσιμη παράμετρος για την υλοποίηση αναζήτησης στην βιβλιοθήκη του Zotero. Στη συγκεκριμένη εργασία χρησιμοποιήθηκε η παράμετρο **q**, με την οποία πραγματοποιείται γρήγορη αναζήτηση στα πεδία των τίτλων και του των δημιουργών σε κάθε στοιχείο της βιβλιοθήκης ξεχωριστά. Επίσης, μπορεί να γίνει χρήση της παραμέτρου **qmode** ώστε να αλλάξει ο τρόπος της αναζήτησης, έτσι με αυτό τον τρόπο μπορούμε να κάνουμε αναζήτηση σε κείμενο με την τιμή της παραμέτρου να έχει οριστεί σε “everything”. Μέχρι στιγμής η γρήγορη αναζήτηση δεν είναι διαθέσιμη για άλλα πεδία.

Άλλες παράμετροι αναζήτησης είναι οι “itemType” και “tag” οι οποίες υποστηρίζουν μόνο boolean αναζητήσεις. Άλλες διαθέσιμες παράμετροι είναι “itemKey” όπου μπορεί να υπάρχει μέχρι και 50 τιμές κλειδιών σε ένα ερώτημα. Τελευταία η παράμετρος “since”, όπου επιστρέφει τα στοιχεία που έχουν μετατραπεί μετά από την συγκεκριμένη έκδοση της βιβλιοθήκης. Η προκαθορισμένη τιμή είναι 0 (μηδέν).

4.4 HTTP Responses

Τέλος, γίνεται αναφορά στην κατάσταση αιτημάτων. Ένα επιτυχημένο read αίτημα θα επιστρέψει ως **status code: 200 OK.** Για οποιοδήποτε read ή write αίτημα ο εξυπηρετητής μπορεί να επιστρέψει απάντηση όπως:

- **400 Bad Request**
- **404 Not Found**
- **405 Method Not Allowed**
- **500 Internal Server Error**

- **503 Service Unavailable**
- **429 Too Many Requests**, καθώς υπάρχει όριο στον αριθμό των αιτημάτων.

Σύνοψη

Στο παραπάνω κεφάλαιο έγινε σύντομη αναφορά στο Zotero και τις δυνατότητες που προσφέρει στο χρήστη αλλά και στον προγραμματιστή. Οι παράμετροι που παρουσιάστηκαν αφορούν κυρίως την παρούσα πτυχιακή εργασία. Όπως παρατηρείτε το zotero παρέχει πλούσια τεκμηρίωση που το καθιστά πολύ ενδιαφέρον αντικείμενο για τον κάθε προγραμματιστή.

ΚΕΦΑΛΑΙΟ 5

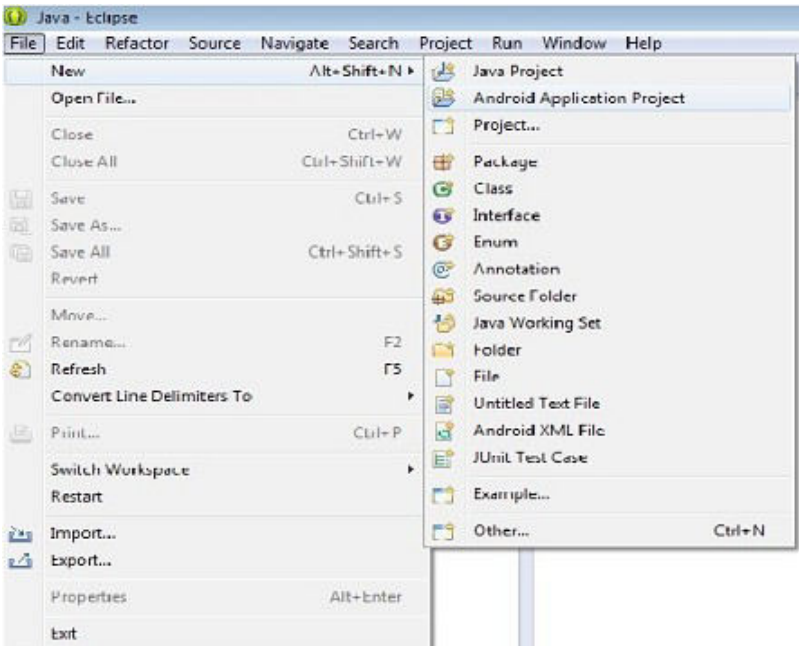
Ανάπτυξη Εφαρμογής

Εισαγωγή

Στο κεφάλαιο που ακολουθεί θα αναλυθεί η διαδικασία ανάπτυξης της εφαρμογής. Αρχικά, θα γίνει αναφορά στον τρόπο που δημιουργούμε ένα Android project στα συστατικά του αλλά και τις ρυθμίσεις που είναι απαραίτητες για να εκτελεστεί. Κατόπιν, γίνεται αναφορά στα στάδια ανάπτυξης της εφαρμογής.

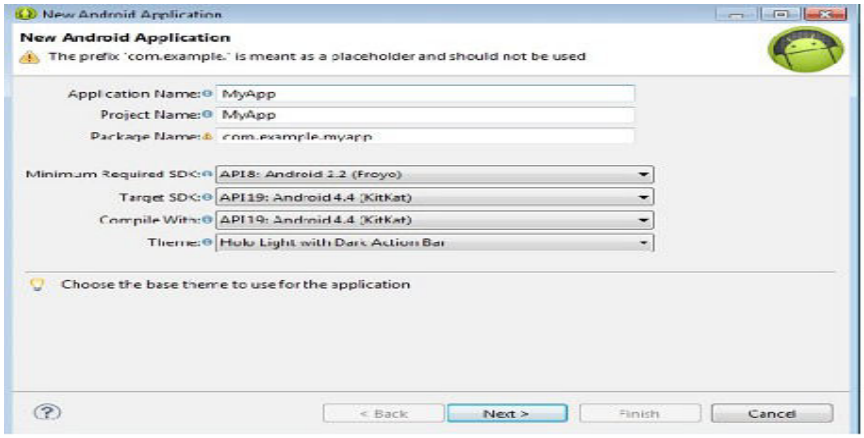
5.1 Δημιουργία Android Project

Αφού πρώτα εγκαταστήσουμε επιτυχημένα το Ecplice IDE με το ADT-plug in και το Android SDK είμαστε έτοιμοι να δημιουργήσουμε το πρώτο Android project. Αρχικά από το μενού του Ecplice πατάμε File-->New-->Android Application Project



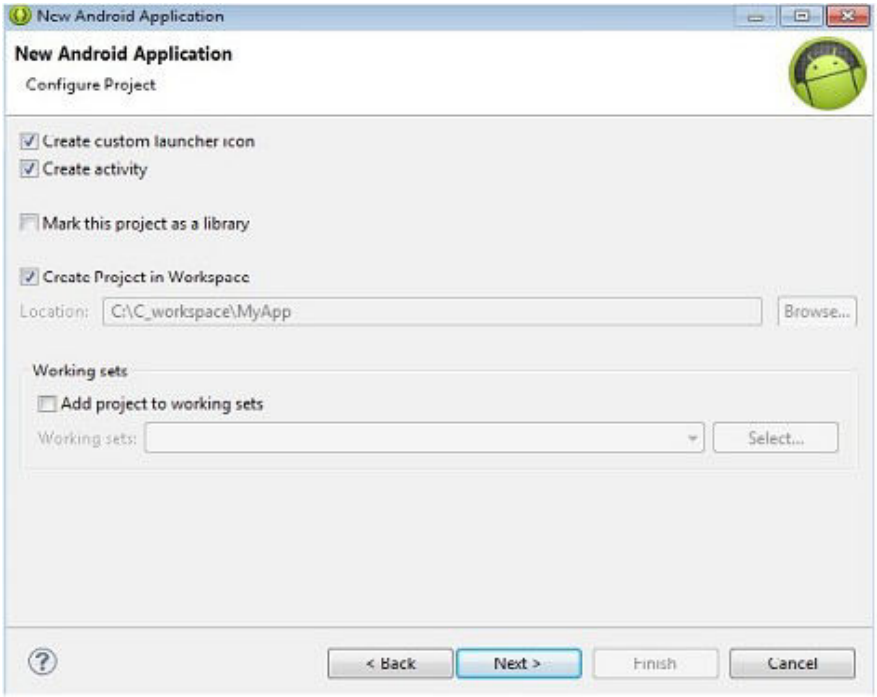
Εικόνα: viii

Στη συνέχεια μας εμφανίζεται το παράθυρο διαλόγου όπου δηλώνουμε το όνομα της εφαρμογής, το όνομα του project και το όνομα του πρώτου πακέτου που δημιουργείται αυτόματα. Δίνεται η δυνατότητα να επιλέξουμε την ελάχιστη και τη μέγιστη έκδοση Android με την οποία θα είναι συμβατή η εφαρμογή αλλά και την τρέχουσα έκδοση (Εικόνα: ix), στη συνέχεια πατάμε κουμπί next.



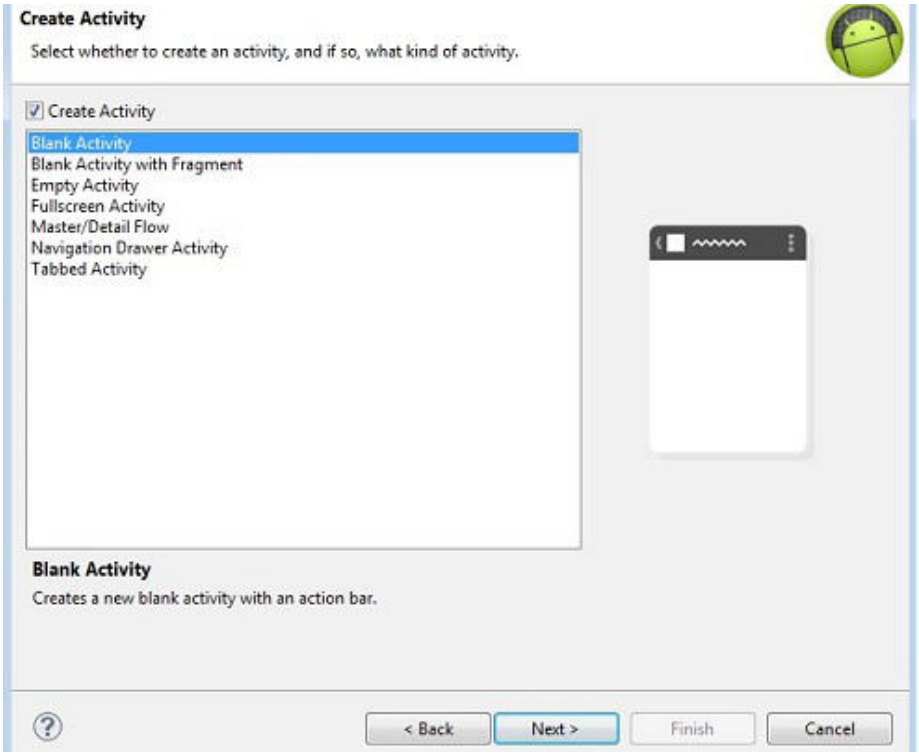
Εικόνα: ix

Στο επόμενο παράθυρο διαλόγου επιλέγουμε την τοποθεσία στο δίσκο που θα είναι αποθηκευμένο το project και πατάμε next (Εικόνα: x), κατόπιν στο επόμενο παράθυρο επιλέγουμε την εικόνα της εφαρμογής και διάφορες ρυθμίσεις σχετικά με αυτή(μέγεθος, σχήμα κ.α.).



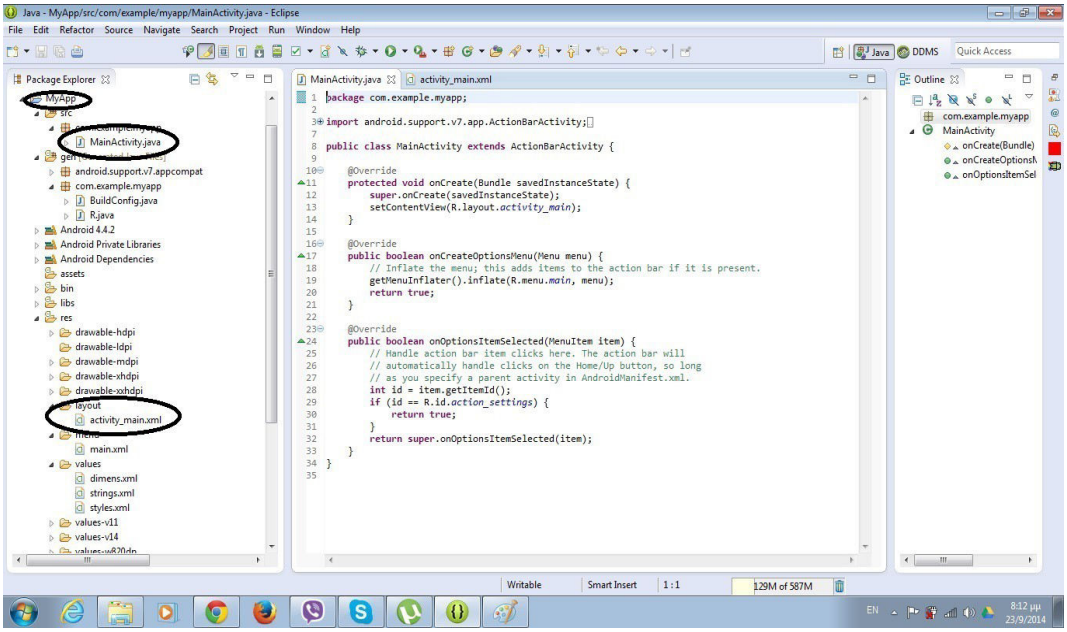
Εικόνα: x

Στο επόμενο παράθυρο διαλόγου επιλέγουμε Blank Activity, δηλαδή το πρώτο layout της εφαρμογής που δημιουργείται θα είναι κενό, χωρίς κανένα συστατικό. Κατόπιν, το επόμενο παράθυρο είναι πολύ σημαντικό, καθώς σε αυτό δηλώνουμε το όνομα της πρώτης μας δραστηριότητας, δηλαδή της πρώτης κλάσης που θα δημιουργηθεί. Ακόμα, δηλώνουμε και το όνομα του πρώτου layout της εφαρμογής, το οποίο όπως προαναφέρθηκε είναι κενό εκτός από την έκφραση *Hello World*.



