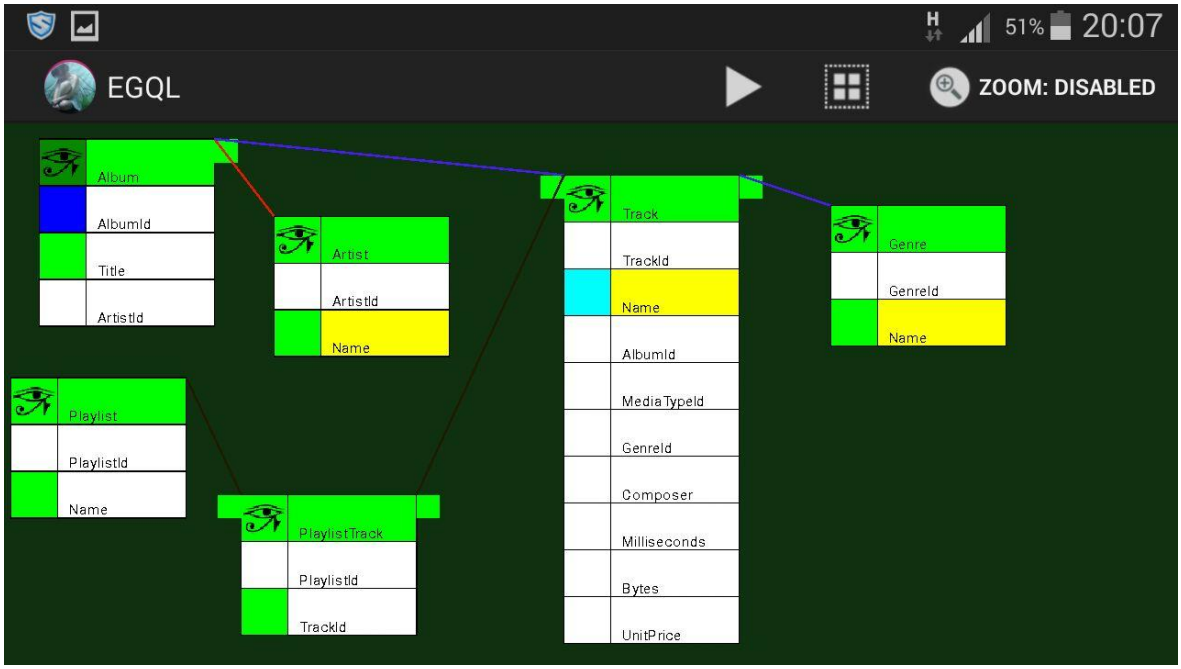
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

EGQL: A Gestural Query Language



Του φοιτητή

Αρέν Μεμέτ Ερκάν

Αρ. Μητρώου: 08/3361

Επιβλέπων καθηγητής

Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2015

ΠΡΟΛΟΓΟΣ

Οι συσκευές με οθόνες αφής μας δίνουν την δυνατότητα να κατασκευάσουμε γραφικά ερωτήματα σχεσιακών βάσεων δεδομένων με την χρήση των χειρονομιών. Στην πτυχιακή εργασία αυτή παρουσιάζουμε μια εφαρμογή γραφικής γλώσσας ερωτημάτων σχεσιακών βάσεων δεδομένων που υλοποιείται με την χρήση των χειρονομιών που βασίζονται στο λειτουργικό σύστημα Android. Η εφαρμογή απευθύνεται στις multi-touch οθόνες μικρών διαστάσεων και μεταφράζει τις χειρονομίες που πραγματοποιούνται στην οθόνη σε ερωτήματα βάσεων δεδομένων. Με την λειτουργία ζουμ της εφαρμογής μπορούν να γίνουν διαμορφώσεις στις διαστάσεις του καμβά που απεικονίζει τους πίνακες σύμφωνα με τις ανάγκες του χρήστη. Ενώ για την καλύτερη κατανόηση του σχήματος της βάσης δεδομένων χρησιμοποιούμε διάφορα χρώματα γεμίσματος για κάθε κατάσταση κελιού ή πίνακα και γραμμές που δηλώνουν τις σχέσεις μεταξύ πινάκων. Η εφαρμογή μας μπορεί να χρησιμοποιηθεί για γρηγορότερη και ευκολότερη κατασκευή ερωτημάτων, για την καλύτερη κατανόηση του σχήματος της βάσης δεδομένων και των σχέσεων μεταξύ πινάκων και για την κατανόηση της λειτουργικότητας της SQL και των σχεσιακών βάσεων δεδομένων.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία έχει γίνει η ανάπτυξη της γραφικής γλώσσας ερωτημάτων EGQL που βασίζεται στις χειρονομίες του λειτουργικού συστήματος Android. Παρουσιάζονται οι δυνατότητες της πλατφόρμας Android και του συστήματος διαχείρισης βάσεων δεδομένων SQLite.

