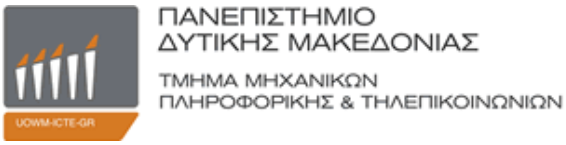
2012

Πανεπιστήμιο Δυτικής Μακεδονίας

Πραμαγκιούλης Ευστάθιος



Επιβλέπων καθηγητής : Δασυγένης Μηνάς

Μέλη Εξεταστικής Επιτροπής : Δασυγένης Μηνάς
Ζυγκιρίδης Θεόδωρος

[ΚΑΤΑΣΚΕΥΗ ΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΣΤΗΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ANDROID ΠΟΥ ΥΠΟΛΟΠΟΙΕΙ ΕΝΑ ΠΑΙΧΝΙΔΙ ΕΡΩΤΟΑΠΑΝΤΗΣΕΩΝ]

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον λέκτορα καθηγητή του τμήματος Μηχανικών Πληροφορικής και Τηλεπικοινωνιών Δασυγένη Μηνά για την καθοδήγηση και το πνεύμα συνεργασίας καθ'όλη την διάρκεια αυτής της διπλωματικής εργασίας καθώς και τον συνάδελφο Σχισμένο Ηλία για την βοήθεια που μου προσέφερε όσον αφορά την δοκιμή της εφαρμογής σε φυσικές συσκευές.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την ανάπτυξη μιας διαδικτυακής ηλεκτρονικής εφαρμογής σε περιβάλλον Android. Η εφαρμογή επιτρέπει στον κάτοχο κινητού τηλεφώνου με τέτοιο λειτουργικό σύστημα να παίξει ένα παιχνίδι ερωτοαπαντήσεων μαζί με άλλους παίκτες – συσκευές έχοντας την δυνατότητα να είναι διακομιστής σε ένα παιχνίδι ή απλά να πάρει μέρος σε ένα.

Το παιχνίδι είναι στην ουσία μια βάση δεδομένων που αποτελείται από έξι στήλες, τον αύξοντα αριθμό, την ερώτηση και τις πιθανές απαντήσεις και η επίσημη έκδοση της βρίσκεται στους servers του Πανεπιστημίου Δυτικής Μακεδονίας και ο καθένας μπορεί να την κατεβάσει, αρκεί να έχει δικαιώματα υπερχρήστη στην συσκευή του.

Αν θελήσουμε να συνοψίσουμε τα βασικά στοιχεία αυτής της διπλωματικής εργασίας, αυτά επικεντρώνονται σε δύο άξονες. Ο πρώτος περιστρέφεται γύρω από τις λειτουργικές παραμέτρους και την αισθητική παρουσίαση της εφαρμογής και οριοθετεί την ευχρηστία και την ευκολία πρόσβασης μέσω του interface ενός έξυπνου κινητού τηλεφώνου (smart phone). Ο δεύτερος αφορά την αξιοπιστία και την διασύνδεση πολλών συσκευών σε μία συγκεκριμένη χρησιμοποιώντας το πρωτόκολλο TCP και την κοινή χρήση μιας αναπτυσσόμενης βάσης δεδομένων.

Συμπερασματικά, πρόκειται για μια πλήρη διαδικτυακή εφαρμογή σε περιβάλλον Android, η οποία αναμένεται να τύχει ευρείας αποδοχής από τους χρήστες smart phones η οποία θα είναι ιδιαίτερα εύπλαστη, όσον αφορά το θέμα της διαχείρισης βάσης δεδομένων και θα είναι σε θέση να ακολουθεί τις εκάστοτε αλλαγές που θα προτείνονται από τα feedback των χρηστών.

Λέξεις Κλειδιά : Android, TCP sockets, SQLite database, server, client.

“Construction of a web-based android application that implements a question game”

Abstract

The main goal of this Diploma Thesis is the development of a web-based application within the Android environment. The development allows the owner of a mobile smart-phone, equipped with the Android operating system, to play a question game together with other players – devices having the possibility to act as a server or participate as a client to a game.

The game, actually, is a database that consists of six columns, identification number, question and possible answers and the official version of the servers is located at the servers of University of Western Macedonia and anyone can download it, as long as he has root privileges to his device.

In order to summarize the key elements of this Diploma Thesis, they focus on two main aspects. The first one has to do with the operational parameters and the aesthetic presentation of the application and defines the usability and the ease of access through the interface of a smart-phone. The second concerns the reliability and the connection of many devices to a single one using the TCP protocol and the common use of a developing database.

In conclusion, this is a complete web application in the Android environment, which is expected to gain broad acceptance by the users of such devices and it will be particularly pliable, regarding the issue of managing a database and it will be able to follow the respective changes that will be proposed by the user’s feedback.

Keywords: Android, TCP sockets, SQLite database, server, client.

Πίνακας Περιεχομένων

1

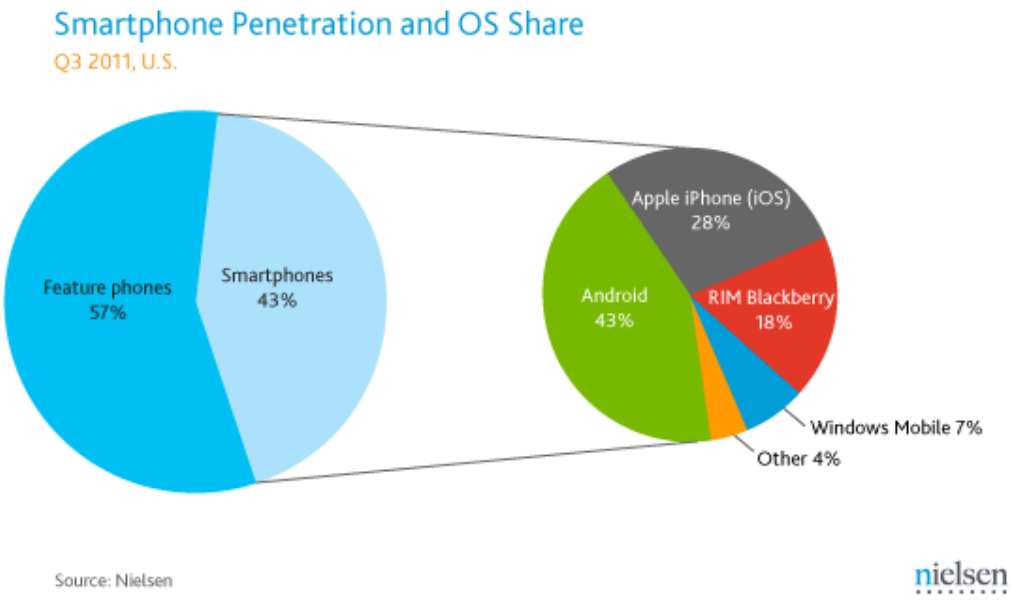
Εισαγωγή

1.1. Αντικείμενο της διπλωματικής

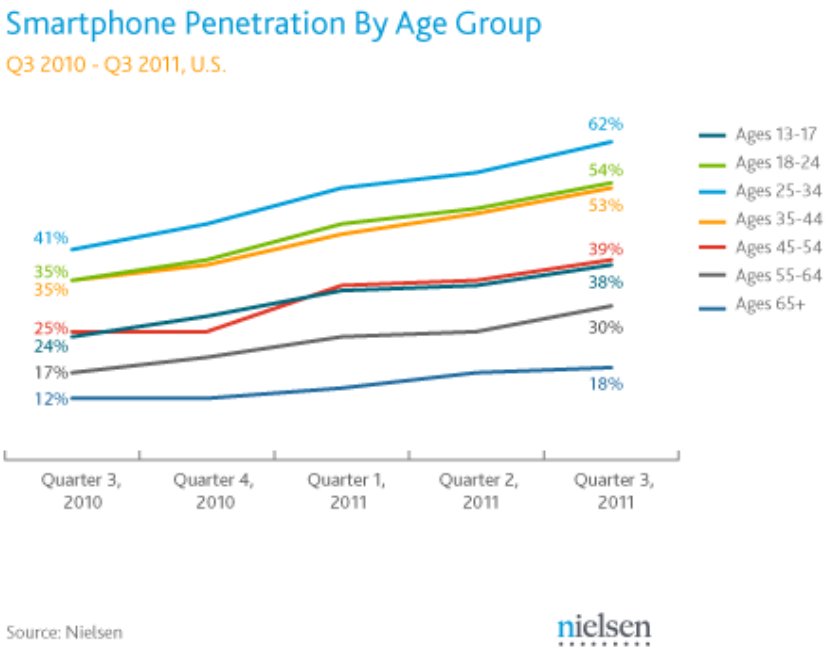
Τα τελευταία χρόνια βιώνουμε μια επανάσταση των «έξυπνων» κινητών τηλεφώνων, τα οποία ενσωματώνουν δυνατότητες που δεν υπήρχαν στα μέχρι σήμερα κινητά τηλέφωνα. Περιλαμβάνουν ολοκληρωμένο λειτουργικό σύστημα, ισχυρό επεξεργαστή ικανό για εκτέλεση χρονοβόρων υπολογισμών, δυνατότητα σύνδεσης στο διαδίκτυο είτε μέσω ασυρμάτων δικτύων (Wi-Fi) είτε δικτύων τρίτης γενιάς (3G networks), κάμερα υψηλής ανάλυσης, συστήματα εντοπισμού θέσης (GPS), ένας αριθμός από αισθητήρες κ.α. Κάθε είδους αισθητήρας μπορεί να υπάρχει σε ένα Smartphone, όπως αισθητήρας θερμοκρασίας περιβάλλοντος, αισθητήρας μέτρησης θορύβου καθώς και πολλοί άλλοι. Το λειτουργικό σύστημα της Google για smartphone έκανε την εμφάνισή του το 2008.Το Android είναι μια πλατφόρμα ανοιχτού λογισμικού που αναπτύχτηκε από προγραμματιστές της ίδιας και προγραμματιστές των Intel, HTC, ARM, Motorola και Samsung. Βέβαια στην αγορά υπάρχουν και άλλα λειτουργικά συστήματα που προορίζονται για κινητά τηλέφωνα με το Android αυτή τη στιγμή να βρίσκεται στην δεύτερη θέση. Περιγραμματικά τα σημαντικότερα λειτουργικά είναι τα εξής :

- Windows Phone
- iOS
- Android
- Blackberry (RIM)
- Bada

Σύμφωνα με επίσημα στοιχεία η ιστοσελίδα της Google, Android Market (πλέον Google Play) περιλαμβάνει αυτή την στιγμή περισσότερες από 250.000 εφαρμογές.



«Εικόνα 1 : Αριστερά: Μερίδιο τυπικών έναντι έξυπνων τηλεφώνων. Δεξιά: Μερίδιο ανά λειτουργικό σύστημα»



«Εικόνα 2 : Σχέση ηλικίας και κατοχής Smartphone»

Στο πλαίσιο αυτό, σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας εφαρμογής, που προορίζεται για τα παραπάνω κινητά τηλέφωνα και όσες συσκευές περιλαμβάνουν το λειτουργικό σύστημα Android (tablets, pc). Με την εφαρμογή αυτή ο χρήστης θα μπορεί να εμπλουτίσει τις γνώσεις του σε καίρια θέματα που συνέβησαν στον κόσμο. Επιπλέον, μπορεί

πολύ εύκολα να χαλαρώσει και να περάσει δημιουργικά και ευχάριστα αρκετό χρόνο, όπως για παράδειγμα τα διαλλείματα του από την δουλειά ή το σχολειό καθώς και ένα πολύωρο ταξίδι.

1.2. Οργάνωση κειμένου

Η διπλωματική αυτή εργασία αναλύεται συνολικά σε 6 κεφάλαια. Συγκεκριμένα :

Στο **1ο Κεφάλαιο** ασχολούμαστε με την εισαγωγή στο αντικείμενο που πραγματεύεται η διπλωματική εργασία. Εισάγουμε τον αναγνώστη στις νέες τεχνολογίες όσον αφορά τα κινητά τηλέφωνα.

Το **2ο κεφάλαιο** περιγράφει το λειτουργικό σύστημα Android. Εξηγούνται τα προγραμματιστικά εργαλεία που χρειάζονται και αναλύονται τα βασικά βήματα που απαιτούνται για την ανάπτυξη μιας εφαρμογής που θα εκτελείται σε αυτό το λειτουργικό σύστημα. Περιγράφονται ορισμένες προγραμματιστικές τεχνικές και αναλύονται τρόποι βελτιστοποίησης κάθε εφαρμογής ώστε να είναι λειτουργική και προσβάσιμη στο ευρύ κοινό. Αποτελεί λοιπόν ένα οδηγό που πρέπει να ακολουθήσει ο προγραμματιστής ώστε να αναπτύξει μια εφαρμογή Android.

Στο **3ο κεφάλαιο** γίνεται περιγραφή στο κομμάτι της διασύνδεσης ανάμεσα σε συσκευές με το λειτουργικό σύστημα Android. Προσπαθούμε να κάνουμε τον χρήστη οικείο με έννοιες, όπως το πρωτόκολλο TCP και ο προγραμματισμός με sockets (socket programming), βασικές για την κατανόηση της λειτουργίας της εφαρμογής που αναπτύξαμε.

Στο **4ο κεφάλαιο** παρουσιάζεται αναλυτικά η αρχιτεκτονική της εφαρμογής που υλοποιήθηκε στο πλαίσιο αυτής της διπλωματικής εργασίας. Αρχικά αναφέρονται οι διάφορες λειτουργίες και δυνατότητες της εφαρμογής και έπειτα περιγράφεται η δομή της, το πως δηλαδή οργανώνονται όλα τα αρχεία που περιέχονται σε αυτήν και ποια είναι η λειτουργία τους.

Στο **5ο κεφάλαιο** παρουσιάζονται διεξοδικά όλες οι λειτουργίες της εφαρμογής και οι επιλογές που έχει ο χρήστης αλληλεπιδρώντας με αυτήν. Περιγράφεται κάθε διαφορετική οθόνη που συναντά καθώς και οι επιλογές που έχει όπως επίσης και ο τρόπος υλοποίησης αυτών των λειτουργιών στο προγραμματιστικό περιβάλλον. Όλα τα αποτελέσματα παρουσιάζονται σε εικόνες, οι οποίες εμφανίζουν ένα στιγμιότυπο από την αντίστοιχη οθόνη της συσκευής που εκτελεί την εφαρμογή και βοηθάνε τον αναγνώστη στην κατανόηση των παραπάνω λειτουργιών.

Στο **6ο κεφάλαιο** αναλύονται τα συμπεράσματα που προκύπτουν από την εκπόνηση της διπλωματικής αυτής εργασίας, καθώς και οι μελλοντικές εξελίξεις της εφαρμογής. Αναφέρονται ακόμα συγκεκριμένες επεκτάσεις που σχεδιάζουμε να συμπεριλάβουμε στο προσεχές διάστημα με σκοπό την δημοσιοποίηση της εφαρμογής στο Android Market όπου μπορεί να είναι διαθέσιμη στο ευρύ κοινό.

Τέλος, παρουσιάζεται η **Βιβλιογραφία** που χρησιμοποιήθηκε για την συγγραφή του παρόντος κειμένου, καθώς και οι ιστοσελίδες και όλες οι πηγές που βοήθησαν στην ανάπτυξη της εφαρμογής.

2

Προγραμματισμός σε περιβάλλον Android

2.1. Εισαγωγή

Το Android είναι ένα λειτουργικό σύστημα για κινητές συσκευές, όπως τα smartphones και τα tablets και είναι βασισμένο στα Linux. Όλες οι λειτουργίες του κινητού τηλεφώνου ελέγχονται από τις εφαρμογές (apps), τις οποίες ο χρήστης μπορεί να κατεβάσει και να εγκαταστήσει ελεύθερα.

Αξίζει να σημειωθεί ότι το Android είναι ένα ελεύθερο λογισμικό ανοικτού κώδικα (free and open source software) πράγμα που επιτρέπει στους χρήστες την ανάπτυξη δωρεάν παιχνιδιών και κάθε είδους εφαρμογής. Αυτή ακριβώς η δυνατότητα συντέλεσε στην ταχεία διάδοση του. Επιπλέον, τα εργαλεία για να αναπτύξει εφαρμογές οποιοσδήποτε προγραμματιστής μπορεί να τα βρει δωρεάν στο Internet.

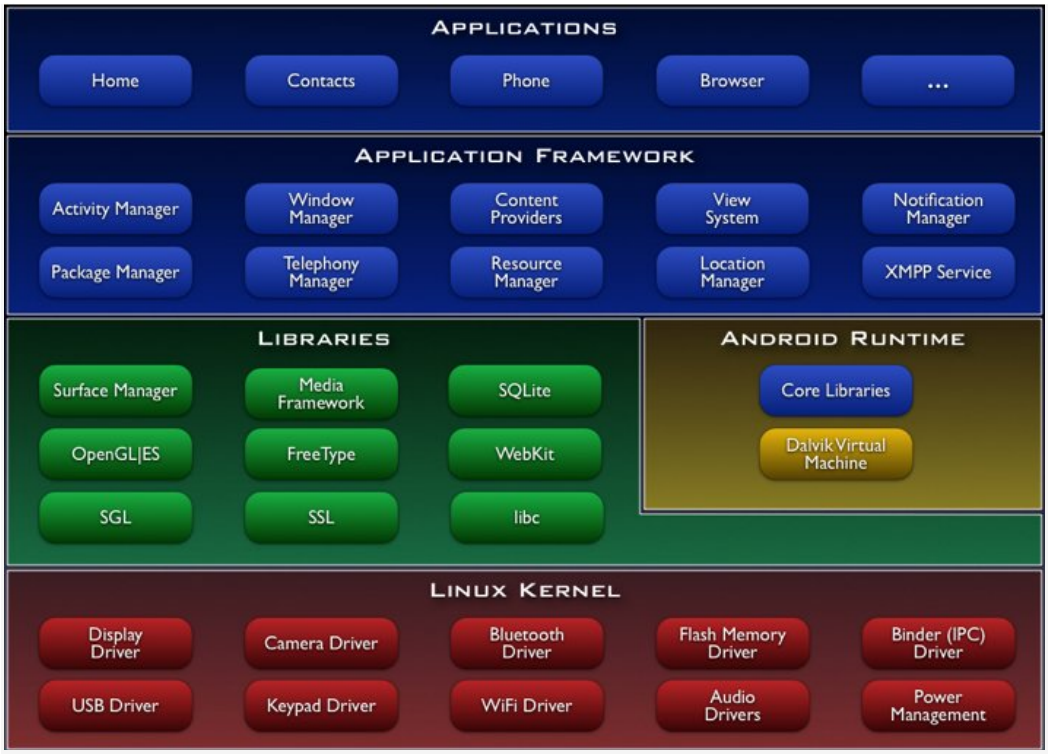
2.2. Γενικά για το Android

2.2.1. Ιστορικά Στοιχεία

Το λειτουργικό σύστημα Android αναπτύχθηκε από την εταιρεία Android Inc. που εξαγοράστηκε πλήρως από την Google τον Αύγουστο του 2005. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Δύο χρόνια αργότερα, τον Νοέμβριο του 2007, έκανε την εμφάνισή της η Open Handset Alliance, μια κοινοπραξία 48 τηλεπικοινωνιακών εταιρειών, εταιρειών λογισμικού και κατασκευής hardware, ανάμεσα τους ονόματα όπως LG, Intel, HTC, Motorola και NVidia. Στόχος της ήταν η δημιουργία ανοικτών προτύπων για κινητές συσκευές. Παράλληλα, δημοσίευσαν το πρώτο τους προϊόν, το Android, μια πλατφόρμα βασισμένη στον πυρήνα του Linux (Linux kernel)
2.6. Η Google έδωσε στην δημοσιότητα και το μεγαλύτερο μέρος του πηγαίου κώδικα του Android υπό τους όρους της Apache License.

2.2.2. Σχεδιασμός

Το Android αποτελείται από έναν kernel βασισμένο σε αυτόν του Linux με το middleware, τις βιβλιοθήκες και τα APIs να είναι γραμμένα σε C και το software των εφαρμογών που τρέχει πάνω σε ένα πλαίσιο (applications framework) που περιλαμβάνει βιβλιοθήκες συμβατές με την Java. Το Android χρησιμοποιεί την εικονική μηχανή Dalvik (Dalvik virtual machine). Η κύρια πλατφόρμα του hardware είναι η ARM αρχιτεκτονική που χρησιμοποιείται σε ευρέως σε 32-bit συστήματα.



«Εικόνα 3 : Διάγραμμα Αρχιτεκτονικής»

2.2.3. Linux

Η αρχιτεκτονική του Linux είναι βασισμένη στις αρχές του λειτουργικού Unix αλλά έχει αναπτυχθεί εκ του μηδενός και δεν περιλαμβάνει κώδικα από το Unix. Η ανάπτυξη του Linux είναι χαρακτηριστικό παράδειγμα εθελοντικής συνεργασίας από διαδικτυακές κοινότητες, ενώ όλο το έργο είναι ανοικτού κώδικα και ελεύθερα προσβάσιμο από όλους για αντιγραφή, τροποποίηση ή αναδιανομή χωρίς περιορισμό. Το Linux είναι διαθέσιμο υπό άδειες όπως η GNU General Public License.

Δημιουργός του πυρήνα Linux είναι ο Linus Torvalds, από το όνομα του οποίου προήλθε και η ονομασία Linux. Ο Torvalds άρχισε να αναπτύσσει έναν kernel το 1991 εμπνευσμένος από το λειτουργικό MINIX και χρησιμοποιώντας πολλά προγράμματα και βιβλιοθήκες από το GNU του Richard Stallman. Πάνω στον αρχικό πυρήνα του Torvalds έχουν εργαστεί

χιλιάδες χρήστες αλλά και εταιρείες. Λόγω των στενότατων σχέσεων μεταξύ Linux και GNU, πολλές φορές το σύστημα αυτό αναφέρεται ως GNU/Linux, ονομασία που είναι πιο ακριβής και την προτιμά και το Ίδρυμα Ελεύθερου Λογισμικού.

Ο kernel του Android είναι βασισμένος σε αυτόν του Linux αλλά έχει αρκετές αρχιτεκτονικές διαφορές κυρίως χάρη στην Google. Δεν υποστηρίζει το πλήρες σύνολο των καθιερωμένων βιβλιοθηκών GNU πράγμα που καθιστά δύσκολη την μεταφορά εφαρμογών και βιβλιοθηκών από το Linux στο Android. Ορισμένα χαρακτηριστικά που η Google έδωσε πίσω στον Linux kernel, ένα σύστημα διαχείρισης ενέργειας που ονομάζεται wake lock, απορρίφτηκε από τους προγραμματιστές του κυρίως kernel επειδή φάνηκε ότι δεν έδIνε και πολλή σημασία στην δικιά τους δουλειά. Σήμερα πολλοί προγραμματιστές εργάζονται για τη τροποποίηση των υφιστάμενων kernel και ROM, για να δημιουργήσουν νέα που θα είναι συμβατά με το νέο λειτουργικό σύστημα Android. Αυτές οι προσπάθειες για το Android γίνονται συνήθως μέσω του XDA-developers και του androidforums.com.

2.2.4. Χαρακτηριστικά

- **Συνδεσιμότητα:** Το Android υποστηρίζει διάφορες τεχνολογίες ανάμεσα στις οποίες είναι GMS/EDGE, Bluetooth, Wi-Fi, WiMAX και LTE.

- **Μηνύματα:** Υποστηρίζει τις κλασσικές μορφές SMS και MMS καθώς επίσης και το Android Cloud To Device Messaging (C2DM), που είναι μια υπηρεσία προώθησης ειδοποιήσεων που βοηθά τους προγραμματιστές να στέλνουν δεδομένα από τον server στις εφαρμογές τους.

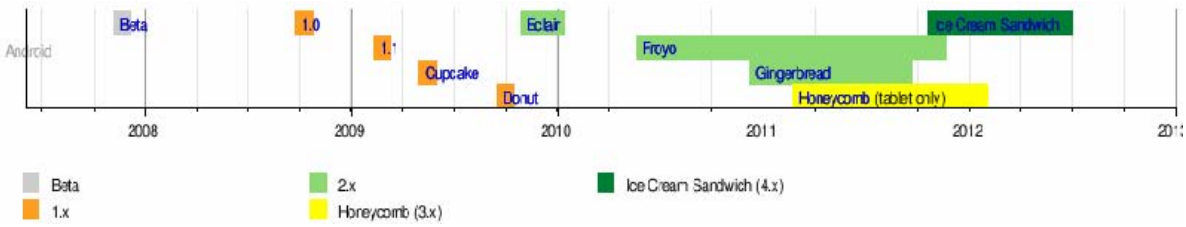
- **Φυλλομετρητής (browser):** Ο browser που είναι διαθέσιμος στο Android βασίζεται στην διάταξη ανοικτού κώδικα WebKit, που είναι μια μηχανή που ενδυναμώνει διάφορους browsers**.**

- **Υποστήριξη Java:** Ενώ οι περισσότερες εφαρμογές είναι γραμμένες σε Java, δεν υπάρχει Java virtual machine και δεν εκτελείται Java byte code. Οι κλάσεις στην Java συντάσσονται σε εκτελέσιμα τα οποία τρέχουν πάνω στο Dalvik, ένα virtual machine ειδικά για το Android που είναι βελτιστοποιημένο για συσκευές που τροφοδοτούνται από μπαταρία με περιορισμένη μνήμη και υπολογιστική ισχύ.

- **Αποθήκευση:** Χρησιμοποιείται το SQLite, μια μικρή σε απαιτήσεις σχεσιακά βάση δεδομένων για την αποθήκευση δεδομένων

- **Επιπλέον:** υπάρχει ένα ευρύ φάσμα από χαρακτηριστικά όπως το Multitasking, κλήσεις μέσω IP, εξωτερική κάρτα μνήμης κ.α.

2.2.5. Εκδόσεις του Android



«Εικόνα 4 : **Χρονοδιάγραμμα των εκδόσεων του Android**»

Μέχρι σήμερα το Android έχει εξελιχθεί αρκετές φορές δημιουργώντας μια πληθώρα εκδόσεων διαθέσιμες στο ευρύ κοινό. Βέβαια δεν ενδείκνυνται όλες για οποιαδήποτε συσκευή έχει κανείς. Παρακάτω θα αναφέρουμε όλες τις υπάρχουσες εκδόσεις και θα προσπαθήσουμε να αναφέρουμε επιγραμματικά τις εξελίξεις της καθεμιάς. Συγκεκριμένα :

- **Android 1.0**

Η πρώτη εμπορική έκδοση του λογισμικού, που κυκλοφόρησε στις 23 Σεπτεμβρίου 2008. Η πρώτη συσκευή Android, το HTC Dream, ενσωματώνει τα ακόλουθα χαρακτηριστικά του Android 1.0:

- Την εφαρμογή Android Market όπου ο χρήστης μπορεί να κατεβάσει νέες εφαρμογές και να αναβαθμίσει αυτές που ήδη έχει.
- Browser για την διαχείριση ιστοσελίδων
- Υποστήριξη κάμερας
- Πρόσβαση σε email-servers, υποστηρίζοντας τα πρωτόκολλα POP-3, IMAP4 και SMTP.
- Ενσωμάτωσε τα Google Contacts, Calendar, Maps, Search, Talk.
- Υποστήριξη Wi-Fi και Bluetooth.

- **Android 1.1**

Στις 9 Φεβρουαρίου 2009, το Android 1.1 κυκλοφόρησε, αρχικά μόνο για το T-Mobile G.. Η ενημέρωση έλυσε σφάλματα, άλλαξε το API και πρόσθεσε μια σειρά από άλλα χαρακτηριστικά:

- Δυνατότητα να αποθηκεύσετε τα συνημμένα σε μηνύματα
- λεπτομέρειες και κριτικές όταν ένας χρήστης ψάχνει για τις επιχειρήσεις στο Google Maps.
- Προστέθηκε υποστήριξη για μαρκίζα σε διατάξεις του συστήματος

- **Android 1.5 (Cupcake)**

Τον Απρίλιο του 2009 κυκλοφόρησε η αναβάθμιση 1.5 βασισμένη στον Linux kernel 2.6.27. Πρόσθεσε τα ακόλουθα χαρακτηριστικά :

- Υποστήριξη για Widgets – μικροσκοπική θεώρηση της εφαρμογής που μπορεί να ενσωματωθεί σε άλλες εφαρμογές (όπως στην αρχική οθόνη) και να λαμβάνει περιοδικές ενημερώσεις.

«Εικόνα 5: Το λογότυπο του Cupcake»

- Προστίθενται χαρακτηριστικά αντιγραφής και επικόλλησης στο πρόγραμμα περιήγησης στο Web.
- Δυνατότητα να ανεβάσει ο χρήστης βίντεο στο YouTube.
- Δυνατότητα να ανεβάσει ο χρήστης φωτογραφίες στο Picasa.
- Αυτόματη αντιστοίχιση (pairing) και υποστήριξη στέρεο για το Bluetooth.

- **Android 1.6 (Donut)**

Στις 15 Σεπτεμβρίου 2009, δημοσιεύθηκε το Android 1.6- που ονομάστηκε Donut- με βάση τον Linux kernel 2.6.29 και περιελάμβανε πολλά νέα χαρακτηριστικά:

- Η αναζήτηση με φωνής και κείμενο ενισχύθηκε με το ιστορικό του σελιδοδείκτη, τις επαφές και το διαδίκτυο.

«Εικόνα 6 : Το λογότυπο του Donut»

- Δυνατότητα για τους προγραμματιστές να συμπεριλάβουν το περιεχόμενό τους στα αποτελέσματα αναζήτησης.
- Ευκολότερη αναζήτηση και την ικανότητα να δει κανείς στιγμιότυπα εφαρμογών στο Android Market.
- Δυνατότητα στους χρήστες να επιλέξουν πολλές φωτογραφίες για διαγραφή.
- Υποστήριξη ανάλυση οθόνης WVGA.
- Βελτίωση στην ταχύτητα όσον αφορά εφαρμογές σχετικά με την αναζήτηση και τη φωτογραφική μηχανή.

- **Android 2.0/2.1 (Éclair)**

Στις 26 Οκτωβρίου 2009, δημοσιεύθηκε το Android 2.0 -με την κωδική ονομασία Éclair -με βάση τον Linux kernel 2.6.29. Ενσωματώθηκαν οι παρακάτω αλλαγές:

- Διευρυμένη δυνατότητα συγχρονισμού λογαριασμών, επιτρέποντας στους χρήστες να προσθέσουν πολλούς λογαριασμούς σε μια συσκευή για το email και το συγχρονισμό επαφών.

«Εικόνα 7 : Το λογότυπο του Éclair»

- Υποστήριξη Bluetooth 2.1
- Δυνατότητα να αξιοποιήσει ο χρήστης μια φωτογραφία από τις επαφές και να επιλέξει να καλέσει αυτήν την επαφή, να στείλει SMS ή email.
- Βελτιωμένη ταχύτητα δακτυλογράφησης σε εικονικό πληκτρολόγιο, με εξυπνότερο λεξικό που μαθαίνει από την χρήση των λέξεων και περιλαμβάνει τα ονόματα των επαφών καθώς και διάφορες προτάσεις.
- Ανανεωμένοι UI του browser με μικρογραφίες σελιδοδείκτη και υποστήριξη για HTML5.
- Βελτιστοποίηση της ταχύτητας του hardware και ανανεωμένο UI.
- Βελτιωμένη έκδοση Google Maps 3.1.2

- **Android 2.2.x (Froyo)**

Στις 20 Μαΐου 2010, δημοσιεύθηκε το Android 2.2 με βάση τον Linux kernel 2.6.32. Περιείχε τις εξής αλλαγές:

- Βελτιστοποιήσεις στην ταχύτητα, στην μνήμη και στην επίδοση.
- Λειτουργικότητα πρόσδεσης (tethering) USB και Wi-Fi hotspot.

«Εικόνα 8 : Το λογότυπο του Froyo»

- Προστέθηκε μια επιλογή για απενεργοποίηση της πρόσβασης σε δεδομένα μέσω δικτύου κινητής τηλεφωνίας.

- Φωνητική κλήση και ανταλλαγή επικοινωνίας μέσω Bluetooth.
- Υποστήριξη για αριθμητικούς και αλφαριθμητικούς κώδικες πρόσβασης.
- Υποστήριξη για την εγκατάσταση εφαρμογών στην επεκτάσιμη μνήμη.
- Υποστήριξη Adobe Flash.
- Βελτιωμένη εκκίνηση εφαρμογών με συντομεύσεις στις εφαρμογές του τηλεφώνου και στις εφαρμογές περιήγησης.

- **Android 2.3.x (Gingerbread)**

Στις 6 Δεκεμβρίου 2010, δημοσιεύθηκε ο kernel 2.6.35. Οι αλλαγές που περιλαμβάνονται είναι οι ακόλουθες:

- Ενημέρωση στον σχεδιασμό διεπαφής του χρήστη (user interface) κάνοντας το πιο απλό και πιο γρήγορο.
- Υποστήριξη για εξαιρετικά μεγάλα μεγέθη οθόνης και μεγάλες αναλύσεις.
- Εγγενής υποστήριξη για VoIP τηλεφωνία.

«Εικόνα 9 : Το λογότυπο του Gingerbread»

- Ενισχυμένη λειτουργικότητα για αντιγραφή / επικόλληση, επιτρέποντας στους χρήστες να επιλέξουν μια λέξη πατώντας και διαλέγοντας αντιγραφή και επικόλληση.
- Νέος διαχειριστής λήψεων, δίνοντας στους χρήστες εύκολη πρόσβαση σε κάθε αρχείο που κατεβάζει ο χρήστης από τον browser, e-mail, ή άλλη εφαρμογή.
- Ενισχυμένη υποστήριξη για την ανάπτυξη εγγενή κώδικα.
- Βελτιώσεις σε ήχο, γραφικά για τους προγραμματιστές παιχνιδιών.
- Υποστήριξη συνομιλίας με φωνή ή βίντεο χρησιμοποιώντας το Google Talk.
- Βελτιωμένη απόδοση μπαταρίας.