Εικόνα: xi

Τέλος, πατάμε Finish και εμφανίζεται μπροστά μας το project με όλα τα συστατικά του (Εικόνα: xii).



Εικόνα: xii

5.2 Εσωτερικό εφαρμογής Android

Κάθε εφαρμογή αποτελείται από ένα σύνολο αρχείων δομημένων σε μορφή project. Από αυτά τα αρχεία προέρχεται το τελικό .apk αρχείο το οποίο μπορούμε να το εγκαταστήσουμε στην συσκευή μας. Παρακάτω, γίνεται αναφορά στα πιο σημαντικά συστατικά ενός Android project.

5.2.1 Το αρχείο AndroidManifest.xml

Κάθε εφαρμογή Android πρέπει να περιέχει το αρχείο **AndroidManifest.xml** με ακριβώς αυτό το όνομα. Σε αυτό βρίσκονται καταχωρημένες οι πιο σημαντικές πληροφορίες της εφαρμογής, απαραίτητες για την εκτέλεση της εφαρμογής. Σε αυτό το αρχείο δηλώνεται το κύριο πακέτο της εφαρμογής. Το κύριο πακέτο λειτουργεί ως αναγνωριστικό της εφαρμογής από το λειτουργικό σύστημα. Περιέχει πληροφορίες για τις εκδόσεις των APIs που χρησιμοποιούνται, τον αριθμό έκδοσης της εφαρμογής.

Είναι πολύ σημαντικό ότι σε αυτό το αρχείο περιγράφονται τα συστατικά της εφαρμογής όπως, οι δραστηριότητες (Activities), πάροχος περιεχομένου (Content Provider), υπηρεσίες (Services). Δηλώνονται οι κλάσεις που υλοποιούν κάθε συστατικό και κάνει γνωστές τις δυνατότητες τους. Ακόμα δηλώνονται οι άδειες που είναι απαραίτητες ώστε να μπορεί η εφαρμογή να έχει πρόσβαση σε προστατευμένα API, αλλά και άδειες που είναι απαραίτητες από άλλους για να έχουν πρόσβαση στα συστατικά της εφαρμογής.

5.2.2 Δομή καταλόγων ενός Android Project.

- **gen:** Ο φάκελος gen περιέχει αρχεία τα οποία δημιουργούνται αυτόματα. Για παράδειγμα το R.java το οποίο είναι μια κλάση που περιέχει αναφορές από διάφορους πόρους. Αυτή η κλάση δημιουργείται αυτόματα από το Eclipse IDE και οι αλλαγές από τον χρήστη είναι περιττές.

- **src:** Προέρχεται από το source. Στο συγκεκριμένο φάκελο περιέχονται όλα τα αρχεία java, τα οποία δομούνται μέσα στο βασικό πακέτο ή και σε περισσότερα πακέτα της εφαρμογής.

- **res:** Προέρχεται από το resources. Εσωτερικά αυτού του φακέλου περιέχονται πόροι τις εφαρμογής όπως αρχεία .xml, Strings, εικόνες κ.α. Όλα τα παραπάνω δομούνται σε υπό φακέλους με βάση τον τύπο τους. Χαρακτηριστικοί είναι ο res/layout, όπου περιέχει τα layout που θα χρησιμοποιήσει η εφαρμογή. Ο res/values όπου μέσα σε αυτό ορίζονται συμβολοσειρές, χρώματα, στατικοί πίνακες, διαστάσεις. Για να μην υπάρχει σύγκρουση τύπων κάθε τύπος αποθηκεύεται σε διαφορετικό αρχείο, έτσι υπάρχει res/value/string.xml, res/value/colours.xml κ.τ.λ. Άλλος ένας υποκατάλογος είναι res/menu όπου περιέχονται οι πόροι που θα χρησιμοποιήσει η εφαρμογή για τα μενού (option menu, contex menu)

- **bin:** Στο φάκελο bin είναι η περιοχή που χρησιμοποιείται από τον compiler για να προετοιμάσει τα αρχεία στο τελικό πακέτο .apk αρχείο. Αυτό

περιλαμβάνει τα αρχεία με κώδικα java σε .class αρχεία και την εισαγωγή σε μορφή zip όλων των πόρων της εφαρμογής.

- **libs:** Περιέχει αρχεία εξωτερικών βιβλιοθηκών
- **assets:** Αυτός ο φάκελος περιέχει αρχεία χωρίς κάποια συγκεκριμένη μορφή χωρίς κάποιες δυνατότητες. Είναι απλά μια μη δομημένη ιεραρχία από αρχεία, που μας επιτρέπει να βάλουμε οτιδήποτε θέλουμε εκεί και αργότερα να ανακτήσουμε.

5.2.3 Εκτέλεση Android project

Τέλος, για να τρέξουμε την εφαρμογή που μόλις δημιουργήσαμε πρέπει να ορίσουμε και την συσκευή που θα εκτελεστεί το πρόγραμμα. Μπορεί να γίνει με παραπάνω από έναν τρόπους.

Αρχικά, υπάρχει η δυνατότητα να εξάγουμε το .apk αρχείο από το φάκελο bin του project και να το εγκαταστήσουμε σε μια συσκευή με λειτουργικό σύστημα Android. Η εφαρμογή συμπεριφέρεται όπως και οι υπόλοιπες εφαρμογές της συσκευής, δηλαδή εμφανίζεται το εικονίδιο το οποίο είχαμε επιλέξει όταν ανοίγαμε το project και με ένα πάτημα ανοίγει και βλέπουμε το προκαθορισμένο layout με την έκφραση Hello World.

Πολύ εύκολα με τη χρήση μια Android συσκευής μπορούμε να εγκαταστήσουμε και να γίνει έλεγχος της λειτουργίας της εφαρμογής. Χρειάζεται να συνδεθεί η συσκευή με τον υπολογιστή μέσω usb καλωδίου. Πρέπει να ρυθμιστεί η συσκευή ώστε να επιτρέπει την αποσφαλμάτωση από usb (Ρυθμίσεις -->επιλογές για προγραμματιστές-->εντοπισμός σφαλμάτων μέσω usb για λειτουργικό 4.0 και άνω). Κατόπιν, από το μενού του Eclipse επιλέγουμε Run-->Run As –>Application project, τότε το Eclipse εγκαθιστά την εφαρμογή στην συνδεδεμένη συσκευή. Με τον παραπάνω τρόπο ενδέχεται εάν χρησιμοποιούνται Windows να χρειάζεται εγκατάσταση των κατάλληλων USB οδηγών (drivers) για την συσκευή.

Εξαιρετικά χρήσιμη είναι η επιλογή δημιουργίας μιας εικονικής συσκευής Android. Το Android SDK παρέχει τον emulator δηλαδή, μία εικονική συσκευή η οποία τρέχει στον υπολογιστή. Ο emulator μπορεί να μιμηθεί όλες τις λειτουργίες μίας συσκευής εκτός από το να πραγματοποιήσει τηλεφωνικές κλήσεις. Η εικονική συσκευή έχει εγκατεστημένο μία πλήρη έκδοση του λειτουργικού Android. Για να χρησιμοποιήσουμε τον emulator χρειάζεται πρώτα να δημιουργηθεί η εικονική συσκευή. Αυτή η δυνατότητα παρέχεται από τον **Android Virtual Device Manager**. Κάθε εικονική μηχανή λειτουργεί ανεξάρτητα, με τον δικό της ιδιωτικό χώρο αποθήκευσης, την SD κάρτα κτλ. Για να έχουμε πρόσβαση στο AVD Manager, από το μενού του eclipse επιλέγουμε **Window--> Android Virtual Device Manager.** Στο πλαίσιο διαλόγου που μας εμφανίζεται επιλέγουμε μία συσκευή από τη λίστα και δημιουργούμε την αντίστοιχη εικονική και μπορούμε να επιλέξουμε τις ρυθμίσεις υλικού και λογισμικού που θα προσομοιωθούν.

5.3 Ανάπτυξη του ZoteroApp

Παραπάνω έγινε αναφορά στη δημιουργία του project και παρουσιάστηκαν συνοπτικά τα συστατικά του. Σε αυτό το σημείο θα αναφερθούμε στην ανάπτυξη της εφαρμογής ZoteroApp. Θα αναπτυχθούν περαιτέρω συστατικά που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής και είναι θεμελιώδη για κάθε εφαρμογή Android.

5.3.1 Η κλάση Activity

Αρχικά θα αναφερθούμε στις δραστηριότητες ή Activities. Η Activity είναι το συστατικό το οποίο παρέχει τη μορφή της οθόνης με την οποία αλληλεπιδρά ο χρήστης. Κάθε δραστηριότητα είναι ένα δοσμένο παράθυρο στο οποίο αναπαρίσταται η διασύνδεση χρήστη. Συνήθως ταιριάζει με το μέγεθος της οθόνης,αλλά σε μερικές περιπτώσεις μπορεί να είναι μικρότερο από την οθόνη. Μία εφαρμογή συνήθως αποτελείται από πολλές δραστηρίοτητες. Είθισται, μια

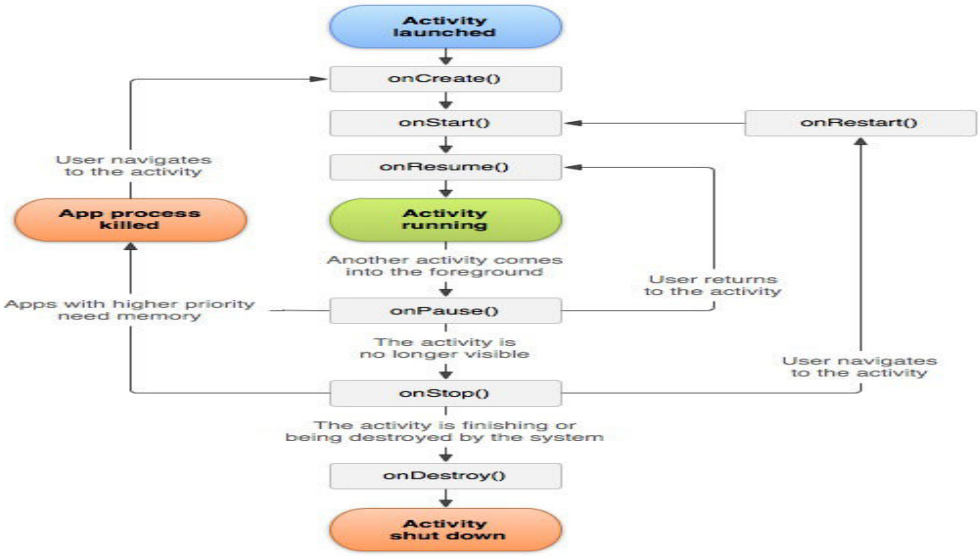
δραστηριότητα μέσα σε μια εφαρμογή να προσδιορίζεται ως κύρια δραστηριότητα ή πιο σωστά main Activity, η οποία εμφανίζεται στο χρήστη όταν η εφαρμογή ξεκινά. Κάθε Activity μπορεί να ενεργοποιήσει τη λειτουργία μιας άλλης Activity, ώστε να επιτελέσει διαφορετικές λειτουργίες. Δεδομένα μπορούν να μεταφέρονται ανάμεσα στις Activities μέσω της κλάσης **Intent,** την οποία αναλύουμε παρακάτω. Κάθε φορά που ξεκινά η νέα δραστηριότητα η προηγούμενη σταματά, αλλά το σύστημα την βάζει μέσα σε μια στοίβα. Εάν ο χρήστης πατήσει το back button τότε η προηγούμενη η Activity προωθείται ξανά.

Όταν μια Activity έχει σταματήσει επειδή μια άλλη έχει έρθει στο προσκήνιο τότε σημειώνεται αυτή η αλλαγή στην κατάστασή της με την κλήση των μεθόδων που διαχειρίζονται των κύκλο ζωής των Activities. Για να δημιουργηθεί μια Activity πρέπει να δημιουργήσουμε μια υποκλάση της κλάσης Activity. Μέσα στην υποκλάση πρέπει να υλοποιείται τουλάχιστον η μέθοδος **onCreate()**.

Η μέθοδος onCreate() είναι μια από τις μεθόδους που καλούνται όταν στη διάρκεια του κύκλου ζωής μιας Activity. Μόλις μια Activity ενεργοποιηθεί τότε καλείται αρχικά η μέθοδος onCreate() και μετά με αυτή την σειρά οι μέθοδοι on-Start() και onResume προτού φτάσει σε κατάσταση που να μπορεί να αλληλεπιδρά με τον χρήστη (Activity Running). Εάν μια Activity πρόκειται να τερματίσει την λειτουργία της τότε θα κληθούν οι μέθοδοι onPause(), onStop() και τελικά η μέθοδος onDestroy() που τερματίζει οριστικά την Activity. Η αλλαγή στην κατάσταση του κύκλου ζωής της εφαρμογής εξαρτάται από τις επιλογές του χρήστη αλλά και από το ίδιο το Android. Για παράδειγμα ο χρήστης μπορεί να πατήσει το κουμπί, για παράδειγμα το Home και η εφαρμογή να μπει στο παρασκήνιο. Σε περίπτωση που μια συσκευή δεν έχει μεγάλη μνήμη το Android μπορεί να αποφασίσει να τερματίσει μια Activity που είδη βρίσκεται σε κατάσταση paused. Υπάρχει και η περίπτωση που μια Activity έχει σταματήσει την λειτουργία (stopped state) της και πρόκειται να ξεκινήσει πάλι την λειτουργία της (start state), τότε καλείται η μέθοδος onRestart().