Στον πρώτο κεφάλαιο της εργασίας κάνουμε μία μικρή εισαγωγή στο λειτουργικό σύστημα Android και παρουσιάζουμε την ιστορική εξέλιξη, τα βασικά χαρακτηριστικά του και την αρχιτεκτονική του. Στο δεύτερο κεφάλαιο αναλύουμε τα βασικά στοιχεία μίας εφαρμογής Android. Τα βασικά των διεπαφών χρηστών του Android και τα γραφικά του συστήματος αναλύονται στο τρίτο κεφάλαιο. Στο τέταρτο κεφάλαιο κάνουμε μία μικρή εισαγωγή στο σύστημα διαχείρισης βάσεων δεδομένων SQLite και βλέπουμε μερικά παραδείγματα για την καλύτερη κατανόηση του συστήματος. Στο Πέμπτο κεφάλαιο κάνουμε εισαγωγή στις γλώσσες γραφικών ερωτημάτων και χειρονομιών. Συγκρίνουμε τις γλώσσες και αναλύουμε την γλώσσα που έχουμε αναπτύξει, την EGQL. Στο έκτο κεφάλαιο αναλύουμε την EGQL, την διεπαφή χρήστη της γλώσσας, την αρχιτεκτονική και τα διάφορα κομμάτια κώδικα που αποτελείται η γλώσσα. Το κεφάλαιο αυτό τελειώνει με διάφορα παραδείγματα που απεικονίζουν την λειτουργικότητα της γλώσσας EGQL.

ABSTRACT

The touch screen devices allow us to construct user friendly graphical user interfaces using the gesture technology which is very popular for novice users. In this thesis, we present a new graphical query language for relational databases which is designed to support gesture design philosophy and the full functionality of SQL and addressed to non-expert users on databases. The EGQL is implemented on Android platform for compact multi-touch screens. Moreover, Metaphors, Color and other functionality like zoom are used for the best comprehension of the database schema and the query development.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Ευκλείδη Κεραμόπουλο για την εμπιστοσύνη και υποστήριξη που μου έδειξε κατά την διάρκεια υλοποίησης της πτυχιακής μου εργασίας…

Θα ήθελα επίσης να απευθύνω τις ευχαριστίες μου στους γονείς και φίλους που έχουν στηρίξει τις σπουδές μου με διάφορους τρόπους…

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο Σχημάτων

ΚΕΦΑΛΑΙΟ 1: ΜΙΑ ΕΠΙΣΚΟΠΗΣΗ ΤΟΥ ANDROID

Εισαγωγή

Το Android είναι ένα λειτουργικό σύστημα ανοιχτής πηγής το οποίο βασίζεται σε Linux πυρήνα. Το Android χρησιμοποιείται στα κινητά τηλέφωνα αφής, tablets, στις τηλεοράσεις (Android TV), στα αυτοκίνητα και στα ρολόγια. Το λειτουργικό σύστημα αυτό χρησιμοποιεί τις χειρονομίες των χρηστών που πραγματοποιούνται στην οθόνη της συσκευής για να χειριστεί τα δεδομένα. Επίσης, μπορούν να χρησιμοποιηθούν οι διάφοροι αισθητήρες των συσκευών για να γίνουν εισαγωγές πληροφοριών, για παράδειγμα η δόνηση της συσκευής μπορεί να χρησιμοποιηθεί για είσοδο για να ξεκινήσει μία διαδικασία σε μία εφαρμογή. Παρά το γεγονός ότι το Android έχει σχεδιαστεί για είσοδο αφής, χρησιμοποιείται επίσης σε κονσόλες παιχνιδιών, σε ψηφιακές φωτογραφικές μηχανές σε υπολογιστές και άλλες ηλεκτρονικές συσκευές. Το Android έχει σχεδιαστεί από την εταιρία Android Inc. Έχει εξαγοραστεί από την Google και έπειτα έχει αναπτυχθεί από την ίδια τη Google και τον οργανισμό Open Handset Alliance. Ο λόγος που χρησιμοποιήσαμε το Android στην ανάπτυξη της εφαρμογής μας είναι ότι το Android μας δίνει τη δυνατότητα εύκολης ανάπτυξης εφαρμογών σε μικρές οθόνες αφής. οι εφαρμογές που αναπτύσσονται για Android είναι ευέλικτες και μπορούν να εκτελεστούν σε διάφορες συσκευές με διάφορα μεγέθη. Στα επόμενα υποκεφάλαια θα εξηγήσουμε αναλυτικά το λειτουργικό σύστημα Android, την ιστορική εξέλιξη, την αρχιτεκτονική και διάφορα άλλα κομμάτια του συστήματος.