- **Android 3.x (Honeycomb)**

Στις 22 Φεβρουαρίου 2011, δημοσιεύθηκε το Android 3.0 (Honeycomb) -η πρώτη έκδοση Android για tablets- με βάση τον πυρήνα του Linux 2.6.36. Η πρώτη συσκευή που διαθέτει αυτή την έκδοση, το Motorola Xoom, κυκλοφόρησε στις 24 Φεβρουαρίου 2011. Οι αλλαγές που περιλαμβάνονται είναι:

«Εικόνα 10 : Το λογότυπο του Honeycomb»

- Βελτιστοποιημένη υποστήριξη για tablet με ένα νέο εικονικό και «ολογραμματικό» user interface.
- Προστέθηκε μια μπάρα συστήματος, που διαθέτει γρήγορη πρόσβαση στις ειδοποιήσει, η εμφάνιση της κατάστασης της συσκευής, και τα μαλακά πλήκτρα πλοήγησης, διαθέσιμα στο κάτω μέρος της οθόνης.
- Απλοποιήθηκε το multitasking, με την μπάρα συστήματος ο χρήστης μπορεί να δει όλες τις εφαρμογές που τρέχουν και να μεταβεί από την μια στην άλλη.
- Επιτάχυνση του hardware.
- Υποστήριξη για multi-core επεξεργαστές.
- Δυνατότητα για την κρυπτογράφηση όλων των δεδομένων του χρήστη.
- Συνδεσιμότητα για αξεσουάρ USB.
- Υποστήριξη για χειριστήρια παιχνιδιών.
- Υψηλή απόδοση για την διατήρηση της σύνδεσης μέσω Wi-Fi όταν η οθόνη της συσκευής είναι απενεργοποιημένη.
- Υποστήριξη HTTP proxy για κάθε συνδεδεμένο σημείο Wi-Fi.
- Βελτιωμένη υποστήριξη Adobe Flash στον browser.

- **Android 4.0.x (Ice Sandwich)**

Το Android 4.0 βασίζεται στον Linux kernel 3.0.1 και δημοσιοποιήθηκε στις 19 Οκτωβρίου του 2011. Ο υπάλληλος της Google Gabe Cohen δήλωσε ότι το Android 4.0 είναι θεωρητικά συμβατό με οποιαδήποτε 2.3.χ Android συσκευή που βρίσκεται στην παραγωγή εκείνη την εποχή. Ο πηγαίος κώδικας για το Android

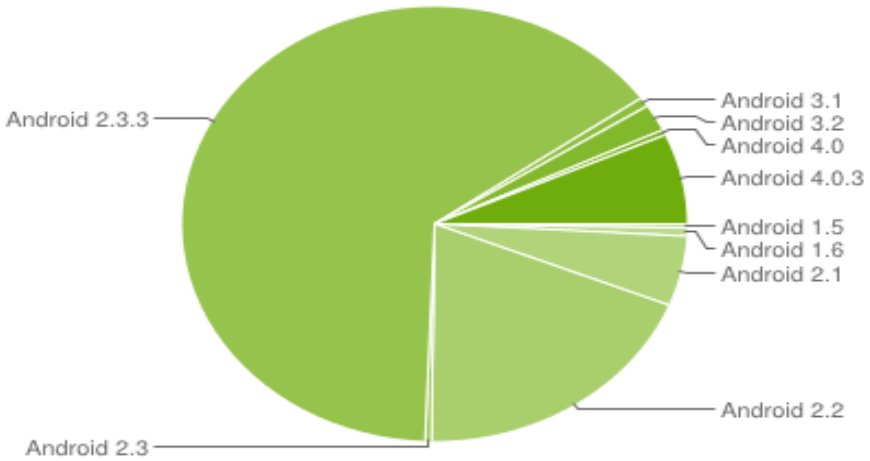
«Εικόνα 11 : Το λογότυπο του Ice Sandwich»

4.0 έγινε διαθέσιμος στις 14 Νοεμβρίου 2011. Αυτή η ενημέρωση παρουσιάζει χαρακτηριστικά όπως:

- Εικονικά κουμπιά στο περιβάλλον εργασίας, αντί για φυσικά

κουμπιά.

- Βελτιωμένη διόρθωση λάθους στο πληκτρολόγιο.
- Δυνατότητα πρόσβασης σε εφαρμογές απευθείας από την οθόνη κλειδώματος.
- Βελτιωμένη λειτουργικότητα αντιγραφής και επικόλλησης.
- Νέος web browser, που επιτρέπει έως και 16 καρτέλες.
- Ενότητα Χρήσης δεδομένων στις ρυθμίσεις που επιτρέπει στους χρήστες να ορίσουν προειδοποιήσεις όταν φτάσουν ένα συγκεκριμένο όριο χρήσης, και να απενεργοποιήσουν τη χρησιμοποίηση των δεδομένων, όταν υπερβούν αυτό το όριο.
- Δυνατότητα να κλείσει ο χρήστης εφαρμογές που χρησιμοποιούν δεδομένα στο παρασκήνιο.
- Wi-Fi Direct.



«Εικόνα 12 : Αριθμός συσκευών που μπήκαν στο Android Market από 16 Ιουνίου – 1 Ιουλίου»

2.3. Προγραμματιστικά εργαλεία

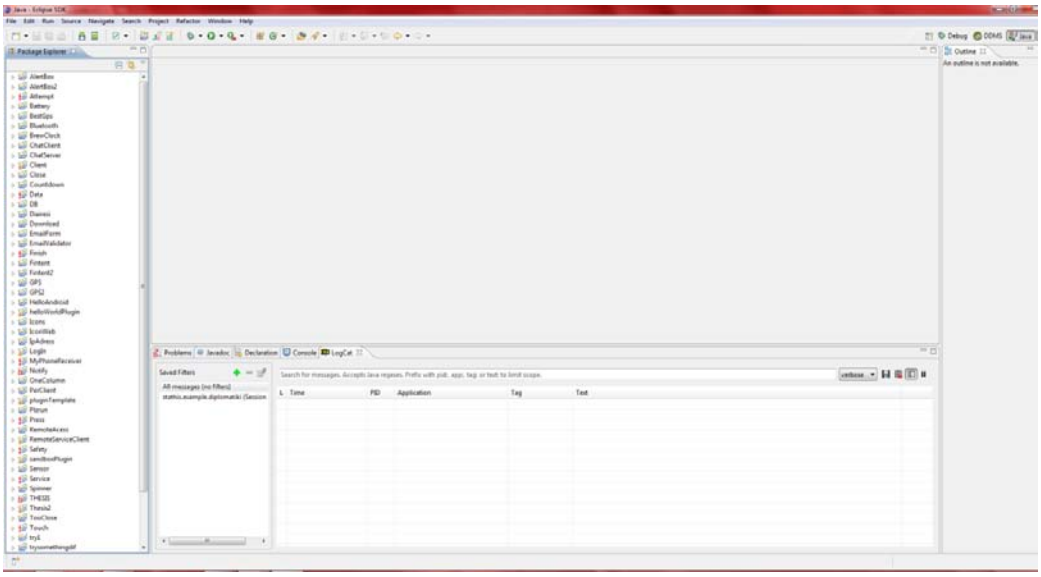
Η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται για την ανάπτυξη εφαρμογών Android είναι η Java, ενώ τους τελευταίους μήνες γίνονται προσπάθειες για να συμπεριληφθούν και άλλες γλώσσες όπως η C και η C++. Οπότε βασική προϋπόθεση είναι να διαθέτουμε τα αντίστοιχα εργαλεία της γλώσσας προγραμματισμού και συγκεκριμένα το Java Development Kit (JDK). Ακόμη χρειαζόμαστε ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment ή IDE) για να μεταγλωττίσουμε και να τρέξουμε τα προγράμματα μας, όπως είναι για παράδειγμα το Eclipse (Εικόνα 13).

Το βασικότερο εργαλείο αποτελεί το Android SDK (software development kit) το οποίο ουσιαστικά μας παρέχει τα επιπλέον εργαλεία ώστε να μπορούμε να γράψουμε κώδικα για την κατασκευή μιας εφαρμογής Android. Η σύνδεση του Android SDK με το γραφικό μας περιβάλλον (Eclipse) γίνεται μέσω μιας επέκτασης (Android Development Tools ή ADT Plug-in) που κάνουμε εγκατάσταση στο Eclipse, ώστε να μπορέσουμε να μεταγλωττίσουμε την εφαρμογή μας και έπειτα να την τρέξουμε.

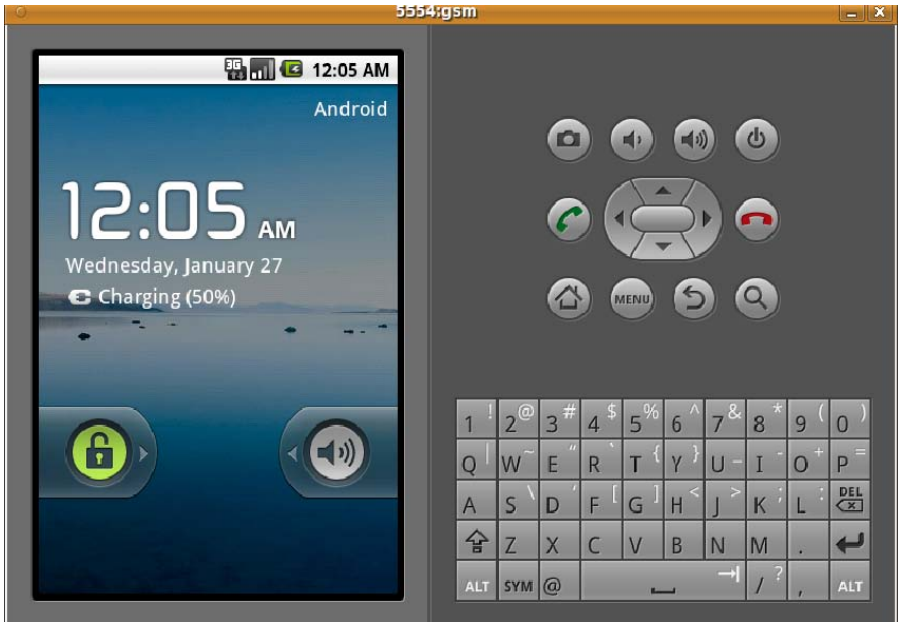
Καθώς το λειτουργικό σύστημα Android κυκλοφορεί σε διάφορες εκδόσεις, όπως αναφέραμε παραπάνω, καθίσταται σαφές ότι η κάθε έκδοση θα χρησιμοποιεί και διαφορετικά προγραμματιστικά εργαλεία. Έτσι μέσα από το ADT Plug-in μπορούμε να εγκαταστήσουμε τα εργαλεία καθώς και την τεκμηρίωση (documentation) αλλά και διάφορα παραδείγματα για την υλοποίηση της εφαρμογής σε οποιαδήποτε έκδοση. Για παράδειγμα, αν θέλουμε η εφαρμογή μας να είναι συμβατή και σε παλαιότερες εκδόσεις του Android τότε θα πρέπει , μαζί με άλλες αλλαγές που θα παρουσιάσουμε στην συνέχεια της διπλωματικής (βλ. Android Manifest), να εγκαταστήσουμε και τα αντίστοιχα εργαλεία και να δοκιμάσουμε την εφαρμογή μας σε αυτές.

Τέλος, για να δοκιμάσουμε την εφαρμογή μας και να δούμε τα αποτελέσματα της θα χρειαστούμε κάποιον προσομοιωτή κινητού τηλεφώνου, αν δεν διαθέτουμε εμείς οι ίδιοι, στον υπολογιστή μας. Τη λύση μας δίνει μια εικονική συσκευή Android (Android Virtual Device ή AVD) η οποία ουσιαστικά αποτελεί προσομοιωτή τόσο software όσο και hardware ενός κινητού τηλεφώνου με λειτουργικό σύστημα Android (Εικόνα 14). Την συσκευή αυτή την εγκαθιστάμε μέσα από το ADT Plug-in, από όπου μπορούμε να ρυθμίσουμε πολλές παραμέτρους της, όπως τι έκδοση Android θα χρησιμοποιεί, το μέγεθος της οθόνης, το μέγεθος της εξωτερικής κάρτας αποθήκευσης και της cache, καθώς και άλλα χαρακτηριστικά που έχουν να κάνουν με την τοποθεσία (GPS), την δυνατότητα λήψης φωτογραφιών, κ.α. Απαιτούνται αρκετοί υπολογιστικοί πόροι και μεγάλη έκταση μνήμης RAM για την εκτέλεση της AVD, πράγμα που καθιστά τις περισσότερες φορές αργή την εκτέλεση της εφαρμογής μας στη συσκευή αυτή. Φυσικά, κάθε φορά μπορούμε να εγκαθιστούμε και να ελέγχουμε τις εφαρμογές μας σε φυσική συσκευή, όπως ένα κινητό τηλέφωνο ή ένα tablet που χρησιμοποιούν λειτουργικό σύστημα Android, συνδέοντας το απλά σε μια θύρα USB του υπολογιστή μας.

Στο σημείο αυτό, πρέπει να σημειώσουμε ότι όλα τα παραπάνω προγραμματιστικά εργαλεία διατίθενται ελεύθερα στο διαδίκτυο από τους κατασκευαστές τους τα οποία μπορούμε να κατεβάσουμε και να χρησιμοποιήσουμε χωρίς κανένα κόστος.



«Εικόνα 13 : Το περιβάλλον ανάπτυξης λογισμικού Eclipse»



«Εικόνα 14 : Εικονική συσκευή με λειτουργικό σύστημα Android (AVD)»

2.4. Συστατικά της εφαρμογής

Με τον όρο αυτό εννοούμε τα απαραίτητα δομικά στοιχεία μιας εφαρμογής Android. Το κάθε στοιχείο στην ουσία είναι ένας τρόπος πρόσβασης του λειτουργικού συστήματος στην εφαρμογή μας.

Μπορούμε να τα διακρίνουμε σε τέσσερα βασικά στοιχεία : τις δραστηριότητες (activities), τις υπηρεσίες (services), τους παρόχους περιεχομένου (content providers) και τους καθολικούς παραλήπτες μηνυμάτων (broadcast receivers).

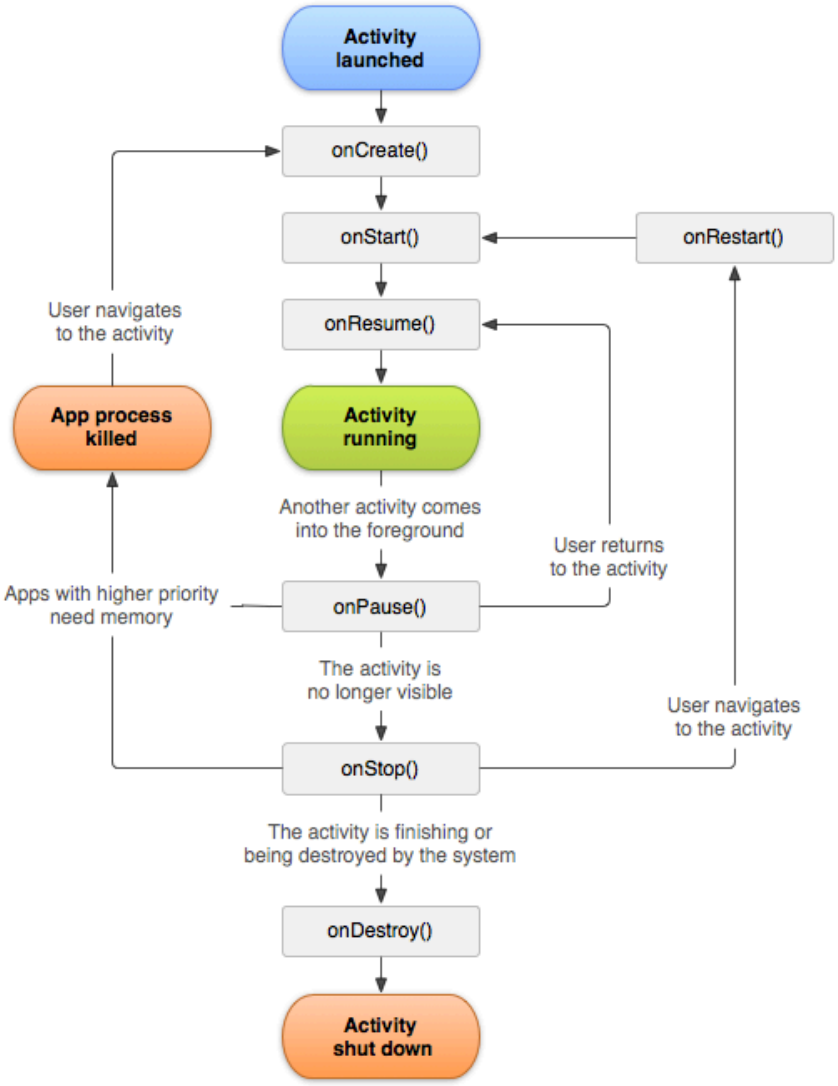
2.4.1. Δραστηριότητες

Αποτελούν το βασικότερο στοιχείο οποιασδήποτε εφαρμογής. Η κάθε δραστηριότητα (activity) αποτελεί μια διαφορετική οθόνη (παράθυρο) της εφαρμογής, με την οποία ο χρήστης αλληλεπιδρά. Σε αυτήν δηλαδή φορτώνεται το γραφικό περιβάλλον το οποίο βλέπει τελικά το χρήστης.

Η διεπαφή χρήστη (user interface) της εφαρμογής εμφανίζεται σε μια συσκευή μέσω μιας activity, συνήθως με μια activity που δημιουργήθηκε για κάθε μοναδική οθόνη. Αύτη την πρώτη οθόνη που βλέπει ο χρήστης την ονομάζουμε αρχική (main) activity. Εσωτερικά υπάρχει μια στοίβα των δραστηριοτήτων
, κατά τη
μετακίνηση από τη μία οθόνη στην άλλη. Συγκεκριμένα, η επόμενη activity που είναι να εμφανιστεί στέλνεται στην κορυφή της στοίβας -με άλλα λόγια, η activity στην κορυφή της στοίβας είναι αυτή που είναι ορατή στην οθόνη. Οι activities βγαίνουν από τη στοίβα πατώντας το πλήκτρο επιστροφής, ενέργεια η οποία συνεχίζει την προηγούμενη activity. Δηλαδή έχουμε μια μορφή στοίβας LIFO (Last In First Out). Αν πατήσουμε το πλήκτρο επιστροφής στην main activity τότε βγαίνουμε από την εφαρμογή μας.

Κάθε activity πρέπει να υλοποιεί την μέθοδο onCreate(), η οποία καλείται την στιγμή που ξεκινά η activity. Η μέθοδος αυτή καλείται μόνο όταν τρέξουμε μια νέα activity είτε από ένα κουμπί είτε από κάποια επιλογή, δηλαδή δεν καλείται αν επιστρέψουμε σε αυτή με το πλήκτρο επιστροφής. Αυτό συμβαίνει γιατί αν συμβεί το παραπάνω σενάριο θα δούμε ένα στιγμιότυπο της activity και αυτή δεν θα δημιουργηθεί ξανά. Επίσης, μέσα σε αυτήν την μέθοδο πρέπει να δημιουργηθεί και το user interface το οποίο δηλώνεται με την εντολή setContentView(). (Το UI θα το εξετάσουμε εκτενέστερα παρακάτω).

Για να ξεκινήσουμε μια activity μέσα από μια άλλη χρησιμοποιούμε την εντολή Intent intent = new Intent (this, ActivityB.class); startActivity(intent);
Για να τερματίσουμε μια activity, αρκεί η εντολή finish(). Σε γενικές γραμμές, το Dalvik διαχειρίζεται τον κύκλο ζωής της κάθε activity οπότε δε χρειάζεται να τις τερματίσουμε εμείς, παρά μόνο όταν θέλουμε οπωσδήποτε να μην μπορεί ο χρήστης να επιστρέψει στο δικό της στιγμιότυπο. Ο κύκλος ζωής καθώς και οι αντίστοιχες μέθοδοι φαίνονται αναλυτικά στην παρακάτω εικόνα.



«Εικόνα 15 : Ο κύκλος ζωής των activities και οι αντίστοιχες μέθοδοι»

2.4.2. Υπηρεσίες

Μια υπηρεσία (service) αποτελεί ένα στοιχείο της εφαρμογής, το οποίο τρέχει στο παρασκήνιο (background) και μπορεί να εκτελέσει αρκετές και χρονοβόρες διαδικασίες. Ειδοποιός διάφορα με την activity είναι ότι η service δεν έχει user interface. Επίσης, μια service μπορεί να τρέχει ακόμα και όταν τρέξουμε μια διαφορετική εφαρμογή από αυτήν που την ξεκίνησε.
Παράδειγμα μιας service είναι η εφαρμογή για την αναπαραγωγή μουσικής. Όταν ακούμε μουσική, είναι πιθανόν να θέλουμε να συνεχίσουμε να την ακούμε ακόμα και όταν τρέξουμε κάποια άλλη εφαρμογή, για παράδειγμα εάν λάβουμε κάποιο γραπτό μήνυμα στο κινητό μας και δίχως να απαιτούμε γραφικό περιβάλλον. Επιπλέον, όλη η επικοινωνία με το διαδίκτυο θέλουμε να συνεχιστεί όταν ανοίξουμε μια άλλη εφαρμογή και να μην σταματήσει, οπότε θα πρέπει και αυτό να λειτουργεί σαν service.

Δύο βασικές λειτουργίες σχετίζονται με τις υπηρεσίες: startService() και η bindService(). Η πρώτη καλείται από κάποια activity για να ξεκινήσει η service και θα συνεχίσει να εκτελείται ακόμα και όταν η activity που την κάλεσε τερματιστεί. Για το λόγο αυτό η service δεν επιστρέφει κάποιο αποτέλεσμα στην activity που την κάλεσε, αλλά εκτελεί κάποια ξεχωριστή διαδικασία, όπου για παράδειγμα το κατέβασμα κάποιου αρχείου από το διαδίκτυο. Η δεύτερη λειτουργία «δένει» την service με κάποιο στοιχείο της εφαρμογής, μια activity, έτσι ώστε να υπάρχει αλληλεπίδραση μεταξύ αυτών και αποστολή ή λήψη αποτελεσμάτων. Στην περίπτωση αυτή, αν τερματιστεί το στοιχείο που έχει δεθεί με τη service, τότε τερματίζεται και αυτή.

Η διάρκεια και ο κύκλος ζωής των services είναι λίγο διαφορετικός από αυτόν των activities για τον λόγο ότι αυτές θα συνεχίσουν να τρέχουν μέχρι να τις ειδοποιήσει ο χρήστης να σταματήσουν με την εντολή stopSelf() ή να τις σταματήσει το ίδιο το λειτουργικό στην προσπάθεια του να εξοικονομήσει RAM. Το να τρέχει κάποια service για μεγάλο χρονικό διάστημα κοστίζει, έτσι θα πρέπει ο προγραμματιστής να μην χρησιμοποιεί πολύ υπολογιστική ισχύ γιατί έτσι η μπαταρία θα εξαντληθεί σε χρόνο μηδέν.

Προηγουμένως αναφέραμε ότι η service δεν παρέχει user interface, επομένως γεννιέται το εύλογο ερώτημα, πως θα ενημερώνεται ο χρήστης για κάποιο αποτέλεσμα της υπηρεσίας. Υπάρχουν δύο τρόποι : η ειδοποίηση Toast και η ειδοποίηση μέσω της Status Bar. Αυτές θα εξεταστούν στο κεφάλαιο 2.5.3, καθώς αφορούν το user interface.

2.4.3. Πάροχοι Περιεχομένου

Οι πάροχοι περιεχομένου (content provides) διαχειρίζονται αποθηκευτικούς χώρους για δεδομένα, οι οποίοι είναι προσπελάσιμοι από οποιαδήποτε εφαρμογή. Αποτελούν τον μοναδικό τρόπο για αποθήκευση δεδομένων, στα οποία θέλουμε πρόσβαση και από άλλες εφαρμογές. Κλασικό παράδειγμα είναι η εφαρμογή για την αναπαραγωγή μουσικής η οποία αποθηκεύει τα αρχεία μας σε συγκεκριμένο χώρο ώστε αυτά να είναι προσβάσιμα από οποιαδήποτε άλλη εφαρμογή.

Σε αυτό το πλαίσιο υπάρχει και η ενσωματωμένη βάση δεδομένων στο λειτουργικό σύστημα Android, η SQLite Database, στην οποία μπορεί να αποθηκεύσουν και να διαβάσουν δεδομένα οι content providers.

2.4.4. Broadcast Receivers

Οι broadcast receivers είναι interfaces που ενημερώνονται για ένα συγκεκριμένο γεγονός από το λειτουργικό σύστημα και τότε ενεργοποιούνται. Τέτοια γεγονότα μπορεί να είναι η κατάσταση της μπαταρίας, η λήψη ενός μηνύματος κλπ. Επίσης και οι εφαρμογές μπορεί να ειδοποιήσουν ένα broadcast receiver, για παράδειγμα να ειδοποιήσει άλλες εφαρμογές ότι κάποια δεδομένα κατέβηκαν από το Internet και είναι διαθέσιμα προς χρήση από αυτές.

Οι broadcast receivers δεν έχουν user interface. Ωστόσο, μπορούν να ξεκινήσουν μια activity σε ανταπόκριση για τις πληροφορίες που έλαβαν, ή μπορούν να ενημερώσουν τον χρήστη μέσω των ειδοποιήσεων status bar.

2.5. Διεπαφή Χρήστη

Η διεπαφή χρήστη (user interface) έχει τεράστια σημασία για κάθε εφαρμογή. Αποτελεί την τελική εικόνα που βλέπει ο χρήστης, το γραφικό περιβάλλον στο οποίο θα περιηγηθεί και ενεργοποιεί όλες τις λειτουργίες της εφαρμογής. Το user interface και η λειτουργικότητα της εφαρμογής αποτελούν αλληλένδετα στοιχεία και χωρίς το ένα δεν μπορεί να υπάρξει το άλλο. Πολλές φορές είναι πιο δύσκολο για τον προγραμματιστή να κατασκευάσει ένα εύχρηστο και όμορφο περιβάλλον εργασίας παρά η ίδια η εφαρμογή. Καθίσταται σαφές λοιπόν, ότι απαιτεί μεγάλη προσπάθεια και προσοχή η δημιουργία ενός γραφικού περιβάλλοντος που θα προσελκύει τους χρήστες και θα τους ωθεί να χρησιμοποιούν μια συγκεκριμένη εφαρμογή έναντι μιας άλλης με τις ίδιες λειτουργίες.

2.5.1. Χωροταξία των αντικειμένων

Σε κάθε οθόνη της εφαρμογής πρωταρχικό στοιχείο του γραφικού περιβάλλοντος αποτελεί η διάταξη των γραφικών στοιχείων (layout). Το layout περιλαμβάνει όλα τα γραφικά στοιχεία της οθόνης τα οποία μπορεί να είναι διατεταγμένα σε επιμέρους layouts.

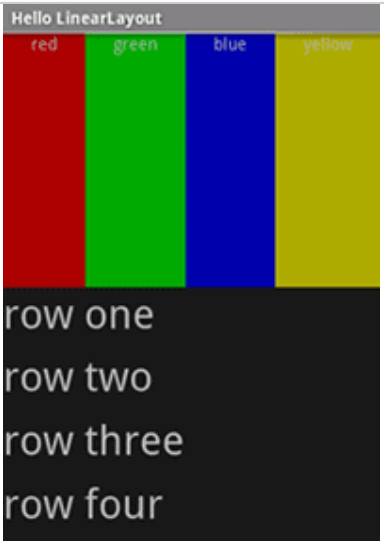
Υπάρχουν τέσσερα είδη διάταξης η γραμμική διάταξη (Linear Layout), η σχετική διάταξη (Relative Layout), η διάταξη πλαισίου (Frame Layout) και η διάταξη πίνακα (Table Layout). Υπάρχει όμως και ένα άλλο είδος διάταξης, το Absolute Layout, το οποίο έχει αποφασιστεί να μην χρησιμοποιείται πλέον γιατί ορίζει την διάταξη των αντικείμενων με βάση την απόλυτη θέσης τους σε σχέση με την οθόνη, η οποία όμως διαφέρει ανάλογα με την συσκευή που έχει ο χρήστης. Το Linear Layout αποτελεί την διάταξη στοιχείων σε οριζόντια ή κατακόρυφη σειρά. Το Relative Layout δίνει στον προγραμματιστή μια μεγαλύτερη ευχέρεια και ελευθερία με την έννοια ότι μπορούμε να εμφανίσουμε οποιοδήποτε αντικείμενο σε όποια θέση θέλουμε, αρχή, μέση και τέλος της οθόνης ακόμα και να ορίσουμε την θέση σε σχέση με κάποιο άλλο αντικείμενο. Το Frame Layout αποτελεί την πιο απλή διάταξη στοιχείων, καθώς δημιουργεί ένα κενό χώρο όπου μπορεί ο προγραμματιστής να βάλει ένα αντικείμενο, για παράδειγμα μια εικόνα, και αυτό το στοιχείο θα έχει τον ρόλο της ρίζας (root) όπου όλα τα υπόλοιπα θα διατάσσονται σύμφωνα με αυτό. Τέλος, το Table Layout, είναι η διάταξη πίνακα, μπορεί δηλαδή να διατάσσει τα παιδιά του (children) σε σειρές και στήλες. Όλα τα παραπάνω μπορούν να πάρουν και διάφορες παραμέτρους όπως το μέγεθος, οριζόντια η κατακόρυφη διάταξη, βαρύτητα (gravity) και άλλα.

Ακολουθούν δύο παραδείγματα διάταξης των στοιχείων της οθόνης :

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

     <LinearLayout
```

```
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1">
        <TextView
            android:text="red"
            android:gravity="center_horizontal"
            android:background="#aa0000"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <TextView
            android:text="green"
            android:gravity="center_horizontal"
            android:background="#00aa00"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <TextView
            android:text="blue"
            android:gravity="center_horizontal"
            android:background="#0000aa"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <TextView
            android:text="yellow"
            android:gravity="center_horizontal"
            android:background="#aaaa00"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
    </LinearLayout>

    <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1">
      <TextView
          android:text="row one"
          android:textSize="15pt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
      <TextView
          android:text="row two"
          android:textSize="15pt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
      <TextView
          android:text="row three"
          android:textSize="15pt"
          android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_weight="1"/>
        <TextView
            android:text="row four"
            android:textSize="15pt"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"/>
      </LinearLayout>

    </LinearLayout>
```



«Εικόνα 16 : Παράδειγμα κατασκευής ενός LinearLayout με οριζόντια και κατακόρυφη σειρά διάταξης»

Σε αυτό το παράδειγμα έχουμε δημιουργήσει δύο διαφορετικές προσεγγίσεις του Linear Layout, την οριζόντια και την κατακόρυφη στοίχιση. Έτσι βλέπουμε τις τέσσερις χρωματισμένες στήλες που βρίσκονται σε οριζόντια στοίχιση, είναι δηλαδή η μία δίπλα στην άλλη. Στην ουσία έχουμε τρία Linear Layout (Εικόνα 16). Έχουμε το ενιαίο το οποίο περιλαμβάνει τα άλλα δύο.

Για να γίνει πιο κατανοητό θα πρέπει ο αναγνώστης να φανταστεί τις χρωματισμένες στήλες και τις αριθμημένες γραμμές σαν δύο διαφορετικά γραφικά στοιχεία. Θα παρατηρήσει ότι βρίσκονται σε κατακόρυφη σειρά. Έτσι έχουμε την γραμμή κώδικα android:orientation=”vertical”.

Αν εξετάσει τώρα το κάθε γραφικό στοιχείο ξεχωριστά εύκολα παρατηρεί την διάταξη με την οποία είναι τοποθετημένα τα στοιχεία του καθενός, το πρώτο βρίσκεται σε οριζόντια διάταξη και το δεύτερο σε κάθετη.

Επιπλέον, παρατηρεί και τα ονόματα που υπάρχουν στα στοιχεία, red, green, row one, row two κλπ, καθώς και το χρώμα που χρησιμοποιείται σε κάθε περίπτωση με την δεκαεξαδική του μορφή, για παράδειγμα το #aa0000 είναι το κόκκινο.

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

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">

    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Open..."
            android:padding="3dip" />
        <TextView
            android:text="Ctrl-O"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Save..."
            android:padding="3dip" />
        <TextView
            android:text="Ctrl-S"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Save As..."
            android:padding="3dip" />
        <TextView
            android:text="Ctrl-Shift-S"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <View
        android:layout_height="2dip"
        android:background="#FF909090" />

    <TableRow>
        <TextView
            android:text="X"
            android:padding="3dip" />
        <TextView
            android:text="Import..."
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="X"
            android:padding="3dip" />
        <TextView
```

```
            android:text="Export..."
            android:padding="3dip" />
        <TextView
            android:text="Ctrl-E"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <View
        android:layout_height="2dip"
        android:background="#FF909090" />

    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Quit"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```

Σε αυτό το παράδειγμα (Εικόνα 17), έχουμε ένα Table Layout, που όπως υποδεικνύει και το όνομα του είναι μια διάταξη που ταξινομεί τα επιμέρους στοιχεία της βάση ενός πίνακα.

Η ιδιότητα gravity Που χρησιμοποιούμε απλά τοποθετεί το αντίστοιχο κείμενο στην θέση που ορίσαμε, στην συγκεκριμένη θα είναι πάντα τέρμα δεξιά.

Μπορούμε να ορίσουμε παραπάνω από μια στήλες, android:layout_column, ανάλογα με το τι στοιχεία θέλουμε να εμφανίσουμε.

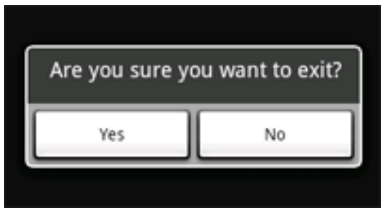
«Εικόνα 17 : Παράδειγμα κατασκευής ενός TableLayout με διάφορες επιλογές για κάθε παιδί»

2.5.2. Παράθυρα Διαλόγου

Ο διάλογος (dialog) είναι συνήθως ένα μικρό παράθυρο μου εμφανίζεται στην οθόνη μπροστά από την activity μέσα από την οποία ο χρήστης το κάλεσε με κάποια ενέργεια του. Η activity αυτή χάνει την εστίαση (focus) που είχε και το παράθυρο του διαλόγου είναι το μοναδικό με το οποίο μπορεί να αλληλεπιδράσει ο χρήστης. Χρησιμοποιείται είτε για ενημέρωση του χρήστη για κάποιο γεγονός είτε για να ορίσει ο χρήστης την επόμενη ενέργεια του. Τα κυριότερα είδη διαλόγων είναι ο διάλογος ειδοποίησης (Alert Dialog) και ο διάλογος προόδου (Progress Dialog).

Ο Alert Dialog (Εικόνα 18) είναι ο πιο συνηθισμένος διάλογος. Αποτελείται από ένα τίτλο, ένα μήνυμα και ορισμένα κουμπιά ή μια λίστα από επιλογές. Για κάθε κουμπί του διαλόγου ορίζουμε στην activity τις ενεργείς που θα ακολουθήσουν όταν το πατήσει ο χρήστης. Ακολουθεί ένα παράδειγμα κώδικα για την δημιουργία ενός Alert Dialog με δυο κουμπιά, καθώς και το αποτέλεσμα που έχει στο user interface.

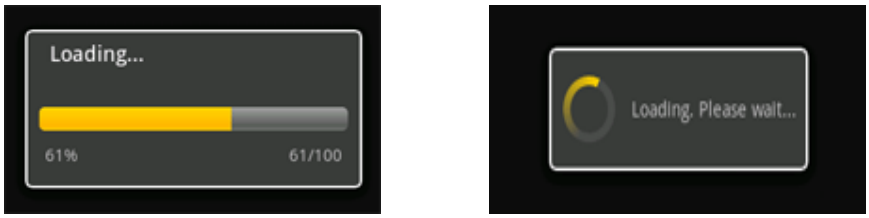
```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")
       .setCancelable(false)
       .setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                 MyActivity.this.finish();
            }
       })
       .setNegativeButton("No", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                 dialog.cancel();
            }
       });
AlertDialog alert = builder.create();
```



«Εικόνα 18 : Παράδειγμα ενός AlerDialog»

Ο Process Dialog (Εικόνα 19) αποτελεί στην ουσία μια επέκταση του Alert Dialog και χρησιμοποιείται όταν θέλουμε να εμφανίσουμε την πρόοδο μιας ενέργειας. Το κλασικότερο παράδειγμα είναι όταν θέλουμε να κατεβάσουμε μια εικόνα ή ένα οποιοδήποτε άλλο αρχείο από το διαδίκτυο και θέλουμε να εμφανίσουμε στο χρήστη το χρονικό διάστημα που θα κάνει να ολοκληρωθεί. Οπότε για να μην βλέπει ο χρήστης μια μαύρη οθόνη όπου δεν θα μπορεί να κάνει τίποτα και θα φαίνεται σαν η εφαρμογή μας να έχει κάποιο bug, εμφανίζουμε ένα Progress Dialog και στο παρασκήνιο εκτελούμε τις χρονοβόρες διαδικασίες. Ας δούμε το παρακάτω παράδειγμα:

```
ProgressDialog dialog = ProgressDialog.show(MyActivity.this, "",
                        "Loading. Please wait...", true);
```



«Εικόνα 19 : Παραδείγματα Progress Dialog»

Υπάρχουν δύο τρόποι να ακυρώσει κανείς ένα διάλογο. Τις περισσότερες φορές θέλουμε να δώσουμε στο χρήστη την δυνατότητα να ακυρώσει με τον πιο φυσικό τρόπο ένα διάλογο, πατώντας το πλήκτρο επιστροφής στην συσκευή του. Αυτό γίνεται εάν δώσουμε στον διάλογο την ιδιότητα να είναι ακυρώσιμος (cancellable). Ακόμη και στις ενέργειες που ακολουθούν όταν ο χρήστης πατήσει κάποιο κουμπί, θα πρέπει να συμπεριλάβουμε και την εντολή *myDialog.dismiss(),* ώστε αυτός να εξαφανιστεί και να συνεχιστεί η κανονική ροή της εφαρμογής.

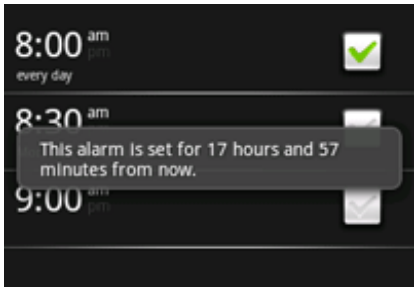
2.5.3. Ειδοποιώντας το χρήστη

Συνήθως θέλουμε να ειδοποιήσουμε τον χρήστη για κάποιο γεγονός ή αποτέλεσμα σχετικά με την εφαρμογή μας. Σε κάποια από αυτά ο χρήστης πρέπει να δώσε μια απάντηση σε άλλα όχι. Για παράδειγμα όταν ο χρήστης αποθηκεύει ένα αρχείο είναι σωστό να τον ενημερώσουμε το πότε αυτό έγινε. Ή όταν η εφαρμογή μας κατεβάζει μια εικόνα στο παρασκήνιο θα πρέπει να ειδοποιήσει τον χρήστη όταν κατέβηκε για το αν θέλει να ανοίξει την εικόνα ή όχι. Στο πρώτο παράδειγμα, όπου δεν είναι αναγκαία η παρέμβαση του χρήστη θα χρησιμοποιήσουμε Toast notification, ενώ στην δεύτερη status bar notification.

Πρώτα θα εξετάσουμε τα Toast notifications. Ένα toast notification είναι ένα μήνυμα που εμφανίζεται για λίγα δευτερόλεπτα στο παράθυρο που βρίσκεται ο χρήστης σε οποιαδήποτε εφαρμογή και αν βρίσκεται. Ο χώρος που καταλαμβάνει είναι ο ελάχιστος απαιτούμενος ώστε αυτό να είναι εμφανές, ενώ ο χρήστης είναι ελεύθερος να αλληλεπιδρά με την activity στην οποία βρίσκεται. Δεν υπάρχει κάποια αλληλεπίδραση του χρήστη με αυτό. Χρησιμοποιείται συνήθως για μικρά μηνύματα που απώτερου σκοπό έχουν μόνο την ενημέρωση του χρήστη και τίποτα άλλο, για παράδειγμα «Το αρχείο κατέβηκε επιτυχώς», «Το ξυπνητήρι ορίστηκε επιτυχώς στις...». Συγκεκριμένα:

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
```

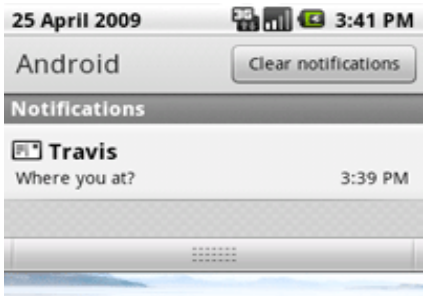
```
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```



«Εικόνα 20 : **Παράδειγμα toast notification**»

Στην συνέχεια έχουμε τα status bar notifications που όπως φανερώνει και το όνομα του είναι μια ειδοποίηση εμφανίζεται στη status bar της συσκευής μας και την οποία μπορούμε να ανοίξουμε είτε βρισκόμαστε στην αρχική οθόνη (home screen), είτε μέσα σε οποιαδήποτε εφαρμογή. Αντίθετα με το προηγούμενο, η status bar notification μπορεί να επιλεχθεί και να ξεκινήσει κάποια λειτουργία ανάλογα με τις ενεργείς που έχουμε ορίσει από την αρχή στον κώδικα μας. Για παράδειγμα όταν υπάρχουν ενημερώσεις για τις εφαρμογές από το Android Market, εμφανίζεται μια τέτοια notification που μας ανακατευθύνει στην σελίδα απ' όπου μπορούμε να συνεχίσουμε με την αναβάθμιση.

```
Notification notification = new Notification(icon, tickerText,
when);


Context context = getApplicationContext();
CharSequence contentTitle = "My notification";
CharSequence contentText = "Hello World!";
Intent notificationIntent = new Intent(this, MyClass.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
notificationIntent, 0);
notification.setLatestEventInfo(context, contentTitle, contentText,
contentIntent);
private static final int HELLO_ID = 1;


mNotificationManager.notify(HELLO_ID, notification);
```



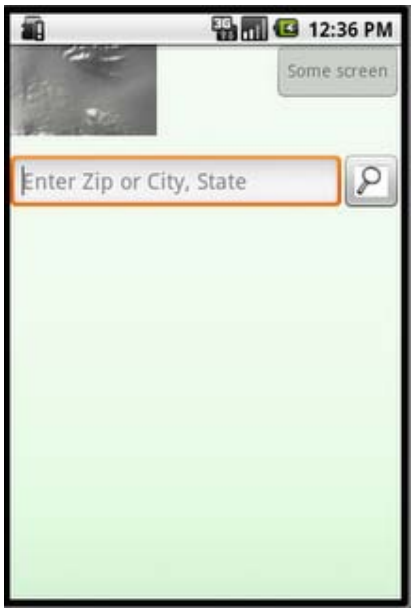
«Εικόνα 21 : Παράδειγμα status bar notification»

2.6. Application Resources και Συμβατότητα

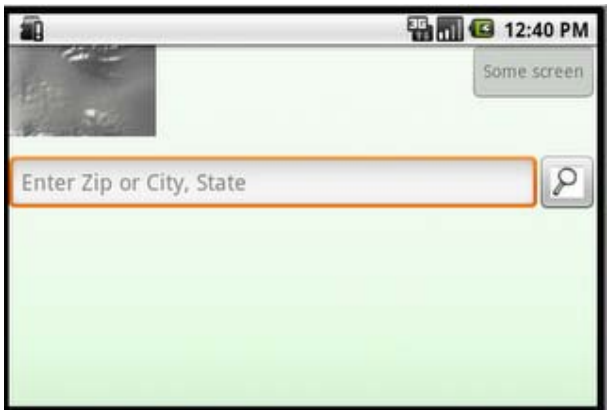
Με τον όρο application resources εννοούμε όλα τα στατιστικά στοιχεία τα οποία χρησιμοποιεί η εφαρμογή μας, όπως τα εικονίδια και τα string που χρησιμοποιεί ο προγραμματιστής στον κώδικα. Όλα αυτά τα στοιχεία θα πρέπει να είναι ανεξάρτητα από τον κώδικα της εφαρμογής, δηλαδή να δηλώνονται σε διαφορετικά σημεία από τις κλάσεις μας για λόγους συμβατότητάς της. Αν εμφανίζουμε μέσα στην εφαρμογή κάποιες εικόνες σε μια οθόνη θα θέλαμε ίσως να τις εμφανίσουμε με διαφορετικό τρόπο ανάλογα με το αν βρισκόμαστε σε portrait mode ή σε landscape mode. Επίσης, αν θέλαμε η εφαρμογή μας να υποστηρίζει αρκετές γλώσσες θα μπορούσαμε να κρατάμε σε διαφορετικό αρχείο τα strings για κάθε γλώσσα και να τα προσπελαύνουμε με τον ίδιο τρόπο από τον κώδικα μας. Έτσι λοιπόν, για κάθε στοιχείο που θα χρησιμοποιήσουμε μπορούμε να ορίσουμε το προεπιλεγμένο (default), να ορίσουμε ένα εναλλακτικό (alternative), τα οποία θα εξαρτώνται από τον προσανατολισμό, το μέγεθος της οθόνης, τη γλώσσα κλπ.

2.6.1. Προσανατολισμός (orientation)

Οι συσκευές με λειτουργικό σύστημα Android μπορούν να τοποθετηθούν είτε σε προσανατολισμό πορτρέτου (portrait mode) είτε σε προσανατολισμό τοπίου (landscape mode). Η πρώτη περίπτωση αφορά και την πιο συνηθισμένη που χρησιμοποιεί κανείς το κινητό του τηλέφωνο δηλαδή την όρθια θέση όπου η μεγαλύτερη πλευρά της οθόνης είναι η κατακόρυφη. Η δεύτερη περίπτωση είναι η ακριβώς αντίστροφη όταν δηλαδή η μεγαλύτερη πλευρά της οθόνης είναι η οριζόντια. Είναι εύκολα κατανοητό λοιπόν ότι τα γραφικά στοιχεία που έχουμε σε κάθε οθόνη θα πρέπει να εμφανίζονται με διαφορετικό τρόπο κάθε φορά. Αυτό συμβαίνει γιατί η οθόνη στο landscape mode αντιστρέφεται. Έχει περισσότερο πλάτος παρά μήκος. Αυτό σημαίνει ότι αν η αντιστροφή γινόταν με την προεπιλεγμένη συμπεριφορά, ο χρήστης μπορεί να δει πολλά κενά στη δεξιά πλευρά. Ο προεπιλεγμένος τρόπος είναι να παρουσιάζεται το user interface με την ίδια διάταξη και με τους δύο προσανατολισμούς. Για παράδειγμα :



«Εικόνα 22 : **Οθόνη σε portrait mode**»



«Εικόνα 23 : **Οθόνη σε landscape mode**»

Η διαφορά είναι εμφανής. Φαίνεται ότι στην «Εικόνα» το κείμενο «some text» είναι πολύ μακριά από την εικόνα της εφαρμογής καθώς και το στοιχείο EditText είναι πάρα πολύ μακρύ.

Προηγουμένως στο κεφάλαιο user interface αναφέραμε ότι η διάταξη των γραφικών μας στοιχείων δηλώνεται αρχικά σε αρχεία xml. Συγκεκριμένα στο project μας υπάρχει ο φάκελος res/layout στο οποίο τοποθετούμε όλα τα αρχεία xml που αφορούν την εφαρμογή μας. Το λειτουργικό σύστημα Android έχει την δυνατότητα να διαχειριστεί αυτό το φαινόμενο με το να καταστρέφει την εφαρμογή και να την δημιουργεί ξανά ψάχνοντας κάθε φορά για το κατάλληλο user interface που θα χρησιμοποιήσει με την εντολή *setContentView().* Έτσι ορίζοντας διαφορετικά αρχεία xml για την κάθε περίπτωση προσανατολισμού μπορούμε να έχουμε μια «όμορφη» εφαρμογή.

2.6.2. Μέγεθος Οθόνης

Παραπάνω μιλήσαμε για την περίπτωση αλλαγής προσανατολισμού της συσκευής. Με τον ίδιο τρόπο μπορούμε να χρησιμοποιήσουμε διαφορετικά γραφικά στοιχεία ανάλογα με το μέγεθος της οθόνης. Στο εμπόριο αυτή την στιγμή υπάρχουν πάρα πολλές συσκευές που χρησιμοποιούν διαφορετικού μεγέθους εικόνες. Αν για παράδειγμα θέλουμε να εμφανίσουμε διάφορες εικόνες στην οθόνη τότε αυτό που έχουμε να κάνουμε είναι να χρησιμοποιήσουμε διαφορετικής ανάλυσης εικόνες ανάλογα με την συσκευή που έχουμε (κινητό τηλέφωνο ή tablet).
Για την λύση αυτού του προβλήματος θα κάνουμε κάτι παρόμοιο με το πρόβλημα του προσανατολισμού. Όλες οι εικόνες αποθηκεύονται στον φάκελο res/drawable. Ανάλογα με την ανάλυση της εικόνας την αποθηκεύουμε και στον κατάλληλο φάκελο, όπως res/drawable/hdpi,

res/drawable/ldpi και res/drawable/mdpi. Το λειτουργικό σύστημα θα εντοπίζει αυτόματα το μέγεθος της οθόνης και θα επιλέγει τα κατάλληλα αρχεία.
Στις activities λοιπόν, θα χρησιμοποιούμε ενιαίες εντολές χωρίς να δηλώνουμε κάτι σχετικό με την οθόνη ή άλλη παράμετρο. Για παράδειγμα :

```
Button button = (Button) findViewById(R.id.button_id);

button.setImageRecource(R.drawable.myimage);
```

Αν έχουμε αποθηκεύσει μια εικόνα με το όνομα myimage.jpg σε διαφορετικούς φακέλους τότε το Android θα εντοπίσει το κατάλληλο αρχείο ανάλογα με το μέγεθος της οθόνης μας για να το εμφανίσει.

2.6.3. Γλώσσα

Ο προγραμματιστής πρέπει να έχει κατά νου ότι οι εφαρμογές Android απευθύνονται σε ένα ευρύ κοινό και όχι μόνο στην χώρα κατασκευής. Το σημαντικότερο ρόλο σε αυτό διαδραματίζει η φυσική γλώσσα που θα είναι γραμμένη η εφαρμογή. Αξίζει, λοιπόν να προσπαθήσουμε να μεταφράσουμε την εφαρμογή μας σε πολλές διαφορετικές γλώσσες, ειδικά μάλιστα όταν το Android μας προσφέρει τέτοια δυνατότητα με αρκετά εύκολο τρόπο.

Το κείμενο που εμφανίζεται στην οθόνη στην εφαρμογή είναι ουσιαστικά ένα σύνολο από strings οπουδήποτε και αν εμφανίζεται αυτό. Υπάρχουν λοιπόν, δύο τρόποι να εμφανίσουμε ένα string στην οθόνη. Ο πρώτος είναι ο συνηθισμένος τρόπος εμφάνισης, δηλαδή ορίζουμε στον κώδικά μας ένα string με την τιμή του, String hello = “Hello World”. Έπειτα για να το εμφανίσουμε αρκεί μόνο να καλέσουμε την μεταβλητή τύπου string hello. Όταν όμως έχουμε χιλιάδες γραμμές κώδικα είναι δύσκολο να εντοπίσουμε όλες αυτές τις μεταβλητές. Τη λύση έρχεται να μας δώσει το Android, η οποία αποτελεί τον δεύτερο και σωστό τρόπο εμφάνισης ενός στοιχείου String. Στο φάκελο res/values του project μας υπάρχει το αρχείο strings.xml στο οποίο μπορούμε να δηλώσουμε όλα τα string που χρησιμοποιεί η εφαρμογή μας και να αναφερόμαστε σε αυτά μέσα από τις κλάσεις μας με τον εξής τρόπο: *<string name = “hello”> Hello World </string>* και αναφερόμαστε σε αυτά μέσα στις κλάσεις με την εντολή getString(R.string.hello).

Έστω λοιπόν, ότι θέλουμε η προεπιλεγμένη γλώσσα να είναι η αγγλική, άρα στο αρχείο res/values/strings.xml δηλώνουμε τα string με το όνομα και την τιμή τους να είναι στην αγγλική γλώσσα. Αν θέλουμε η εφαρμογή μας να είναι συμβατή και με την ελληνική γλώσσα τότε δημιουργούμε τον φάκελο res/values-el και μέσα σε αυτόν το αρχείο strings.xml. Διατηρούμε τα ονόματα των string ακριβώς τα ίδια και αλλάζουμε τις τιμές τους. Όταν ο χρήστης εκτελέσει την εφαρμογή, το λειτουργικό σύστημα θα εντοπίσει αυτόματα τη γλώσσα που χρησιμοποιεί στην συσκευή του και ανάλογα θα επιλέξει και την γλώσσα της εφαρμογής. Αν δεν βρει φάκελος με την γλώσσα του χρήστη τότε θα χρησιμοποιηθούν οι προεπιλεγμένες τιμές.

«Εικόνα 24 : Η εφαρμογή όπως φαίνεται σε συσκευή που χρησιμοποιεί οποιαδήποτε γλώσσα εκτός της ελληνικής»

«Εικόνα 25 : Η εφαρμογή όπως φαίνεται σε συσκευή που χρησιμοποιεί ελληνική γλώσσα»

Συμπερασματικά, αυτό που προκύπτει για τον σωστό προγραμματισμό της εφαρμογής μας είναι πάντα να εξωτερικεύουμε τα application resources από το κομμάτι του κώδικα των κλάσεων. Υπάρχουν κατάλληλοι φάκελοι όπου μπορούμε να αποθηκεύσουμε ή να δηλώσουμε τα παραπάνω στοιχεία και πρέπει να χρησιμοποιούνται. Με αυτόν τον τρόπο, η εφαρμογή μας γίνεται κατανοητή στον προγραμματιστή και εύκολα επεξεργάσιμη, ενώ παράλληλα γίνεται συμβατή με όλες τις παραμέτρους της εκάστοτε συσκευής του χρήστη.

2.7. Android Manifest

Όταν ένας προγραμματιστής ξεκινήσει ένα καινούριο project δημιουργείται αυτόματα ένα αρχείο που ονομάζεται AndroidManifest.xml. Αυτό περιέχει βασικές πληροφορίες σχετικά με την εφαρμογή, τις οποίες το λειτουργικό σύστημα πρέπει να γνωρίζει προτού τρέξει οποιοδήποτε άλλο κομμάτι κώδικα. Οι σημαντικότερες από αυτές τις πληροφορίες περιγράφονται παρακάτω :

- Η ονομασία του πακέτου της Java της εφαρμογής (Java package).
- Η έκδοση της εφαρμογής (πχ 1.0, 2.7, 3.2)
- Η ελάχιστη έκδοση του λειτουργικού συστήματος Android που απαιτεί η εφαρμογή (min SDK version). Για παράδειγμα, αν έχει δηλωθεί ως min SDK version το 7 που ισοδυναμεί με την έκδοση Android 2.1, τότε αυτή

μπορεί να εκτελεστεί σε συσκευές με έκδοση Android 2.1 ή μεγαλύτερη καθώς υπάρχει συμβατότητα προς τα επάνω.

- Το όνομα της εφαρμογής καθώς και το εικονίδιο της.

- Οι άδειες (licenses) που απαιτούνται για να εκτελεστούν ορισμένες λειτουργίες της εφαρμογής. Για παράδειγμα αν η εφαρμογή μας χρησιμοποιεί το διαδίκτυο θα πρέπει να δηλώνεται και η αντίστοιχη άδεια. Το ίδιο ισχύει αν θέλουμε να έχει πρόσβαση στην SD card, αν θέλουμε να στέλνουμε μηνύματα κ.α. Αν δεν δηλώσουμε τις κατάλληλες άδειες τότε η εφαρμογή μας δεν θα δουλεύει όπως προβλέπεται και θα δημιουργεί σφάλμα. Όταν ο χρήστης πάει να εγκαταστήσει την εφαρμογή μας αυτές οι άδειες αναφέρονται και αν δεν συμφωνήσει τότε αυτή δεν θα εγκατασταθεί. Αυτός ο τρόπος είναι και ένα είδος ασφάλειας απέναντι σε κακόβουλες εφαρμογές που μπορεί να υπάρχουν.

- Όλα τα συστατικά στοιχεία (activities, services, content providers, broadcast receivers) της εφαρμογής. Για παράδειγμα, αν δημιουργήσουμε μια activity χωρίς να την δηλώσουμε στο Android Manifest.xml αυτή δε θα μπορέσει να λειτουργήσει.

- Οι εξωτερικές βιβλιοθήκες που χρησιμοποιεί η εφαρμογή μας. Για παράδειγμα, αν η εφαρμογή μας χρησιμοποιεί το google maps θα πρέπει να δηλωθεί η βιβλιοθήκη com.google.android.maps.

Ακολουθεί ένα παράδειγμα του Android Manifest.xml μιας εφαρμογής.

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.samplesync"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-permission
        android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission
        android:name="android.permission.USE_CREDENTIALS" />
    <uses-permission
        android:name="android.permission.MANAGE_ACCOUNTS" />
    <uses-permission
        android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

<application
        android:icon="@drawable/icon"
        android:label="@string/label">
        <!-- The authenticator service -->
        <service
            android:name=".authenticator.AuthenticationService"
            android:exported="true">
            <intent-filter>
                <action

android:name="android.accounts.AccountAuthenticator" />
            </intent-filter>
            <meta-data
```

```
android:name="android.accounts.AccountAuthenticator"
                android:resource="@xml/authenticator" />
        </service>

<activity
            android:name=".authenticator.AuthenticatorActivity"
            android:label="@string/ui_activity_title"
            android:theme="@android:style/Theme.Dialog"
            android:excludeFromRecents="true"
            android:configChanges="orientation"
            >
        </activity>

  </application>
</manifest>
```

2.8. Βέλτιστος Σχεδιασμός

Μπορεί μια εφαρμογή λειτουργικά να έχει πετύχει τον σκοπό για τον οποίο σχεδιάστηκε, ωστόσο είναι σχεδόν πάντα βέβαιο ότι υπάρχουν περιθώρια βελτίωσης, ώστε να γίνει γρηγορότερη, να απαιτεί λιγότερη μνήμη και να απευθύνεται σε ευρύτερο κοινό χωρίς να χάνει κάποιο στοιχείο από την λειτουργικότητά της. Για τον λόγο αυτό ο προγραμματιστής πρέπει να λαμβάνει υπόψη διάφορους παράγοντες, ώστε ο κώδικάς του να είναι βελτιστοποιημένος.

2.8.1. Δυνατότητα Πρόσβασης

Καθώς το Android θεωρείται ένα από τα «έξυπνα» λειτουργικά είναι πολλοί οι χρήστες που έχουν κάποιο είδος αναπηρίας, το οποίο τους αναγκάζει να χρησιμοποιούν την συσκευή με διαφορετικό τρόπο. Για παράδειγμα, είναι πιθανόν να μην μπορούν να χρησιμοποιήσουν αποτελεσματικά μια οθόνη αφής. Το Android περιλαμβάνει λειτουργίες που βοηθάνε τέτοιους χρήστες να πλοηγούνται πιο εύκολα στις συσκευές, όπως υπηρεσίες κείμενο-σε-ομιλία (text-to-speech) ή μια ιχνόσφαιρα (trackball) για πλοήγηση. Αρκούν ορισμένες ενέργειες από τον χρήστη ώστε η εφαρμογή να γίνει προσβάσιμη σε όλους τους δυνατούς χρήστες.

- Πολλές συσκευές Android έχουν επιπλέον, clickable trackball (με δυνατότητα κλικ) ή clickable d-pad (σταυρό κατεύθυνσης) πέρα από τα βασικά κουμπιά. Έτσι ο χρήστης μπορεί να πλοηγηθεί πλήρως χωρίς να χρησιμοποιεί την οθόνη αφής. Δεν απαιτούνται από τον προγραμματιστή κάποια ενέργεια για την χρήση των παραπάνω αλλά οφείλει να το επιβεβαιώσει δοκιμάζοντας την εφαρμογή του σε τέτοιες συσκευές.

- Άτομα με προβλήματα όρασης ίσως δεν έχουν την δυνατότητα να δουν όλα τα γραφικά στοιχεία της οθόνης κάθε εφαρμογής ώστε να καταλάβουν την χρήση του καθενός. Η παράμετρος android:contentDescription που παρέχεται από το λειτουργικό

σύστημα και μπορούμε να την χρησιμοποιήσουμε σε όλα τα αρχεία xml λύνει το παραπάνω πρόβλημα. Αν ο χρήστης έχει ενεργοποιήσει τα εργαλεία βοηθητικής πρόσβασης στην συσκευή του, όταν εστιάσει σε κάποιο στοιχείο θα ακούει την φράση την οποία έχουμε δηλώσει. Για παράδειγμα το
<EditText
android:id="@+id/name"
android:contentDescription="@string/insert_name"/>
όταν γίνει focused από τον χρήστη θα ακουστεί η φράση (string) που έχει οριστεί στο αρχείο strings.xml.

- Το Talk Back είναι μια εφαρμογή η οποία είναι προεγκατεστημένη σε πολλές συσκευές Android ή μπορεί απλά να την κατεβάσει κανείς από το Android Market. Με αυτήν την εφαρμογή ενεργοποιημένη, ο χρήστης μπορεί να ακούει την περιγραφή της οθόνης στην οποία βρίσκεται κάθε φορά που αλλάζει οθόνη. Συγκεκριμένα μέσα στις εφαρμογές ακούει όλα τα είδη κειμένου που έχει η οθόνη ώστε να ξέρει που βρίσκεται και ποιες είναι οι επιλογές του. Με αυτόν τον τρόπο, μαζί με αυτούς που περιγράφηκαν πιο πάνω, θα γνωρίζουμε κάθε φορά την προσβασιμότητα της εφαρμογής μας και τις ενέργειες που πρέπει να κάνουμε ώστε να βελτιώσουμε την εμπειρία του χρήστη.

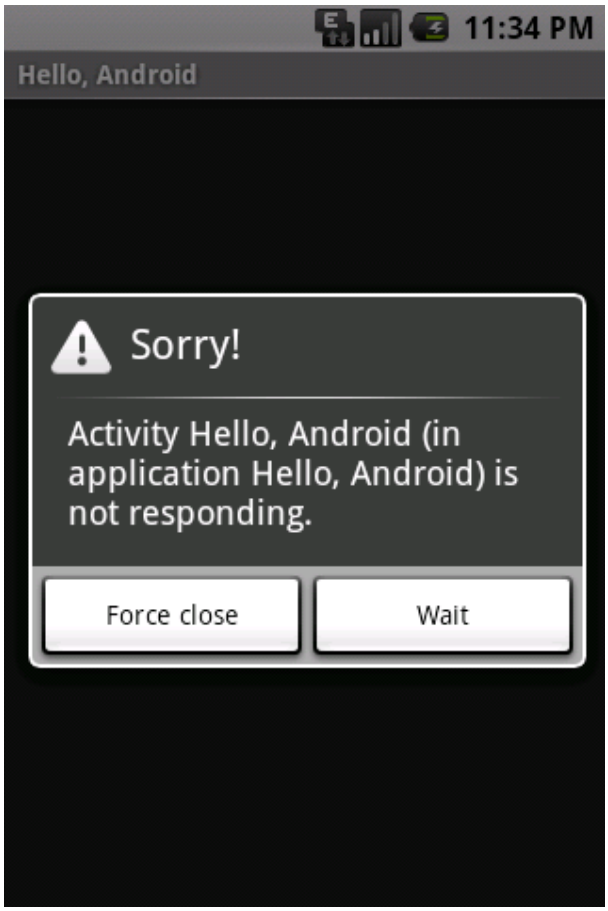
- Τέλος, επισημαίνεται ότι υπάρχουν και εφαρμογές οι οποίες όταν ενεργοποιηθούν δίνουν την δυνατότητα στον χρήστη να εισάγει κείμενο σε πλαίσια μόνο με την ομιλία του (speech-to-text) χωρίς την χρήση του εικονικού πληκτρολογίου.

«Εικόνα 26 : **Δυνατότητα φωνητικής εισαγωγής κειμένου**»

2.8.2. Ανταπόκριση

Όσο βελτιωμένος και αν είναι ο κώδικας της εφαρμογής μας, δεν θα έχει σημασία αν σε κάποιο χρονικό σημείο σταματήσει να αποκρίνεται. Όταν συμβεί κάτι τέτοιο η εφαρμογή μας θα βγάλει ένα παράθυρο διαλόγου ότι δεν αποκρίνεται και ο χρήστης έχει την δυνατότητα να τερματίσει την εφαρμογή ή να περιμένει. Ο διάλογος αυτός σίγουρα δεν είναι κάτι ευχάριστο για τους χρήστες γι' αυτό τον λόγο πρέπει να προσέχουμε στον σχεδιασμό ώστε να μην συμβαίνει κάτι τέτοιο. Γενικά, το σύστημα εμφανίζει έναν τέτοιο διάλογο, όταν η εφαρμογή δεν ανταποκρίνεται σε ενέργειες του χρήστη. Μια τέτοια περίπτωση είναι ένα exception που ο προγραμματιστής έχει παραλείψει να πιάσει. Ομοίως, αν η εφαρμογή υλοποιεί πολύπλοκους και χρονοβόρους υπολογισμούς, ή αν έχει εισέλθει σε έναν άπειρο βρόγχο, τότε έχει κολλήσει και τελικά εμφανίζεται αυτός ο διάλογος. Τέτοια προβλήματα μπορούμε να τα ξεπεράσουμε με τους εξής τρόπους:

- Αρχικά θα πρέπει να ελέγξουμε αν ο κώδικάς μας είναι σωστός και όπου υπάρχει περίπτωση να συμβεί ένα exception να το χειριστούμε κατάλληλα.
- Οι εφαρμογές Android τρέχουν σε μια (κύρια) διεργασία (main thread) η οποία είναι υπεύθυνη και για το user interface. Αυτό σημαίνει ότι πρέπει να εκτελεί όσο το δυνατόν λιγότερες ενέργειες. Επομένως, όλες οι χρονοβόρες διεργασίες, για παράδειγμα οι λειτουργίες που έχουν να κάνουν με το διαδίκτυο ή με την προσπέλαση μιας βάσης δεδομένων, θα πρέπει να εκτελούνται στο παρασκήνιο. Αυτό γίνεται είτε με το να γίνουν σε μια ξεχωριστή thread ή σε μια ειδική ασύγχρονη εργασία που περιλαμβάνει το Android και ονομάζεται AsyncTask. Σε αυτές τις περιπτώσεις για να είναι εμφανές στο χρήστη ότι εκτελείται μια χρονοβόρος εργασία εμφανίζουμε ενα ProgressDialog (π.χ με το μήνυμα Loading…).
- Αν η εφαρμογή μας, πχ ένα παιχνίδι, χρειάζεται αρκετό χρόνο για τις αρχικές λειτουργίες θα ήταν καλό να εμφανίζουμε στον χρήστη μια αρχική οθόνη για όσο διάστημα χρειαστεί.
- Δεν πρέπει να ξεχνάμε ότι το Android είναι λειτουργικό σύστημα για κινητές συσκευές. Αυτό σημαίνει ότι οποιαδήποτε στιγμή μπορεί κάποια activity μιας διαφορετικής εφαρμογής από αυτήν που βρισκόμαστε να ενεργοποιηθεί. Για παράδειγμα, να ενεργοποιηθεί η εφαρμογή που χειρίζεται τις εισερχόμενες τηλεφωνικές κλήσεις και η δικιά μας να περάσει στο παρασκήνιο. Το γεγονός αυτό ενεργοποιεί τις μεθόδους *onSaveInstanceState()* και *onPause().* Αν και τις περισσότερες φορές δεν θα αντιμετωπίσουμε κάποιο πρόβλημα στην εφαρμογή μας θα ήταν προτιμότερο να έχουμε ελέγξει και αυτή την περίπτωση ώστε να προλάβουμε τυχόν σφάλματα.



«Εικόνα 27 : **Ο διάλογος που εμφανίζεται το χρήστη, όταν μια εφαρμογή σταματήσει να ανταποκρίνεται**»

3

Διασύνδεση (Networking)

3.1. Εισαγωγή

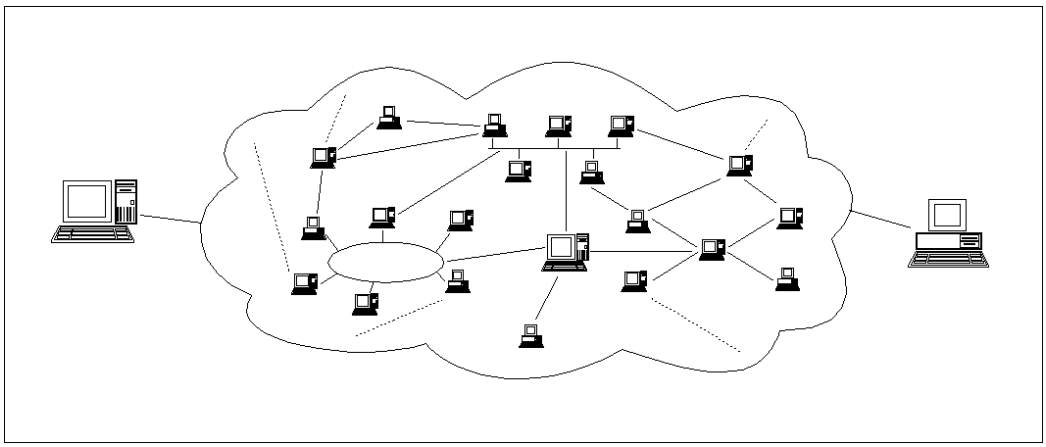
Στην καθημερινή μας ζωή, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς πρέπει να πραγματοποιηθεί κάποια διαδικασία. Στον κόσμο των δικτύων, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα οι υπολογιστές του δικτύου. Το πρωτόκολλο είναι αυτό που καθορίζει το πώς διακινούνται τα δεδομένα, το πώς γίνεται ο έλεγχος και ο χειρισμός των λαθών, κλπ. Το Internet δεν είναι ένα απλό δίκτυο, αλλά ένα διαδίκτυο. Χρειάζεται επομένως ένα σύνολο από συμβάσεις που να καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα υπολογιστές που μπορεί να είναι διαφορετικού τύπου και να ανήκουν σε διαφορετικά δίκτυα.