Συνοπτικά οι λειτουργίες που επιτελεί η κάθε μέθοδος του κύκλου ζωής μιας Activity:

- onCreate(): Καλείτε όταν μια Activity δημιουργείται, ετοιμάζει την προβολή του περιεχομένου της δραστηριότητας.
- onStart(): Η Activity πρόκειται να γίνει ορατή αλλά δεν είναι ακόμα έτοιμη να αλληλεπιδράσει με τον χρήστη.
- onResume(): Η Activity είναι ορατή και πρόκειται να αλληλεπιδράσει με τον χρήστη
- onPause(): Καλείται όταν η Activity πρόκειται να γίνει μη ορατή. Σε αυτή την περίπτωση τερματίζονται λειτουργίες που είναι στο προσκήνιο, όπως ένα animation.
- onStop(): Η Activity δεν είναι πλέον ορατή στον χρήστη. Σε περίπτωση το Android αποφασίσει ότι πρέπει να τερματίσει την λειτουργία μιας Activity τότε αυτή η μέθοδος παραλείπεται.
- onRestart(): Καλείτε όταν μια Activity έχει σταματήσει (stopped state) και πρόκειται να ξεκινήσει πάλι την λειτουργία της (start state).
- onDestroy(): Καλείται όταν μια Activity πρόκειται να τερματιστεί οριστικά. Για να ξεκινήσει την λειτουργία της πρέπει να δημιουργηθεί ξανά.



Εικόνα: xiii

Πριν προχωρήσουμε στην ενότητα συνοψίζουμε επιγραμματικά τα εξής.

- Οι Activities είναι βασικό δομικό στοιχείο των Android εφαρμογών
- Καθώς ο χρήστης περιηγείται στις διασυνδέσεις επιφάνειας, εκκινούν και οι Activities η μία μετά την άλλη.
- Κάθε Activity έχει τον δικό της κύκλο ζωής ο οποίος είναι ανεξάρτητος από τις άλλες Activities της εφαρμογής.
- Το Android κρατά ένα γραμμικό ιστορικό των Activities που έχει επισκεφθεί ο χρήστης (task backstack).

Η διασύνδεση χρήστη (User Interface) για μία δραστηριότητα παρέχεται από από την ιεραρχία των όψεων, δηλαδή τα αντικείμενα **Views.** Αυτά τα αντικείμενα προέρχονται από την κλάση **View**. Κάθε view control δεσμεύει ένα ορισμένο χώρο στο παράθυρο της δραστηριότητας και μπορεί να αλληλεπιδρά στις ενέργειές του χρήστη.

Το Android παρέχει ένα μεγάλο αριθμό από έτοιμα views τα οποία μπορούν να συνθέσουν ένα layout. Τα layouts είναι views τα οποία προέρχονται από **ViewGroup** το οποίο παρέχει ένα μοναδικό μοντέλο layout για τα view παιδιά του.

Ο πιο συνηθισμένος τρόπος για να οριστεί ένα layout με τη χρήση των views είναι μέσω ενός XML layout αρχείου, το οποίο είναι αποθηκευμένο στους πόρους της εφαρμογής. Με αυτό τον τρόπο μπορούμε να χειριστούμε την σχεδίαση της εφαρμογής ξεχωριστά από τον κώδικά της, ο οποίος καθορίζει την συμπεριφορά της. Μπορούμε να θέσουμε το layout ως διασύνδεση χρήστη με την μέθοδο **setContentView()** περνώντας ως παράμετρο το id του συγκεκριμένου layout.

Σε κάθε εφαρμογή Android, οι πόροι όπως εικόνες, strings κ.α. πρέπει να εξωτερικεύονται και να είναι χωριστά από τον κώδικα της εφαρμογής. Με αυτό τον

τρόπο μπορούμε να τους διατηρούμε και να του επεξεργαζόμαστε ανεξάρτητα. Δίνεται η δυνατότητα να ομαδοποιήσουμε τους πόρους δίνοντάς τους διαφορετικό όνομα και να χρησιμοποιούνται για συγκεκριμένες ρυθμίσεις της συσκευής. Για παράδειγμα ο φάκελος layout-land, στον οποίο είναι αποθηκευμένο το layout της δεύτερης Activity και εμφανίζεται όταν η συσκευή είναι σε θέσης landscape mode (οριζόντιο προσανατολισμό).Επιπλέον, είναι σημαντικό να παρέχονται εναλλακτικοί πόροι για διαφορετικές συσκευές με διαφορετικά χαρακτηριστικά. Στο χρόνο εκτέλεσης, το Android χρησιμοποιεί τους κατάλληλους πόρους, βασιζόμενο στα χαρακτηριστικά της συσκευής. Αφού γίνει η ανεξαρτητοποίηση των πόρων από τον κώδικα, η πρόσβαση σε αυτούς γίνεται με την χρήση του id, τα οποία δημιουργούνται αυτόματα ως αναφορές στο αρχείο R class. Τα αρχεία οργανώνονται στον υποφάκελο της εφαρμογής /res. Μέσα στον res υπάρχουν υποφάκελοι που περιέχουν τους πόρους.

Οι πιο σημαντικοί από αυτούς είναι οι εξής:

- **layout:** περιέχει αρχεία XML τα οποία καθορίζουν την διεπιφάνεια χρήστη.
- **menu:** περιέχει XML αρχεία τα οποία καθορίζουν τα μενού της εφαρμογής.
- **values:** περιέχει απλές τιμές, όπως strings, integers. Κάθε αρχείο σε αυτό τον φάκελο αποτελεί ένα ξεχωριστό πόρο. Για παράδειγμα ένα στοιχείο string δημιουργεί R.string πόρος.

Όπως σημειώθηκε και πιο πάνω κάθε δραστηριότητα (Activity) είναι ουσιαστικά και μία οθόνη με την εμφανιζόμενη διεπιφάνεια χρήστη. Στην εφαρμογή αυτή η οθόνη αναπαρίσταται μέσω της δραστηριότητας MainZoteroActivity.java. Πηγαίνοντας στον κώδικα της δραστηριότητας, παρατηρούμε τις μεθόδους onCreate() και onCreateOptionsMenu() συνθέτοντας τον κώδικα που παρουσιάζεται παρακάτω.

```
	public class MainZoteroActivity extends Activity implements
OnClickListener {


	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main_zotero);
}


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

Η κλάση MainActivity πρέπει να επεκτείνει την κλάση Activity. Στην πρώτη μέθοδο ορίζεται το layout που θα εμφανιστεί κατά την δημιουργία της δραστηριότητας. Το layout βρίσκεται στο φάκελο layouts κάτω από τον φάκελο res. Για να πραγματοποιηθεί η συγκεκριμένη λειτουργία απαιτείται η βιβλιοθήκη android.os.

Στην δεύτερη μέθοδο μπορούμε να εισάγουμε τις επιλογές που θα εμφανίζονται αν ο χρήστης κατά την περιήγιση του στην εφαρμογή πατήσει το κουμπί Menu. Για την χρήση αυτής της μεθόδου γίνεται απαραίτητη η βιβλιοθήκη android.view. Η βιβλιοθήκη αυτή περιέχει κλάσεις διαχείρισης της οθόνης. Με την χρήση αυτής της κατηγορίας βιβλιοθηκών μας δίνεται για παράδειγμα η δυνατότητα να επεξεργαστούμε τις διαστάσεις της οθόνης.

Η διασύνδεση χρήστη σε αυτή την δραστηριότητα είναι πολύ απλή. Περιέχει έξι συστατικά view, δηλαδή ένα κουμπί (Button) μία εικόνα (ImageView) και δύο περιοχές κειμένου (edittext) με τις περιγραφές τους (TextViews). Όπως αναφέραμε το layout δημιουργείται μέσω τη γλώσσας σήμανσης XML, ώστε να

αποσυνδεθεί η μορφή της παρουσίασης από την λογική της εφαρμογής. Τονίζεται ότι τα κείμενα που υπάρχουν στις όψεις της εκάστοτε οθόνης δεν ορίζονται μέσα στον κώδικα της Java αλλά αντλούνται μέσα από τα αρχεία των Strings που βρίσκονται στον φάκελο res/values (πόροι της εφαρμογής). Αυτό είναι καλή πρακτική για να πετύχουμε την διεθνοποίηση της εφαρμογής. Κάθε συστατικό που ορίζουμε στην xml του ορίζουμε και ένα μοναδικό χαρακτηριστικό ως αναγνωριστικό το **id.** Το επόμενο βήμα είναι να συνδέσουμε τα συστατικά της διασύνδεσης χρήστη με τον κώδικα. Αυτό επιτυγχάνεται με τη χρήση της μεθόδου findViewById(). Παρακάτω παρουσιάζεται το κομμάτι του κώδικα που συνδέει το κουμπί login της οθόνης με τον κώδικα.

```
bttn_next = (Button)findViewById(R.id.Login);
```

Η μεταβλητή bttn_next είναι τύπου Button, μία κλάση που περιέχεται στο πακέτο android.app.widget. Το id όμως του κάθε συστατικού είναι μεταβλητή τύπου integer. Για αυτό είναι απαραίτητο η μετατροπή τύπων (casting) όπως παρατηρούμε. Η λειτουργία που επιτελεί το κουμπί είναι εξής: πρώτα αποστέλλει στον server του Zotero ένα GET HTTPS Request ζητώντας όλες τις συλλογές του χρήστη και κατόπιν μας μεταφέρει στην επόμενη δραστηριότητα, ενώ παράλληλα μεταφέρει τα δεδομένα που έχει αποκομίσει.

Για να επιτευχθεί η σύνδεση με την βιβλιοθήκη του χρήστη που είναι αποθηκευμένη στον server, πρέπει πρώτα να συνταχθεί το αίτημα GET. Σε αυτό το αίτημα πρέπει να περιλαμβάνεται το USERID και το USERKEY τα οποία είναι μοναδικά για κάθε χρήστη, όπως αναφέρεται και στο Κεφάλαιο 3. Αρχικά, ο χρήστης πρέπει να πληκτρολογήσει το USERID και το USERKEY. Κάθε φορά που θα επιθυμεί ο χρήστης να συνδεθεί με την βιβλιοθήκη θα πρέπει να πληκτρολογεί το USERID, ενώ το USERKEY είναι θα αποθηκεύεται μόνιμα στην συσκευή του και η εφαρμογή θα το αντλεί από την αυτήν. Σε περίπτωση που θέλουμε σύνδεση ως άλλος χρήστης, τότε με το πάτημα του κουμπιού “change user” δίνεται η δυνατότητα να πληκτρολογήσουμε νέο USERKEY το οποίο θα αντικαταστήσει το προηγούμενο. Η αποθήκευση τον USERID και USERKEY γίνεται στην εσωτερική

μνήμη της συσκευής (Internal Storage). Τα αρχεία που αποθηκεύονται είναι ιδιωτικά στην εφαρμογή ZoteroApp, δεν έχουν πρόσβαση σε αυτά άλλες εφαρμογές αλλά ούτε και ο χρήστης.

Για τον χειρισμό των click σε κάποιο συστατικό της διεπιφάνειας χρήστη υλοποιούμε την διασύνδεση **OnClickListener**. Η συγκεκριμένη διασύνδεση περιέχει μόνο μία μέθοδο την **onClick()**, η οποία καλείται όταν υπάρχει κάποιο συμβάν click. Παρακάτω το απόσπασμα κώδικα από την onClick().

```
public void onClick(View v){

        if(v.getId()==R.id.login){

              performItem();

}
```

Η μεταφορά δεδομένων, από δραστηριότητα σε δραστηριότητα, επιτυγχάνεται μέσω τις κλάσης Intent. Σε αυτή τη λειτουργία θα αναφερθούμε σε επόμενο κεφάλαιο (Ενεργοποιώντας νέες δραστηριότητες).

5.3.2 Χρήση του HTTP API

Πριν προχωρήσουμε στην ανάλυση της δεύτερης δραστηριότητας είναι απαραίτητο να αναφερθούμε στη χρήση του HTTP API. Παραπάνω, στο κεφάλαιο που έγινε αναφορά στο Zotero παρουσιάστηκε και ο τρόπος που συντάσσεται ένα URL.Για το χειρισμό HTTP αιτημάτων/αποκρίσεων σε περιβάλλον Android μπορεί να γίνει χρήση πακέτου **java.net**. Ωστόσο για την υλοποίηση των αιτημάτων στη συγκεκριμένη εφαρμογή χρησιμοποιήθηκε το πλαίσιο Apache HTTP Client (http://hc.apache.org). Η πραγματοποίηση των HTTP αιτημάτων γίνεται μέσω της κλάσης HttpRetriever. Η κλάση αυτή είναι υπεύθυνη για την εκτέλεση των αιτημάτων αλλά και για τον χειρισμό των αποκρίσεων, δηλαδή την εκχώρηση των αποτελεσμάτωνσε μία μεταβλητή τύπου String.

Για την εκτέλεση των HTTP αιτημάτων χρησιμοποιούμε ένα στιγμιότυπο της κλάσης DedaultHttpClient.

```
DefaultHttpClient client = new DefaultHttpClient();
```

Για την αναπαράσταση ενός GET αιτήματος το οποίο δέχεται ως όρισμα το URL στόχο, χρησιμοποιούμε την κλάση HttpGet.