1.1 Ιστορική εξέλιξη

Το Android είναι μια πλατφόρμα καινοτόμος και ανοιχτή που επιχειρεί να καλύψει τις ανάγκες της αγοράς κινητών τηλεφώνων. Πριν το Android, η ενσωμάτωση πολλών κοινών εργασιών σε μια εφαρμογή, όπως η αποστολή μηνυμάτων και η πραγματοποίηση κλήσεων με εύκολο τρόπο συχνά ήταν ένας μη ρεαλιστικός στόχος για τους προγραμματιστές εφαρμογών κινητών τηλεφώνων. Το πρώτο κινητό τηλέφωνο, το Motorola DynaTAC 8000X, εμφανίστηκε στην αγορά το 1983. Τα πρώτα κινητά τηλέφωνα δεν είχαν πολλές λειτουργίες εκτός από την πραγματοποίηση και λήψη κλήσεων. Ο ανταγωνισμός ήταν μεγάλος και για αυτό το λόγο οι κατασκευαστές κινητών τηλεφώνων δεν ήθελαν να αποκαλύψουν τις εσωτερικές λειτουργίες των συσκευών τους. Αν δεν συμμετείχατε ως προγραμματιστής στον κλειστό κύκλο ανάπτυξης λογισμικού των εταιριών, δεν είχατε καμία δυνατότητα να γράψετε εφαρμογές για τα κινητά τηλέφωνα. Κατόπιν εμφανίστηκαν τα πρώτα παιχνίδια των κινητών τηλεφώνων, το 1997, η Nokia [1] εγκατέστησε το παιχνίδι Φιδάκι σε μερικά απ' τα πρώτα τηλέφωνα με ασπρόμαυρη οθόνη, στο Nokia 6110. Οι πελάτες ήταν πάντα απαιτητικοί και ήθελαν περισσότερα, έτσι οι κατασκευαστές τηλεφωνικών συσκευών συνειδητοποίησαν ότι πρέπει να αλλάξουν την πολιτική προστατευτισμού που ακολουθούσαν στη

σχεδίαση συσκευών και να αποκαλύψουν σε κάποιον βαθμό τις εσωτερικές λειτουργίες των τηλεφώνων τους. Εκείνη την εποχή εμφανίστηκαν διάφορες ιδιωτικές πλατφόρμες και η Sun Microsystems προσάρμοσε τη δημοφιλή πλατφόρμα Java και παρουσίασε το J2ME.

Στη συνέχεια, εμφανίστηκαν τα πρώτα κινητά που είχαν οθόνες αφής. Το πρώτο iPhone της κυκλοφόρησε στις 29 Ιουνίου 2007 [2]. Ενώ η πρώτη συσκευή Android, που κυκλοφόρησε τον Οκτώβριο του 2008, ήταν το T-Mobile G1 το οποίο κατασκευάστηκε από την HTC και T-Mobile. Η εικόνα 1 απεικονίζει την εξέλιξη των κινητών τηλεφώνων με τη χρήση των διάφορων μοντέλων εταιριών.