Ακριβώς αυτό το σύνολο συμβάσεων προσφέρει το TCP/IP. Όλοι οι συσκευές που είναι συνδεδεμένοι στα χιλιάδες μικρότερα δίκτυα του Internet τρέχουν το πρωτόκολλο TCP/IP κι έτσι μιλούν μια κοινή γλώσσα που τους επιτρέπει να συνεννοούνται παρά τις διαφορές τους.

3.2. Πρωτόκολλο TCP

Το TCP (Transmission Control Protocol - Πρωτόκολλο Ελέγχου Μεταφοράς) είναι ένα από τα κυριότερα πρωτόκολλα της Σουίτας Πρωτοκόλλων Διαδικτύου. Βρίσκεται πάνω από το IP protocol (πρωτόκολλο IP). Οι περισσότερες σύγχρονες υπηρεσίες στο Διαδίκτυο βασίζονται στο TCP. Για παράδειγμα το SMTP (port 25), το παλαιότερο (και μη-ασφαλές) Telnet (port 23), το FTP και πιο σημαντικό το HTTP (port 80), γνωστό ως υπηρεσίες World Wide Web (WWW - Παγκόσμιος Ιστός). Το TCP χρησιμοποιείται σχεδόν παντού, για αμφίδρομη επικοινωνία μέσω δικτύου. Αρχικά το Transmission ήταν Transfer, ένας όρος που προσδιόριζε την μεταβίβαση του ελέγχου στα άκρα του δικτύου TCP/IP πριν αποσπαστεί το IP.

Ας υποθέσουμε ότι θέλουμε να μεταφέρουμε δεδομένα από έναν υπολογιστή που είναι συνδεδεμένος στο Internet και βρίσκεται π.χ. στην Αμερική, στο MIT, σε έναν άλλον που είναι επίσης συνδεδεμένος στο Internet και βρίσκεται π.χ. στην Ελλάδα, στο Πανεπιστήμιο Δυτικής Μακεδονίας. Μεταξύ των δύο υπολογιστών παρεμβάλλεται το “σύννεφο” του Internet, δηλ. ένα πλέγμα από συνδέσεις και ενδιάμεσους υπολογιστές.



«Εικόνα 28 : **Οι δύο τελικοί υπολογιστές και το "σύννεφο" του Internet**»

Το Internet χρησιμοποιεί την τεχνολογία μεταγωγής πακέτων για τη μεταφορά των δεδομένων: τα δεδομένα κόβονται σε κομμάτια που ονομάζονται πακέτα και σε κάθε πακέτο μπαίνει μια "επικεφαλίδα" με τις διευθύνσεις του υπολογιστή - αποστολέα και του υπολογιστή - παραλήπτη. Σημειώνουμε ότι σε κάθε υπολογιστή του Internet αντιστοιχίζεται μία διεύθυνση που ονομάζεται διεύθυνση IP.

Το πρωτόκολλο IP είναι υπεύθυνο για το πέρασμα του πακέτου από υπολογιστή σε υπολογιστή μέσα από το "σύννεφο" των συνδέσεων. Καθώς το IP δρομολογεί το κάθε πακέτο μέσα στο δίκτυο, προσπαθεί να το παραδώσει, αλλά δεν μπορεί να εγγυηθεί ούτε ότι το πακέτο θα φτάσει στον προορισμό του ούτε ότι τα διάφορα πακέτα που αποτελούν τα αρχικά δεδομένα θα φτάσουν με τη σειρά με την οποία στάλθηκαν ούτε ότι το περιεχόμενο των πακέτων θα φτάσει αναλλοίωτο.

Το TCP προσφέρει ένα αξιόπιστο πρωτόκολλο πάνω από το IP. Εγγυάται ότι τα πακέτα θα παραδοθούν στον προορισμό τους, ότι θα φτάσουν με τη σειρά με την οποία στάλθηκαν και ότι τα περιεχόμενα των πακέτων θα φτάσουν αναλλοίωτα (δηλ. όπως στάλθηκαν). Το TCP δουλεύει ως εξής: το κάθε πακέτο δεδομένων αριθμείται. Ο υπολογιστής - παραλήπτης και ο υπολογιστής - αποστολέας, αλλά όχι οι ενδιάμεσοι υπολογιστές, παρακολουθούν τους αριθμούς των πακέτων και ανταλλάσσουν μεταξύ τους πληροφορίες. Ο παραλήπτης λαμβάνει το πρώτο πακέτο, το δεύτερο, κλπ. Σε περίπτωση που παρουσιαστεί κάποιο πρόβλημα στο δίκτυο είτε χαθεί κάποιο πακέτο κατά τη διάρκεια της μετάδοσης, το ξαναζητάει και ο αποστολέας είναι υπεύθυνος για την αναμετάδοση του. Ο παραλήπτης ελέγχει επίσης αν το περιεχόμενο των πακέτων φτάνει σωστά. Η μέθοδος αυτή εξασφαλίζει αξιοπιστία και ταχύτητα διότι οι ενδιάμεσοι υπολογιστές δεν εκτελούν ελέγχους.

3.3. TCP sockets

Όταν δημιουργείται ένα socket, το πρόγραμμα πρέπει να διευκρινίζει τον τομέα διεύθυνσης (address domain) και τον τύπο της. Δύο διαδικασίες μπορούν να επικοινωνούν μεταξύ τους μόνο αν τα socket τους είναι του ίδιου τύπου και στον ίδιο τομέα.

Υπάρχουν δύο address domain που χρησιμοποιούνται ευρέως, τον τομέα Unix, στην οποία δύο διαδικασίες που μοιράζονται ένα κοινό σύστημα αρχείων επικοινωνούν, και ο τομέας του Διαδικτύου, στην οποία δύο διαδικασίες που τρέχουν σε οποιουσδήποτε οικοδεσπότες (hosts) στο Διαδίκτυο επικοινωνούν. Καθένα από αυτά έχει τη δική του μορφή διεύθυνσης.

Η διεύθυνση μιας socket στον τομέα του Unix είναι σειρά χαρακτήρων, που είναι βασικά μια καταχώρηση στο σύστημα αρχείων.

Η διεύθυνση μιας socket στον τομέα του Διαδικτύου αποτελείται από την διεύθυνση διαδικτύου της μηχανής που εκτελεί χρέη host (ο κάθε υπολογιστής στο Internet έχει μια μοναδική διεύθυνση 32 bit, που συχνά αναφέρεται ως διεύθυνση IP). Επιπλέον, κάθε socket χρειάζεται μια θύρα σε αυτόν τον κεντρικό υπολογιστή. Οι αριθμοί θύρας είναι 16-bit μη προσημασμένοι ακέραιοι. Τα χαμηλότερα νούμερα είναι αποκλειστικά στο Unix για τις τυποποιημένες υπηρεσίες. Για παράδειγμα, ο αριθμός της θύρας του διακομιστή FTP είναι 21. Είναι σημαντικό οι τυποποιημένες υπηρεσίες να είναι στην ίδια θύρα σε όλους τους υπολογιστές έτσι ώστε οι πελάτες να γνωρίζουν τις διευθύνσεις τους. Ωστόσο, οι αριθμοί θύρας πάνω από 2000 είναι γενικά διαθέσιμοι.

Υπάρχουν δύο τύποι sockets που χρησιμοποιούνται ευρέως, stream sockets και datagram sockets. Τα stream sockets χειρίζονται τις επικοινωνίες ως μια συνεχή ροή χαρακτήρων, ενώ τα datagram sockets πρέπει να διαβάσουν ολόκληρα τα μηνύματα ταυτόχρονα. Το καθένα χρησιμοποιεί το δικό του πρωτόκολλο επικοινωνίας.

Τα stream sockets χρησιμοποιούν το TCP (Transmission Control Protocol), και τα datagram sockets χρησιμοποιούν το UDP (Unix Datagram Protocol.

Τα TCP Sockets χρησιμοποιούνται στην επικοινωνία μέσω TCP (και UDP) για τον προσδιορισμό μοναδικών end-to-end συνδέσεων. Καλούνται «εικονικοί λιμένες (virtual ports)», διότι μια ενιαία φυσική σύνδεση μπορεί να εξυπηρετήσει πολλαπλές συνδέσεις. Κάθε πλευρά της σύνδεσης χρησιμοποιεί το δικό της αριθμό θύρας, που δεν αλλάζει κατά τη διάρκεια αυτής της σύνδεσης. Ο αριθμός θύρας και τη διεύθυνση IP από κοινού προσδιορίζουν μοναδικά ένα τελικό σημείο. Μαζί, τα δύο άκρα θεωρούνται μια «socket».

3.4. Το μοντέλο πελάτη – διακομιστή

Το ευρύτερα διαδεδομένο μοντέλο ανάπτυξης δικτυακών εφαρμογών (δηλαδή εφαρμογών που εκτελούνται σε ξεχωριστούς υπολογιστές και ανταλλάσσουν μηνύματα μεταξύ τους μέσω δικτύου) είναι το μοντέλο του πελάτη - εξυπηρετητή (client - server). Οι όροι αυτοί αφορούν τις δύο διαδικασίες που θα πρέπει να επικοινωνούν μεταξύ τους.

Μια διεργασία που την χαρακτηρίζουμε ως εξυπηρετητή αρχίζει να εκτελείται σε κάποιον υπολογιστή. Μετά την αρχικοποίηση της, πέφτει σε "λήθαργο", αναμένοντας μία διεργασία - πελάτη να επικοινωνήσει μαζί της και να της ζητήσει κάποια υπηρεσία. Μία διεργασία - πελάτης αρχίσει να εκτελείται, είτε στον ίδιο υπολογιστή, είτε σε κάποιο απομακρυσμένο υπολογιστή. Η διεργασία πελάτης επικοινωνεί με τον υπολογιστή στον οποίο "τρέχει" ο εξυπηρετητής μέσω δικτύου. Αξίζει να σημειωθεί ότι ο πελάτης χρειάζεται να γνωρίζει την διεύθυνση του διακομιστή, ενώ ο διακομιστής δεν χρειάζεται να γνωρίζει ούτε την «ύπαρξη» του πελάτη πριν γίνει η σύνδεση.

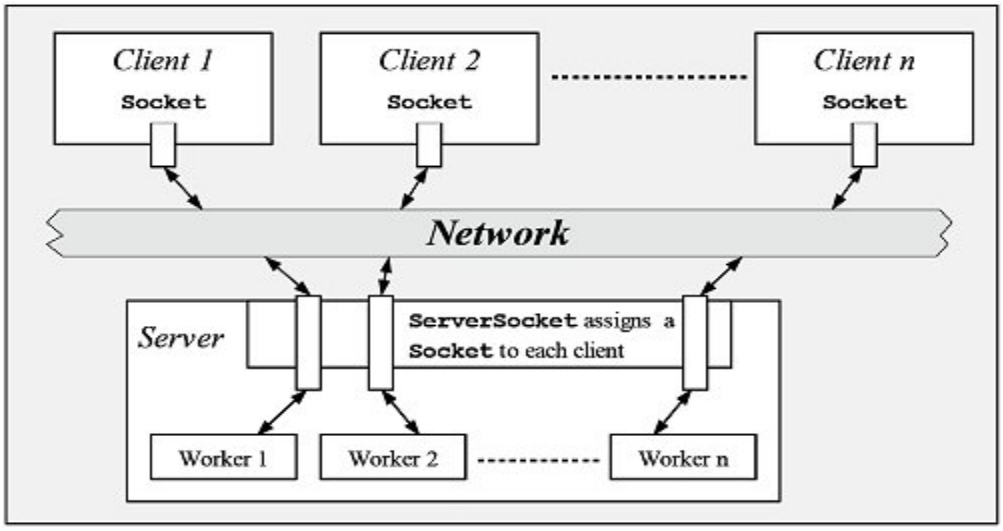
Η διεργασία πελάτης στέλνει μια αίτηση, μέσω του δικτύου, στον εξυπηρετητή, ζητώντας του κάποιου είδους υπηρεσία (π.χ. μεταφορά αρχείου, απομακρυσμένη εκτύπωση, ανάγνωση και αποστολή mail και άλλες). Ταυτόχρονα με την εξυπηρέτηση κάποιου πελάτη, ο server έχει την δυνατότητα να δέχεται και αιτήσεις άλλων πελατών προς εξυπηρέτηση. Όταν ο εξυπηρετητής τελειώσει με όλους τους πελάτες, τότε ξαναπέφτει σε "λήθαργο", περιμένοντας για μια καινούργια αίτηση και η διαδικασία ξαναρχίζει από την αρχή.

Τα βήματα που χρειάζονται για την εγκαθίδρυση ενός socket στην πλευρά του πελάτη είναι τα εξής :

- Δημιουργία του socket με την μέθοδο *new Socket(IP address, port),* όπου δημιουργούμε ένα socket με τον διακομιστή στην συγκεκριμένη διεύθυνση και θύρα που επιλέξαμε.
- Σύνδεση του socket στην διεύθυνση του διακομιστή χρησιμοποιώντας την μέθοδο *connect().*
- Αποστολή και λήψη δεδομένων. Υπάρχουν πολλοί τρόποι για να το κάνει κανείς αυτό, αλλά ο πιο απλός είναι το *read()* και *write().*

Τα βήματα που χρειάζονται για την εγκαθίδρυση ενός socket στην πλευρά του διακομιστή είναι τα εξής :

- Δημιουργία του socket με την μέθοδο *new ServerSocket(port),* όπου δημιουργούμε ένα socket που «ακούει» για συνδέσεις στην συγκεκριμένη θύρα που επιλέξαμε.
- Αποδοχή μιας σύνδεσης με την μέθοδο *accept().* Σε αυτήν συνήθως ο διακομιστής μπλοκάρει μέχρι ένας πελάτης να συνδεθεί.
- Αποστολή και λήψη δεδομένων με τον ίδιο τρόπο όπως και στον πελάτη.



«Εικόνα 29 : **Αλληλεπίδραση πελατών - διακομιστή**»

4

Αρχιτεκτονική της εφαρμογής

4.1. Εισαγωγή

Στο κεφάλαιο που ακολουθεί αρχικά θα περιγράψουμε την εφαρμογή που αναπτύχθηκε για το λειτουργικό σύστημα Android αναφέροντας όλες τις δυνατότητες που αυτή προσφέρει. Έπειτα θα αναλύσουμε την δομή της, τα αρχεία δηλαδή στα οποία οργανώνεται καθώς και την χρήση του καθενός.

Η βάση που χρησιμοποιείται για αυτήν την εφαρμογή φιλοξενείται στον server του Πανεπιστημίου Δυτικής Μακεδονίας και συγκεκριμένα στην διεύθυνση http://zafora.icte.uowm.gr/~ictest00032/DATABASE.

4.2. Περιγραφή

Κύριος σκοπός της εφαρμογής είναι να δημιουργηθεί ένα παιχνίδι γνώσεων, μέσω του οποίου, όσοι συμμετέχουν σε αυτό ως παίκτες να έχουν την ευκαιρία, αφενός, να χαλαρώσουν από το άγχος της καθημερινότητας και να ψυχαγωγηθούν, αφετέρου, να εμπλακούν ευχάριστα στη διαδικασία της μάθησης, καθώς οι ερωτήσεις καλύπτουν ένα ευρύ πεδίο γνωστικών αντικειμένων.

Ο χρήστης λοιπόν αφού εγκαταστήσει την εφαρμογή, αν αποφασίσει να παίξει, έχει δύο επιλογές. Πρώτον, πρέπει να διαλέξει πως θέλει να συμπεριφερθεί η συσκευή του, σαν διακομιστής ή σαν πελάτης. Βασική προϋπόθεση για να λειτουργήσει σαν διακομιστής πρέπει να έχει δικαιώματα πρόσβασης υπερχρήστη στην συσκευή του, θα το αναλύσουμε σε παρακάτω κεφάλαιο. Αφού τα αποκτήσει είτε μπορεί να ξεκινήσει ένα παιχνίδι διαλέγοντας τον αριθμό των ερωτήσεων αλλά και τον αριθμό των παικτών, συμπεριλαμβανομένου και του εαυτού του, είτε μπορεί να τροποποιήσει την βάση δεδομένων των ερωτήσεων.

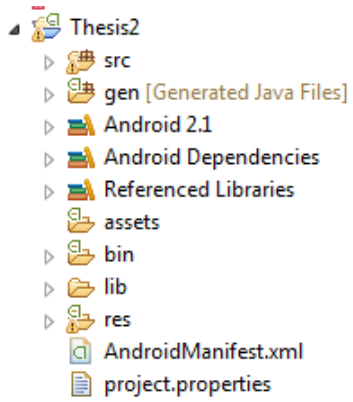
Την πρώτη φορά που γίνεται εγκατάσταση στην εφαρμογή δίνεται η δυνατότητα στον χρήστη να δημιουργήσει μια δική του βάση δεδομένων βάζοντας τις δικές του ερωτήσεις, τροποποιώντας τις ήδη υπάρχουσες ή απλά διαγράφοντας όποιες αντιπροσωπεύουν το γνωστικό του επίπεδο. Ωστόσο, αν το επιθυμεί μπορεί με το κουμπί update να κατεβάσει την επίσημη βάση, η οποία ανανεώνεται συχνά, από τους servers του Πανεπιστημίου Δυτικής Μακεδονίας. Προσοχή όμως, αυτό είναι μια ενέργεια που θα διαγράψει οποιαδήποτε άλλη βάση δεδομένων υπάρχει στην συσκευή και

σχετίζεται με την εφαρμογή και θα χρησιμοποιεί μόνο την συγκεκριμένη που κατέβηκε.

Εάν ο χρήστης επιλέξει να συμπεριφέρεται η συσκευή του σαν πελάτης τότε δεν χρειάζονται δικαιώματα υπερχρήστη και απλά με την διεύθυνση IP που θα πάρει από μια συσκευή – διακομιστή μπορεί να συνδεθεί και όταν συμπληρωθεί ο αριθμός των παικτών τότε το παιχνίδι θα αρχίσει.

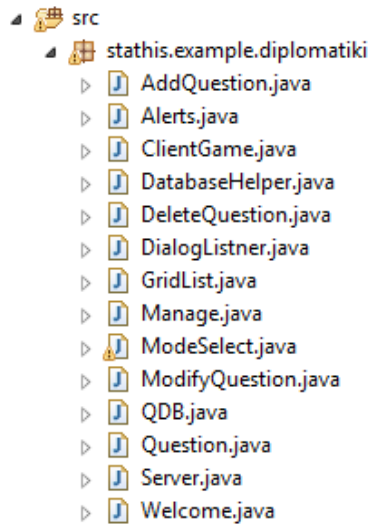
4.3. Δομή

Πριν προχωρήσουμε στην λεπτομερή ανάλυση των διαφόρων λειτουργιών της εφαρμογής που αναπτύχθηκε θα ήταν χρήσιμο να παρουσιάσουμε πρώτα την δομή της, δηλαδή το πως οργανώνονται όλα τα αρχεία που περιέχονται σε αυτήν και ποια είναι η λειτουργία τους. Στο προγραμματιστικό περιβάλλον Eclipse όπου αναπτύχθηκε η εφαρμογή, το project εμφανίζεται με την παρακάτω δομή :



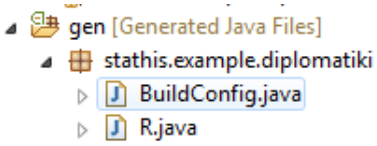
«Εικόνα 30 : **Δομή της εφαρμογής**»

Ακολουθεί η εξήγηση των παρακάτω φακέλων και αρχείων καθώς και του περιεχόμενού τους :



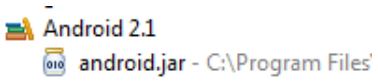
«Εικόνα 31 : **Περιεχόμενα φακέλου src**»

Αυτός ο φάκελος περιέχει τα πακέτα τα οποία έχουμε δηλώσει στην εφαρμογή μας. Έχουμε δηλώσει μόνο ένα πακέτο το stathis.example.diplomatiki οπότε μόνο αυτό εμφανίζεται. Έπειτα στο πακέτο αυτό περιέχονται όλες οι κλάσεις της εφαρμογής μας. Οι κλάσεις χωρίζονται σε activities, δηλαδή στις διαφορετικές οθόνες και στις υπόλοιπες που εκτελούν κάποια ειδική λειτουργία.



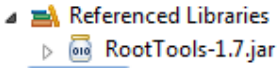
«Εικόνα 32 : **Περιεχόμενα φακέλου gen**»

Ο φάκελος αυτός για το πακέτο που χρησιμοποιούμε στην εφαρμογή μας , το αρχείο R.java το οποί είναι ουσιαστικά είναι μια κλάση η οποία κατασκευάζεται μόνη αυτόματα από το σύστημα. Η κλάση αυτή συνδέει όλα τα application resources με τον κώδικα της εφαρμογής μας. Έχουμε εξηγήσει ότι όλα αυτά δηλώνονται σε ξεχωριστά αρχεία και όχι μέσα στις κλάσεις της εφαρμογής μας. Η κλάση R δηλώνει κάθε στοιχείο από τις εικόνες, τα string, τα style σαν μια ακέραια σταθερά και με τιμή ένα μοναδικό δεκαεξαδικό αριθμό, ώστε να αναφερόμαστε σε αυτές μέσα από τις δικές μας κλάσεις σαν δημόσιες σταθερές (public constants).



«Εικόνα 33 : **Βιβλιοθήκες του λειτουργικού συστήματος Android**»

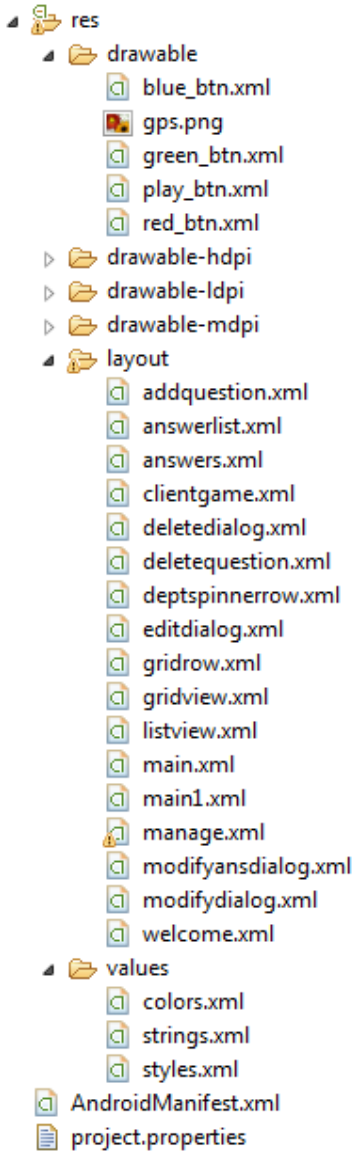
Πρόκειται για τις βιβλιοθήκες που επιλέγονται αυτόματα από το σύστημα, όταν εμείς δηλώνουμε την ελάχιστη έκδοση Android με την οποία θα είναι συμβατή η εφαρμογή μας. Συγκεκριμένα όταν ξεκινάμε ένα project για την κατασκευή Android application τότε ανάμεσα στις διάφορες επιλογές πρέπει να επιλέξουμε και το Build Target της εφαρμογής, που σημαίνει την ελάχιστη έκδοση με την οποία θέλουμε να είναι συμβατή. Έτσι, το σύστημα τοποθετεί τις κατάλληλες βιβλιοθήκες για να λειτουργήσει η εφαρμογή. Συγκεκριμένα η εφαρμογή μας είναι συμβατή από την έκδοση 2.1 και πάνω με την διαφορά ότι από την έκδοση 2.3.3 και πάνω ανοίγουν και άλλες δυνατότητες.



«Εικόνα 34 : **Επιπλέον βιβλιοθήκες που χρησιμοποιεί η εφαρμογή**»

Οι βιβλιοθήκες αυτές προστίθενται από τον προγραμματιστή για την χρήση ορισμένων λειτουργιών, οι οποίες δεν είναι ενσωματωμένες στις βιβλιοθήκες του λειτουργικού συστήματος, ούτε στην έκδοση του JDK που χρησιμοποιούμε. Η βιβλιοθήκη RootTools μας παρέχει τρόπους ελέγχου

ύπαρξης δικαιωμάτων υπερχρήστη στην συσκευή και υπάρχει αυτή η δυνατότητα τα εξασφαλίζει για την εφαρμογή μας.



«Εικόνα 35 : **Όλα τα resources της εφαρμογής**»

Στο φάκελο res αποθηκεύονται όλα τα application resources τα οποία χρησιμοποιούμε. Συγκεκριμένα, στο φάκελο drawable αποθηκεύουμε όλα τα αρχεία εικόνας που χρησιμοποιούνται στην εφαρμογή. Στο φάκελο layout αποθηκεύονται τα αρχεία xml στα οποία δηλώνουμε το user interface της κάθε οθόνης. Ο φάκελος values περιλαμβάνει το αρχείο strings.xml με όλα τα string της εφαρμογής δηλωμένα εκεί, το αρχείο colors.xml όπου γίνεται η διαβάθμιση των διαφορετικών χρωμάτων που χρησιμοποιεί η εφαρμογή και το αρχείο styles.xml στο οποία δηλώνονται τα διαφορετικά styles που χρησιμοποιεί κάθε γραφικό στοιχείο ή ολόκληρη η οθόνη.

Τέλος, παρατηρούμε 2 αρχεία τα οποία δημιουργούνται αυτόματα ξεκινάμε ένα καινούργιο project για μια εφαρμογή Android. Το αρχείο Android Manifest.xml έχει αναλυθεί σε προηγούμενη ενότητα. Το άλλο αρχείο αφορά το σύστημα και δεν πρέπει να το επεξεργαζόμαστε.

5

Βασικές λειτουργίες και σενάριο χρήσης εφαρμογής

5.1. Εισαγωγή

Αφού παρουσιάστηκε η δομή της εφαρμογής, μπορούμε τώρα να παρουσιάσουμε όλες τις λειτουργίες που περιλαμβάνει η εφαρμογή που αναπτύχθηκε, καθώς και τον τρόπο υλοποίησης αυτών στο προγραμματιστικό περιβάλλον. Σε αυτό το κεφάλαιο, θα περιγραφούν όλες οι διαφορετικές οθόνες της εφαρμογής, ο τρόπος σύνδεσης μεταξύ τους, οι κλάσεις στις οποίες υλοποιούνται και όλες οι δυνατές επιλογές που έχει ο χρήστης όταν βρίσκεται σε συγκεκριμένη οθόνη. Τα διάφορα στιγμιότυπα της εφαρμογής παρουσιάζονται σε εικόνες και έπειτα εξηγούνται οι λειτουργίες για να διευκολύνεται ο χρήστης στην κατανόηση τους. Για διευκόλυνση αναγνωσιμότητας του κειμένου, θα παρουσιαστούν κομμάτια κώδικα από τα σημαντικότερα αρχεία της εφαρμογής.

5.2. Σενάριο Χρήσης

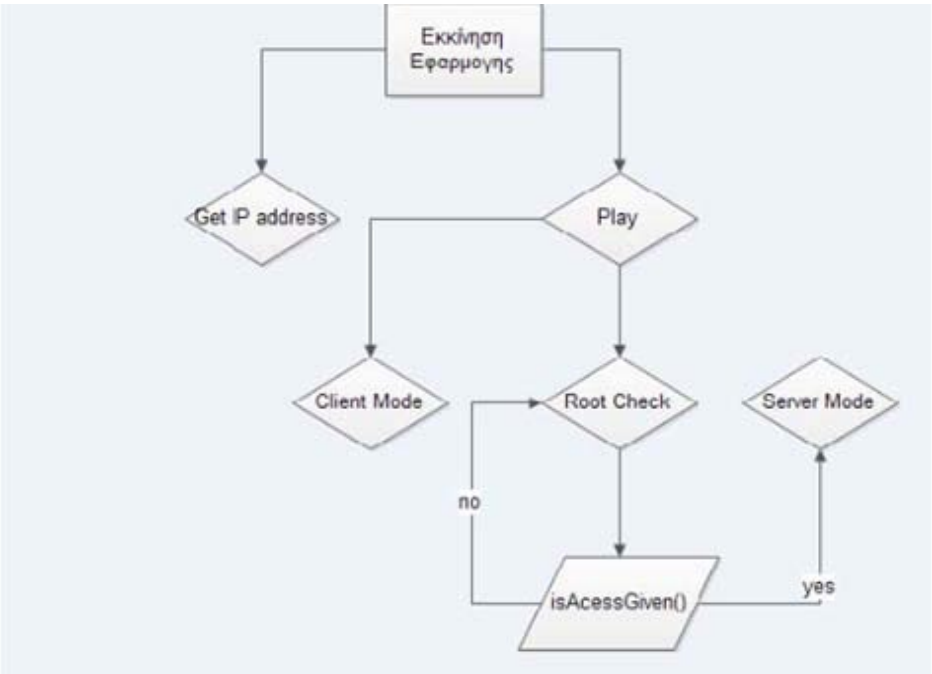
Για να γίνει πιο κατανοητή η εφαρμογή στον αναγνώστη παρουσιάζονται παρακάτω τρία διαγράμματα ροής που αναλύουν τον τρόπο λειτουργίας της εφαρμογής και τις ενέργειες του χρήστη.

Αρχικά, (Εικόνα 36), παρουσιάζεται οι επιλογές του χρήστη όσον αφορά την επιλογή της συμπεριφοράς της συσκευής του. Μπορεί να επιλέξει την επιλογή του πελάτη (client mode) ή την επιλογή του διακομιστή (server mode) εφόσον έχει αποκτήσει πρόσβαση υπερχρήστη (root access) στην συσκευή του.

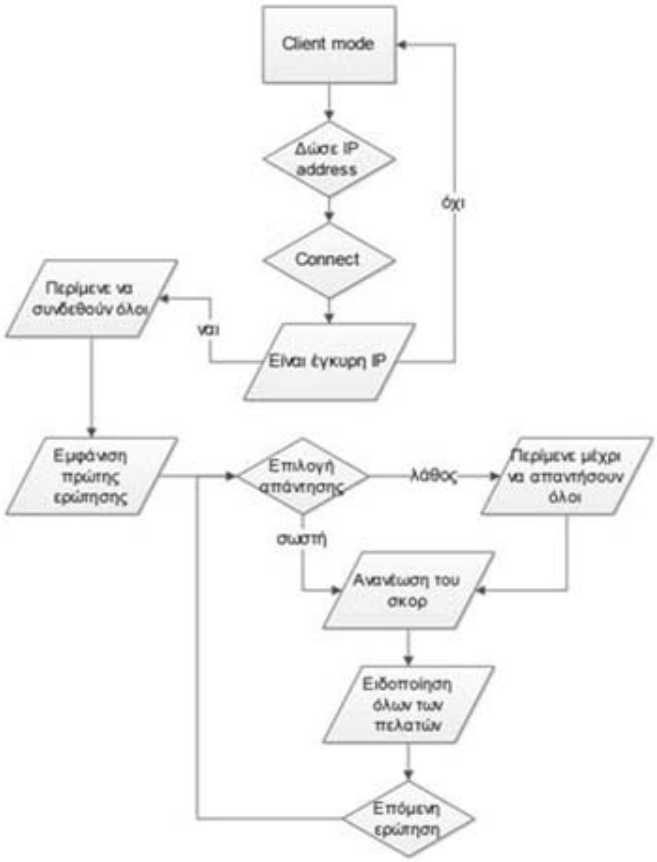
Στην συνέχεια (Εικόνα 37) αναλύονται οι επιλογές που έχει ο χρήστης όταν επιλέξει το client mode συνδεόμενος σε μια διεύθυνση IP και πως λειτουργεί η εφαρμογή με περισσότερους από έναν πελάτες.

Τέλος, παρουσιάζεται ο server mode και οι δυνατότητες του χρήστη που έχει σε αυτήν την περίπτωση (Εικόνα 38). Μπορεί να επιλέξει να γίνει οικοδεσπότης (host) σε ένα παιχνίδι διαλέγοντας τον αριθμό των πελατών και των ερωτήσεων αλλά αποκτά και πρόσβαση στην βάση δεδομένων με τις ερωτήσεις δίνοντας του την δυνατότητα να δημιουργήσει, να διαγράψει ή να

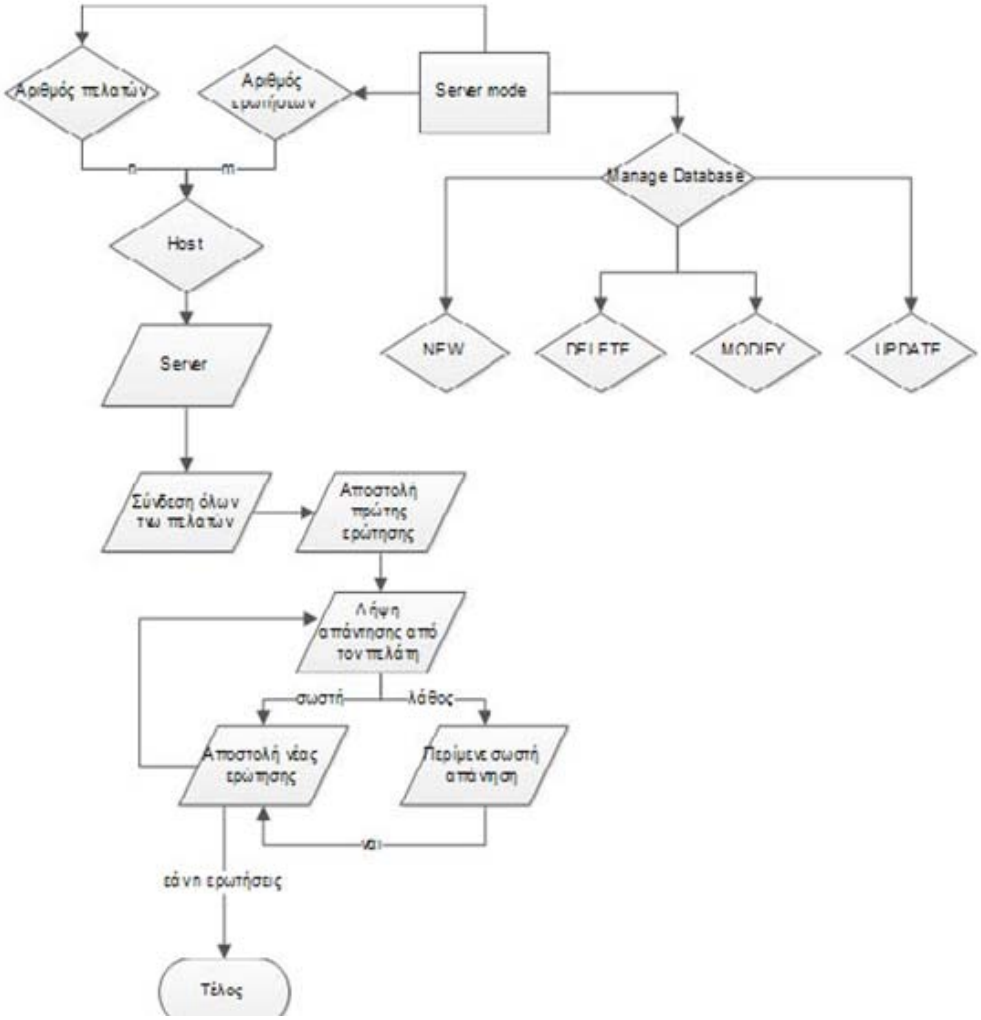
τροποποιήσει μια υπάρχουσα ερώτηση. Ακόμα μπορεί να κατεβάσει την τελευταία επίσημη έκδοση της βάσης από τους servers του Πανεπιστημίου Δυτικής Μακεδονίας.