```
HttpGet getRequest  = new HttpGet(url);
```

Ο HTTP client εκτελεί το αίτημα και επιστρέφει ένα αντικείμενο HttpResponse που ουσιαστικά περιέχει την απόκριση του server στο δοσμένο αίτημα.

```
HttpResponse getResponse = client.execute(getRequest);

 statusCode = getResponse.getStatusLine().getStatusCode();
```

Μπορούμε να ανακτήσουμε την κατάσταση του αιτήματος και να ελέγξουμε εάν είναι επιτυχημένη. Σε περίπτωση που το αίτημα είναι επιτυχημένο, χρησιμοποιούμε την αναφορά του αντικειμένου HttpEntity, το οποίο μας δίνει πρόσβαση στα δεδομένα που πραγματικά μας ενδιαφέρουν και τα μετατρέπουμε σε String.

```
HttpEntity getResponseEntity = getResponse.getEntity();

            if(getResponseEntity!=null){

            s =  EntityUtils.toString(getResponseEntity);

            Log.d(getClass().getCanonicalName(),"status code "+statusCode);
```

Στο παραπάνω απόσπασμα παρουσιάστηκε ο μηχανισμός που δημιουργήθηκε για την πραγματοποίηση όλων των αιτημάτων GET της εφαρμογής. Παρακάτω θα παρουσιαστεί ο τόπος ανάλυσης των xml αρχείων που λάβαμε ως απάντηση από τον server.

5.3.3 XML Parsing

Όπως αναφέρθηκε πιο πάνω απόκριση του Zotero server είναι ένα αρχείο xml σε μορφή Atom. Επίσης, το πρώτο αίτημα προς τον server είναι ένα get request το οποίο ζητά τις συλλογές που έχει δημιουργήσει ο χρήστης. Ένα χαρακτηριστικό παράδειγμα απόκρισης φαίνεται παρακάτω.

```
<?xml version="1.0" encoding="UTF-8"?>
<feed                                    xmlns="http://www.w3.org/2005/Atom"
xmlns:zapi="http://zotero.org/ns/api">
  <title>Zotero / Pappas Vagelis / Collections</title>
  <id>http://zotero.org/users/1737513/collections</id>
               <link        rel="self"        type="application/atom+xml"
href="https://api.zotero.org/users/1737513/collections"/>
               <link        rel="first"        type="application/atom+xml"
href="https://api.zotero.org/users/1737513/collections"/>
               <link        rel="last"        type="application/atom+xml"
href="https://api.zotero.org/users/1737513/collections"/>
                  <link         rel="alternate"         type="text/html"
href="https://www.zotero.org/users/1737513/collections"/>
  <zapi:totalResults>17</zapi:totalResults>
  <zapi:apiVersion>1</zapi:apiVersion>
  <updated>2014-09-30T14:25:24Z</updated>
  <entry>
    <title>Thesis_references</title>
    <author>
      <name>Pappas Vagelis</name>
      <uri>https://www.zotero.org/pappas_vagelis</uri>
    </author>
    <id>https://www.zotero.org/pappas_vagelis/collections/4U92HZWB</id>
    <published>2014-09-24T15:01:51Z</published>
    <updated>2014-09-30T14:25:24Z</updated>
                    <link     rel="self"     type="application/atom+xml"
href="https://api.zotero.org/users/1737513/collections/4U92HZWB"/>
                        <link      rel="alternate"      type="text/html"
href="https://www.zotero.org/pappas_vagelis/collections/4U92HZWB"/>
    <zapi:key>4U92HZWB</zapi:key>
    <zapi:version>295</zapi:version>
```

```
    <zapi:numCollections>0</zapi:numCollections>
    <zapi:numItems>3</zapi:numItems>
  </entry>
  <entry> ......
```

Στο συγκεκριμένο παράδειγμα εμφανίζονται όλες οι συλλογές του χρήστη. Η πληροφορία που μας ενδιαφέρει περιστοιχίζεται μέσα στις ταμπέλες (tags). Συγκεκριμένα, ελέγχουμε το όνομα της συλλογής, το κλειδί που της έχει ανατεθεί από τον εξυπηρετητή, των αριθμό των υπο - συλλογών που βρίσκονται μέσα σε κάθε συλλογή και τον αριθμό των στοιχείων μέσα στην συλλογή.

Για την ανάλυσή του xml εγγράφου και την εξαγωγή των πληροφοριών προτιμήθηκε η χρήση του **XMLPullParser**. Το Android SDK περιλαμβάνει υποστήριξη του **XMLPullParser** μέσω του πακέτου XML Pull package. Το πρώτο βήμα είναι να αποκτήσουμε ένα στιγμιότυπο του XmlPullParserFactory class. Αυτή η κλάση χρησιμοποιείται για για να δημιουργήσει υλοποιήσεις του XML Pull Parser όπως περιγράφονται στο XMPULL V1 API. Απενεργοποιούμε των χώρο ονομάτων (namespaces) καθώς δεν υπάρχει ανάγκη στη συγκεκριμένη εφαρμογή. Αυτή η επιλογή κάνει την ανάλυση του εγγράφου πιο γρήγορη. Παρακάτω παρατίθεται το απόσπασμα του κώδικα.

```
XmlPullParser parser = Xml.newPullParser();
parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
```

Στη συνέχεια δημιουργήθηκε ένα αντικείμενο XMLPullParser καλώντας την μέθοδο newPullParser. Η είσοδος του εγγράφου σε μορφή String επιτυχγάνεται μέσω της μεθόδου setInput(). Ο σχετικός κώδικας είναι παρακάτω.

```
parser.setInput(new StringReader(xmlToParse));

parser.nextTag();
```

Ο XML Pull Parser βασίζεται στα γεγονότα, για να μπορέσουμε να ανακτήσουμε τις πληροφορίες που επιθυμούμε πρέπει να δημιουργήσουμε βρόγχους επανάληψης μέχρι να φτάσουμε στο τέλος του εγγράφου.

Τα γεγονότα που μας ενδιαφέρουν είναι τα εξής:

- START_TAG, tag αρχής διαβάστηκε.
- END_TAG, tag τέλους διαβάστηκε.
- TEXT, το περιεχόμενο κείμενο διαβάστηκε
- START_DOCUMENT, ο parser βρίσκεται στην αρχή του εγγράφου.
- END_TAG, ο parser έχει φτάσει στο τέλος του εγγράφου.

Η ανάλυση του εγγράφου είναι στη λογική της αναδρομής. Κάθε tag entry που εντοπίζεται είναι ένα στοιχείο γονιός. Μέσα σε κάθε στοιχείο γονέα υπάρχουν οι πληροφορίες που ανακτάμε. Για τον έλεγχο των ονομάτων των στοιχείων γίνεται χρήση της μεθόδου getName() και για την ανάκτηση του κειμένου που περιέχεται στα tags η μέθοδος nextText(). Ακολουθεί παράδειγμα με απόσπασμα κώδικα

```
String name = parser.getName();

if(name.equals("title")){

title = readTag(parser,TAG_TITLE);

}

else if(name.equals("zapi:key")){

	key = readTag(parser, TAG_KEY);

}

else if(name.equals("zapi:itemType")){

	type = readTag(parser, TAG_TYPE);

}
```

Στο σημείο αυτό θα γίνει αναφορά σε άλλες μεθόδους ανάλυσης ενός εγγράφου xml που εξετάστηκαν αλλά δεν χρησιμοποιήθηκαν στη παρούσα πτυχιακή εργασία.

Αρχικά, ο DOM(Document Object Model) parser χρησιμοποιείται πιο συχνά για την ανάλυση ενός εγγράφου xml. Αναπαριστά ένα xml έγγραφο σε μορφή δένδρου το οποίο το δημιουργεί στη μνήμη. Είναι αρκετά γρήγορος εάν το έγγραφο είναι σχετικά μικρό. Το μειονέκτημα είναι όταν το έγγραφο είναι μεγάλο και πρέπει να αναπαρασταθεί ολόκληρο στη μνήμη, τότε ο DOM parser μπορεί να αποβεί υπερβολικά χρονοβόρος. Ένας ακόμα κίνδυνος είναι να μην μπορέσει να το φορτώσει ολόκληρο στη μνήμη εξαιτίας του όγκου του.

Άλλη μία δυνατότητα είναι η χρήση του SAX (Simple API for XML) Parser. Βασίζεται στα γεγονότα, αναλύει το έγγραφο βήμα προς βήμα και ειδοποιεί όταν συναντήσει την ετικέτα αρχής (start tag, <), την ετικέτα τέλους (end tag, />), κείμενο, ενός στοιχείου XML και καλούνται μέθοδοι που έχει υλοποιήσει ο χρήστης, χωρίς να απαιτείται να το δημιουργήσει ολόκληρο στη μνήμη. Σε αντίθεση με τον DOM Parser είναι κατάλληλος για μεγάλα αρχεία. Ο SAX parser χρησιμοποιεί ένα διαχειριστή “κλήσεων” όταν φτάνει σε ορισμένα τμήματα του εγγράφου. Αυτή η υλοποίηση κώδικα μπορεί όμως να γίνει περίπλοκη και αναποτελεσματική. Για τους παραπάνω λόγους επιλέχθηκε ο XMLPullParser.

5.3.4 API Request

Σε προηγούμενο κεφάλαιο της παρούσας πτυχιακής εργασίας αναφερθήκαμε στην σύνταξη του αιτήματος προς τον Zotero server. Έγινε ανάλυση των αναγκαίων παραμέτρων, ώστε να συνταχθεί το URL που θα φέρει τα επιθυμητά αποτελέσματα. Στο παρακάτω κεφάλαιο θα αναλυθεί ο τρόπος υλοποίησης ενός API Request μέσα στον κώδικα java.

Μία πολύ σημαντική πτυχή στην ανάπτυξη εφαρμογών για κινητές εφαρμογές είναι η συμπεριφορά της εφαρμογής. Δηλαδή η εφαρμογή πρέπει να συμπεριφέρεται “αθόρυβα”. Η απόκριση της εφαρμογής στην εισαγωγή του χρήστη πρέπει να είναι γρήγορη και αποτελεσματική. Αυτός είναι ο λόγος για

τον οποίο η εφαρμογή εκτελεί τα αιτήματα στο “παρασκήνιο”,δηλαδή η εκτέλεση των αιτημάτων γίνεται σε διαφορετικό νήμα από αυτό του User Interface.

Παρότι οι ταχύτητες των διαδικτυακών συνδέσεων σε κινητές συσκευές έχουν σημειώσει τεράστια πρόοδο, παραμένει γεγονός ότι η μεταφόρτωση δεδομένων από το διαδίκτυο μπορεί να αποτελέσει μια χρονοβόρα διαδικασία. Άρα, είναι απαραίτητο να μην απασχολούμε το κύριο νήμα της εφαρμογης (UI thread) ενώ ο HTTP client περιμένει μέχρι τα δεδομένα να μεταφερθούν. Σε αντίθετη περίπτωση μπορεί να εμφανιστεί ένα παράθυρό διαλόγου (Android not Responding) που δίνει στον χρήστη τη δυνατότητα να σταματήσει την εφαρμογή. Γενικά,το σύστημα εμφανίζει το παράθυρο διαλόγου όταν δεν μπορεί να ανταποκριθεί σε δεδομένα εισόδου από τον χρήστη. Πρακτικά, στην εφαρμογή για την παρούσα πτυχιακή εργασία θα εμφανιστεί το παράθυρο διαλόγου Android not Responding εάν απασχολήσουμε το κύριο νήμα για πάνω από πέντε δευτερόλεπτα.

Για του παραπάνω λόγους θα επεκτείνουμε την κλάση που είναι υπεύθυνη για την υλοποιήσει του αιτήματος με την ενσωματωμένη στο Android κλάση AsyncTask. Αυτή η κλάση μας επιτρέπει να επιτελούμε λειτουργίες στο “παρασκήνιο” και να εμφανίζουμε τα αποτελέσματα τους στο κύριο νήμα (UI thread), χωρίς να χρειάζεται ουσιαστικά να χειριστούμε νήματα. Υπάρχουν τρεις μέθοδοι που μπορούμε να υλοποιήσουμε . Αρχικά, η onPrexecute, η οποία εμφανίζεται στο κύριο νήμα και χρησιμοποιείται κυρίως για αρχικοποίηση. Κατόπιν, η μέθοδος doInBackground, όπου οι λειτουργίες της επιτελούνται στο “παρασκήνιο” επιστρέφοντας ένα αποτέλεσμα (Αντικείμενο Java). Τέλος, η μέθοδος onPostexecute η οποία εμφανίζει τα αποτελέσματα στο κύριο νήμα. Παρακάτω, παρουσιάζεται απόσπασμα του κώδικα με την υλοποίηση λειτουργίας παρασκηνίου.