Εικόνα 1: Η εξέλιξη του κινητού τηλεφώνου

Η πλατφόρμα Android έχει αναπτυχθεί από τη Google και τον οργανισμό Open Handset Alliance. Το Open Handset Alliance (OHA) είναι ένας οργανισμός επιχειρήσεων για την ανάπτυξη ανοικτών προτύπων για τις κινητές συσκευές [3]. Στα μέλη του οργανισμού περιλαμβάνονται εταιρίες όπως η Google, η HTC, η Sony, η Dell, η Intel, η Motorola, η Qualcomm, η Samsung Electronics, η LG Electronics, η T-Mobile, η Sprint Corporation, η Nvidia, και η Wind River Systems [4]. Η OHA ιδρύθηκε στις 6 Νοεμβρίου 2007, υπό την ηγεσία της Google με 47 μέλη, συμπεριλαμβανομένων των κατασκευαστών κινητών τηλεφώνων, προγραμματιστών εφαρμογών, ορισμένων εταιρειών κινητής τηλεφωνίας και κατασκευαστών τσιπ [5]. Τα μέλη του OHA, βάσει του συμβολαίου τους, απαγορεύετε να παράγουν ασυμβίβαστες εκδόσεις του Android μεταξύ τους[6][7].

Παράλληλα με την ανακοίνωση του σχηματισμού του Open Handset Alliance στις 5 Νοεμβρίου 2007, παρουσιάστηκε το Android, μια πλατφόρμα ανοιχτού κώδικα κινητής τηλεφωνίας που βασίζεται σε Linux πυρήνα. [8]. Το πρώτο εμπορικά διαθέσιμο τηλέφωνο Android ήταν το T-Mobile G1 (επίσης γνωστό ως HTC Dream). Εγκρίθηκε από την Federal Communications Commission (FCC) στις 18 Αυγούστου 2008 [9], και ήταν διαθέσιμο στις 22 Οκτωβρίου [10].

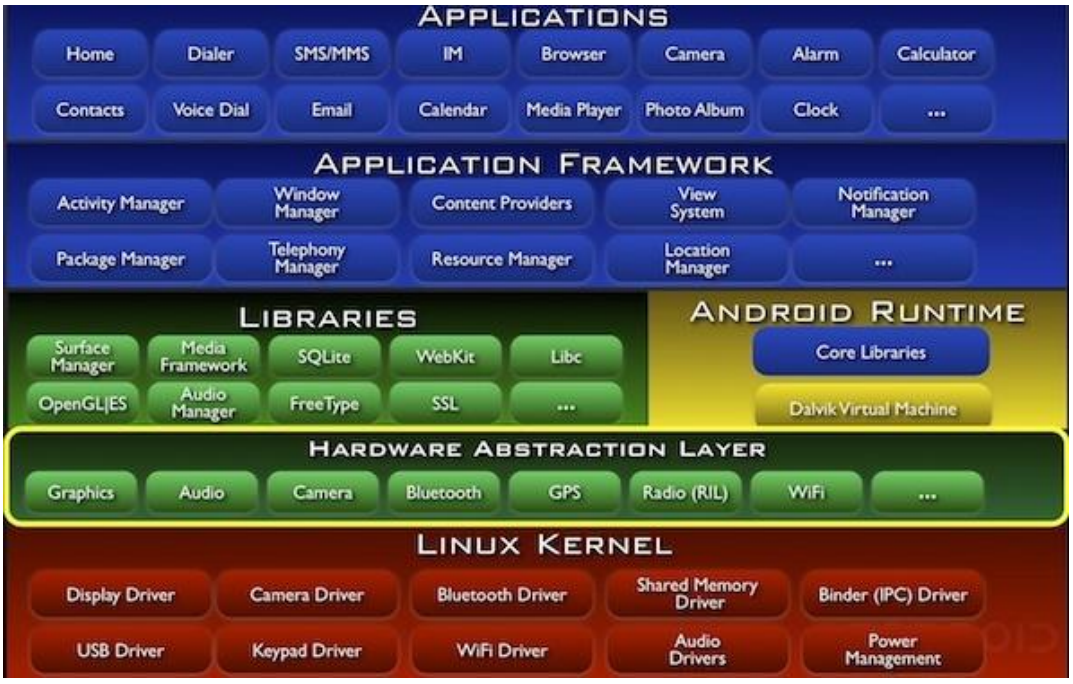
Το Android αναπτύσσεται σήμερα από την Google και είναι το πιο δημοφιλές λειτουργικό σύστημα κινητής τηλεφωνίας. Μέχρι το 2013, οι Android συσκευές έχουν κάνει περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS μαζί[11][12][13][14], επίσης οι πωλήσεις του 2012, 2013 και 2014 πλησίαζαν

στις πωλήσεις όλων των εγκατεστημένων βάσεων των υπολογιστών. Τον Ιούλιο του 2013, το Google Play Store είχε πάνω από 1 εκατομμύριο εφαρμογές Android που δημοσιεύθηκαν, και πάνω από 50 δισεκατομμύρια εφαρμογές που έχουν γίνει download[15]. Μια έρευνα για προγραμματιστές που πραγματοποιήθηκε τον Απρίλιο-Μάιο του 2013 διαπίστωσε ότι το 71% των προγραμματιστών κινητών αναπτύσσουν εφαρμογές για Android [16] .