«Εικόνα 36 : **Διάγραμμα ροής επιλογών πελάτη – διακομιστή**»



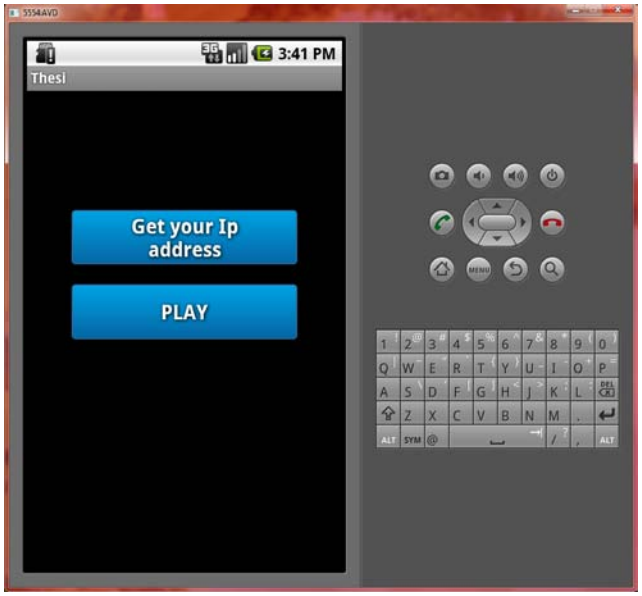
«Εικόνα 37 : **Διάγραμμα ροής επιλογών πελάτη**»



«Εικόνα 38 : **Διάγραμμα ροής επιλογών διακομιστή**»

5.3. Αρχική Οθόνη

Όταν εκτελεστεί η εφαρμογή εμφανίζεται η αρχική οθόνη με δύο επιλογές. Αυτή φαίνεται στην παρακάτω εικόνα.



«Εικόνα 39 : **Αρχική οθόνη εφαρμογής**»

Πρόκειται για την κλάση Welcome που την έχουμε ορίσει σαν main activity στο Android Manifest.xml ως εξής :

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

Όταν ξεκινάει η activity δηλώνουμε πρώτα τη διεπαφή χρήστη της οθόνης με την εντολή *setContentView(R.layout.welcome)*. Στο αρχείο welcome.xml έχουμε δηλώσει όλο το γραφικό περιβάλλον της οθόνης.

Είναι μια αρκετά απλή οθόνη που δίνει στον χρήστη δύο επιλογές. Εάν έχει συνδεθεί σε ένα Wi-fi hotspot μπορεί να δει αμέσως ποια είναι η IP του ώστε να μπορεί να την διαδώσει στους υπόλοιπους παίκτες για να συνδεθούν και μπορεί να ξεκινήσει το παιχνίδι. Η εύρεση της IP γίνεται με το παρακάτω κομμάτι κώδικα :

```
public String getLocalIpAddress()
      {
        try
        {
            for         (Enumeration<NetworkInterface>         en         =
NetworkInterface.getNetworkInterfaces(); en.hasMoreElements();)
             {
```

```
                NetworkInterface intf = en.nextElement();
                for       (Enumeration<InetAddress>       enumIpAddr       =
intf.getInetAddresses(); enumIpAddr.hasMoreElements();)
                        {
                    InetAddress inetAddress = enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress())  {
                        return inetAddress.getHostAddress().toString();

            }
                        }
            }
        } catch (SocketException ex) { Log.e(LOG_TAG, ex.toString()); }
        return null;
           }
```

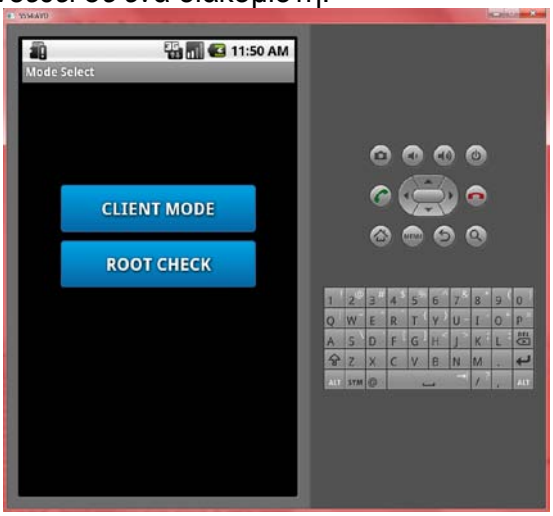
Με αυτή την μέθοδο ψάχνουμε όλες τις διεπαφές δικτύου και στην συνέχεια ψάχνουμε σε αυτές για διευθύνεις IP. Επιστρέφει null αν δεν βρει κάποια διεύθυνση, αλλιώς επιστέφει ένα string με την διεύθυνση της συσκευής είτε την απέκτησε από ένα Wi-Fi δίκτυο είτε από το 3G. Ειδοποιούμε τον χρήστη για την διεύθυνση του μέσω του Notification Manager για να είναι εύκολο για αυτόν να την βρει ανά πάσα στιγμή.

Αν ο χρήστης επιλέξει το κουμπί play τότε μεταβιβάζεται στην δεύτερη οθόνη όπου καλείται να διαλέξει τον ρόλο της συσκευής του, δηλαδή να διαλέξει αν θέλει να είναι διακομιστής ή πελάτης. Η αλλαγή οθόνης γίνεται με την μέθοδο

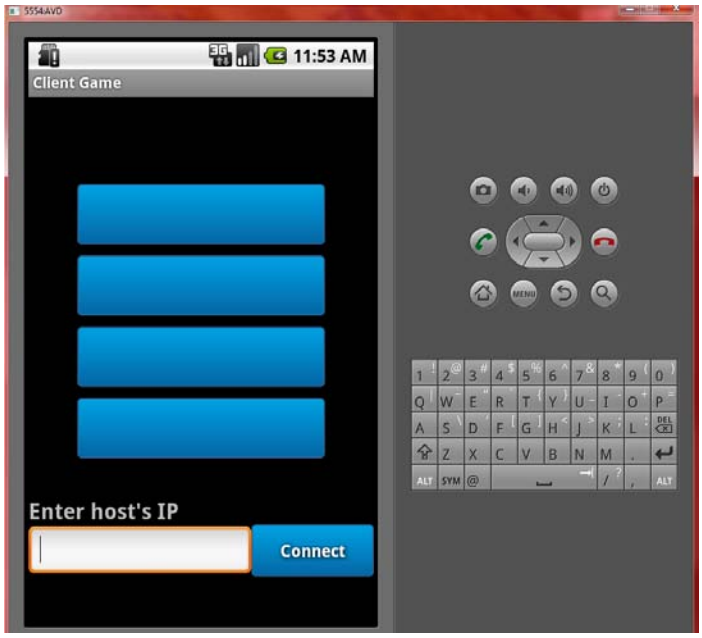
```
Intent ini = new Intent(Welcome.this, ModeSelect.class);
                        startActivity(ini);
```

5.4. Επιλογή συμπεριφοράς πελάτη (client mode)

Σε αυτήν την οθόνη ο χρήστης μπορεί να διαλέξει το κινητό του να συμπεριφέρεται σαν πελάτης και να περιμένει μια διεύθυνση IP ώστε να συνδεθεί σε ένα διακομιστή.



«Εικόνα 40 : Οθόνη επιλογής συμπεριφοράς συσκευής»



«Εικόνα 41 : Οθόνη εισαγωγής διεύθυνσης IP»

Σε αυτήν την περίπτωση (Εικόνα 41) γίνεται η σύνδεση των δύο συσκευών με τα sockets που έχουμε περιγράψει σε προηγούμενο κεφάλαιο. Δημιουργούμε μια persistent socket, ένα socket δηλαδή που παραμένει ανοικτό και συνδεδεμένο με την άλλη συσκευή. Αφού ο χρήστης γράψει την IP την μετατρέπουμε σε string και μετά κάνουμε την σύνδεση με τον εξής τρόπο :

```
private String serverIpAddress;
          Socket nsocket;
      try
                    {
   InetAddress serverAddr = InetAddress.getByName(serverIpAddress);
   SocketAddress sockaddr = new InetSocketAddress(serverAddr, 5000); //
χρησιμοποιούμε την Ip και την θύρα 5000
                    nsocket = new Socket();
                    nsocket.connect(sockaddr);
                    }
      catch(Exception e)
        {Log.i("Connect", "Connection Error");}
```

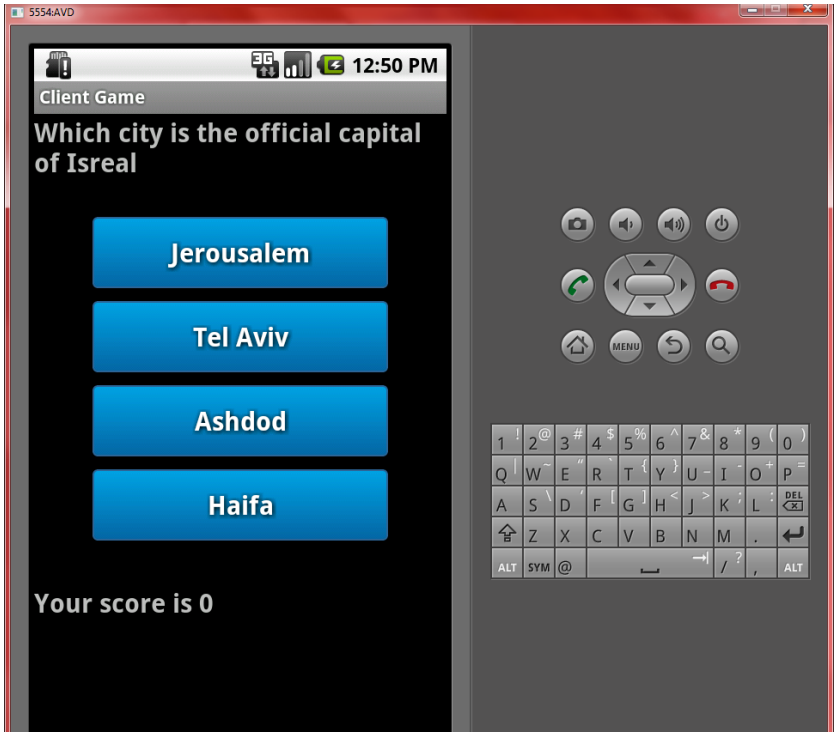
Ο τρόπος αυτός είναι και ο προτιμότερος για την αποφυγή επιπλέον εργασίας δημιουργώντας και καταστρέφοντας συνδέσεις και την επιπλέον καθυστέρηση σε όλα τα TCP πακέτα καθώς και την αποφυγή δημιουργίας πολλών socket είτε στον πελάτη είτε στον διακομιστή. Δεν υπάρχει απλά κανένας καλός λόγος για να δημιουργούνται και να καταστρέφονται συνδέσεις. Έτσι δημιουργούμε το persistent socket με την global μεταβλητή nsocket.

Ο άλλος τρόπος υλοποίησης αυτού του θέματος είναι οι πελάτες να δημοσκοπούν (polling) τον διακομιστή κάτι το οποίο είναι ανεπαρκές. Πρέπει ο διακομιστής να δημοσιεύει τα δεδομένα στους πελάτες όταν υπάρχουν διαθέσιμα δεδομένα. Αυτό θα επιτρέπει στον διακομιστή να ελέγχει κάπως τον φόρτο εργασίας διαλέγοντας το χρονικό διάστημα που θα στείλει δεδομένα

στους πελάτες. Μπορεί να στέλνει δεδομένα κάθε φορά ή να περιμένει να συγκεντρώσει δεδομένα περιμένοντας κάποιο χρονικό διάστημα.

Εάν κάθε πελάτης δημοσκοπεί τον διακομιστή δημιουργείται περισσότερος θόρυβος δικτύου καθώς κάθε πελάτης στέλνει μήνυμα στον διακομιστής για το αν υπάρχει κάτι να του στείλει και δημιουργείται περισσότερη δουλειά στον διακοσμητή καθώς χρειάζεται να απαντά σε αυτά τα ερωτήματα. Ο διακομιστής ξέρει πότε υπάρχουν διαθέσιμα δεδομένα οπότε η πιο σωστή προσέγγιση θα ήταν να είναι αυτός υπεύθυνος για να ενημερώσει τους πελάτες.

Αφού λοιπόν γίνει η σύνδεση το παιχνίδι αρχίζει και η πρώτη ερώτηση περιμένει να απαντηθεί.



«Εικόνα 42 : Παράδειγμα εμφάνισης ερώτησης»

Πρέπει να σημειωθεί εδώ το εξής. Κατά την διάρκεια της ανάπτυξης της εφαρμογής ο προγραμματιστής μπορεί να μην διαθέτει συσκευές για να ελέγξει την συνδεσιμότητα μεταξύ τους και επομένως δεν μπορεί να συνεχίσει την εφαρμογή. Την λύση την προσφέρει το προγραμματιστικό εργαλείο Eclipse μαζί με την AVD και με λίγη βοήθεια από το λειτουργικό σύστημα του κάθε υπολογιστή στον οποίο δουλεύει ο προγραμματιστής. Με την βοήθεια, λοιπόν του telnet ο χρήστης μπορεί να χρησιμοποιήσει όσες εικονικές συσκευές θέλει που θα παίξουν τον ρόλο του πελάτη.

Το Telnet, ακρωνύμιο των αγγλικών λέξεων TELecommunication NETwork, είναι ένα πρωτόκολλο επικοινωνίας διασυνδεδεμένων (σε δίκτυο) υπολογιστών. Δημιουργήθηκε αρχικά ως πρωτόκολλο επικοινωνίας σε τοπικά δίκτυα το 1969 και επεκτάθηκε και στο Διαδίκτυο το 1975. Ο όρος καλύπτει επίσης την υπηρεσία του Διαδικτύου αλλά και το λογισμικό που την υποστηρίζει. Με το Telnet ο χρήστης που συνδέεται με κάποιον υπολογιστή

μπορεί να τον "ελέγχει" (όσο του επιτρέπεται από το διαχειριστή της υπηρεσίας/δικτύου) σαν να ήταν καθισμένος σε κάποιο τερματικό του. Αυτό, πρακτικά, σημαίνει ότι από ένα προσωπικό υπολογιστή με λειτουργικό σύστημα Windows, ο χρήστης μπορεί να χειριστεί έναν υπολογιστή με λειτουργικό σύστημα Unix.

Αν και στο Διαδίκτυο δεν είναι πλέον ιδιαίτερα δημοφιλής υπηρεσία (το βοηθητικό πρόγραμμα Telnet.exe έχει αφαιρεθεί από την εξ ορισμού εγκατάσταση των Windows Vista), χρησιμοποιείται από τεχνικούς για τον έλεγχο άλλων πρωτοκόλλων, όπως το SMTP, το POP3 κτλ. Χρησιμοποιείται επίσης πολύ από μεγάλους υπολογιστές (mainframes) εκπαιδευτικών ιδρυμάτων, μεγάλων εταιρειών και παρόμοιων φορέων, επειδή, όπως προαναφέρθηκε, παρέχει εξαιρετικές δυνατότητες ελέγχου στα επιμέρους στοιχεία ενός δικτύου. Για το σημερινό χρήστη ιδιαίτερα χρήσιμη θα φανεί η υπηρεσία για πρόσβαση σε υλικό απομακρυσμένων βιβλιοθηκών.

Όπως αναφέραμε και παραπάνω αυτή η υπηρεσία είναι απενεργοποιημένη στα Windows Vista και στα Windows 7. Πρέπει μέσα από τον πίνακα ελέγχου να μπούμε στην καρτέλα προσθαφαίρεσης προγραμμάτων

Έχοντας, λοιπόν ανοίξει 2 συσκευές AVD παίρνοντας η καθεμία ένα μοναδικό αριθμό πάνω αριστερά (5554, 5556). Πρέπει να διαλέξουμε ποια συσκευή θα έχει τον ρόλο του διακομιστή και ποια το ρόλο του πελάτη. Για παράδειγμα, ας πούμε ότι ο διακομιστής θα είναι η συσκευή με τον αριθμό 5554. Ανοίγουμε ένα παράθυρο command line και γράφουμε την εντολή *telnet localhost 5554.* Έτσι μπαίνουμε στο περιβάλλον του telnet όπου γράφουμε την εντολή *redir add tcp:5000:6000* με την οποία προωθούμε όλη την επικοινωνία tcp μέσω της θύρας 5000 στην θύρα 6000.

Όταν λοιπόν έχουμε να κάνουμε με φυσικές συσκευές δεν χρειάζεται η προώθηση της επικοινωνίας με tcp σε κάποια άλλη θύρα. Δηλαδή και στον κώδικα μας πρέπει να έχουμε τις ίδιες θύρες και στον πελάτη και στον διακομιστή. Επίσης, βάζουμε την διεύθυνση του διακομιστή που είναι του τύπου 192.168.x.x. Όταν όμως έχουμε εικονικές συσκευές η διεύθυνση που ακούει ο διακομιστής είναι η τοπική. Μπορούμε δηλαδή να χρησιμοποιήσουμε την 127.0.0.1 ή και την 10.0.2.2. Ωστόσο, αν προσπαθήσει κανείς να συνδεθεί με την 127.0.0.1 δεν θα μπορέσει. Ο λόγος είναι ότι αυτή αναφέρεται στην συσκευή και όχι στον ίδιο τον server.

Για να κάνουμε λίγο πιο ενδιαφέρον το παιχνίδι κάθε φορά στην ίδια ερώτηση οι απαντήσεις εμφανίζονται με διαφορετική σειρά ώστε ο χρήστης να μην μπορεί να απομνημονεύσει την απάντηση. Αυτό παρουσιάζεται στον παρακάτω κώδικα :

```
      int[] btnViews = new int[] { R.id.button1, R.id.button2, R.id.button3, R.id.button4 };

List<Integer> intList = new ArrayList<Integer>(Arrays.asList(0,1,2,3));
                    Collections.shuffle(intList);
                    List<String> ansList = new ArrayList<String>(Arrays.asList(mssg[1], mssg[2], mssg[3], mssg[4]));


((Button)findViewById(btnViews[intList.get(0)])).setText(ansList.get(0));
```

```
                    ((View)
findViewById(btnViews[intList.get(0)])).setTag("wrong");


((Button)findViewById(btnViews[intList.get(1)])).setText(ansList.get(1));
                    ((View)
findViewById(btnViews[intList.get(1)])).setTag("wrong");


((Button)findViewById(btnViews[intList.get(2)])).setText(ansList.get(2));
                    ((View)
findViewById(btnViews[intList.get(2)])).setTag("wrong");


((Button)findViewById(btnViews[intList.get(3)])).setText(ansList.get(3));
                          ((View)
findViewById(btnViews[intList.get(3)])).setTag("right");
```

Στην ουσία αυτό που κάνουμε είναι να δημιουργούμε μια λίστα που αποτελείται από τους ακέραιους αριθμούς (int) 0,1,2,3 τοποθετημένους στην λογική τους σειρά. Η μεταβλητή btnViews είναι ένας πίνακας ακεραίων όπου έχουμε τοποθετήσει μέσα τις ακέραιες τιμές των κουμπιών της οθόνης όπως έχουν δοθεί αυτόματα από το σύστημα. Ανακατεύουμε αυτή την λίστα. Στην συνέχεια τοποθετούμε σε μια άλλη λίστα τις απαντήσεις. Από την πρώτη λίστα παίρνουμε τον αριθμό που βρίσκεται στην θέση μηδέν, το πρώτο στοιχείο του πίνακα, και ότι αριθμό πάρουμε σε εκείνο το κουμπί τοποθετούμε το πρώτο στοιχείο του άλλου πίνακα και ούτω καθεξής. Στην τελευταία απάντηση , που είναι και πάντα η σωστή τοποθετούμε μια σήμανση. Αυτό αποτελεί και τον έλεγχο για το αν ο χρήστης έχει δώσει την σωστή απάντηση.

Στον χρήστη παρουσιάζονται η ερώτηση και οι απαντήσεις στα κουμπιά. Αυτός καλείται να απαντήσει πατώντας κάποιο κουμπί και ανάλογα αν η απάντηση που έδωσε είναι σωστή ή λανθασμένη παίρνει και τους ανάλογους πόντους, δέκα για κάθε σωστή απάντηση και μείον πέντε για κάθε λανθασμένη. Παρατίθεται ο κώδικας για το πάτημα σε κάθε κουμπί από τον χρήστη.

```
if(view == answer1)
            {
                  if(view.getTag().toString().equalsIgnoreCase("right"))
                  {
                              sendMessageToServer("right");

      answer1.setBackgroundResource(R.drawable.green_btn);
                        delay();
                        sc = sc + 10;
                        score.setText("Your score is " + sc);
                  }
                  else
                  {
                        sendMessageToServer("wrong");

      answer1.setBackgroundResource(R.drawable.red_btn);
                        delay();
                        EnableButtons(false);
                        sc = sc - 5;
```

```
                    score.setText("Your score is " + sc);
                }
                }
```

Έχοντας πατήσει στην συγκεκριμένη περίπτωση το πρώτο κουμπί το πρώτο που ελέγχουμε είναι να δούμε τι σήμανση έχει. Αν έχει την σήμανση right, δηλαδή η απάντηση που έδωσε ο χρήστης είναι η σωστή τότε στέλνουμε στον διακομιστή το string “right” και αυτός πράττει ανάλογα, θα το εξετάσουμε παρακάτω. Επίσης, αν ο χρήστης έχει δώσει σωστή απάντηση τότε στο ήδη υπάρχων σκορ του θα προστεθούν 10 βαθμοί και το κουμπί θα αλλάξει το χρώμα του σε πράσινο.

Αντίθετα, αν η σήμανση που έχει είναι wrong τότε η απάντηση που έδωσε ο χρήστης είναι λανθασμένη, αφαιρούνται πέντε βαθμοί από το συνολικό σκορ του, το κουμπί θα γίνει κόκκινο και στον διακομιστή στέλνεται το string “wrong”.

Αν ο χρήστης έδωσε λανθασμένη απάντηση τότε αποκόβεται από την δυνατότητα να απαντήσει ξανά μέχρι να εμφανιστεί καινούργια ερώτηση. Για να το πετύχουμε αυτό χρησιμοποιούμε την παρακάτω μέθοδο :

```
public void EnableButtons(Boolean state)
        {
            answer1.setEnabled(state);
            answer2.setEnabled(state);
            answer3.setEnabled(state);
            answer4.setEnabled(state);
              }
```

Τα answer1, answer2, answer3, answer4 είναι τα ονόματα των κουμπιών και ανάλογα τι state θα βάλουμε , true ή false καθώς είναι μεταβλητή τύπου Boolean, ενεργοποιούνται ή απενεργοποιούνται τα κουμπιά.

Επιπλέον, με την εντολή `setBackgroundResource(R.drawable.green_btn);` το κουμπί θα παρέμενε πράσινο ή κόκκινο ανάλογα την απάντηση του χρήστη. Αυτό δεν πρέπει να γίνει. Θα πρέπει αφού γίνει ο έλεγχος το κουμπί να πάρει το φυσικό του χρώμα όταν εμφανιστεί καινούργια ερώτηση. Αυτό επιτυγχάνεται με την μέθοδο *delay() :*

```
public void delay()
       {
             final Handler handler = new Handler();
            Timer t = new Timer();
            t.schedule(new TimerTask() {
                    public void run()         {
                            handler.post(new Runnable() {
                                    public void run()            {

answer1.setBackgroundResource(R.drawable.blue_btn);

answer2.setBackgroundResource(R.drawable.blue_btn);

answer3.setBackgroundResource(R.drawable.blue_btn);

answer4.setBackgroundResource(R.drawable.blue_btn);
                                                                        }
                                                                        });
                                    }      },80);}
```

Δημιουργούμε ένα καινούργιο στιγμιότυπου τύπου Handler. Ένα Handler μας επιτρέπει να στείλουμε μηνύματα επεξεργασίας και αντικείμενα Runnable, κομμάτι κώδικα που μπορεί εκτελεστεί σε ίδιο ή σε διαφορετικό νήμα (thread) από αυτό που δουλεύουμε στην ουρά εκτέλεσης. Όταν δημιουργούμε ένα καινούργιο Handler είναι δεμένο με το thread που το δημιούργησε και παραδίδει σε αυτό τα μηνύματα και τα Runnable καθώς βγαίνουν από την ουρά εκτέλεσης.

Επίσης, δημιουργούμε ένα καινούργιο στιγμιότυπου τύπου Timer. Οι χρονοδιακόπτες (timers) οργανώνουν για εκτέλεση εργασίες που χρειάζονται να επαναληφθούν ή όχι. Κάθε timer έχει ένα νήμα στο οποίο οι διεργασίες εκτελούνται διαδοχικά. Όταν αυτό το νήμα είναι απασχολημένο με το να εκτελεί μια άλλη διεργασία, τότε έχουμε καθυστερήσεις.

Αυτό που κάνουμε εμείς εδώ είναι να καθυστερούμε εσκεμμένα την επαναφορά του φυσικού χρώματος στα κουμπιά για 80 milisecond. Αυτό δίνει την ψευδαίσθηση στον χρήστη της εναλλαγής των ερωτήσεων.

Το σημαντικότερο κομμάτι κώδικα όμως σε αυτό το σημείο είναι η αποστολή των string στον διακομιστή ανάλογα κάθε φορά την απάντηση που έχει δώσει ο χρήστης σε κάθε ερώτηση. Αυτό γίνεται με την μέθοδο *sendMessageToServer(String str).* Συγκεκριμένα :

```
public void sendMessageToServer(String str)
        {

              final String str1 = str;
              new Thread(new Runnable() {

                    @Override
                    public void run() {
                          PrintWriter out;
                          try {
                                out                    =                    new
PrintWriter(nsocket.getOutputStream());
                                out.println(str1);
                                Log.d("Message Sent", str1);
                                out.flush();
                          } catch (UnknownHostException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                                Log.d("Error", e.toString());
                          } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                                Log.d("Error", e.toString());
                          }
                    }
              }).start();
                          }
```

Όπως παρατηρεί κανείς δημιουργούμε ένα καινούργιο thread στο οποίο εκτελείται αυτή η μέθοδος η οποία παίρνει σαν όρισμα μια μεταβλητή τύπου string που είναι στην ουσία η λέξη που θέλουμε να στείλουμε. Δηλώνουμε την τοπική μεταβλητή out που είναι τύπου PrintWriter. Σε αυτό το σημείο πρέπει να κάνουμε σαφές πως γίνεται η επικοινωνία μεταξύ πελάτη – διακομιστή όσον αφορά τα string. Αφού ο πελάτης συνδεθεί σε ένα διακομιστή

για να αρχίσει η επικοινωνία θα πρέπει να χρησιμοποιήσουμε τις μεθόδους *getOutputStream()* και *getInputSream().* Την δεύτερη θα την εξετάσουμε στο κομμάτι του διακομιστή.

Αυτές οι δύο ανοίγουν stream πάνω στο persistent socket πάνω στα οποία μπορεί ο πελάτης να γράψει δεδομένα, στην περίπτωση μας string, και να τα στείλει στον διακομιστή. Με την εντολή *out.flush() «*καθαρίζουμε» το stream από εναπομείναντα δεδομένα έτσι ώστε ο διακομιστής να λάβει οτιδήποτε του έχει στείλει ο κάθε πελάτης. Εδώ αποκτά νόημα και η έννοια του persistent socket γιατί δεν χρειάζεται κάθε φορά να ανοίγουμε και να κλείνουμε διαφορετικά sockets ώστε να στείλουμε κάτι στον διακομιστή. Αυτό παραμένει συνέχεια ανοικτό μέχρι να το κλείσει ο χρήστης με κάποια ενέργεια του, δημιουργώντας πάνω του πολλαπλά stream.

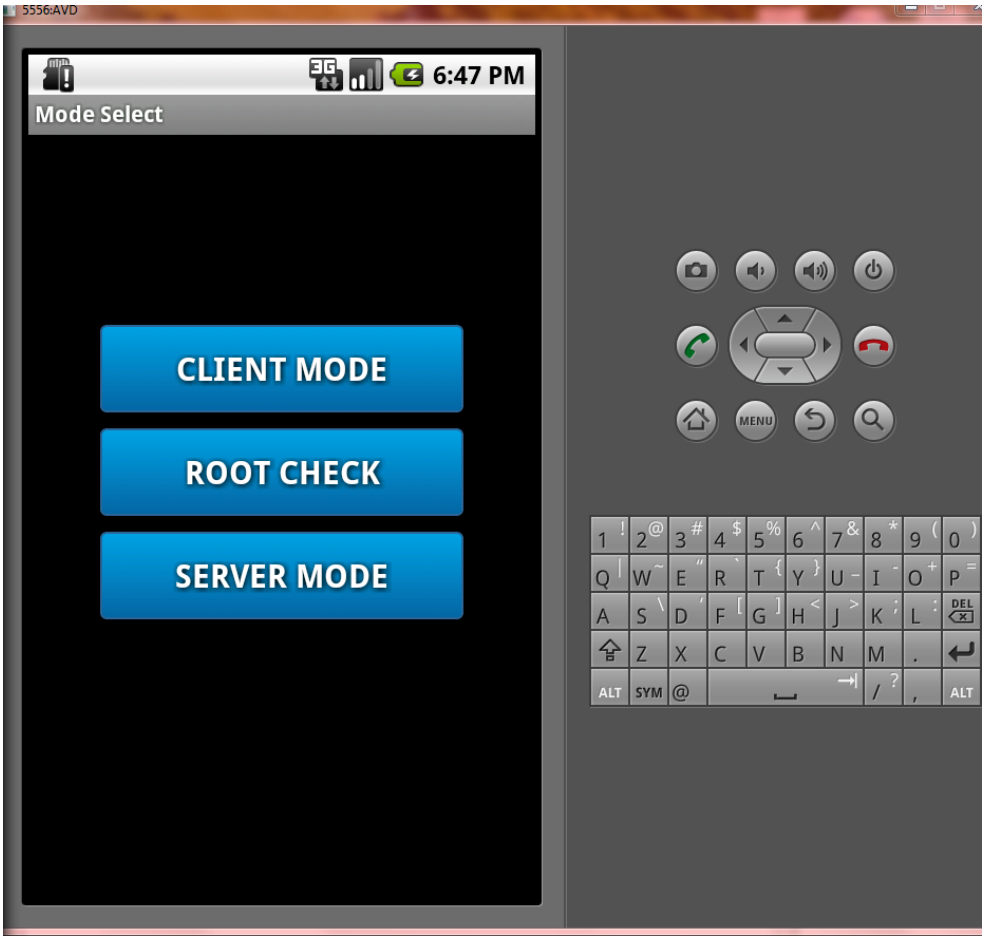
5.5. Έλεγχος δικαιωμάτων υπερχρήστη

Το routing είναι μια διαδικασία που επιτρέπει στους χρήστες smartphones, tablets, και άλλων συσκευών που τρέχουν το λειτουργικό σύστημα Android την επίτευξη προνομιακού ελέγχου (γνωστή ως "root πρόσβαση") στο υποσύστημα του Android. Το routing συχνά γίνεται με στόχο να ξεπεραστούν οι περιορισμοί που ορισμένοι κατασκευαστές υλικού βάζουν σε ορισμένες συσκευές, με αποτέλεσμα την δυνατότητα να μεταβάλει ή να αντικαθιστά ο χρήστης τις ρυθμίσεις συστήματος, να εκτελεί εξειδικευμένες εφαρμογές που απαιτούν δικαιώματα επιπέδου διαχειριστή, ή να εκτελέσει άλλες εργασίες που είναι απρόσιτες για έναν κανονικό χρήστη του Android. Το routing είναι ανάλογοο με το jailbreaking που υπάρχει σε συσκευές που τρέχουν το λειτουργικό σύστημα της Apple iOS ή το Sony PlayStation 3. Στο Android, το routing μπορεί επίσης να διευκολύνει την πλήρη αφαίρεση και αντικατάσταση του λειτουργικού συστήματος της συσκευής.

Καθώς το Android προήλθε από τον πυρήνα του Linux, το να κάνει κανείς routing μια συσκευή Android είναι μια πράξη παρόμοια με την πρόσβαση σε διοικητικά δικαιώματα σε Linux ή σε οποιοδήποτε άλλο σύστημα τύπου Unix όπως το FreeBSD ή το OS X.