```
private class ItemTask extends AsyncTask<String, Void , ArrayList<Item>>{

protected ArrayList<Item> doInBackground(String... params) {

                    Log.d(getClass().getName(), "doInbackground()");
```

```
        // TODO Auto-generated method stub
        String query = params.toString();
        ArrayList<Item> it_seeker = (ArrayList<Item>)
                itemseeker.find(query);
        return it_seeker;
}

protected void  onPostExecute(final ArrayList<Item> result) {
        Log.d(getClass().getCanonicalName(),"on post execute");
        runOnUiThread(new Runnable() {
                @Override
                public void run() { ....
```

Από το παραπάνω απόσπασμα παρατηρούμε ότι η παράμετρος που δέχεται ως είσοδο για εκτελεστεί η εργασία είναι τύπου String, αυτό το URL προς τον server. Δεν καθορίζεται κάποια μονάδα που μας δείχνει την πρόοδο της λειτουργίας παρασκηνίου (Void). Τα αποτελέσματα της διεργασίας παρασκηνίου τοποθετούνται σε μια λίστα.

Στην ενότητα API Request παρουσιάστηκε ο αναγκαίος τρόπος, ώστε να εκτελείται ένα αίτημα προς τον server του Zotero χωρίς να επιβαρύνεται η λειτουργία του βασικού νήματος. Παρακάτω θα παρουσιαστεί ένα βασικό δομικό στοιχείο των Android εφαρμογών, οι Δραστηριότητες (Activities).

5.3.5 Activity m_screen

Σε αυτή την ενότητα θα παρουσιαστεί η δεύτερη Activity αλλά και τρόπος που μπορούμε να μεταφέρουμε δεδομένα από την πρώτη στη δεύτερη.

Σε αυτό το σημείο πρέπει να γίνει αναφορά στην κλάση **Intent**. Το Intent είναι ένα αντικείμενο μηνυμάτων, το οποίο χρησιμοποιείται για την αίτηση μιας ενέργειας από ένα συστατικό σε ένα άλλο συστατικό της εφαρμογής. Το Intent έχει πεδία τα οποία καθορίζουν συγκεκριμένες πληροφορίες.Τα Intents διευκολύνουν την επικοινωνία μεταξύ των συστατικών της εφαρμογής με διάφορους τρόπους.

- **Εκκίνηση μιας Activity**: Μπορεί να γίνει η εκκίνηση μιας Activity περνώντας ένα Intent ως παράμετρο στη μέθοδο startActivity(). Το Intent περιγράφει την Activity που ακολουθεί και μεταφέρει δεδομένα. Ανάλογη περίπτωση χρήσης υπάρχει και παρακάτω,στην εκκίνηση της Activity m_screen. Εάν θέλουμε να λάβουμε αποτελέσματα από μια Activity όταν τελειώσει, τότε καλούμε την μέθοδο **startActivityForResults()**. Η Activity λαμβάνει τα αποτελέσματα ως ένα ξεχωριστό Intent αντικείμενο.

- **Εκκίνηση ενός Service**: Το Service είναι ένα συστατικό που πραγματοποιεί λειτουργίες στο παρασκήνιο χωρίς την ύπαρξη της διασύνδεσης χρήστη. Για παράδειγμα ένα service μπορεί να εκκινήσει για να επιτελέσει τη μεταφόρτωση ενός αρχείου από το διαδίκτυο, περνώντας ως παράμετρο Intent στη μέθοδο **startService()**. Το Intent περιγράφει πιο service πρέπει να εκκινήσει και μεταφέρει δεδομένα που είναι απαραίτητα.

- **Παράδοση μηνυμάτων broadcact:** Ένα broadcast είναι ένα μήνυμα που μπορεί να λάβει η εφαρμογή. Το σύστημα παραδίδει διάφορα μηνύματα για γεγονότα που έχουν συμβεί, όπως όταν το σύστημα εκκινεί ή όταν η συσκευή είναι στη φόρτιση. Μπορούμε να παραδώσουμε μηνύματα broadcast σε άλλες εφαρμογές περνώντας ως παράμετρο ένα Intent στις μεθόδους **sentBroadcast(), sendOrderdBroadcast()** ή στη μέθοδο **sendStickyBroadcast().**

Υπάρχουν δύο τύποι Intent τα άμεσα και τα έμμεσα:

- **Άμεσα Intents (Explicit):** Καθορίζουν το συστατικό που θα εκκινήσει άμεσα με τη χρήση του ονόματός του. Συνήθως χρησιμοποιούμε άμεσα intent γιατί

γνωρίζουμε το όνομα του συστατικού που θέλουμε να εκκινήσει. Για παράδειγμα η εκκίνηση μιας νέας Activity στην επίδραση του χρήστη.

- **Έμμεσα Intents (Implicit):** Αυτά τα intent δεν δηλώνουν κάποιο συγκεκριμένο όνομα συστατικού. Αντιθέτως, δηλώνουν μια γενική ενέργεια που πρέπει να γίνει, η οποία επιτρέπει σε άλλο συστατικό άλλης εφαρμογής να το χειριστεί.

Όταν δημιουργείται ένα έμμεσο Intent με στόχο να εκκινήσει μια Activity ή ένα Service το Android σύστημα βρίσκει το κατάλληλο συστατικό που πρέπει να εκκινήσει συγκρίνοντας τα περιεχόμενα των Intents. Εάν περισσότερα από ένα είναι κατάλληλα τότε το σύστημα εμφανίζει παράθυρο διαλόγου και δίνει τη δυνατότητα στο χρήστη να επιλέξει.

Μέσα στα Intents υπάρχουν πληροφορίες οι οποίες καθορίζονται από τα εξής πεδία:

- **Action**: Είναι μια συμβολοσειρά (String) η οποία αναπαριστά μια λειτουργία που που θα εκτελεστεί.
- **Data**: Αναπαριστά δεδομένα που σχετίζονται με το Intent.
- **Category**: Προσθέτει πληροφορίες σχετικά με το είδος του συστατικού που θα χειριστεί το Intent.
- **Type**: Καθορίζει τον Multipurpose Internet Mail Extension τύπο των δεδομένων των Intent.
- **Component**: Καθορίζει ακριβώς ποιο συστατικό θα λάβει το συγκεκριμένο Intent.
- **Extras**: Αποθηκεύουν επιπλέον δεδομένα που σχετίζονται με το Intent. Τα δεδομένα είναι ζεύγη κλειδιού – τιμή (key-value pairs). Η Activity που πρόκειται να χρησιμοποιήσει αυτά τα δεδομένα πρέπει να γνωρίζει τον τύπο και το όνομα των δεδομένων.

- **Flags**: Αναπαριστούν πληροφορίες που καθορίζουν το τρόπο που τα Intents πρέπει να χρησιμοποιηθούν.

Σε περίπτωση που ένα Intent δεν ονομάζει ρητά ένα συστατικό, δηλαδή μια Activity που πρόκειται να ξεκινήσει τη λειτουργία της, τότε το Android ξεκινά την διαδικασία ανάλυσης του Intent. Η ανάλυση ενός Intent βασίζεται σε ειδών πληροφορίες. Το πρώτο είναι το Intent να περιγράφει την λειτουργία που πρέπει να γίνει. Χαρακτηριστικό παράδειγμα που συντάμε στην εφαρμογή ZoteroApp είναι η παρακάτω εντολή.

```
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(url)));
```

Στο συγκεκριμένο κομμάτι κώδικα δεν ονομάζεται κάποια εντολή. Μόλις ο χρήστης της εφαρμογής πατήσει πάνω σε ένα συγκεκριμένο view που αναπαριστά το url ενός συγκεκριμένου στοιχείου της βιβλιογραφίας τότε το Android λαμβάνει αυτό το Intent που καθορίζει την ενέργεια που πρέπει να γίνει, δηλαδή πληροφορία που πρέπει να προβάλει στον χρήση ενεργοποιώντας τον browser.

Η άλλη επιλογή είναι η κάθε Activity να καθορίζει φίλτρα Intent (Intent filters) τα οποία περιγράφουν ποια Intents ή λειτουργίες μπορεί να διαχειριστεί.Τα Intents filters δηλώνονται στο αρχείο AndroidManifest.xml. Χαρακτηριστικό παράδειγμα στην εφαρμογή ZoteroApp είναι η δήλωση της MainZoteroActivity στο αρχείο AndroidManifest.xml.

```
<activity
            android:name="com.example.zoteroapp.MainZoteroActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
</activity>
```

Στο παραπάνω απόσπασμα xml, με το πεδίο action δηλώνεται ότι η συγκεκριμένη Activity είναι το κύριο πεδίο εισόδου της εφαρμογή. Το πεδίο category προσθέτει πληροφορία και δηλώνει ότι είναι η αρχική Activity.

Μέχρι στιγμής αναφερθήκαμε μόνο στην πρώτη Activity της εφαρμογής. Σε αυτό το σημείο θα προχωρήσουμε περισσότερο. Θα παρουσιάσουμε την δεύτερη και τελευταία Activity της εφαρμογής. Να σημειώσουμε ότι δεν υπάρχει κάποιος περιορισμός στη χρήση των Activities. Η συγκεκριμένη εφαρμογή προορίζεται για συσκευές ταμπλέτες (tablets). Δηλαδή, προορίζεται για μεγάλη οθόνη που μπορεί να εμφανίσει πολλά συστατικά (views) μαζί και δεν κρίθηκε αναγκαία η δημιουργία περισσότερων Activities. Η μεταφορά από την μία Activity στην άλλη γίνεται μέσω της κλάσης Intent (πρόθεση). Ακολουθεί η χαρακτηριστική εντολή από απόσπασμα κώδικα η οποία είναι ένα παράδειγμα άμεσου (explicity) Intent .

```
Intent intent = new Intent(MainZoteroActivity.this , m_screen.class);
```

Μέσω της κλάσης Intent και της κλήσεις της μεθόδου putExtra() γίνεται και η μεταφορά των δεδομένων. Ακολουθεί απόσπασμα κώδικα.

```
intent.putExtra("items", result);
```

Η μέθοδος putExtra παρέχεται από την κλάση Intent. Μεταφέρει μια λίστα δεδομένων τύπου Item.

Για να ολοκληρωθεί η μεταφορά σε άλλη Activity και των δεδομένων που την συνοδεύουν χρειάζεται η εντολή startActivity().

```
startActivity(intent);
```

Κάθε Activity πρέπει να δηλώνεται στο αρχείο AndroidManifest.xml. Παρακάτω το απόσπασμα του κώδικα.

```
 <activity android:name="com.example.zoteroapp.m_screen"

android:label="@string/app_name" >.
```

5.3.6 Listview

Στην παρακάτω ενότητα θα δείξουμε τον τρόπο για να εμφανίσουμε τα ονόματα όλων των συλλογών που έχει δημιουργήσει ο χρήστης. Τα ονόματα θα τοποθετηθούν σε μία λίστα. Το απαραίτητο συστατικό για την δημιουργία αυτής της όψης είναι η τοποθέτηση ενός widget Listview. Αρχικά, πρέπει η νέα κλάση να υποδεχτεί τα δεδομένα. Για να επιτευχθεί αυτό πρώτα παίρνουμε μια αναφορά του Intent που εκίνησε τη δεύτερη Activity χρησιμοποιώντας την μέθοδο getIntent(). Από αυτή ανακτούμε τα δεδομένα με την κλήση της μεθόδου getSerializableExtra(). Παρακάτω το απόσπασμα κώδικα.

```
item_list = (ArrayList<Item>) getIntent().getSerializableExtra("items");
```

Η μέθοδος getSerializableExtra() έχει ως παράμετρο ένα String το οποίο είναι το όνομα των δεδομένων που έχουν σταλεί από την προηγούμενη Activity.

Η παρουσίαση των δεδομένων τύπου Items, δηλαδή οι συλλογές με τα χαρακτηριστικά τους, αποθηκευμένα σε μια λίστα στην οθόνη απαιτεί την δημιουργία της λίστας ως όψη. Αρχικά δηλώνουμε την λίστα στο αρχείο m_screen.xml. Της αναθέτουμε μια μεταβλητή ως το μοναδικό της αναγνωριστικό μέσα στην εφαρμογή (id) και καθορίζουμε και το μέγεθός της.

```
<ListView
                android:id="@+id/list1"
                android:layout_width="250dp"
                android:layout_height="match_parent"
                   >
            </ListView>
```

Κατόπιν πρέπει να δώσουμε μορφή στις γραμμές της λίστας. Δημιουργούμε ένα νέο αρχείο XML. Σε αυτό θα προσθέσουμε τις όψεις ώστε να δώσουμε την επιθυμητή μορφή σε κάθε γραμμή της λίστας. Τοποθετούμε δύο περιοχές κειμένου (Textview) και μία περιοχή που θα υποδεχτεί εικόνα. Στις δύο υποδοχές κειμένου θα μπουν το όνομα και η ημερομηνία δημιουργίας κάθε συλλογής αντίστοιχα. Στην υποδοχή της εικόνας θα μπει το χαρακτηριστικό εικονίδιο με τον φάκελο ώστε να γίνεται εύκολα αντιληπτό ότι πρόκειται για συλλογές.

```
 <TextView
            android:id="@+id/tv_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium" />


        <TextView
            android:id="@+id/tv_date"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="@string/No_Available"
            android:textAppearance="?android:attr/textAppearanceSmall" />
<ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="match_parent" />
```

Στο παραπάνω απόσπασμα xml φαίνονται τα id των όψεων και τα μεγέθη τους. Για τις περιοχές κειμένου δηλώνονται και το μέγεθος των γραμματοσειρών που θα έχει καθεμιά. Οι τιμές "match_parent" στα μεγέθη των όψεων δηλώνουν ότι ο χώρος που θα καταλαμβάνουν θα είναι όσος του επιτρέπει το γονικό στοιχείο. Η τιμή "wrap_content" δηλώνει ότι η όψη θα καταλάβει χώρο αναλόγως του στοιχείου που θα υποδεχτεί.

Για να είμαστε σε θέση να χειριστούμε το Listview χρειαζόμαστε ένα προσαρμογέα (adapter). Το ρόλο αυτό θα παίξει η κλάση ItemAdapter. Η συγκεκριμένη κλάση επεκτείνει την κλάση ArrayAdpater. Στον ItemAdapter θα αποδίδονται τα δεδομένα της λίστας που έχουν ανακτηθεί. Κατόπιν γίνεται η χρήση της μεθόδου setListAdapter ώστε να συνεργαστεί με το Listview.

Αρχικά στη κλάση ItemAdapter δηλώνουμε τον δομητή της. Παρακάτω το απόσπασμα του κώδικα.