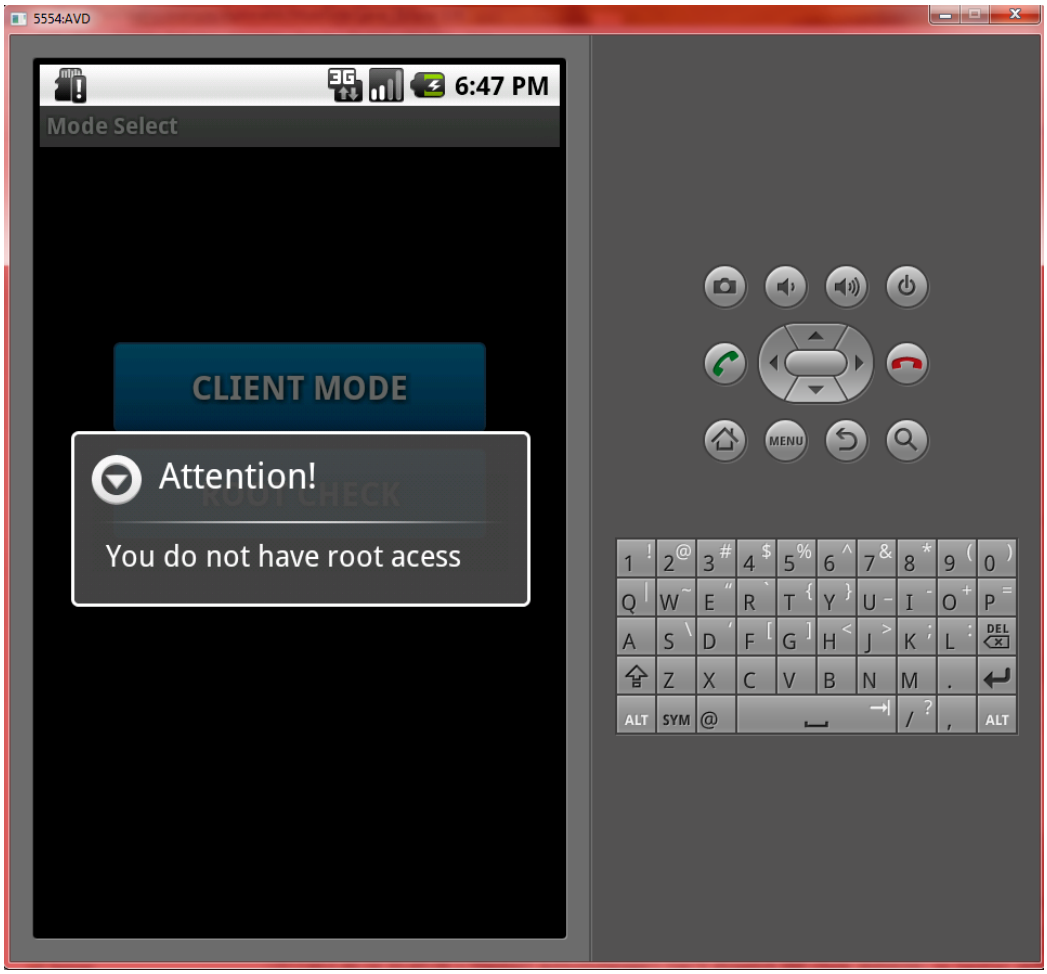
Η εφαρμογή απαιτεί από τον χρήστη να έχει δικαιώματα πρόσβασης υπερχρήστη για να μπορεί να κάνει την συσκευή του να συμπεριφέρεται σαν διακομιστής. Έτσι το κουμπί RootCheck ελέγχει για την ύπαρξη αυτών των δικαιωμάτων και αν υπάρχει η δυνατότητα πρόσβασης σε αυτά τότε μέσω της εφαρμογής superuser, μια εφαρμογή που την εγκαθιστά ο χρήστης στην προσπάθεια του να δώσει root access στην συσκευή του δίνονται δικαιώματα διαχειριστή στην εφαρμογή μας.

Έτσι εμφανίζεται αυτή η οθόνη :



«Εικόνα 43 : **Οθόνη που εμφανίζεται αν έχουμε δικαιώματα υπερχρήστη**»

Εάν η συσκευή μας δεν έχει τέτοια δικαιώματα τότε στον χρήστη εμφανίζεται ένα μήνυμα διαλόγου που τον ειδοποιεί για το γεγονός, Συγκεκριμένα :



«Εικόνα 44 : **Μήνυμα διαλόγου ειδοποίησης του χρήστη ότι δεν έχει δικαιώματα υπερχρήστη**»

Ο έλεγχος για το αν μια συσκευή είναι rooted ή όχι κρύβει πάρα πολλές παγίδες από την φύση της. Όπως είδαμε το να έχει ένας χρήστης root access στην συσκευή του ρου δίνει την δυνατότητα να αλλάξει τις ρυθμίσεις ακόμα και το λειτουργικό σύστημα και τον kernel που έχει η συσκευή του. Έτσι για παράδειγμα αν προσπαθήσουμε εκτελέσουμε στον κώδικά μας την εντολή *su, Process proc = Runtime.getRuntime ().exec ("su"),* και να ελέγξουμε τι επιστρέφει και ανάλογα να πράξουμε δεν είναι και ότι πιο σωστό υπάρχει.

Το αποτέλεσμα αυτής της εντολής, όποιο και αν είναι αυτό, σημαίνει ότι η πρόσβαση δεν επετράπη σε αυτήν την εφαρμογή για την συγκεκριμένη προσπάθεια. Ακόμα για να ψάχνουμε να βρούμε αν η εφαρμογή superuser, που αναφέραμε πιο πάνω είναι εγκατεστημένη και αυτό δεν θα αποτελούσε την ασφαλέστερη επιλογή γιατί κάποιος χρήστης μπορεί απλά να την έχει απεγκαταστήσει.

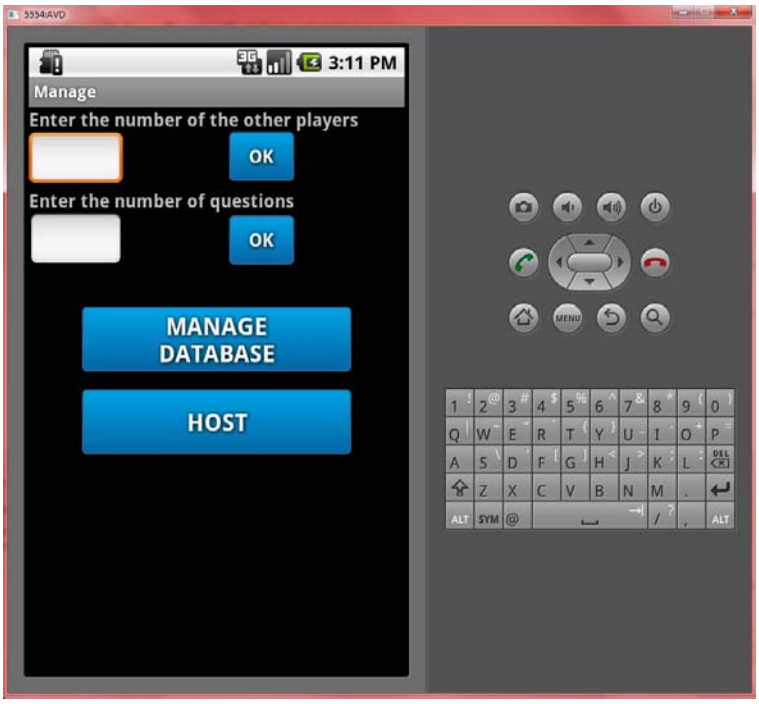
Κάθε διαφορετική συσκευή και κάθε διαφορετική έκδοση του λειτουργικού συστήματος Android έχει και διαφορετικό τρόπο απόκτησης δικαιωμάτων υπερχρήστη. Αλλάζοντας για παράδειγμα την ROM της συσκευής μας εγκαθιστούμε στο 80% των περιπτώσεων το BusyBox, μια εφαρμογή που έχει συνδυάσει πολλές λειτουργίες του Unix σε ένα εκτελέσιμο.

Ακόμη ένας τρόπος θα ήταν να ελέγχουμε την ύπαρξη του αλλά και πάλι δεν θα μπορούσαμε να είμαστε απολύτως σίγουροι. Η πιο διαδεδομένη και η πιο ασφαλής μέθοδος στους προγραμματιστές είναι η χρήση της βιβλιοθήκης RootTools.jar. Διανέμεται κάτω από την Apache Licence όπως και το λειτουργικό σύστημα και προσφέρει στους προγραμματιστές εργαλεία ανάπτυξης rooted εφαρμογών. Έτσι με την μέθοδο `RootTools.isAccessGiven()` Ελέγχουμε όλα τα παραπάνω. Δηλαδή δεν ελέγχει μόνο εάν η συσκευή είναι rooted αλλά καλεί την εντολή *su* για την εφαρμογή, ζητά άδεια και επιστρέφει true αν έχουν γίνει όλα με επιτυχία.

Αυτό που πρέπει να κάνει ο προγραμματιστής είναι να κατεβάσει από την σελίδα του το RootTools.jar και να το ενσωματώσει στην εφαρμογή του στον φάκελο lib όπως έχουμε δει σε προηγούμενο κεφάλαιο.

5.6. Επιλογή συμπεριφοράς διακομιστή (server mode)

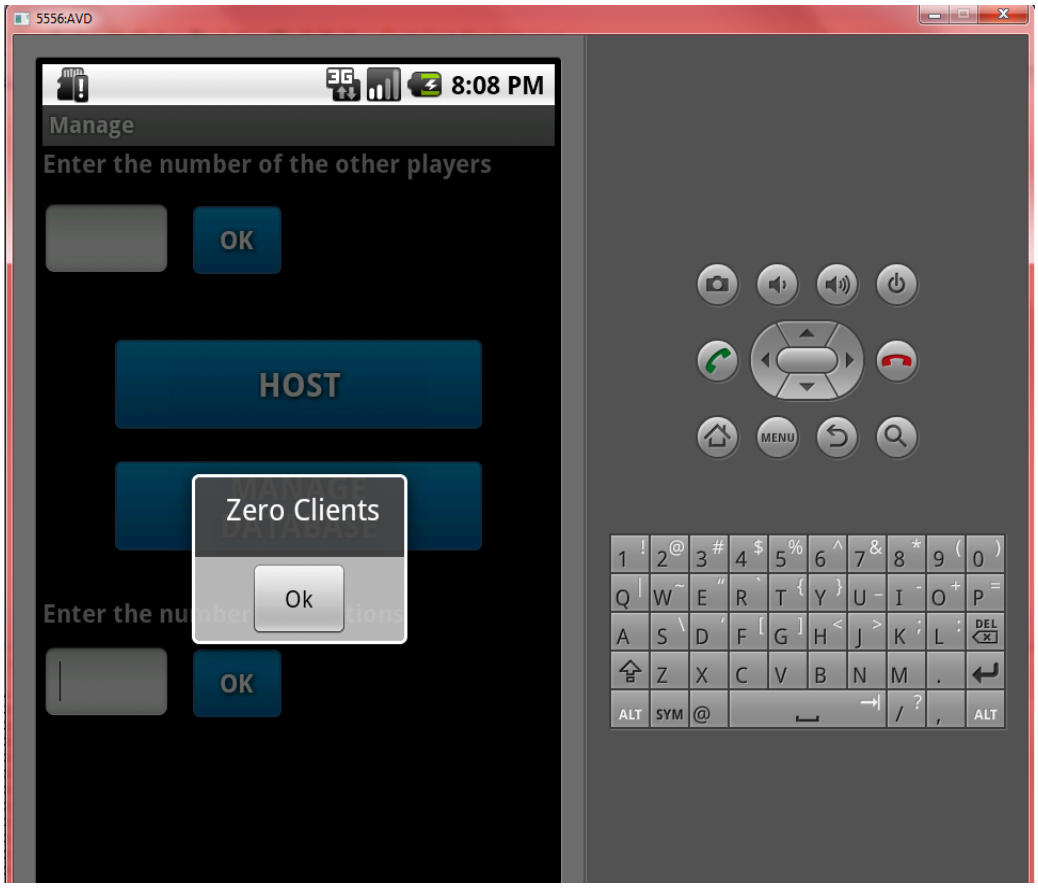
Σε αυτήν την οθόνη ο χρήστης μπορεί να διαλέξει το κινητό του να συμπεριφέρεται σαν διακομιστής και να περιμένει να πάρει μια διεύθυνση IP την οποία και θα διαμοιράσει ώστε οι υπόλοιπες συσκευές πελάτες να συνδεθούν σε αυτή.



«Εικόνα 45 : Οθόνη διακομιστή»

Αρχικά ο διακομιστής πρέπει να διαλέξει τον αριθμό των συσκευών που θα συνδεθούν σε αυτόν, μαζί με τον εαυτό του βέβαια. Αφού διαλέξει τον αριθμό πατήσει το κουμπί OK και μετά το κουμπί HOST είναι έτοιμος να

δεχτεί συνδέσεις. Αν καταλάθος πατηθεί το κουπί HOST χωρίς να ο χρήστης να προλάβει να βάλει τον αριθμό των πελατών τότε εμφανίζεται ένα παράθυρο διαλόγου που δεν επιτρέπει περαιτέρω ενέργειες στον χρήστη μέχρι να βάλει ένα αριθμό.



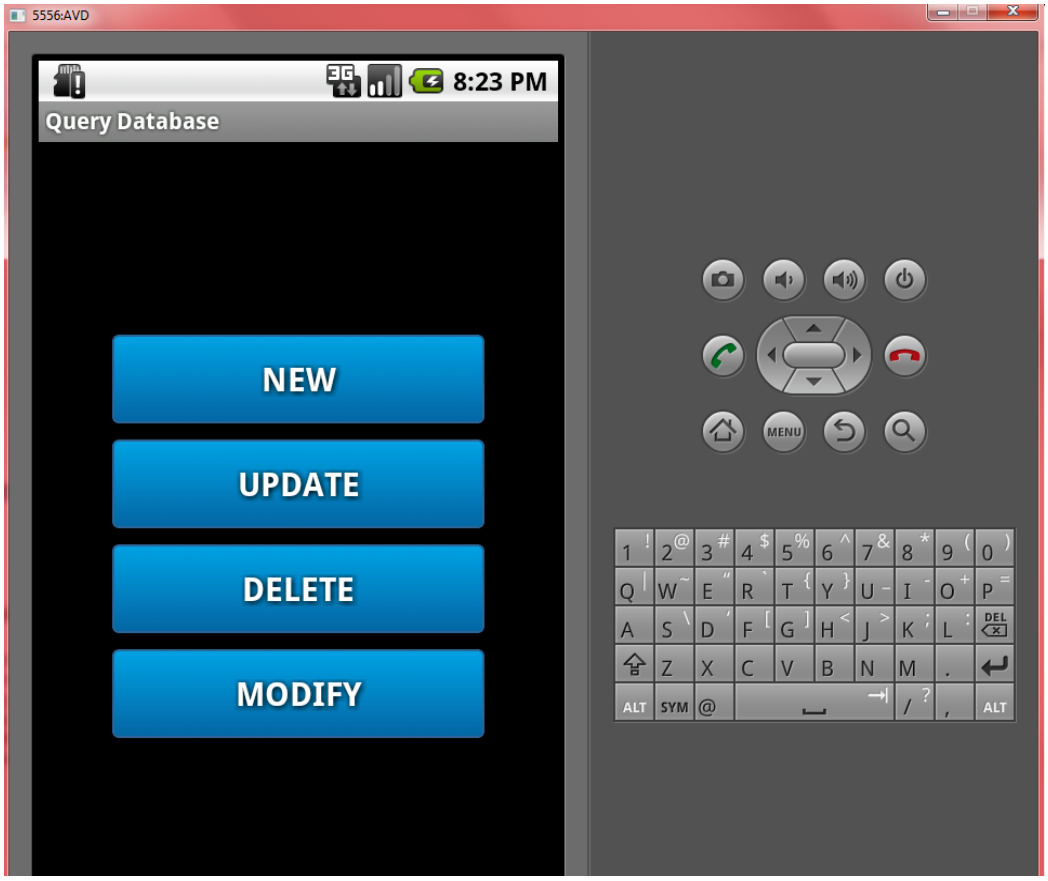
«Εικόνα 46 : **Μήνυμα διαλόγου με μηδενικό αριθμό πελατών»**

Επίσης ο χρήστης πρέπει να διαλέξει και τον αριθμό των ερωτήσεων με τον οποίο επιθυμεί να παίξει με ανάλογο μήνυμα να εμφανίζεται σε περίπτωση μηδενικής εισαγωγής.

Επιπλέον, ο χρήστης μπορεί να διαλέξει το κουμπί MANAGE DATABASE όπου εκεί μπορεί να διαχειριστεί την βάση δεδομένων που περιέχει τις ερωτήσεις και τις απαντήσεις καθώς και να αναβαθμίσει αυτή που έχει ήδη (Εικόνα 45).

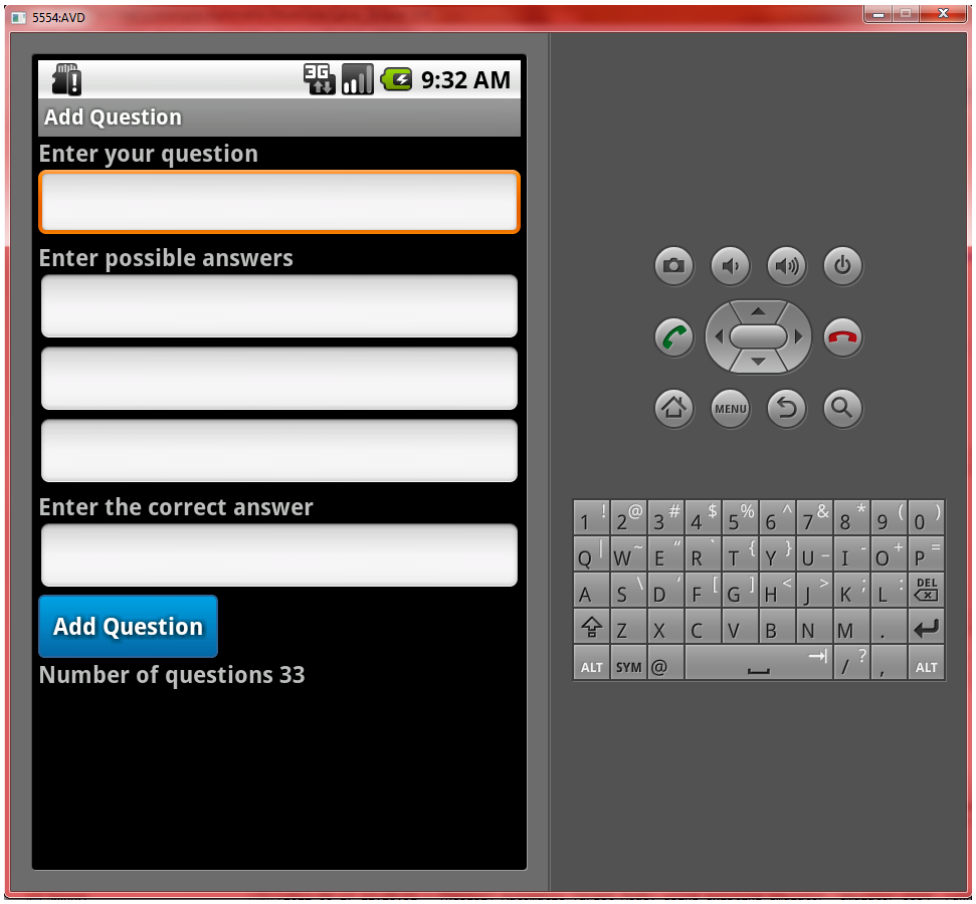
5.6.1. Δημιουργία νέας ερώτησης

Αν ο χρήστης επιλέξει το κουμπί MANAGE DATABASE θα εισέλθει σε μια οθόνη όπου μπορεί να διαχειριστεί την βάση δεδομένων. Όταν εγκαθιστά για πρώτη φορά την εφαρμογή τότε η βάση που δημιουργείται περιέχει μηδέν ερωτήσεις. Ο χρήστης τότε έχει δύο επιλογές, ή να κατεβάσει την επίσημη βάση από τους servers του Πανεπιστημίου Δυτικής Μακεδονίας ή να φτιάξει την δικιά του βάση με τις δικές του ερωτήσεις και απαντήσεις.



«Εικόνα 47 : **Οθόνη διαχείρισης βάσης δεδομένων»**

Διαλέγοντας στην συνέχεια από το παραπάνω μενού το κουμπί NEW ο χρήστης μπαίνει στην φόρμα δημιουργίας νέας ερώτησης η οποία φαίνεται στην παρακάτω εικόνα. Ο χρήστης έχει μπροστά του μια φόρμα όπου μπορεί να προσθέσει την ερώτηση και τις απαντήσεις που θέλει, βάζοντας την σωστή απάντηση τελευταία στο κατάλληλο πεδίο. Στο τέλος της οθόνης φαίνεται και ο αριθμός των ερωτήσεων που έχει ο χρήστης στην συσκευή του.



«Εικόνα 48 : Φόρμα προσθήκης νέας ερώτησης.»

Η οθόνη αυτή είναι η κλάση AddQuestion. Όταν ξεκινάει η activity δηλώνουμε πρώτα την διεπαφή χρήστη (user interface) με την εντολή *setContentView(R.layout.addquestion)*. Σε αυτό το αρχείο, έχουμε δηλώσει τα γραφικά στοιχεία τις οθόνης. Τα EditText, τα TextView και το κουμπί. Έτσι με την σειρά ο χρήστης βάζει την ερώτηση που επιθυμεί, στην συνέχεια τις πιθανές απαντήσεις και τέλος την σωστή απάντηση. Πατώντας το κουμπί Add Question η ερώτηση προστίθεται στην βάση καλώντας την παρακάτω μέθοδο:

```
public void btnAddQuest_Click(View view)
	{
		boolean ok = true;
		try
		{

			String question = Question.getText().toString();
			String answer1 = Answer1.getText().toString();
			String answer2 = Answer2.getText().toString();
			String answer3 = Answer3.getText().toString();
			String answer4 = Answer4.getText().toString();
			Question nquest = new
Question(question,answer1,answer2,answer3,answer4);  //  Στιγμιότυπο  του
constructor Question
			dbHelper.AddQuestion(nquest);
```

```
		}
		catch(Exception ex)
		{
			ok = false;
			CatchError(ex.toString());
		}
		finally
		{
			if(ok)
			{
			Alerts.ShowQuestAddedAlert(this); // Παράθυρο διαλόγου
				txtQuests.setText("Number of Questions
"+String.valueOf(dbHelper.getQuestionCount()));
			}
		}
		}
```

Παίρνουμε λοιπόν οτιδήποτε έχει βάλει ο χρήστης στα κατάλληλα πεδία και τα μετατρέπουμε σε μεταβλητές τύπου string για να μπορούμε να τα διαχειριστούμε. Χρησιμοποιώντας τον constructor Question που έχουμε ορίσει σε ξεχωριστό αρχείο δημιουργούμε μια μεταβλητή τέτοιου τύπου όπου τοποθετούμε τα στοιχεία που έχει εισάγει ο χρήστης.

Σε αυτό το σημείο πρέπει να αναφερθούμε σε δύο αρχεία που ανήκουν στο project και έχουν να κάνουν με την βάση δεδομένων. Αυτά είναι ο constructor Question και το αρχείο DatabaseHelper.

```
public class Question {

	int _id;
	String _question;
	String _answer1;
	String _answer2;
	String _answer3;
	String _answer4;

	public Question(String Answer1, String Answer2, String Answer3,
String Answer4)
	{
		this._answer1 = Answer1;
		this._answer2 = Answer2;
		this._answer3 = Answer3;
		this._answer4 = Answer4;
	}
	public Question(String Question,String Answer1)
	{
		this._question = Question;
		this._answer1 = Answer1;
	}

	public Question(String Question, String Answer1, String Answer2,
String Answer3, String Answer4)
	{
		this._question = Question;
		this._answer1 = Answer1;
		this._answer2 = Answer2;
		this._answer3 = Answer3;
		this._answer4 = Answer4;
```

```
	}

	public int getID()
	{
		return this._id;
	}
	public void SetID(int ID)
	{
		this._id = ID;
	}

	public String getQuestion()
	{
		return this._question;
	}

	public String getAnswer1()
	{
		return this._answer1;
	}
	public String getAnswer2()
	{
		return this._answer2;
	}
	public String getAnswer3()
	{
		return this._answer3;
	}
	public String getAnswer4()
	{
		return this._answer4;
	}

	public void setQuestion(String question)
	{
		this._question = question;
	}
	public void setAnswer1(String Answer1)
	{
		this._answer1 = Answer1;
	}
	public void setAnswer2(String Answer2)
	{
		this._answer2 = Answer2;
	}
	public void setAnswer3(String Answer3)
	{
		this._answer3 = Answer3;
	}
	public void setAnswer4(String Answer4)
	{
		this._answer4 = Answer4;
	}

}
```

Αποτελεί το καλούπι μας και περιέχει τα δεδομένα που θα αποθηκεύσουμε στην βάση δεδομένων και θα εμφανίσουμε στο user interface. Κάθε φορά λοιπόν που θέλουμε να τροποποιήσουμε δεδομένα στην βάση, είτε δηλαδή να προσθέσουμε, είτε να αφαιρέσουμε θα πρέπει να αναφερόμαστε σε αυτόν τον constructor γιατί αλλιώς δεν μπορεί η εφαρμογή μας να αναγνωρίσει το τι διάλεξε ο χρήστης για ερώτηση και τι για απαντήσεις.

Στην συνέχει με την εντολή *dbHelper.AddQuestion(nquest)* προσθέτουμε την ερώτηση με τις απαντήσεις στην βάση. Η global μεταβλητή dbHelper είναι τύπου DatabaseHelper που τον παρουσιάζουμε παρακάτω.

```
public class DatabaseHelper extends SQLiteOpenHelper {

	static final String dbName="DATABASE";
	static final String questionTable="THESIS";
	static final String colID = "_id";
	static final String colQuestion = "QUESTIONS";
	static final String colAnswer1 = "ANSWER1";
	static final String colAnswer2 = "ANSWER2";
	static final String colAnswer3 = "ANSWER3";
	static final String colAnswer4 = "ANSWER4";

	public DatabaseHelper(Context context) {
		super(context, dbName, null,33);

		// TODO Auto-generated constructor stub
	}

	@Override
	public void onCreate(SQLiteDatabase db) {
		// TODO Auto-generated method stub

		db.execSQL("CREATE TABLE "+questionTable+" ("+colID+" INTEGER
PRIMARY KEY AUTOINCREMENT, "+
				colQuestion+" TEXT NOT NULL, "+colAnswer1+" TEXT
NOT NULL,"+colAnswer2+" TEXT NOT NULL,"+colAnswer3+" TEXT NOT
NULL,"+colAnswer4+" TEXT NOT NULL);");
	}

	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
		// TODO Auto-generated method stub

		db.execSQL("DROP TABLE IF EXISTS "+questionTable);
		onCreate(db);
		}
```

Αυτή η κλάση είναι υπεύθυνη για την δημιουργία της βάσης και χειρίζεται όλη την επικοινωνία χρήστη – βάσης δεδομένων. Η μέθοδος *onUpgrade* απλά θα διαγράψει όλα τα υπάρχοντα δεδομένα και θα δημιουργήσει ξανά τον πίνακα. Επίσης εδώ δίνουμε το όνομα στην βάση και στις στήλες που θα έχουμε σαν constants μεταβλητές. Η βάση λοιπό αποτελείται από πέντε στήλες. Την στήλη των ερωτήσεων , τρεις στήλες με τις πιθανές απαντήσεις και την στήλη της σωστής απάντησης.

Στην συνέχεια έχουμε τον κώδικα με τον οποίο η ερώτηση που έχει βάλει ο χρήστης προστίθεται στην βάση. Συγκεκριμένα η μέθοδος *AddQuestion(Question nquest).*

```
void AddQuestion(Question nquest)
	{
		SQLiteDatabase db= this.getWritableDatabase();
		ContentValues cv = new ContentValues();
		cv.put(colQuestion, nquest.getQuestion());
		cv.put(colAnswer1, nquest.getAnswer1());
		cv.put(colAnswer2, nquest.getAnswer2());
		cv.put(colAnswer3, nquest.getAnswer3());
		cv.put(colAnswer4, nquest.getAnswer4());

		db.insert(questionTable, colAnswer1, cv);
		db.close();}
```

Με την μέθοδο *getWritableDatabase()* παίρνουμε δικαιώματα εγγραφής και ανάγνωσης (read and write) στην βάση μας. Χωρίς αυτά θα παίρναμε ένα Runtime Exception. Επιπλέον δημιουργούμε μια μεταβλητή τύπου ContentValues και την αρχικοποιούμε δημιουργώντας ένα καινούργιο στιγμιότυπο. Το ContentValues είναι μια κλάση που μας επιτρέπει να αποθηκεύσουμε ένα σύνολο από δεδομένα που ο ContentResolver μπορεί να επεξεργαστεί. Ο ContentResolver είναι μια διεπαφή που μπορεί να διαχειριστεί δεδομένα μιας διεργασίας με τον κώδικα που τρέχει σε μια άλλη.

Έτσι αποθηκεύουμε στην μεταβλητή cv τις τιμές των μεθόδων *getQuestion(), getAnswer1(), getAnswer2(), getAnswer3(), getAnswer4(),* που επιστρέφουν την ερώτηση και τις τέσσερις απαντήσεις αντίστοιχα για κάθε στιγμιότυπο που δημιουργείται. Τέλος, με την μέθοδο insert*(tablename, nullHack, values)* βάζουμε τις καινούργιες γραμμές στον πίνακα που έχουμε δημιουργήσει. Όπως μπορεί να παρατηρήσει κανείς στα ορίσματα τις *insert* έχουμε το όνομα του πίνακα που χρησιμοποιούμε, τις τιμές που θέλουμε να εισάγουμε και ένα τρίτο όρισμα που αποτελεί την είσοδο στον πίνακα εάν υπάρχει μηδενική εισαγωγή. Η SQLite δεν επιτρέπει την μηδενική εισαγωγή μιας γραμμής χωρίς μια τιμή έστω σε μια στήλη.

Όπως αναφέραμε και στην αρχή αυτής της ενότητας, όταν δημιουργείται η οθόνη που περιέχει την φόρμα προσθήκης νέας ερώτησης στο κάτω μέρος της αναγράφεται ο αριθμός των ερωτήσεων. Το μέτρημα όλων αυτών γίνεται με την μέθοδο *getQuestioncount()* που αναλύεται ως εξής.

```
int getQuestionCount()
	{
		SQLiteDatabase db = this.getWritableDatabase();
		Cursor cur= db.rawQuery("Select * from "+questionTable, null);
		int x = cur.getCount();
		cur.close();
		return x;
		 }
```

Παρατηρούμε ότι είναι μια μέθοδος int δηλαδή επιστρέφει ένα ακέραιο αριθμό που είναι ο αριθμός όλων των ερωτήσεων. Όπως και προηγουμένως

αφού έχουμε να κάνουμε με στοιχεία της βάσης πρέπει να πάρουμε και την κατάλληλη άδεια. Σε αυτήν την μέθοδο συναντάμε για πρώτη φορά το στοιχείο Cursor.

Ένας Cursor είναι μια δομή ελέγχου που επιτρέπει την μετάβαση στις εγγραφές σε μια βάση δεδομένων. Οι cursor διατελούν και κάποιες εργασίες σε συνδυασμό με το να διασχίζει μια βάση, όπως την προσθήκη, διαγραφή και ανάκτηση εγγραφών στην βάση. Χρησιμοποιούνται συνήθως από τους προγραμματιστές για την επεξεργασία μεμονωμένων γραμμών που επιστρέφονται από τα queries σε μια βάση.

Στο κώδικά μας λοιπόν χρησιμοποιούμε την μέθοδο rawQuery που μας επιτρέπει να βάλουμε ένα ερώτημα στην βάση σε μορφή MySQL και χρησιμοποιούμε ένα Cursor για να διαχειριστούμε αυτό που θα επιστρέψει. Έτσι επιστρέφουμε τα πάντα από τον πίνακα μας και αποθηκεύουμε σε μια μεταβλητή x τύπου int ότι επιστρέψει η μέθοδος *getCount()*. Αυτή η μέθοδος επιστρέφει τον αριθμό των γραμμών του πίνακα όποτε είναι ακριβώς ότι χρειαζόμαστε. Τέλος, κλείνουμε τον cursor, μια ενέργεια που πρέπει ο προγραμματιστής να μην ξεχνάει γιατί οδηγεί σε σφάλμα και ξαφνικό τερματισμό της εφαρμογής, και επιστρέφουμε την τιμή του x.

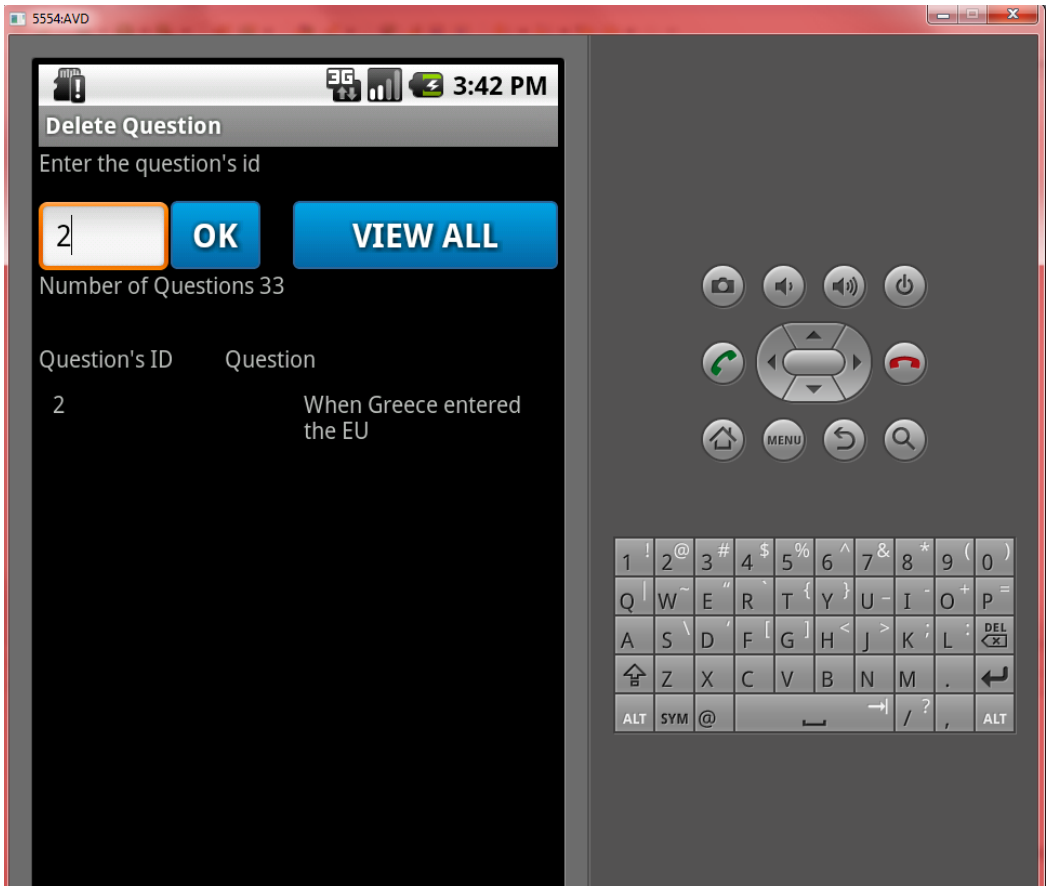
Έτσι όταν ξεκινά η activity AddQuestion εμφανίζουμε τον αριθμό των ερωτήσεων στην οθόνη με τον εξής τρόπο :

```
public void onStart()
	{
		try
		{
		super.onStart();
		dbHelper=new DatabaseHelper(this);

	txtQuests.setText(txtQuests.getText()+String.valueOf(dbHelper.getQue
stionCount()));
		//never close cursor
		}
		catch(Exception ex)
		{
			CatchError(ex.toString());
		}
		}
```

Δημιουργώντας ένα στιγμιότυπο της κλάσης DatabaseHelper και μετατρέποντας τον αριθμό int που επιστρέφει σε string το εμφανίζουμε με την βοήθεια του widget TextView.