```
public ItemAdapter(Activity context, int resource, List<Item> items) {

            super(context, resource, items);

            // TODO Auto-generated constructor stub

            this.context = context;

            this.itemList = items;


      }
```

Η μεταβλητή resource είναι το αναγνωριστικό πόρου για ένα αρχείο διάταξης που περιέχει ένα TextView.

Η μεταβλητή context δηλώνει την τρέχουσα δραστηριότητα.

Η μεταβλητη items είναι μια λίστα αντικειμένων με τα δεδομένα που θα αναπαρασταθούν στο Listview.

Δεύτερη μέθοδος είναι η getView η οποία επιστρέφει την μία όψη, δηλαδή είναι η υπεύθυνη για την αναπαράσταση των δεδομένων σε κάθε γραμμή της ListView. Παρακάτω ο κώδικας της μεθόδου.

```
public View getView(int position, View convert, ViewGroup group){

            View view = convert;


            if(view==null) {

                  LayoutInflater li = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

                  view = li.inflate(R.layout.lista_item, null);

                                          }

            Item item = itemList.get(position);

            if(item!=null){

            TextView tv_name = (TextView)view.findViewById(R.id.tv_name);

                  tv_name.setText(item.getTitle());


            TextView tv_date = (TextView)view.findViewById(R.id.tv_date);

                  String s;

                  s = item.getPublished().toString();

                  String new_s;

                   new_s = s.substring(0, 10);



ImageView imageView = (ImageView)view.findViewById(R.id.imageView1);
```

```
            imageView.setImageResource(item.imageRetriever(type));


ImageView imageView2 =    (ImageView)view.findViewById(R.id.imageView2);
                    try{
                          if(MainZoteroActivity.flag<2){
                                imageView2.setVisibility(View.GONE);

      tv_date.setText(String.valueOf("Created:"+new_s));
else {
      imageView2.setImageResource(item.connecterRetrieve(childNum));
                                String itemAuthor =
item.getAuthor().toString();
                                if(String.valueOf(itemAuthor) != null){


                                tv_date.setText(String.valueOf("Creator:
"+itemAuthor));
                                }


                          }
                    }catch(NullPointerException e){
                          e.printStackTrace();
                    }
            }
            return view;
      }
```

Στη κλάση m_screen ετοιμάζουμε τον ItemAdapter με τα εξής τμήματα κώδικα.

```
item_adapter = new ItemAdapter(this,
android.R.layout.simple_expandable_list_item_1, item_list);
```

και με `list1.setAdapter(item_adapter);`

Για να είμαστε σε θέση να χειριστούμε γεγονότα που προκαλούνται από το πάτημα σε μία θέση της λίστας πρέπει να προσπελάσουμε την μέθοδο setOnItemClickListener(). Μέσα από αυτή την μέθοδο μπορούμε να ανακτήσουμε το αντικείμενο που είναι τοποθετημένο στη γραμμή που σημειώθηκε το γεγονός. Σε περίπτωση που μια συλλογή περιλαμβάνει άλλες συλλογές εκτός από μεμονωμένα στοιχεία τότε θα εμφανίζεται στη γραμμή και το κουμπί το οποίο μας δίνει τη δυνατότητα να εμφανίσουμε και τις εμφωλευμένες συλλογές. Αυτό επιτυγχάνεται κάνοντας αρχικά εισαγωγή στη λίστα με τα δεδομένα τις μεταβλητές τύπου Items και κατόπιν την αναπαράσταση τους στο wigdet Listview. Αυτό το καταφέρνουμε με την κλήση της μεθόδου `notifyDataSetChanged()`. Η συγκεκριμένη μέθοδος ενημερώνει ότι το σετ των δεδομένων έχει αλλάξει και κάθε όψη (View) πρέπει η ίδια να ανανεώνεται.

Η μέθοδος που επιτελεί αυτή τη λειτουργία είναι η εξής:

```
public void updateList(ArrayList<Item> data){

            itemList = data;

            notifyDataSetChanged();

      }
```

Σε αυτή την ενότητα παρουσιάστηκε ο τρόπος χειρισμού του συστατικού Listview. Παρακάτω θα παρουσιαστεί η μορφοποίηση κειμένου σε JSON.

5.3.7 JSON

Προχωρώντας στην ανάπτυξη της εφαρμογής διαπιστώθηκε η ανάγκη για την εξαγωγή περισσότερων πληροφοριών από τα xml αρχεία που λαμβάνονται ως

απάντηση από τον server. Ανάλυση ενός εγγράφου XML όταν περιέχει πολλά εμφωλευμένα στοιχεία ακόμα και με τον XMLPullparser μπορεί να γίνει περίπλοκη διαδικασία. Για αυτό έγινε επιλογή της μορφοποίησης κειμένου JSON. Σημειώνεται ότι η κατασκευή της εφαρμογής ξεκίνησε όταν default έκδοση του Zotero ήταν η δεύτερη (v=2), την στιγμή συγγραφής του κειμένου η default έκδοση είναι η τρίτη (v=3), όπου η μορφή του κειμένου της απόκρισης του server είναι η JSON.

Το JSON προέρχεται από Javascript Object Notation. Είναι ένας ελαφρύς μηχανισμός μορφοποίησης δεδομένων που μπορούν εύκολα να διαβαστούν από ανθρώπους και μηχανές. Το JSON είναι μια μορφοποίηση κειμένου η οποία είναι εντελώς ανεξάρτητη από γλώσσες προγραμματισμού. Η αναπαράσταση JSON δεδομένων έχει δύο μορφές, δηλαδή, ενός αντικειμένου ή ενός πίνακα. Τα αντικείμενα JSON είναι μία μη-ταξινομημένη συλλογή από ζεύγη τιμής κλειδιού. Από τη άλλη πλευρά ο πίνακας είναι μία ταξινομημένη συλλογή από τιμές. Η κύρια διαφορά των δύο παραπάνω είναι στη αναπαράσταση, τα αντικείμενα ξεκινάνε με αριστερή αγκύλη ({)και ολοκληρώνονται με δεξιά αγκύλη (}) ενώ οι πίνακες με τετράγωνες αγκύλες ([) και (]) αντίστοιχα. Η τιμή μπορεί να είναι μια συμβολοσειρά(String) μέσα σε διπλά εισαγωγικά ή ένας αριθμός ή αληθής(true) ή ψευδής(false) ή μηδενική (null) ή ένα αντικείμενο ή ένας πίνακας.

Όπως αναφέρεται σε προηγούμενο κεφάλαιο το Zotero υπάρχει η δυνατότητα στα αιτήματα του χρήστη να συμπεριληφθεί και η παράμετρος json. Ο server απαντά επιστρέφοντας ένα αρχείο σε μορφή JSON. Παρακάτω παρατίθεται ένα παράδειγμα αιτήματος και της απόκρισης του server.

https://api.zotero.org/users/1737513/items/R4HJIZRH/?format=json

Το αίτημα αυτό θα μας επιστρέψει το στοιχείο με κλειδί **R4HJIZRH** του χρήστη με ταυτότητα 1737513 σε μορφή json. Η σχετική απάντηση παρατίθεται παρακάτω.


```
{
    "key": "R4HJIZRH",
    "version": 247,
    "library": {
        "type": "user",
```

```
        "id": 1737513,
        "name": "Pappas Vagelis",
        "links": {
            "alternate": {
                "href": "https://www.zotero.org/pappas_vagelis",
                "type": "text/html"
            }
        }
    },
    "links": {
        "self": {
                                                                "href":
"https://api.zotero.org/users/1737513/items/R4HJIZRH",
            "type": "application/json"
        },
        "alternate": {
                                                                "href":
"https://www.zotero.org/pappas_vagelis/items/R4HJIZRH",
            "type": "text/html"
        }
    },
    "meta": {
        "creatorSummary": "Antoniou and Pappas",
        "parsedDate": "2014-01-07",
        "numChildren": 1
    },
    "data": {
        "key": "R4HJIZRH",
        "version": 247,
        "itemType": "journalArticle",
        "title": "Test article",
        "creators": [
            {
                "creatorType": "author",
                "firstName": "E",
                "lastName": "Antoniou"
            },
```

```
            {
                "creatorType": "author",
                "firstName": "E",
                "lastName": "Pappas"
            }
        ],
          "abstractNote": "Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.",
        "publicationTitle": "IEEE trans. java",
        "volume": "13",
        "issue": "1",
        "pages": "135-358",
        "date": "1/7/2014",
        "series": "",
        "seriesTitle": "",
        "seriesText": "",
        "journalAbbreviation": "IEEE tr.j",
        "language": "",
        "DOI": "",
        "ISSN": "",
        "shortTitle": "",
        "url": "",
        "accessDate": "",
        "archive": "",
        "archiveLocation": "",
        "libraryCatalog": "",
        "callNumber": "",
        "rights": "",
        "extra": "",
        "dateAdded": "2014-07-03T07:44:25Z",
        "dateModified": "2014-07-03T07:52:16Z",
        "tags": [

        ],
        "collections": [
            "N2IPNFGD"
        ],
```

```
        "relations": {

        }
    }
}
```

Στο συγκεκριμένο έγγραφο φαίνεται ότι υπάρχουν αντικείμενα και πίνακες JSON. Τα αντικείμενα μπορεί να είναι εμφωλευμένα σε άλλα αντικείμενα όπως αντίστοιχα και πίνακες.

Στον κώδικα της java δημιουργήθηκε το αρχείο JsonParser.java, ώστε η εφαρμογή να είναι σε θέση να αναλύσει το json δεδομένα. Πρώτα πρέπει να δημιουργηθεί το αντικείμενο που θα δέχεται ως όρισμα την απάντηση του server, ως μεταβλητή τύπου String.

```
JSONObject jobj = new JSONObject(jsonToParse);
```

Μετά από αυτή την εντολή κοιτάμε μέσα στο αρχείο και μπορεί να αρχίσει η εξαγωγή των δεδομένων. Για παράδειγμα εάν θέλουμε να εντοπίσουμε το στοιχείο με το όνομα data τότε πρέπει να γραφτεί η εξής δήλωση.

```
JSONObject subobj = jobj.getJSONObject("data");
```

Όπως παρατηρείται στο data υπάρχουν άλλα ζεύγη αντικειμένων τα οποία “κρέμονται” από κάτω. Για να έχουμε πρόσβαση, για παράδειγμα, στη τιμή του αντικειμένου με το όνομα title πρέπει να συγγραφεί το εξής κομμάτι κώδικα. Διευκρινίζεται ότι γίνεται πρώτα έλεγχος εάν υπάρχει το συγκεκριμένο όνομα.

```
if(data.has("title")){

                        title = data.getString("title");

                }
```

Παρόμοιες είναι και οι εντολές για να εξαχθούν τιμές ενός πίνακα json. Παρακάτω ακολουθεί απόσπασμα κώδικα που αφορά τον πίνακα creators.

```
if(data.has("creators")){
                        JSONArray jArray = data.getJSONArray("creators");
                        for(int i=0;i<jArray.length();i++){
                              JSONObject obj = jArray.getJSONObject(i);


                              f_name = obj.getString("firstName");
                              l_name = obj.getString("lastName");
                              creator_type = obj.getString("creatorType");
```

Στην παραπάνω ενότητα έγινε παρουσίαση του μηχανισμού που χρησιμοποιήθηκε για τις ανάγκες της εφαρμογής. Η μορφοποίηση JSON είναι πολύ εύκολη στη χρήση, αποτελεί εξαιρετική εναλλακτική στη XML. Εύκολη είναι και η διαδικασία παραγωγής JSON δεδομένων από αντικείμενα της Java.

5.3.8 Fragment

Στη δεύτερη Activity της εφαρμογής συναντάμε ένα ενδιαφέρον συστατικό του Android. Η αναφορά γίνεται για τα **Fragment.** Στην παρακάτω ενότητα θα αναλυθεί ο τρόπος λειτουργίας τους στην εφαρμογή και γενικότερα η χρησιμότητα τους στις εφαρμογές Android.

Τα Fragment αναπαριστούν τη συμπεριφορά ενός κομματιού της διασύνδεσης χρήσης. Υπάρχει η δυνατότητα με τα fragment να χωριστεί η διασύνδεση σε διαφορετικά κομμάτια με διαφορετικές λειτουργίες, μέσα στην ίδια Activity. Ένα fragment μπορεί να επαναχρησιμοποιηθεί σε διαφορετικές Activities.

Μπορούμε να πούμε ότι τα Fragment είναι μια ξεχωριστή μονάδα που φιλοξενείται μέσα σε μια Activity.

Το fragment πρέπει να είναι ενσωματωμένο στην Activity. Ο κύκλος ζωής του επηρεάζεται άμεσα από την Activity που το φιλοξενεί. Για παράδειγμα, όταν μια Activity μπει σε κατάσταση Paused τότε στην ίδια κατάσταση περιέρχονται και όλα τα Fragments που φιλοξενεί. Ωστόσο, εάν η Activity επανέρθει σε λειτουργία μπορούμε να χειριστούμε το κάθε fragment ανεξάρτητα. Όταν εισάγεται ένα fragment μέσα σε μια Activity, τότε αυτό ζει μέσα στην ιεραρχία όψεων της Activity. Για το κάθε fragment καθορίζεται και το δικό του layout. Μπορούμε να ορίσουμε το fragment μέσα στο layout της Activity, δηλώνοντας το tag <fragment>.

Τα fragment εισήχθησαν στην έκδοση 3.0 (API level 11) του Android. Ο σκοπός της δημιουργίας ήταν να υποστηρίξουν δυναμικά και ευέλικτα φύλλα μορφοποίησης (layout). Ειδικά όταν πρόκειται για συσκευές με μεγάλη οθόνη όπως τα tablet, η ανάγκη για πιο δυναμικά layouts γίνεται ακόμα πιο φανερή.

Για τη δημιουργία ενός Fragment, πρέπει να δημιουργηθεί μια κλάση που να επεκτείνει την κλάση **Fragment**. Αυτή η κλάση θα μοιάζει σαν μια κλάση Activity. Μπορεί να περιέχει μεθόδους που αναπαριστούν τον κύκλο ζωής. Σίγουρα πρέπει να υλοποιηθεί η μέθοδος onCreateView() ώστε το να εμφανιστεί το layout που έχει δημιουργηθεί στη διασύνδεση χρήστη. Στην συγκεκριμένη εφαρμογή η κλάση που υλοποιήθηκε είναι η FragmentTab. Μέσω της onCreateView() το fragment τοποθετήται στο συστατικό γονέα, δηλαδή στο layout της Activity.

Η τοποθέτηση του fragment στη διασύνδεση χρήστη έγινε με τη χρήση ενός Framelayout. Γενικά, το παραπάνω συστατικό είναι σχεδιασμένο να δεσμεύει ένα χώρο στην οθόνη ώστε να υποδεχθεί ένα άλλο συστατικό μέσα του, συνήθως ένα. Στο Framelayout ορίζεται id ως μοναδικό αναγνωριστικό. Στο παρακάτω απόσπασμα κώδικα φαίνεται ο τρόπος που προσθέτουμε το fragment στην Activity της εφαρμογής.

```
fragmentTab = new FragmentTab();

ft = getFragmentManager().beginTransaction();
```

```
ft.replace(R.id.framelayout1, fragmentTab);

ft.commit();
```

Η διαχείριση των fragments γίνεται από το αντικείμενο FragmentManager. Για να ανακτήσουμε όλα τα fragments καλούμε τη μέθοδο getFragmentManager(). Τη συγκεκριμένη μέθοδο την χρησιμοποιούμε και για να ανοίξουμε μια FragmentTransaction, η οποία μας επιτρέπει να πραγματοποιούμε συναλλαγές με τα fragment, όπως προσθήκες, διαγραφές, αντικαταστάσεις fragment. Η εφαρμογή όλων αυτών των διαδικασιών επιτελείται με την κλήση της μεθόδου commit().

5.3.9 Διαχείριση γεγονότων πληκτρολογίου

Η εφαρμογή υποστηρίζει και την δυνατότητα αναζήτησης στοιχείων βιβλιογραφίας της βιβλιοθήκης του χρήστη. Η δυνατότητα αναζήτησης αλλά και η και ο τρόπος που επιτυγχάνεται να εμφανίζουμε τα αποτελέσματα έχει αναλυθεί σε προηγούμενα κεφάλαια. Στη διαδικασία της αναζήτησης συναντάμε την δυνατότητα διαχείρισης γεγονότων πληκτρολογίου. Συγκεκριμένα, μόλις εστιάσουμε στην περιοχή κειμένου προς αναζήτηση τότε εμφανίζεται και το πληκτρολόγιο. Παρατηρούμε ότι το κουμπί κάτω δεξιά, το οποί είναι το “enter” έχει την εικόνα ενός μεγεθυντικού φακού. Μόλις το πατήσουμε ενεργοποιείται η αναζήτηση και το πληκτρολόγιο φεύγει από την οθόνη. Παρακάτω το απόσπασμα κώδικα.

```
et_search.setOnKeyListener(new OnKeyListener() {

                public boolean onKey(View v, int keyCode, KeyEvent event) {

                    // TODO Auto-generated method stub
```

```
                if(keyCode == KeyEvent.KEYCODE_ENTER){
                        getEditText = et_search.getText();
                        performSearch(getEditText);
                        InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
                imm.hideSoftInputFromWindow(et_search.getWindowToken(), 0);
                }
```

Ακόμη, θέτουμε και ως χαρακτηριστικό της περιοχής κειμένου (EditText) τη μέθοδο εισόδου, παρακάτω απόσπασμα κώδικα.