5.6.2. Διαγραφή ερώτησης

Από το μενού της Εικόνας 47 μπορούμε να διαλέξουμε την διαγραφή μιας ερώτησης που δεν την γνωρίζουμε ή μιας στην οποία έχουμε βάλει λάθος στοιχεία. Επιλέγοντας, λοιπόν, το κουμπί DELETE τότε ο χρήστης οδηγείται στην παρακάτω οθόνη.



«Εικόνα 49 : Οθόνη διαγραφής ερώτησης»

Ο χρήστης, στην παραπάνω οθόνη, μπορεί να επιλέξει το κουμπί VIEW ALL που θα του επιτρέψει να δει όλες τις ερωτήσεις που έχει η βάση εκείνη την στιγμή. Αυτό γίνεται με την μέθοδο *ViewAll_Click(View view) :*

```
public void ViewAll_Click(View view)
	{
		try
		{
		Cursor cur = dbHelper.getAllQuestions();
		startManagingCursor(cur);
		String [] from = new String [] {DatabaseHelper.colID,
DatabaseHelper.colQuestion};
		int [] to = new int [] {R.id.colID,R.id.colName};
		SimpleCursorAdapter sca = new SimpleCursorAdapter(this,
R.layout.gridrow, cur, from, to);
		grid.setAdapter(sca);
		}
		catch(Exception e)
		{
			CatchError(e.toString());
		}
	}
```

Χρησιμοποιούμε την μέθοδο getAllQuestions που βρίσκεται στην κλάση DatabaseHelper και συντάσσεται ως εξής :

```
Cursor getAllQuestions()
	{
		SQLiteDatabase db=this.getWritableDatabase();
		Cursor cur = db.rawQuery("Select "+colQuestion+", "+colID+"
from "+questionTable, new String [] {});
			return cur;

	}
```

Χρησιμοποιούμε πάλι το στοιχείο cursor για να διασχίσουμε την βάση και, έχοντας πάρει την κατάλληλη άδεια χρησιμοποιούμε ένα ερώτημα MySQL που επιστρέφει όλες τις εγγραφές από τις στήλες colQuestion και colID όπου έχουμε αποθηκεύσει την ερώτηση και τον αύξοντα αριθμό της. Επιστρέφουμε τον cursor για να μπορούμε να διαχειριστούμε και πάλι τα αποτελέσματα της αναζήτησης που κάναμε.

Στην συνέχεια δημιουργούμε ένα πίνακα string με στοιχεία του το ID και την ερώτηση που έχει επιστρέψει ο cursor και ένα πίνακα int με στοιχεία του τα γραφικά στοιχεία της οθόνης όπου θα τα εμφανίσουμε. Με τον constructor SimpleCursorAdapter τοποθετούμε τις στήλες που έχει επιστρέψει ο cursor στο κατάλληλο GridView που έχουμε ορίσει στο xml αρχείο, στην συγκεκριμένη περίπτωση το gridrow.xml και στην συνέχεια συνδέουμε τα στοιχεία με την μέθοδο setAdpater για να τα εμφανίσουμε στην οθόνη. Το GridView είναι ένα τρόπος απεικόνισης των γραφικών στοιχείων στην οθόνη, όπως είναι το ListView, που παραθέτει τα αντικείμενα σε ένα πλέγμα.

Μια δεύτερη επιλογή του χρήστη είναι να εισάγει το ID της ερώτησης που θέλει και με το κουμπί OK θα εμφανιστεί μόνο η συγκεκριμένη ερώτηση. Αυτό γίνεται με την μέθοδο *LoadGrid()* που χρησιμοποιεί την σειρά της την μέθοδο getQuestion(String id) που την έχουμε ορίσει στην κλάση DatabaseHelper. Συγκεκριμένα :

```
public Cursor getQuestion(String id)
	{

		SQLiteDatabase db = this.getReadableDatabase();
		String [] columns = new String[]{colID, colQuestion};
		Cursor c = db.query(questionTable, columns, colID+"=?", new
String[]{id}, null, null, null);
		c.moveToFirst();
		return c;
		}
```

Είναι παρόμοια μέθοδος με τις προηγούμενες με την διαφορά ότι εδώ χρησιμοποιούμε ένα διαφορετικό τρόπο να θέσουμε το ερώτημα στην βάση μας. Χρησιμοποιούμε την μέθοδο *query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)* η οποία δέχεται σαν ορίσματα το όνομα του πίνακα, μια λίστα από τις στήλες που θα επιστρέψει, και στην συνέχεια χρησιμοποιεί κάποια φίλτρα για να μειώσει τις στήλες που θα επιστραφούν ανάλογα με αυτά της MySQL. Στην συνέχεια χρησιμοποιούμε την μέθοδο *moveToFirst()* για να σιγουρευτούμε ότι ο cursor που επιστρέφεται δείχνει στην πρώτη καταχώρηση. Έτσι συνεχίζουμε με την μέθοδο *LoadGrid :*

```
public void LoadGrid()
    {
```

```
        dbHelper = new DatabaseHelper(this);
        try
        {
                String id = String.valueOf(ID.getText());
                Cursor cur = dbHelper.getQuestion(id);
                startManagingCursor(cur);
                String [] from = new String [] {DatabaseHelper.colID,
DatabaseHelper.colQuestion};
                int [] to = new int [] {R.id.colID,R.id.colName};
                SimpleCursorAdapter sca = new SimpleCursorAdapter(this,
R.layout.gridrow, cur, from, to);
                grid.setAdapter(sca);
                txtNoQuest.setText("Number of Questions
"+String.valueOf(dbHelper.getQuestionCount()));
        }
        catch(Exception ex)
        {
                AlertDialog.Builder b=new AlertDialog.Builder(this);
                b.setMessage(ex.toString());
                b.show();
        }
            }
```

Όπως πάντα δημιουργούμε ένα στιγμιότυπο του DatabaseHelper για να συνδεθούμε με την βάση μας γιατί αλλιώς θα παίρναμε μια NullPointerException καθώς δεν θα υπήρχε βάση να διαχειριστούμε. Μετατρέπουμε το ID που έβαλε ο χρήστης σε string και το χρησιμοποιούμε σαν όρισμα στην *getQuestion()* για να επιστρέψει την συγκεκριμένη ερώτηση. Όπως και παραπάνω τοποθετούμε την ερώτηση και τον αύξοντα αριθμό της στο πλέγμα με την μέθοδο *SimpleCursorAdapter().* Επίσης με την μέθοδο *getQuestionCount()* που αναλύθηκε παραπάνω ενημερώνουμε πάντα τον χρήστη για το πόσες ερωτήσεις υπάρχουν στην βάση.

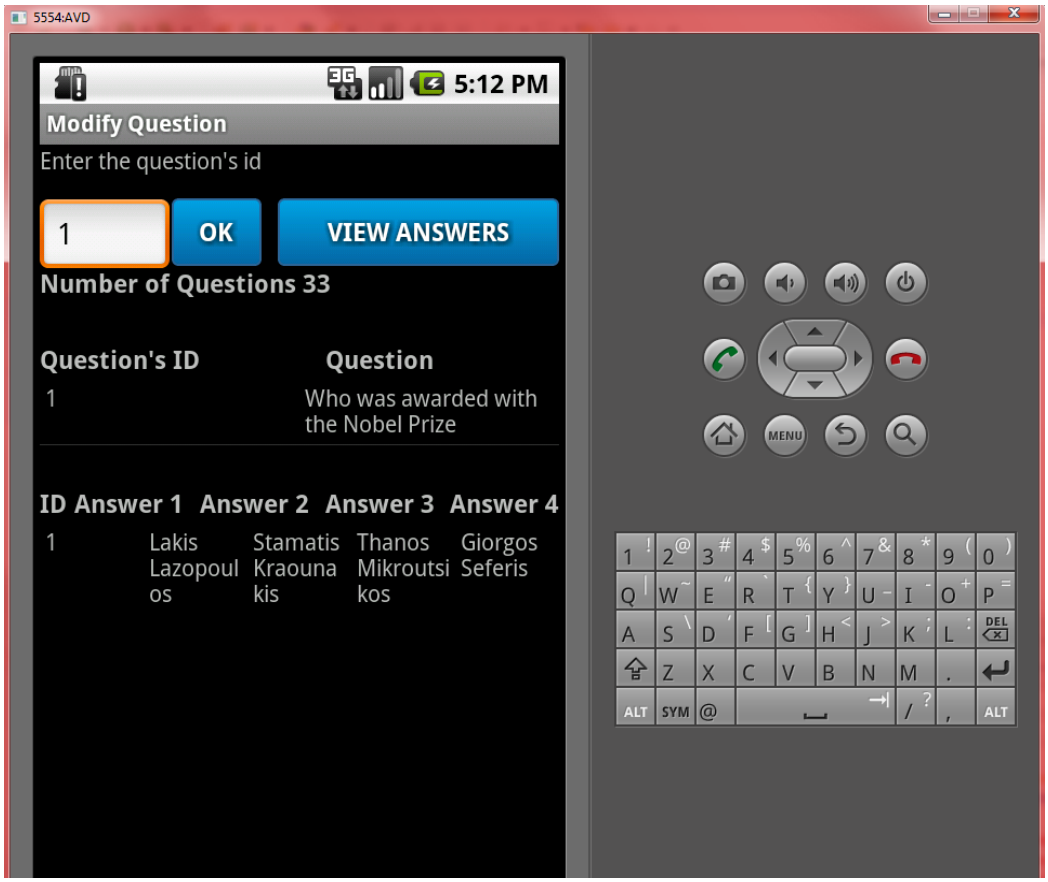
Όταν ο χρήστης εμφανίσει την ερώτηση που θέλει μπορεί να την διαλέξει κάνοντας ένα απλά κλικ πάνω της και θα εμφανιστεί ένα παράθυρο διαλόγου με τις επιλογές Delete και Cancel. Διαλέγοντας την επιλογή Delete η ερώτηση διαγράφεται χρησιμοποιώντας την μέθοδο DeleteQusetion(Question nquest).

```
public void DeleteQuestion(Question nquest)
       {
                SQLiteDatabase db = this.getWritableDatabase();
                db.delete(questionTable,colID+"=?", new String []
{String.valueOf(nquest.getID())});
                db.close();
                }
```

Αφού έχουμε πάρει δικαιώματα με τον γνωστό πλέον τρόπο χρησιμοποιούμε την μέθοδο delete*(String table, String whereClause, String[] whereArgs)* που παρέχεται από την κλάση *SQLiteOpenHelper.* Δέχεται σαν ορίσματα το όνομα του πίνακα και την στήλη που θα διαγράψει. Άμα βάλουμε την τιμή Null θα διαγράψει όλες τις στήλες. Ο τρόπος που έχουμε βάλει το στοιχείο colID σημαίνει ότι περιμένει την εισαγωγή του από τον χρήστη για να το διαγράψει.

5.6.3. Τροποποίηση ερώτησης

Από το μενού της Εικόνας 47 φαίνεται μια ακόμα επιλογή για τον χρήστη, αυτή της τροποποίησης μιας ερώτησης καθώς υπάρχει μεγάλη δυνατότητα λάθους κατά την εισαγωγή των ερωτήσεων. Έτσι όταν ο χρήστης διαλέξει το κουμπί MODIFY θα μεταβεί στην παρακάτω οθόνη :



«Εικόνα 50 : Οθόνη τροποποίησης ερώτησης»

Αφού ο χρήστης επιλέξει το ID της ερώτησης που επιθυμεί με τον ίδιο τρόπο όπως και στην διαγραφή ερώτησης με το πάτημα του κουμπιού OK εμφανίζεται η ερώτηση και το ID. Μπορεί στην συνέχεια να πατήσει το κουμπί VIEW ANSWERS και να δει όλες τις απαντήσεις της συγκεκριμένης ερώτησης. Αυτό γίνεται με την μέθοδο *LoadAnswerList()* :

```
public void LoadAnswerList()
    {
       dbHelper = new DatabaseHelper(this);
       try
       {
              String id = String.valueOf(ID.getText());
              Cursor cur = dbHelper.getAnswers(id);
              startManagingCursor(cur);
```

```
            String [] from = new String []{DatabaseHelper.colID,
DatabaseHelper.colAnswer1,                    DatabaseHelper.colAnswer2,
DatabaseHelper.colAnswer3, DatabaseHelper.colAnswer4};
            int [] to = new int [] {R.id.colID, R.id.colAnswer1,
R.id.colAnswer2, R.id.colAnswer3, R.id.colAnswer4};
            SimpleCursorAdapter sca = new SimpleCursorAdapter(this,
R.layout.answerlist, cur, from, to);
            answerlist.setAdapter(sca);
      }
      catch(Exception ex)
      {
            AlertDialog.Builder b=new AlertDialog.Builder(this);
            b.setMessage(ex.toString());
            b.show();
      }
          }
```

Στην ουσία δεν κάνουμε κάτι διαφορετικό από την προηγούμενη ενότητα. Στο μόνο που διαφέρει αυτή η μέθοδος είναι ότι χρησιμοποιεί την μέθοδο *getAnswers(id)* που είναι παρόμοια με την προηγούμενη μέθοδο *getQuestion(id):*

```
public Cursor getAnswers(String id)
       {

            SQLiteDatabase db = this.getReadableDatabase();
            String [] columns = new String[]{colID, colAnswer1,
colAnswer2, colAnswer3, colAnswer4};
            Cursor c = db.query(questionTable, columns, colID+"=?", new
String[]{id}, null, null, null);
            c.moveToFirst();
            return c;
       }
```

Σε αυτήν την περίπτωση χρειαζόμαστε τις στήλες με τις απαντήσεις και την στήλη με το ID οπότε η σύνταξη της query() θα αλλάξει.

Διαλέγοντας λοιπόν ο χρήστης την ερώτηση που θέλει μπορεί είτε να την τροποποιήσει είτε να τροποποιήσει τις απαντήσεις της. Εμφανίζεται ένα παράθυρο διαλόγου με τα πεδία της ερώτησης και των απαντήσεων και δύο επιλογές, η επιλογή Modify που καλεί τις μεθόδους UpdateQuestion ή UpdateAnswers ανάλογα με το τι θέλει ο χρήστης να αλλάξει και η επιλογή Cancel που επιστρέφει τον χρήστη στην προηγούμενη οθόνη.

```
public int UpdateQuestion(Question nquest)
       {
            SQLiteDatabase db = this.getWritableDatabase();
            ContentValues cv = new ContentValues();

                  cv.put(colQuestion, nquest.getQuestion());
                  return db.update(questionTable, cv, colID+"=?", new
String []{String.valueOf(nquest.getID())});

       }

public int UpdateAnswers(Question nquest)
       {
```

```
                SQLiteDatabase db = this.getWritableDatabase();
                ContentValues cv = new ContentValues();

                      cv.put(colAnswer1, nquest.getAnswer1());
                      cv.put(colAnswer2, nquest.getAnswer2());
                      cv.put(colAnswer3, nquest.getAnswer3());
                      cv.put(colAnswer4, nquest.getAnswer4());
                      return  db.update(questionTable,  cv,  colID+"=?",  new
String []{String.valueOf(nquest.getID())});

        }
```

Στην πραγματικότητα δεν διαφέρουν και πολύ από τα προηγούμενα με την διαφορά ότι εδώ αντί να χρησιμοποιήσουμε την μέθοδο *insert()* ή *delete()* χρησιμοποιούμε την *update()* που αλλάζει τις καταχωρήσεις μας στην βάση.

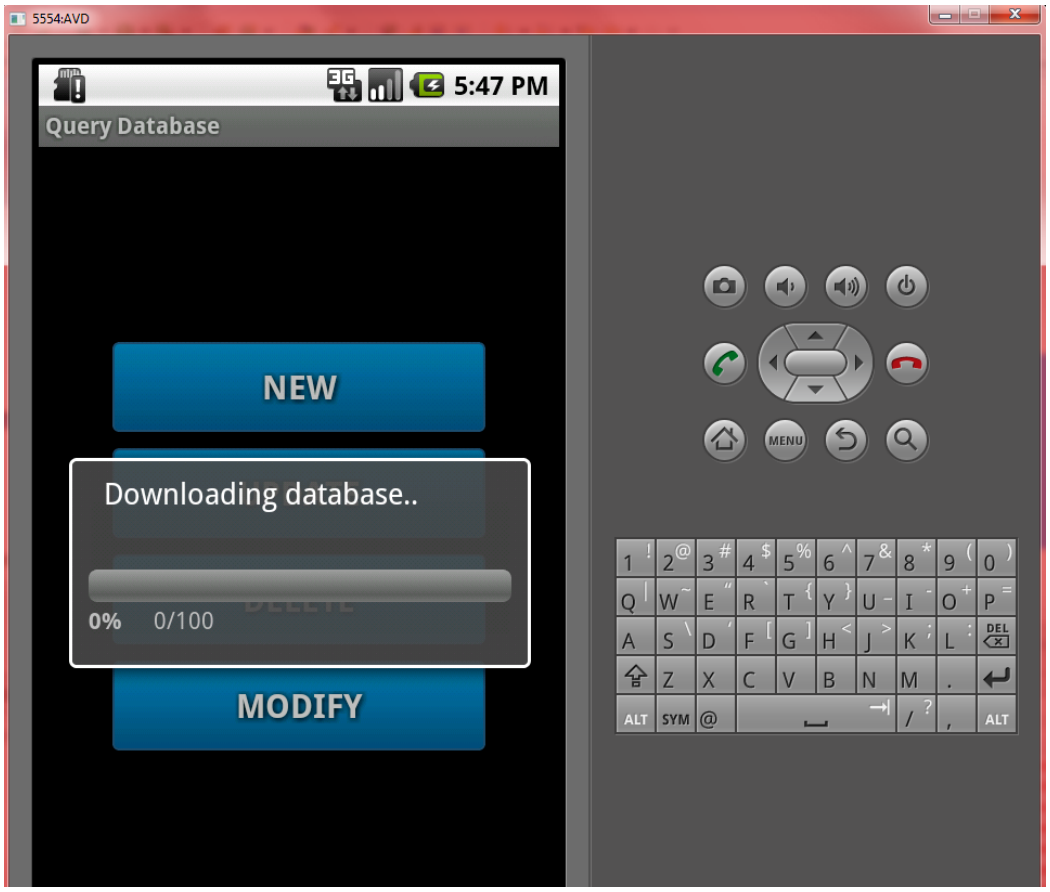
5.6.4. Αναβάθμιση βάσης δεδομένων

Τελευταία επιλογή από το μενού της Εικόνας 47 είναι το κουμπί Update που δίνει την δυνατότητα στον χρήστη να αναβαθμίσει αμέσως την βάση δεδομένων με τις ερωτήσεις κατεβάζοντας την επίσημη βάση που υπάρχει στους servers του Πανεπιστημίου Δυτικής Μακεδονίας.

Όπως έχουμε αναφέρει και σε προηγούμενο κεφάλαιο το να κατεβάζει ο χρήστης ένα αρχείο από το Internet είναι μια χρονοβόρα διαδικασία και πρέπει να εκτελείται σε διαφορετικό νήμα (thread) από αυτό που τρέχει η εφαρμογή μας. Για την καλύτερη εμπειρία του χρήστη χρησιμοποιείται ένας ProgressDialog με μια μπάρα ποσοστού που γεμίζει μέχρι να κατέβει η βάση δεδομένων. Συγκεκριμένα :

```
protected Dialog onCreateDialog(int id) {
                    switch (id) {
                         case DIALOG_DOWNLOAD_PROGRESS:
                               mProgressDialog           =           new
ProgressDialog(this);
                               mProgressDialog.setMessage("Downloading
database..");

      mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                               mProgressDialog.setCancelable(false);
                               mProgressDialog.show();
                               return mProgressDialog;
                         default:
                               return null;
                    }
                       }
```

Με αυτόν τον κώδικα δημιουργούμε το ProgressDialog με το κείμενο “Downloading database” σε οριζόντια μορφή και του δίνουμε την ιδιότητα να μην μπορεί ο χρήστης να το ακυρώσει μέχρι να ολοκληρωθεί η λήψη.



«Εικόνα 51 : Οθόνη με το μήνυμα διαλόγου για το κατέβασμα της βάσης»

Η διαδικασία λοιπόν του κατεβάσματος της βάσης (Εικόνα 51) «τρέχει» στο παρασκήνιο με την μέθοδο AsyncTask, που την περιγράψαμε σε προηγούμενο κεφάλαιο. Συγκεκριμένα :

```
class DownloadFileAsync extends AsyncTask<String, String, String> {

                    @Override
                    protected void onPreExecute() {
                          super.onPreExecute();
                          showDialog(DIALOG_DOWNLOAD_PROGRESS);
                    }

                    @Override
                    protected String doInBackground(String... aurl) {
                          int count;

                    try {

                    URL url = new URL(aurl[0]);
                    URLConnection conexion = url.openConnection();
                    conexion.connect();

                    int lenghtOfFile = conexion.getContentLength();
                    Log.d("ANDRO_ASYNC",    "Lenght    of    file:    "    +
lenghtOfFile);
```

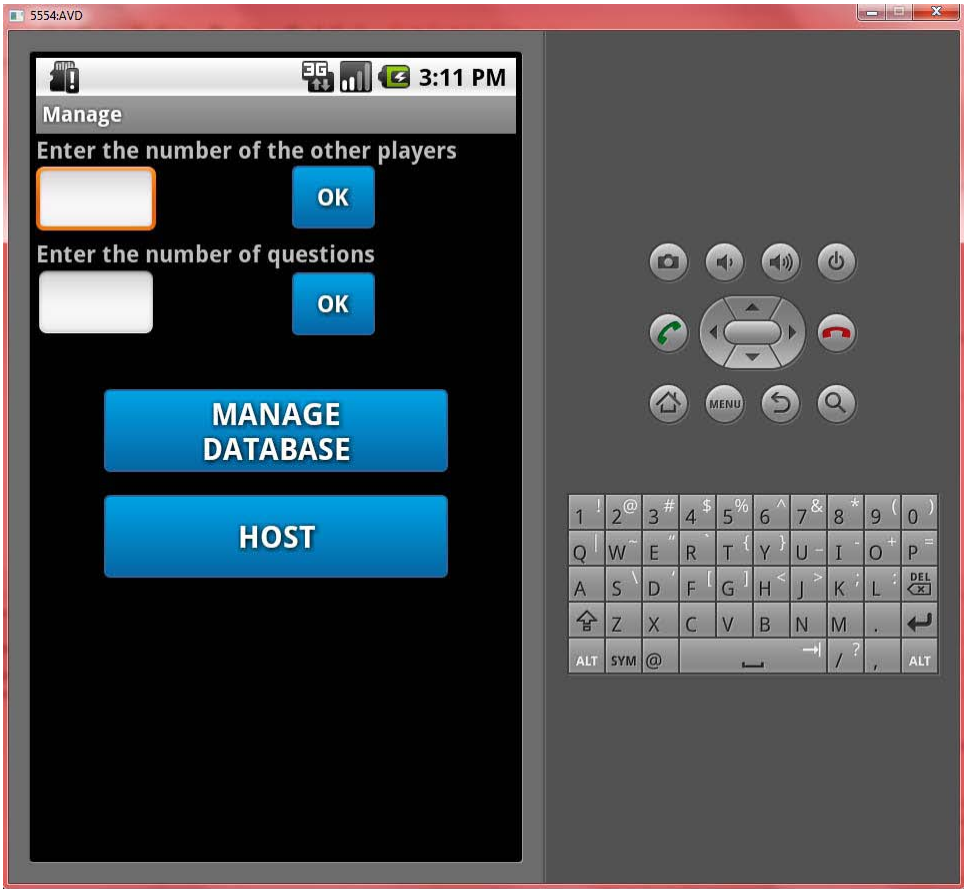
```
                    InputStream            input            =            new
BufferedInputStream(url.openStream());
                    OutputStream           output           =            new
FileOutputStream("/data/data/stathis.example.diplomatiki/databases/DATABASE
");

                    byte data[] = new byte[1024];

                    long total = 0;

                          while ((count = input.read(data)) != -1) {
                                total += count;

      publishProgress(""+(int)((total*100)/lenghtOfFile));
                                output.write(data, 0, count);
                          }

                          output.flush();
                          output.close();
                          input.close();
                    } catch (Exception e) {}
                    return null;

                    }
                    protected void onProgressUpdate(String... progress) {
                           Log.d("ANDRO_ASYNC",progress[0]);

mProgressDialog.setProgress(Integer.parseInt(progress[0]));
                    }

                    @Override
                    protected void onPostExecute(String unused) {
                          dismissDialog(DIALOG_DOWNLOAD_PROGRESS);
                          Toast.makeText(getApplicationContext(), "Database
downloaded sucessfully", Toast.LENGTH_LONG).show();
                    }
              }
```

Αυτή η μέθοδος αποτελείται στην ουσία από τρεις υπομεθόδους. Την *doInBackground()*, την *OnProgressUpdate()* και την *onPostExecute()*. Η πρώτη μέθοδος ασχολείται με το background thread και είναι αυτή που αναλαμβάνει όλη την δουλειά. Αυτή διαχειρίζεται την σύνδεση με το server στην σελίδα `url = "http://zafora.icte.uowm.gr/~ictest00032/DATABASE".`
Το αρχείο αποθηκεύεται στον φάκελο του συστήματος data και συγκεκριμένα στο path : `/data/data/stathis.example.diplomatiki/databases/DATABASE .`

Στον φάκελο αυτό δεν έχουμε δικαιώματα πρόσβασης και εγγραφής αν δεν έχουμε root access όπως περιγράψαμε σε προηγούμενο κεφάλαιο. Επίσης αυτή η μέθοδος μετράει το μέγεθος του αρχείου που θέλει να κατεβάσει ο χρήστης και με την μέθοδο *onProgressUpdate()* ενημερώνει τον χρήστη για το ποσοστό του αρχείου το οποίο έχει κατεβάσει. Η μέθοδος *onPostExecute()* ειδοποιεί τον χρήστη ότι η διαδικασία έχει τελειώσει με ένα μήνυμα ότι το κατέβασμα της βάσης ολοκληρώθηκε με επιτυχία.

5.7. Οικοδεσπότης ενός παιχνιδιού (host)

Φτάνοντας στο σημαντικότερο κομμάτι της εφαρμογής πρέπει να θυμίσουμε στον αναγνώστη σε ποια οθόνη ακριβώς είμαστε.



«Εικόνα 52 : Hosting a game»

Ο χρήστης σε αυτήν την οθόνη πρέπει να διαλέξει τον αριθμό των παικτών μαζί με τον εαυτό του καθώς και τον αριθμό των ερωτήσεων με τις οποίες θέλει να παίξει. Αφού τα επιλέξει και πατήσει τα αντίστοιχα κουμπιά OK τότε μπορεί να επιλέξει την επιλογή HOST. Αυτό το κουμπί ξεκινάει μια service στην οποία τρέχει ο server που διαχειρίζεται την επικοινωνία με τους άλλους παίκτες – clients. Παρακάτω παρουσιάζουμε τα σημαντικότερα κομμάτια του κώδικα :

```
public class Server extends Service {

    public static final int SERVERPORT = 6000;
    private ServerSocket serverSocket;
    static Vector<Socket> mclients;
    private List<PrintWriter> mouts;
    static DatabaseHelper dbHelper;
    Cursor cur;
    String numberofclients, numberofquestions;
    int aa, bb, counter;
```

```
    @Override
       public void onCreate()
    {
              Thread server = new Thread(new ServerThread());
              server.start();
              dbHelper = new DatabaseHelper(this);
          }
```

Αρχικά, πρέπει να ορίσουμε την θύρα διαμέσω της οποίας γίνεται η επικοινωνία πελάτη – διακομιστή. Αξίζει να σημειωθεί ότι αν ο προγραμματιστής χρησιμοποιεί φυσικές συσκευές για να ελέγξει την εφαρμογή του ο αριθμός θύρας πρέπει να είναι ο ίδιος και σε πελάτη και σε διακομιστή, ενώ αν χρησιμοποιεί εικονικές συσκευές θα πρέπει να έχει διαφορετικό αριθμό θύρας και να προωθήσει όλη την επικοινωνία μέσω TCP στον server σε αυτή που θα επιλέξει μέσω του telnet.

Όταν λοιπόν δημιουργηθεί η service δημιουργείται και ένα καινούργιο thread, το ServerThread, που θα διαχειρίζεται την επικοινωνία με τους πελάτες γιατί όπως έχουμε αναφέρει όλες οι χρονοβόρες διαδικασίες θα πρέπει να τις επεξεργαζόμαστε σε διαφορετικό thread από το UIthread. Αν δεν γίνει αυτό η συσκευή διακομιστής θα φαίνεται ότι έχει κολλήσει μέχρι να συνδεθεί ο πρώτος χρήστης πράγμα που μπορεί να οδηγήσει σε τερματισμό της εφαρμογής μας.

```
public void run() {
                    try {
                   serverSocket = new ServerSocket(SERVERPORT);

                   mouts = new LinkedList<PrintWriter>();
                   for (;;)
                    {
                     Socket listener = serverSocket.accept();
                     Thread       client       =       new       Thread(new
ClientHandler(listener));
                      client.start();
                    }
                   } catch (IOException ex) { ex.printStackTrace(); }

                    }}
```

Σε κάθε thread υπάρχει υποχρεωτικά και μια μέθοδος run όπου εκεί δώνουμε το κομμάτι του κώδικα όπου μπορούμε να είμαστε δημιουργικοί. Έτσι δημιουργούμε ένα ServerSocket στην θύρα που έχουμε ορίσει. Αρχικοποιούμε την μεταβλητή mouts που είναι τύπου LinkedList καθώς εδώ τοποθετούμε όλα τα outputStreams που δημιουργούνται από κάθε πελάτη.

Έτσι για κάθε σύνδεση – πελάτη που έχουμε δημιουργούμε ένα καινούργιο thread το οποί θα χειρίζεται τον κάθε πελάτη ξεχωριστά.

```
public class ClientHandler implements Runnable
       {
       private Socket client;
       public ClientHandler(Socket listener)  {
           this.client = listener;

             @Override
```

```
             public void run() {
            PrintWriter out = null;
            try {
                out         =         new         PrintWriter(         new
OutputStreamWriter(client.getOutputStream()));

                // inform the server of this new client
                if(mouts.size() < aa)
                {
                    Server.this.addClient(out);
                }
                else
                {
                    Server.this.removeClient(out);
                    client.close(); // handle it more gently
                }
```

Στο νέο thread, το ClientHandler, έχουμε την μέθοδο *run().* Για κάθε πελάτη δημιουργούμε ένα OutputStream πάνω στο αρχικό ServerSocket όπως κάναμε και για την περίπτωση του client. Έχουμε αναφέρει πιο πάνω ότι ο χρήστης διαλέγει τον αριθμό των πελατών και περνάει τον αριθμό αυτό στο service που ξεκινά τον server.

Ο τρόπος μεταφοράς δεδομένων από μια activity σε μια service είναι η εξής. Στο κομμάτι της activity χρησιμοποιούμε τον παρακάτω κώδικα :

```
Bundle bundle = new Bundle();
                    bundle.putCharSequence("NumberOfClients", a);
                    bundle.putCharSequence("NumberOfQuestions", b);
                    Intent intent = new Intent(Manage.this, Server.class);
                    intent.putExtras(bundle);
                    startService(intent);
```

Δημιουργούμε ένα νέο Bundle, ένα χάρτη δηλαδή που μεταβιβάζεται από activity σε activity και μεταφέρει δεδομένα string σε άλλες μετατροπίσιμες μορφές. Μετατρέπουμε την ακέραια μεταβλητή (int) που περιέχει την επιλογή του χρήστη όσον αφορά τον αριθμό των πελατών και τον αριθμό των ερωτήσεων σε sting και αρχίζει την service με αυτήν την μεταβλητή.

Το πρόβλημα που προκύπτει είναι το εξής: Η service δεν έχει γραφικό περιβάλλον και επομένως δεν έχει και Bundle. Την λύση έρχεται να ΄δωσει η μέθοδος *onStartCommand().*

```
public int onStartCommand(Intent intent, int flags, int startId) {
        // TODO Auto-generated method stub
        super.onStartCommand(intent, flags, startId);
          Bundle bundle = new Bundle();
          bundle = intent.getExtras();
          numberofclients=(String)
bundle.getCharSequence("NumberOfClients");
       numberofquestions=(String)
bundle.getCharSequence("NumberOfQuestions");
          Int a = Integer.parseInt(numberofclients);
           int b = Integer.parseInt(numberofquestions);
          aa = a;
          bb = b;

          Log.d(numberofclients, numberofclients);
```

```
        return START_STICKY;
            }
```

Έτσι παίρνουμε την μεταβλητή string πλέον a με την ονομασία NumberOfClients που παίζει τον ρόλο της σήμανσης και τίποτα άλλο και την μετατρέπουμε πάλι σε μεταβλητή τύπου int. Η παράμετρος που επιστρέφεται, return START_STICKY, τοποθετείται για την περίπτωση που η service τερματιστεί από το λειτουργικό σύστημα λόγω έλλειψης πόρων μνήμης. Χρησιμοποιώντας λοιπόν αυτό δείχνουμε στο λειτουργικό ότι όταν ξαναβρεί τους πόρους να ξεκινήσει ξανά την service με το συγκεκριμένο Bundle. Με τον ίδιο τρόπο γίνεται και η επιλογή του αριθμού των ερωτήσεων. Συγκριμένα, ορίζουμε μια global μεταβλητή idList τύπου ArrayList(string), δηλαδή τα στοιχεία που δέχεται θα είναι string. Την αρχικοποιούμε με την εντολή : if (idList == null) {idList = new ArrayList<String>(); } όπου απλά δημιουργούμε μια ένα νέο ArrayList αν η μεταβλητή μας δεν έχει κανένα στοιχείο. Στην συνέχεια παίρνουμε το ID και κάνουμε τον εξής έλεγχο. Εάν αυτό το ID δεν υπάρχει μέσα στην λίστα τότε προχώρησε σε παρακάτω ενέργειες, διαφορετικά ανασύρουμε άλλη ερώτηση από την βάση για αν συνεχίσει το παιχνίδι. Έτσι διασφαλίζουμε ότι δεν υπάρχουν ίδιες ερωτήσεις στην λίστα και κάθε πελάτης θα λαμβάνει και διαφορετική ερώτηση.