```
android:imeOptions="actionSearch"
```

Με αυτό τον τρόπο καθορίζουμε ποια ενέργεια θα γίνει μόλις ολοκληρωθεί η είσοδος της λέξεις προς αναζήτηση από τον χρήστη.

5.3.10 Log

Κλείνοντας το κεφάλαιο, αξίζει να αναφερθούμε στην κλάση Log. Δεν έχει άμεσο ενδιαφέρον για την ανάπτυξη της εφαρμογής αλλά μπορεί να φανεί πολύ χρήσιμη στην ανεύρεση σφαλμάτων. Με την κλάση αυτή μπορούμε να εμφανίζουμε μηνύματα κατά την διάρκεια εκτέλεσης στο παράθυρο LogCat του Eclipse. Κάποιες από τις προσφερόμενες μεθόδους, οι οποίες χρησιμοποιήθηκαν και στην συγκεκριμένη εφαρμογή είναι

- **i()** (info): Μας δίνει απλές απλές πληροφορίες το χρώμα τη γραμματοσειράς του μηνύματος είναι πράσινο.
- **e()** (error): Χρησιμοποιείται για κάποιο σφάλμα, εμφανίζει μήνυμα με κόκκινο χρώμα.

- **d()** (debug): Αν θέλετε να εκτυπώσετε μια δέσμη των μηνυμάτων, έτσι ώστε να μπορεί να καταγράψει την ακριβή ροή του προγράμματός σας, χρησιμοποιήστε αυτή

Το LogCat είναι ένα ισχυρό εργαλείο που σίγουρα αξίζει να ενσωματωθεί στη διαδικασία ελέγχου που έχουμε επιλέξει για τις εφαρμογές μας.

Σύνοψη

Στο κεφάλαιο **Ανάπτυξη του ZoteroApp**, παρουσιάστηκαν τα βασικά βήματα ανάπτυξης της εφαρμογής. Έγινε εκτενείς αναφορά σε κάποια συστατικά της εφαρμογής τα οποία είναι πολύ σημαντικά για κάθε εφαρμογή Android.

Επίλογος - Προτάσεις

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας είναι η προσπάθεια για να έρθουμε πιο κοντά στην δημιουργία εφαρμογών Android και να κατανοήσουμε τον τρόπο που λειτουργούν. Στόχος ήταν ο χρήστης της εφαρμογής να μπορεί να περιηγηθεί στα περιεχόμενα της βιβλιογραφίας του, να μπορεί να δει τις πληροφορίες του καθενός, αλλά και να κατεβάσει συνημμένα αρχεία. Η εφαρμογή προορίζεται για συσκευές tablet, δηλαδή για συσκευές με μεγάλη οθόνη. Μελλοντικά θα μπορούσε να αναπτυχθεί η εφαρμογή ώστε να μπορεί να αποθηκεύει τα δεδομένα της βιβλιογραφίας οργανωμένα σε βάση δεδομένων (SQLite Database). Αυτό θα επιτρέψει την εργασία χωρίς σύνδεση στο διαδίκτυο, που μέχρι τώρα είναι απαραίτητη. Η βάση που θα αναπτυχθεί θα

μπορεί να συγχρονίζει με την βάση που είναι στον server. Επιπλέον, θα βοηθούσε στην περαιτέρω διάδοση της εφαρμογής η συμβατότητα με διάφορα μεγέθη οθόνης, δηλαδή η εφαρμογή να είναι διαθέσιμη για smartphone και για tablet.

Αναφορές

1: https://www.zotero.org/support/dev/web_api/v3/start

2: https://www.zotero.org/support/dev/web_api/v3/basics

3: http://www.javacodegeeks.com/2010/10/android-full-application-tutorial.html

4:http://www.javacodegeeks.com/2010/10/android-full-app-part-1-main-activity.html

5:http://www.javacodegeeks.com/2010/10/android-full-app-part-2-using-http-api.html

6:http://www.javacodegeeks.com/2010/10/android-full-app-part-4-asynchronous.html

7:http://www.javacodegeeks.com/2010/11/android-full-app-part-5-launch-activity.html

8:http://www.javacodegeeks.com/2010/11/android-full-app-part-6-customized-list.html

9: http://www.javacodegeeks.com/2010/11/boost-android-xml-parsing-xml-pull.html

10: http://developer.android.com/guide/topics/manifest/manifest-intro.html

11: http://developer.android.com/guide/components/fundamentals.html

12: http://developer.android.com/guide/topics/ui/declaring-layout.html

13: http://developer.android.com/guide/topics/data/index.html

14: http://developer.android.com/guide/practices/tablets-and-handsets.html

15: http://developer.android.com/guide/topics/resources/providing-resources.html

16: http://developer.android.com/guide/topics/ui/index.html

17: http://developer.android.com/reference/android/util/Log.html

18:http://tools.android.com/tech-docs/new-build-system/migrating-from-eclipse-projects

19: http://el.wikipedia.org/wiki/RSS

20: http://el.wikipedia.org/wiki/XML

21: http://en.wikipedia.org/wiki/Android_version_history

22:http://javarevisited.blogspot.gr/2011/12/parse-xml-file-in-java-example-tutorial.html

23:http://javarevisited.blogspot.gr/2013/06/introduction-of-how-android-works-Java-programmers.html

24:http://javarevisited.blogspot.gr/2011/12/parse-read-xml-file-java-sax-parser.html

25: http://www.w3schools.com/json/

26: http://json.org/

27: www.eclipse.org/downloads/

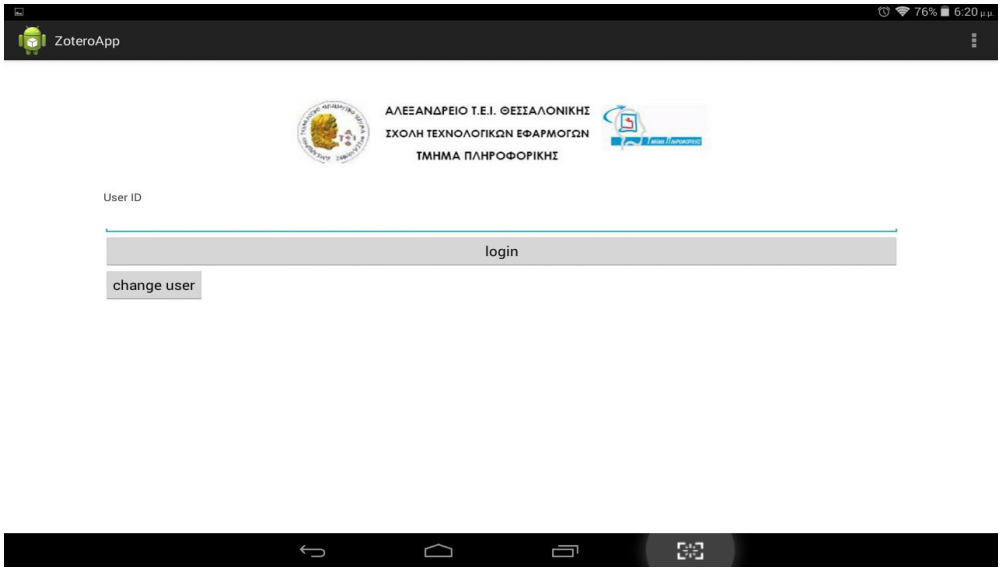
28:http://developer.android.com/training/basics/network-ops/xml.html

Οδηγός χρήσης λογισμικού

Το ZotreroApp είναι η εφαρμογή που αναπτύχθηκε στο πλαίσιο της πτυχιακής εργασίας με τίτλο "*Δημιουργία εφαρμογής Android για το σύστημα διαχείρισης βιβλιογραφικών αναφορών Zotero*". Η εφαρμογή έχει πρόσβαση στο λογαριασμό Zotero του χρήστη. Στόχος της είναι ο χρήστης να μπορεί δει τα στοιχεία της βιβλιογραφίας που έχει αποθηκευμένα στο λογαριασμό του. Η λειτουργία της εφαρμογής απαιτεί την ύπαρξη σύνδεσης στο internet.

Εκκίνηση εφαρμογής

Μόλις εκκινήσει η εφαρμογή ο χρήστης αντικρίζει την εξής εικόνα.



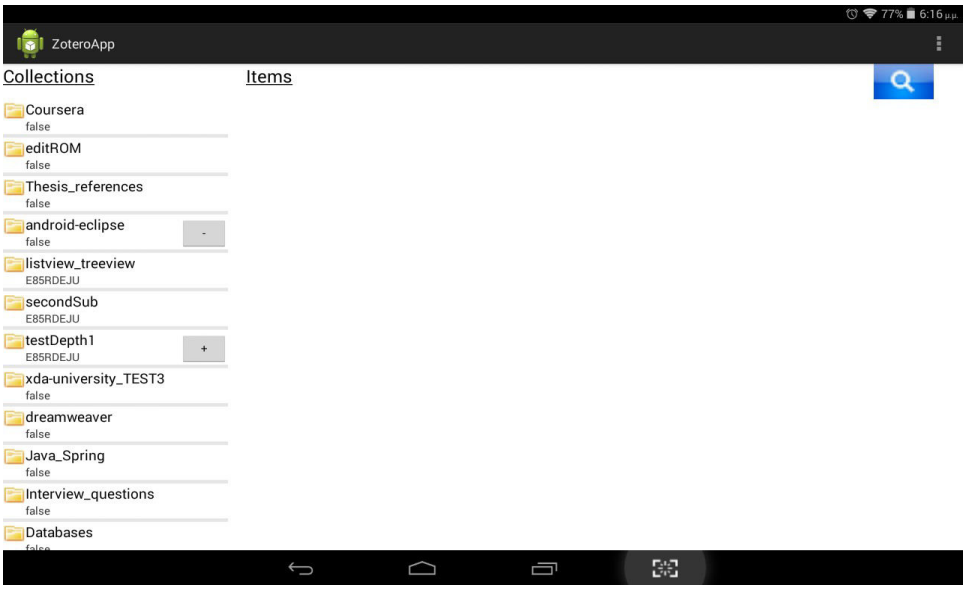
Εικόνα: xiv

Κατόπιν, πληκτρολογούμε το userID και πατάμε login ώστε να μεταφερθούμε στην επόμενη Activity.

Σημείωση: Εάν δεν έχουμε καταχωρήσει το userKEY τότε πατάμε το κουμπί change user, ώστε να καταχωρήσουμε το κλειδί του χρήστη που παρέχεται από το zotero. Αυτό αποθηκεύεται μόνιμα στη συσκευή και δεν είναι απαραίτητο να το πληκτρολογήσουμε κάθε φορά. Εάν θέλουμε να κάνουμε login ως διαφορετικός χρήστης τότε ακολουθούμε την ίδια διαδικασία.

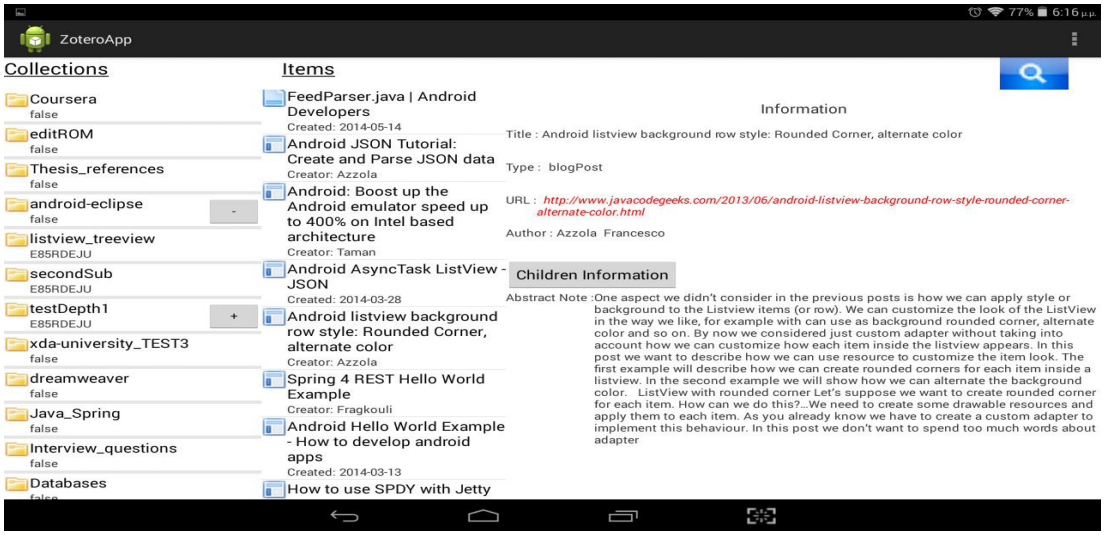
Κεντρική οθόνη

Η επόμενη οθόνη που αντικρίζει ο χρήστης φαίνεται παρακάτω.



Εικόνα: xv

Η εικόνα δείχνει μία λίστα με όλες τις συλλογές κορυφή του χρήστη. Όταν ο χρήστης επιλέξει να δει τα περιεχόμενα μιας συγκεκριμένης συλλογής τότε εμφανίζεται ακριβώς δίπλα της μια δεύτερη λίστα με τα περιεχόμενα της εκάστοτε συλλογής. Εάν σε μια συλλογή υπάρχουν υποσυλλογές τότε εμφανίζεται δίπλα στο όνομα της ένα κουμπί με το σύμβολο σύν (+), όπου πατώντας πάνω σε αυτό εμφανίζονται ακριβώς από κάτω οι υποσυλλογές, με τον ίδιο τρόπο μπορούμε να κρύψουμε τις υποσυλλογές. Η εφαρμογή υποστηρίζει την αναπαράσταση των συλλογών με βάθος μεγαλύτερο του ένα.

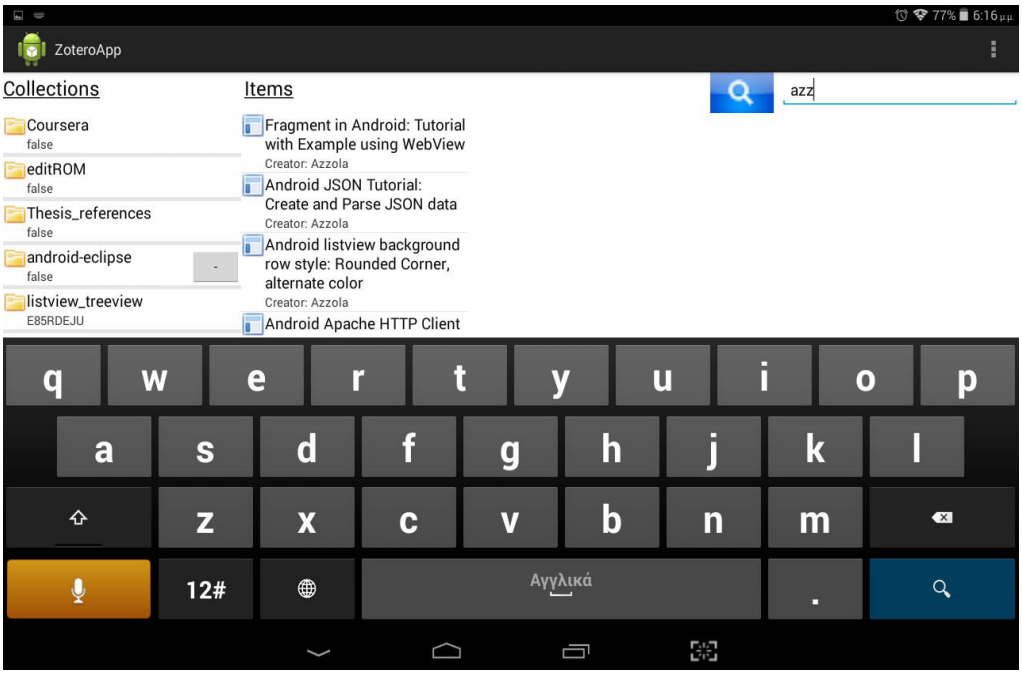


Εικόνα: xvi

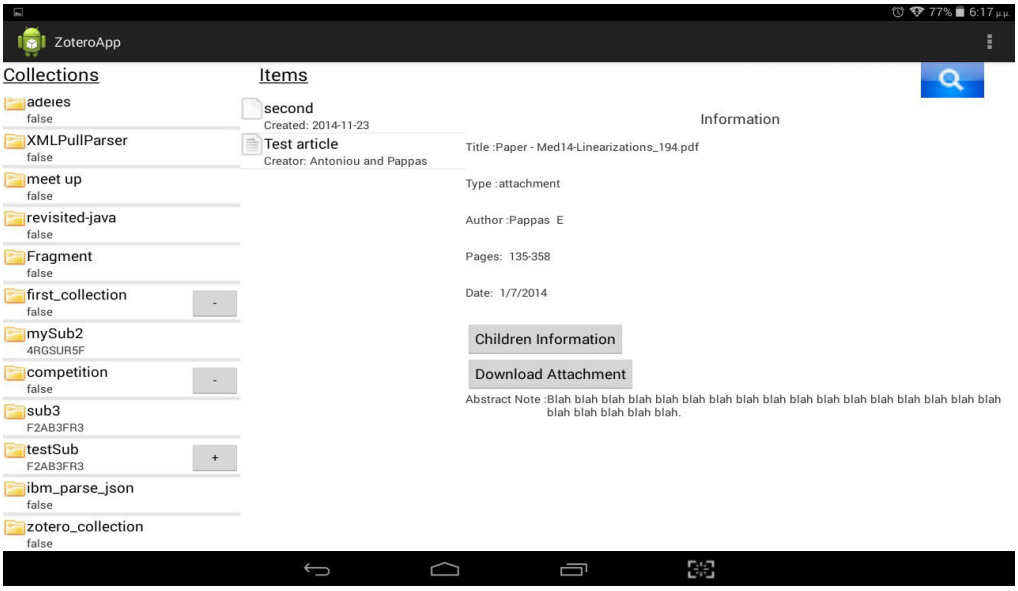
Στο δεξί κομμάτι της κάθε λίστας υπάρχει εικονίδιο το οποίο αναπαριστά το είδος του στοιχείου που αναγράφεται π.χ. Συλλογή, άρθρο εφημερίδας, ιστοσελίδα, αρχείο pdf κ.ο.κ.

Επιλέγοντας κάποιο στοιχείο από τη λίστα εμφανίζονται για αυτό στο δεξιό μέρος της οθόνης πληροφορίες.

Εμφανίζονται βασικές πληροφορίες που μπορεί να φανούν χρήσιμες στο χρήστη . Αυτές είναι ο τύπος και το όνομα του στοιχείου, το URL εάν υπάρχει, το οποίο είναι link. Πολύ σημαντική πληροφορία είναι και τα DOI (Digital Object Identifier) ISSN και ISBN εάν υπάρχουν. Στο κάτω μέρος της οθόνης το κείμενο με την περίληψη του στοιχείου που αναφέρεται. Τέλος, το κουμπί ***Children Information***. Πατώντας το συγκεκριμένο κουμπί εμφανίζονται πληροφορίες για στοιχεία παιδιά και εάν υπάρχει κάποιο συνημμένο αρχείο τότε εμφανίζεται το κουμπί ***Download Attachment*** με το οποίο μπορούμε να κατεβάσουμε το συνημμένο αρχείο. Δίνεται η δυνατότητα στον χρήστη να κάνει και αναζήτηση στοιχείου σε ολόκληρη τη βιβλιοθήκη. Εφόσον επιθυμεί να πραγματοποιήσει αναζήτηση τότε πρέπει να κάνει κλικ στο εικονίδιο με τον μεγεθυντικό φακό ώστε να εμφανιστεί ο χώρος του κειμένου προς αναζήτηση. Στην παρακάτω εικόνα η λειτουργία της αναζήτησης εμφανίζεται στο πάνω δεξιό μέρος της οθόνης. Άλλη μια δυνατότητα από το μενού της εφαρμογής είναι ανανέωση (refresh), η οποία μας εμφανίζει την αρχική εικόνα της κεντρικής οθόνης, με τις όποιες αλλαγές έχουν υπάρξει στις συλλογές. Παρακάτω εικόνες με τις λειτουργίες που περιγράφονται

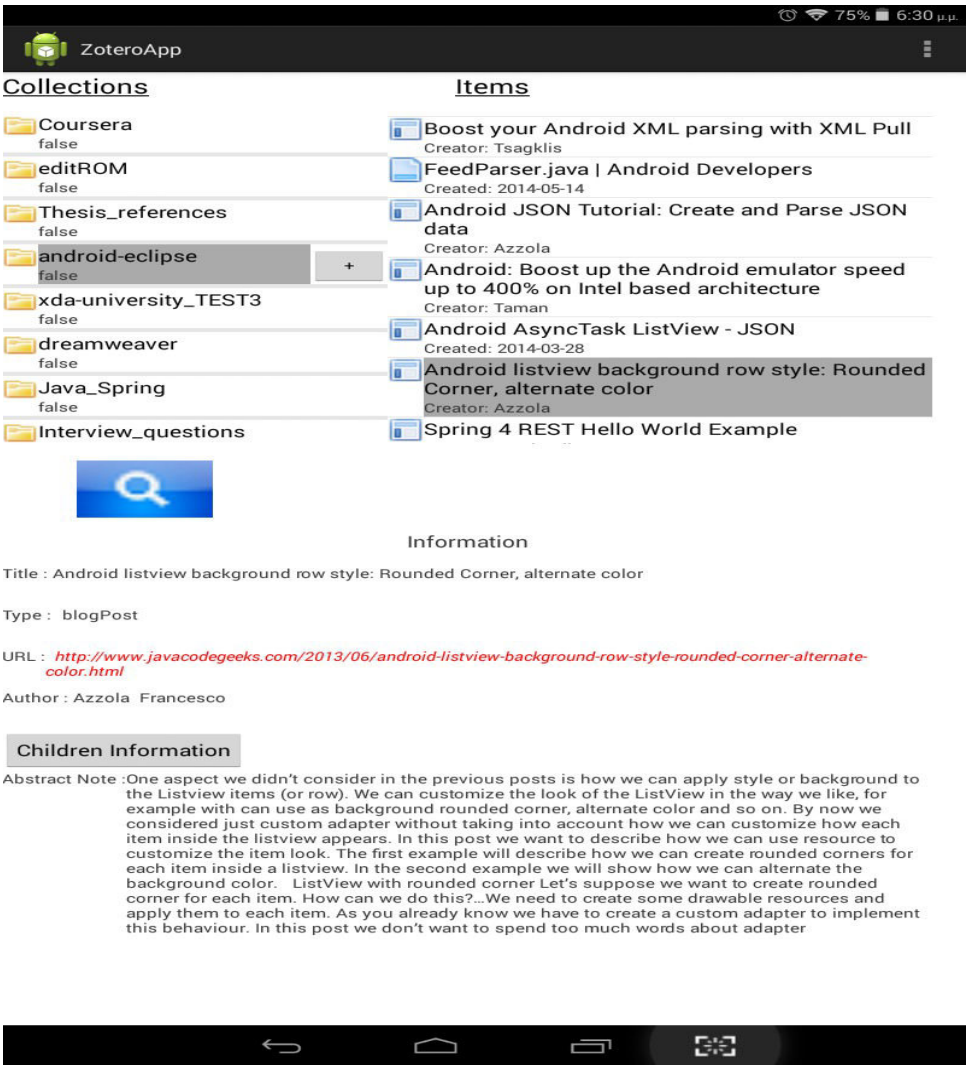


Εικόνα: xvii



Εικόνα: xviii

Τέλος, η εφαρμογή υποστηρίζει και την περιστροφή της οθόνης. Μέχρι στιγμής παρουσιάστηκαν εικόνες από *landscape mode.* Παρακάτω, η εικόνα με την *portrait mode* της εφαρμογής.



Εικόνα: xiv