Επομένως, έχοντας αποθηκεύσει στην λίστα όλα τα OutputStreams που δημιουργούνται αυτόματα με την σύνδεση του πελάτη στον διακομιστή μπορούμε να ελέγχουμε το αν και πότε ο αριθμός των πελατών έχει ξεπερασθεί. Αν όχι τότε προσθέτουμε στην λίστα τον νέο πελάτη. Αν ναι τότε τον διαγράφουμε και κλείνουμε και το socket που έχει δημιουργηθεί. Η πρόσθεση και η διαγραφή γίνονται με τον παρακάτω κώδικα :

```
private void addClient(PrintWriter out) {
        synchronized(mouts) {
             mouts.add(out);
        }
    }

    /** Adds the client with given print writer. */
    private void removeClient(PrintWriter out) {
        synchronized(mouts) {
             mouts.remove(out);
        }
          }
```

Οι μέθοδοι *add()* και *remove()* είναι μέρος της LinkedList που αναφέραμε παραπάνω και προσθέτουν και αφαιρούν αντίστοιχα τα OutputStreams από την λίστα.

```
                if(mouts.size() == aa)
                {
                    cur = dbHelper.getRandomQuestion(); // gia na bgazei
tis idies erotiseis kai sta 2
                            String                  question                =
cur.getString(cur.getColumnIndex("QUESTIONS"));
```

```
                        String                  answer1                 =
cur.getString(cur.getColumnIndex("ANSWER1"));
                        String                  answer2                 =
cur.getString(cur.getColumnIndex("ANSWER2"));
                        String                  answer3                 =
cur.getString(cur.getColumnIndex("ANSWER3"));
                        String                  answer4                 =
cur.getString(cur.getColumnIndex("ANSWER4"));
                        String confirmsg = "Game Begins. Have fun!!";
                        String output = question +(":") + answer1 +(":")
+ answer2 + (":") +answer3 + (":") + answer4 + (":") + confirmsg;
                    // broadcast the receive message
                    Server.this.broadcast(output);
                }
```

Όταν ο αριθμός των πελατών έχει φθάσει τον επιθυμητό τότε σε όλους του χρήστες στέλνεται η ίδια τυχαία ερώτηση και το παιχνίδι είναι έτοιμο να ξεκινήσει. Η επιλογή της τυχαίας ερώτησης γίνεται με την μέθοδο *getRandomQuestion()* που παρουσιάζεται παρακάτω:

```
public Cursor getRandomQuestion()
       {

             SQLiteDatabase db = this.getReadableDatabase();
             Cursor  cursor  =  db.query("THESIS  Order  BY  RANDOM()  LIMIT
1",new String[] { "*" }, null, null, null, null, null);
             cursor.moveToFirst();
             return cursor;
             }
```

Είναι παρόμοια με τις υπόλοιπες μεθόδους που προσπελαύνουν την βάση δεδομένων. Διαφέρουν στο σημείο ότι αυτή χρησιμοποιεί ένα φίλτρο για να ανακατέψει τις ερωτήσεις και να τις επιστρέψει με την καινούργια σειρά που θα έχουν. Χρησιμοποιούμε μια μέθοδο παρόμοια με αυτή της MySQL order by rand().

Αποθηκεύουμε το περιεχόμενο κάθε στήλης σε διαφορετική περιγραφική μεταβλητή και στέλνουμε τις ερωτήσεις και τις απαντήσεις σε κάθε πελάτη με την μέθοδο *broadcast(),* που περιγράφεται ως εξής:

```
private void broadcast(String msg) {
       for (PrintWriter out : mouts)
       {
                   out.println(msg);
              out.flush();
```

Δηλαδή για κάθε στοιχείο της λίστας mouts στείλε το string που έχεις λάβει σαν όρισμα. Ωστόσο κάθε φορά μπορούμε να στέλνουμε ένα στοιχείο πάνω από κάθε Stream. Αν θέλουμε να στείλουμε πολλά στοιχεία πρέπει να κάνουμε ένα πολλά stream ή στην περίπτωσή μας να δημιουργήσουμε ένα string που αποτελείται από την ερώτηση και τις απαντήσεις καθώς και ένα μήνυμα καλωσορίσματος διαχωρισμένα από ένα συγκεκριμένο στοιχείο που ονομάζεται SPACER. Στην συγκεκριμένη περίπτωση έχουμε την άνω και κάτω τελεία («:»). Δημιουργούμε λοιπόν το ενιαίο string output και το στέλνουμε με την μέθοδο broadcast. Ο πελάτης θα λάβει αυτό το string, θα το

χωρίσει σε επιμέρους βάση του SPACER και θα αποθηκεύσει το καθένα από τα string σε ένα πίνακα ώστε να μπορεί να τα εμφανίσει.

```
                BufferedReader in = new BufferedReader( new
InputStreamReader(client.getInputStream()));
                for (;;)
                {
                    String msg = in.readLine();
                    if(msg.equals("wrong"))
                    {
                          counter ++;
                          String confirmsg = "Client from " +
client.getPort() +(" gave the wrong answer") + counter;

                        System.out.println("Received: " + msg);
                        // broadcast the receive message
                        Server.this.broadcast(confirmsg);
                        if(counter == aa)
                        {
                          cur = dbHelper.getRandomQuestion(); // για να
βγάλει τις ίδιες ερωτήσεις και στις 2 συσκευές
                                String question =
cur.getString(cur.getColumnIndex("QUESTIONS"));
                                String answer1 =
cur.getString(cur.getColumnIndex("ANSWER1"));
                                String answer2 =
cur.getString(cur.getColumnIndex("ANSWER2"));
                                String answer3 =
cur.getString(cur.getColumnIndex("ANSWER3"));
                                String answer4 =
cur.getString(cur.getColumnIndex("ANSWER4"));
                                String confirmsg1 = "Noone Answer
correctly, let's move on";
                                String output = question +(":") + answer1
+(":") + answer2 + (":") +answer3 + (":") + answer4 + (":") + confirmsg1;
                            System.out.println("Received: " + msg);
                            counter = 0;
                            // broadcast the receive message
                            Server.this.broadcast(output);
                        }
                    }
                    else
                    {
                          if(msg.equals("right"))
                          cur = dbHelper.getRandomQuestion(); // gia na
bgazei tis idies erotiseis kai sta 2
                          String question =
cur.getString(cur.getColumnIndex("QUESTIONS"));
                          String answer1 =
cur.getString(cur.getColumnIndex("ANSWER1"));
                          String answer2 =
cur.getString(cur.getColumnIndex("ANSWER2"));
                          String answer3 =
cur.getString(cur.getColumnIndex("ANSWER3"));
                          String answer4 =
cur.getString(cur.getColumnIndex("ANSWER4"));
```

```
                        String confirmsg = "Client from " +
client.getPort() +(" gave the correct answer");
                        String output = question +(":") + answer1 +(":")
+ answer2 + (":") +answer3 + (":") + answer4 + (":") + confirmsg;
                    System.out.println("Received: " + msg);
                    counter = 0;
                    // broadcast the receive message
                    Server.this.broadcast(output);
                }
                  }
```

Το κομμάτι του παραπάνω κώδικα διαχειρίζεται την διαδραστικότητα μεταξύ πελάτη και διακομιστή. Δημιουργεί δηλαδή ένα Inputstream πάνω στο οποίο στέλνονται τα δεδομένα και ακούει για τις απαντήσεις των πελατών.

Όταν λοιπόν ο πελάτης διαλέξει μια απάντηση στην εκάστοτε ερώτηση τότε ανάλογα την ορθότητά της στέλνει στον διακομιστή και το string right, αν η απάντηση είναι σωστή, το string wrong, αν η απάντηση είναι λανθασμένη. Ο διακομιστής τότε πρέπει συμπεριφέρεται ανάλογα την απάντηση του πελάτη. Αν αυτή είναι λάθος τότε στέλνει το μήνυμα σε όλους ότι ο συγκεκριμένος πελάτης έστειλε λάθος απάντηση με αποτέλεσμα για την ίδια ερώτηση αυτός να μην μπορεί να δώσει και άλλη απάντηση. Σε κάθε λοιπόν λανθασμένη απάντηση αυξάνει και ένα μετρητή (counter). Αν όλοι οι πελάτες απαντήσουν λανθασμένα, δηλαδή ο counter γίνει ίσος με τον αριθμό των πελατών, τότε ο server στέλνει το κατάλληλο μήνυμα σε όλους τους πελάτες μαζί με μια νέα ερώτηση.

Στην περίπτωση που κάποιος χρήστης δώσει την σωστή απάντηση τότε όλοι οι χρήστες ενημερώνονται για αυτό το γεγονός και στέλνεται σε όλους μια νέα ερώτηση προς απάντηση.

Αν τον κώδικα για την επεξεργασία κάθε τυχαίας ερώτησης τον είχαμε τοποθετήσει μέσα στην μέθοδο που στέλνει τα δεδομένα στους πελάτες, δηλαδή στην *broadcast()*, τότε κάθε πελάτης θα έπαιρνε διαφορετική τυχαία ερώτηση γιατί σε κάθε κάλεσμα της μεθόδου θα δημιουργούνταν διαφορετικό στιγμιότυπο της μεθόδου για κάθε πελάτη.

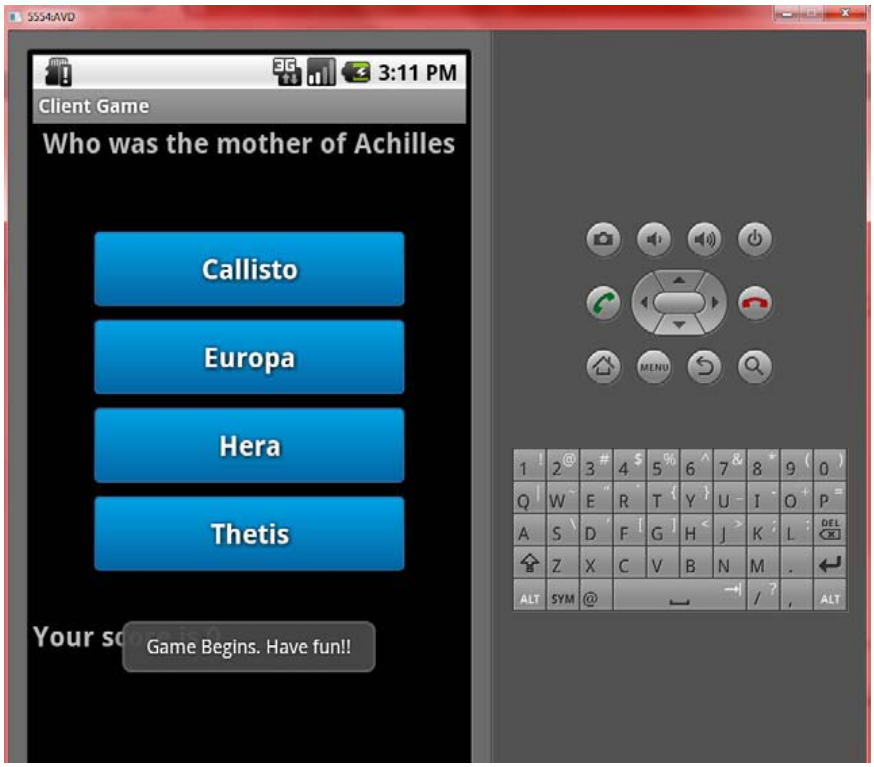
5.8. Ειδοποίησης του χρήστη

Είδαμε ως τώρα τις βασικότερες λειτουργίες της εφαρμογής οι οποίες συνοψίζονται στην συμπεριφορά της συσκευής ανάλογα με την επιλογή του χρήστη. Μπορεί να δηλαδή η συσκευή να συμπεριφέρεται σαν πελάτης περιμένοντας να συνδεθεί σε ένα διακομιστή ή να συμπεριφέρεται σαν διακομιστής και να περιμένει άλλες συσκευές να συνδεθούν σε αυτή.
Ένα εξίσου σημαντικό τμήμα της εφαρμογής που θα αναλύσουμε αμέσως είναι το κομμάτι της αλληλεπίδρασης με τον χρήστη. Για παράδειγμα η αποστολή μηνυμάτων στον χρήστη ότι μια ενέργεια ολοκληρώθηκε με επιτυχία ή ότι κάποιος πελάτης απάντησε λανθασμένα κλπ.

Ας δούμε λοιπόν τα πράγματα από την μεριά του χρήστη ο οποίος έχει επιλέξει το client mode για την συσκευή του. Αμέσως μόλις συνδεθεί ο κατάλληλος αριθμός πελατών στον κάθε πελάτη εμφανίζεται η ερώτηση και ένα μήνυμα καλωσορίσματος (Εικόνα 42). Το μήνυμα αυτό αποστέλλεται από

τον διακομιστή σε όλους τους πελάτες μαζί με την ερώτηση, σε μορφή ενός μεγάλου string που αναλύεται κατάλληλα. Συγκεκριμένα :
Toast.makeText(getApplicationContext(),mssg[0],
Toast.LENGTH_SHORT).show();

Όπως έχουμε αναφέρει και σε προηγούμενη ενότητα η μέθοδος displayMsg(String [] msg), τοποθετεί οτιδήποτε στείλει ο διακομιστής σε ένα πίνακα και τα εμφανίζει στην οθόνη του χρήστη. Έτσι το πρώτο στοιχείο του πίνακα είναι το μήνυμα καλωσορίσματος (Εικόνα 53). Με ανάλογο τρόπο εμφανίζονται και τα μηνύματα για το ποιος χρήστης έδωσε σωστή ή λανθασμένη απάντηση.



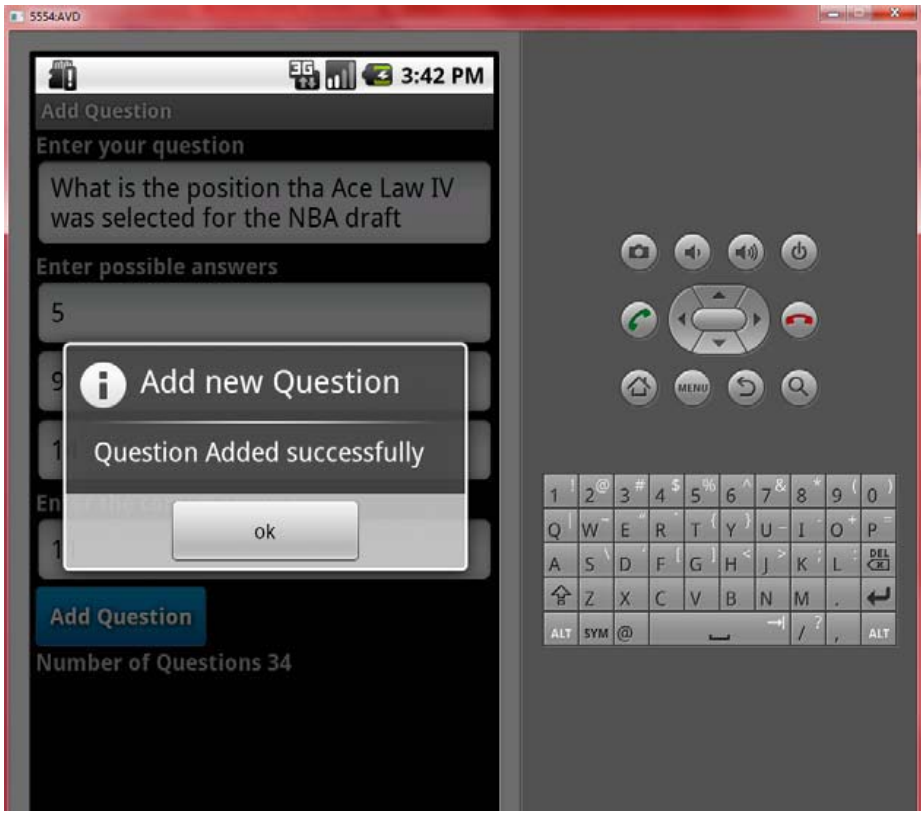
«Εικόνα 53 : **Εμφάνιση μηνύματος καλωσορίσματος**»

Στην συνέχεια, όταν ο χρήστης είναι σε server mode μπορεί να διαχειριστεί και την βάση δεδομένων που περιέχει και τις ερωτήσεις και τις απαντήσεις προσθέτοντας, αφαιρώντας ή τροποποιώντας μια ερώτηση (Εικόνα 47).

Όταν λοιπόν ο χρήστης στην φόρμα συμπλήρωσης ερώτησης συμπληρώσει όλα τα πεδία και πατήσει το κουμπί Add Question τότε ο χρήστης, αν δεν δημιουργηθεί κάποιο πρόβλημα, θα ειδοποιηθεί για την επιτυχή προσθήκη της ερώτησης μέσω κάποιου παράθυρου διαλόγου (Εικόνα 54). Επίσης το ίδιο θα συμβεί και όταν προσπαθήσει να διαγράψει ή να τροποποιήσει μια ερώτηση (Εικόνες 49 και 50). Αυτές οι ειδοποιήσεις τις χειρίζεται το αρχείο Alerts.java. Συγκεκριμένα για τον διάλογο στην προσθήκη ερώτησης έχουμε την μέθοδο *ShowQuestAddedAlert()* :
public static void ShowQuestAddedAlert(Context con)

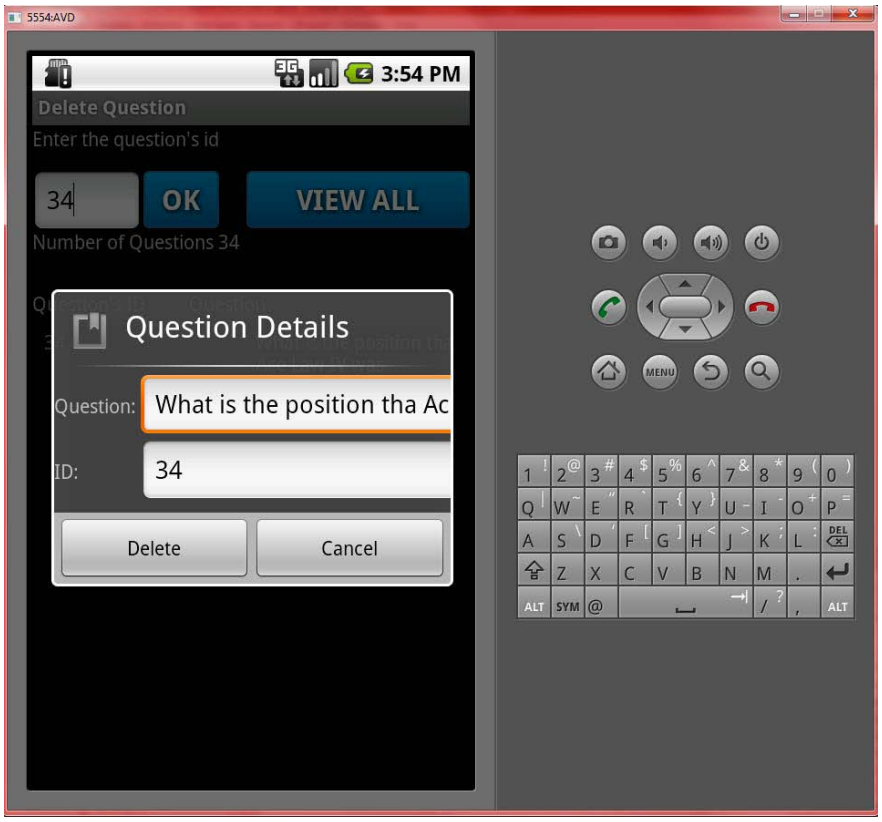
{ AlertDialog.Builder builder = new AlertDialog.Builder(con); builder.setTitle("Add new Question"); builder.setIcon(android.R.drawable.ic_dialog_info); DialogListner listner = new DialogListner(); builder.setMessage("Question Added successfully"); builder.setPositiveButton("ok", listner);
AlertDialog diag = builder.create(); diag.show(); }

Δημιουργούμε απλά ένα παράθυρο διαλόγου με την επικεφαλίδα “Question Added successfully” και με την επιλογή OK που μας επιτρέπει να ενημερώσουμε τον χρήστη ότι όλα πήγαν καλά και η ερώτηση του προστέθηκε στην βάση (Εικόνα 54).



«Εικόνα 54 : **Μήνυμα ειδοποίησης του χρήστη για την επιτυχή προσθήκη της ερώτησης**»

Επιπλέον όταν ο χρήστης είναι στην οθόνη διαγραφής μιας ερώτησης αφού επιλέξει την ερώτηση που θέλει να διαγράψει εμφανίζεται ένα παράθυρο με το ID και την ερώτηση. Εκεί ο χρήστης έχει δύο επιλογές, το κουμπί delete, που διαγράφει την ερώτηση και το κουμπί cancel, που ακυρώνει το παράθυρο και επιστρέφει στην προηγούμενη οθόνη (Εικόνα 55).



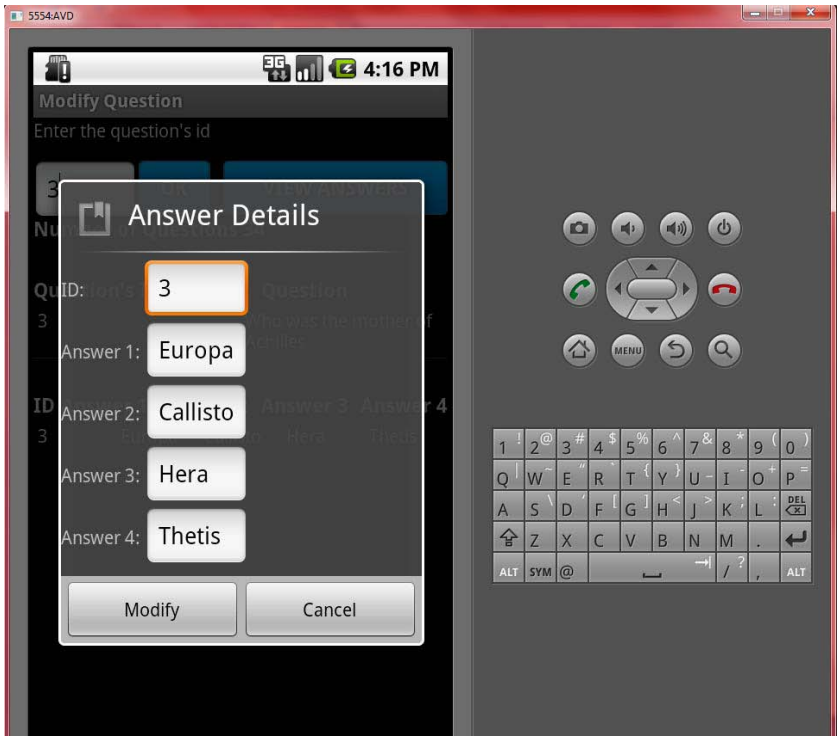
«Εικόνα 55 : **Παράθυρο διαλόγου για την διαγραφή της ερώτησης**»

Ο παραπάνω διάλογος δημιουργείται από την μέθοδο *ShowDeleteDialog()* που αναλύεται παρακάτω:

```
public static AlertDialog ShowDeleteDialog(final Context context,final Question nquest)
{ AlertDialog.Builder build = new AlertDialog.Builder(context); build.setTitle("Question Details");
        LayoutInflater li = LayoutInflater.from(context);  View v = li.inflate(R.layout.deletedialog,
        null);

         build.setIcon(android.R.drawable.ic_input_get);

         build.setView(v); final TextView txtName = (TextView)v.findViewById(R.id.txtDelName);
        final TextView txtID = (TextView)v.findViewById(R.id.txtDelID);

         txtName.setText(nquest.getQuestion()); txtID.setText(String.valueOf(nquest.getID());

         build.setNeutralButton("Delete", new OnClickListener() {

                @Override
                public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                    DatabaseHelper db = new DatabaseHelper(context);
                   db.DeleteQuestion(nquest); }

                }); build.setNegativeButton("Cancel", null);
        return build.create(); }
```

Γενικά σε ένα τέτοιο διάλογο που ζητάμε την επιλογή του χρήστη μπορούμε να βάλουμε μέχρι τρία κουμπιά τα οποία χαρακτηρίζονται από default positive, neutral και negative κουμπιά. Αυτό δεν είναι τίποτα άλλο από μια τυπική ονομασία που χρησιμοποιείται για την διευκόλυνση του προγραμματιστή. Στο παραπάνω χρησιμοποιούμε τα neutral και negative κουμπιά που είναι το delete και το cancel αντίστοιχα. Παρατηρούμε ότι καλούμε ένα ξεχωριστό αρχείο xml το deletedialog.xml που ορίζει ακριβώς τα πεδία που θα εμφανίζονται στον διάλογο που είναι η ερώτηση και το ID της.

Τέλος στην περίπτωση που ο χρήστης έχει εντοπίσει κάποιο λάθος στις ερωτήσεις της βάσης δεδομένων και επιθυμεί να το αλλάξει θα πρέπει τότε από το μενού στην Εικόνα 47 να πατήσει το κουμπί Modify και να εμφανιστεί μια οθόνη παρόμοια με αυτή της διαγραφής ερώτησης με την διαφορά ότι εκτός από την ίδια την ερώτηση μπορεί να δει και τις απαντήσεις σε κάθε μια ώστε να ελέγχει πιο αποτελεσματικά για κάποιο λάθος. Συγκεκριμένα :



«Εικόνα 56 : **Παράδειγμα αλλαγής απαντήσεων σε μια ερώτηση**»

Αφού ο χρήστης επιλέξει είτε την ερώτηση, όπου εμφανίζεται ένα παράθυρο παρόμοιο με αυτό που εμφανίζεται στην διαγραφή ερώτησης, είτε τις απαντήσεις όπου εμφανίζεται το παράθυρο της Εικόνας 56 τότε μπορεί να κάνει τις τροποποιήσεις που θέλει πατώντας το κουμπί Modify. Το παράθυρο δημιουργείται από την μέθοδο ShowModifyAnsDialog().

```
public static AlertDialog ShowModifyAnsDialog(final Context con,final Question quest)
{ AlertDialog.Builder build = new AlertDialog.Builder(con); build.setTitle("Answer Details");
        LayoutInflater li = LayoutInflater.from(con);  View v = li.inflate(R.layout.modifyansdialog,
        null);

         build.setIcon(android.R.drawable.ic_input_get);

         build.setView(v); final TextView txtid = (TextView)v.findViewById(R.id.txtModID); final
        TextView txtAns1 = (TextView)v.findViewById(R.id.txtModAns1); final TextView txtAns2 =
        (TextView)v.findViewById(R.id.txtModAns2); final TextView txtAns3 =
        (TextView)v.findViewById(R.id.txtModAns3); final TextView txtAns4 =
        (TextView)v.findViewById(R.id.txtModAns4);

         txtAns1.setText(quest.getAnswer1());  txtAns2.setText(quest.getAnswer2());
        txtAns3.setText(quest.getAnswer3());  txtAns4.setText(quest.getAnswer4());
        txtid.setText(String.valueOf(quest.getID())); build.setPositiveButton("Modify", new
        OnClickListener() {

                @Override
                public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                         quest.setAnswer1(txtAns1.getText().toString());
                        quest.setAnswer2(txtAns2.getText().toString());
                        quest.setAnswer3(txtAns3.getText().toString());
                        quest.setAnswer4(txtAns4.getText().toString());

                        try
                         { DatabaseHelper db = new DatabaseHelper(con);
                        db.UpdateAnswers(quest);

                        } catch(Exception ex) {
                        CatchError(con, ex.toString()); }
                }
        });
        build.setNegativeButton("Cancel", null);

        return build.create(); }.
```

6

Συμπεράσματα και μελλοντικές εξελίξεις

6.1. Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με την ανάπτυξη μιας εφαρμογής, η οποία θα εκτελείται σε κινητές συσκευές που χρησιμοποιούν το λειτουργικό σύστημα Android. Αποτελείται από 39 συνολικά αρχεία, συμπεριλαμβανομένου και των αρχείων μορφοποίησης και εμφάνισης στην οθόνη, από 6500 γραμμές κώδικα περίπου και από 40 περίπου μεθόδους. Η επιλογή της ανάπτυξης της εφαρμογής για το συγκεκριμένο λειτουργικό σύστημα κρίνεται επιτυχής για αρκετούς λόγους:

- Το Android αυτή τη στιγμή χρησιμοποιείται στο μεγαλύτερο ποσοστό smartphones της αγοράς, ενώ περίπου 550.000 νέες συσκευές Android ενεργοποιούνται καθημερινά.
- Ο αριθμός αυτός αυξάνεται ανά εβδομάδα περίπου κατά 4% με πρόβλεψη για 1 εκατομμυρίου ενεργοποιήσεων νέων συσκευών μέχρι το τέλος του 2012.
- Οι μεγαλύτερες κατασκευαστικές εταιρίες κινητών τηλεφώνων, όπως οι HTC, LG, Motorola, ZTE, υιοθετούν το λειτουργικό σύστημα στις συσκευές τους.
- Το Android Market τη στιγμή αυτή διαθέτει πάνω από 250.000 εφαρμογές.
- Σε γενικές γραμμές, η εκμάθηση των εργαλείων για την ανάπτυξη εφαρμογών κρίνεται εύκολη.
- Όλα τα προγραμματιστικά εργαλεία που χρειάζονται για την ανάπτυξη εφαρμογών μπορεί εύκολα να τα βρει κανείς στο διαδίκτυο και ο καθένας μπορεί να τα κατεβάσει και να τα χρησιμοποιήσει χωρίς κανένα κόστος.
- Υπάρχουν πάρα πολλές διαδικτυακές κοινότητες που ασχολούνται καθαρά με τον προγραμματισμό εφαρμογών για Android. Χαρακτηριστικό παράδειγμα ότι στις περισσότερες σελίδες προγραμματιστικού ενδιαφέροντος, η «ετικέτα» Android βρίσκεται ανάμεσα στις δημοφιλέστερες.

Η έρευνα που πραγματοποιήθηκε για την εκπόνηση της διπλωματικής αυτής εργασίας κρίνεται άκρως ενδιαφέρουσα. Κατανοήθηκε ο τρόπος

επικοινωνίας μεταξύ τέτοιων συσκευών και εφαρμόστηκαν βασικές έννοιες σχετικά με την αλληλεπίδραση ανθρώπου μηχανής. Αποκτήθηκαν πολύ σημαντικές γνώσεις για τις τεχνολογίες των έξυπνων κινητών τηλεφώνων και τον τρόπο ανάπτυξης εφαρμογών σε αυτά. Προτείνεται ανεπιφύλακτα η ενασχόληση με τις τεχνολογίες αυτές, σε φοιτητές, προγραμματιστές και μηχανικούς καθώς αποτελεί ένα καινούργιο και συνεχώς αναπτυσσόμενο κλάδο της επιστήμης υπολογιστών ο οποίος δημιουργεί νέες θέσεις εργασίας παρά τις δυσκολίες που αντιμετωπίζουμε στην εποχή μας.

6.2. Μελλοντικές εξελίξεις

Το Android Market δίνει την δυνατότητα στους προγραμματιστές να δημοσιεύσουν ελεύθερα και χωρίς κόστος τις εφαρμογές τις οποίες έχουν κατασκευάσει. Η διαδικασία είναι απλή και το μόνο αρχείο που χρειάζεται για την υπηρεσία αυτή είναι το εκτελέσιμο apk της εφαρμογής. Με τον τρόπο αυτό, η εφαρμογή γίνεται προσβάσιμη σε όλους τους χρήστες Android, οι οποίοι μπορούν να την κατεβάσουν ανά πάσα στιγμή, να την βαθμολογήσουν και να γράψουν κάποια κριτική.

Στο πλαίσιο αυτό σχεδιάζεται η δημοσίευση της εφαρμογής στο Android Market καθώς και η συνεχή ενημέρωση της βάσης δεδομένων με τις ερωτήσεις ώστε ο χρήστης να ανακαλύπτει νέα πράγματα κάθε φορά. Μέσα από τις κριτικές των χρηστών (feedback) θα μπορέσουμε να την κάνουμε ακόμα πιο φιλική προς τον χρήστη και να διορθώσουμε τυχόν σφάλματα που μπορούν να εμφανιστούν.

Συγκεκριμένα προτείνονται οι παρακάτω επεκτάσεις και βελτιώσεις προτού δημοσιευτεί η εφαρμογή:

- Μετάφραση της γλώσσας της εφαρμογής στις πιο δημοφιλείς ξένες γλώσσες με στόχο να αποκτήσουν οι αντίστοιχοι χρήστες επιπλέον κίνητρο να την χρησιμοποιήσουν
- Δημιουργία μεγάλης βάσης δεδομένων όπου θα δίνεται η δυνατότητα στον χρήστη να επιλέξει το πεδίο των ερωτήσεων, για παράδειγμα επιστήμες, αθλητικά κλπ.
- Δυνατότητα διαδικτυακού παιχνιδιού ανεξαρτήτου απόστασης. Ο κάθε παίκτης θα συνδέεται σε ένα Wi-Fi hotspot και θα μπορεί να παίζει με τους φίλους του ανεξάρτητα με το πόσο μακριά βρίσκεται ο ένας από τον άλλο.

Βιβλιογραφία – Ιστοσελίδες

[1]. Android Developers, The Developer's Guide, http://developer.android.com/index.html , Ιούνιος 2012

[2]. Google Groups, Android Developers, http://groups.google.com/group/android-developes , Ιούνιος 2012

[3]. Android Development Community, http://www.anddev.org, Ιούνιος 2012

[4]. Ed Burnette. "Hello Android Introducing Google's Mobile Development Platform.

[5]. Laurent Darcey and Shane Conder "Ανάπτυξη εφαρμογών σε Android" (Second Edition).

[6]. StackOverflow, http://stackoverflow.com, Φεβρουάριος 2012-Ιούνιος 2012

[7]. Βικιπαιδεία – Wikipedia, http://el-wikipedia.org, Ιούνιος 2012