1.2 Η Ανατομία του Android

Η πλατφόρμα του Android έχει σχεδιαστεί έτσι ώστε να είναι ανθεκτική στα σφάλματα. Οι συσκευές Android βασίζονται στο λειτουργικό σύστημα Linux, πάνω στο οποίο οι εφαρμογές λειτουργούν με ασφαλή τρόπο. Το λειτουργικό σύστημα Android είναι μια στοίβα στοιχείων λογισμικού που χωρίζεται σε πέντε τμήματα και τέσσερα στρώματα (Εικόνα 2)[17]. Τα τμήματα που αποτελείται μία πλατφόρμα Android είναι:

1. Linux Kernel
2. Libraries
3. Android Runtime
4. Application Framework
5. Applications



Εικόνα 2: Η ανατομία του Android

1.2.1 LINUX Kernel

Το Android αποτελείται από έναν πυρήνα που βασίζεται στο Linux. Οι συσκευές Android, από το 2014 και μετά θα βασίζονται σε Linux πυρήνα 3.4 ή και σε

νεότερο[18][19], ο συγκεκριμένος αριθμός της έκδοσης του πυρήνα θα εξαρτάται από την συσκευή και το chipset[20][21][22] . Ο Linux πυρήνας χειρίζεται τις βασικές υπηρεσίες του Android και χρησιμοποιείται ως επίπεδο αφαίρεσης υλικού (HAL – hardware abstraction layer) ανάμεσα στη στοίβα λογισμικού και στο φυσικό υλικό της συσκευής [23].

Κάποιες από τις βασικές λειτουργίες που χειρίζονται από το Linux πυρήνα είναι:

- Επιβολή δικαιωμάτων χρήσης και ασφάλειας εφαρμογών.
- Διαχείριση μνήμης χαμηλού επιπέδου.
- Διαχείριση διεργασιών και αλληλουχία ενεργειών.
- Η στοίβα δικτύου.
- Οθόνη, είσοδος από πληκτρολόγιο, κάμερα, Wifi , μνήμη Flash, ήχος και πρόσβαση προγράμματος οδήγησης λειτουργίας σύνδεσης (επικοινωνία μεταξύ διεργασιών).

1.2.2 Libraries

Πάνω από το Linux πυρήνα βρίσκονται οι εγγενείς βιβλιοθήκες που είναι γραμμένες σε C και C++ .Οι βιβλιοθήκες αυτές παρέχουν λειτουργίες χαμηλού επιπέδου και υπολογισμούς εντατικών υπηρεσιών για την πλατφόρμα Android.

Οι εγγενείς βιβλιοθήκες χωρίζονται σε 4 βασικές κατηγορίες:

- Bionic Libc
- Function Libraries
- Native Servers
- Hardware Abstraction Libraries

1.2.2.1 Bionic Libc

Είναι μία προσαρμοσμένη runtime libc υλοποίηση πού είναι τροποποιημένη για ενσωματωμένη χρήση στις κινητές συσκευές βασισμένες στο Linux. Η κάθε διαδικασία περιλαμβάνει και εγγενείς βιβλιοθήκες, για αυτό το λόγο το μέγεθος των libc είναι μικρό. Επίσης παρέχει ενσωματωμένη υποστήριξη για κάποιες συγκεκριμένες υπηρεσίες Android. Έτσι μπορούμε να ενημερωθούμε εύκολα και γρήγορα για τις προεπιλεγμένες ιδιότητες του συστήματος (getprop(“my.system.property”, buff, default);) και να έχουμε δυνατότητες καταγραφής μηνυμάτων αποσφαλμάτωσης (LOGI(“Logging a message with priority ‘Info’”);.). Η Bionic Libc δεν είναι συμβατή με την GNU Libc, υποστηρίζει κάποιες επεκτάσεις της GNU Libc αλλά όχι όλες.

1.2.2.2 Function Libraries

Οι Function Libraries είναι οι βασικές βιβλιοθήκες που χρησιμοποιούνται από τις εφαρμογές. Μερικές από αυτές είναι:

Surface Manager: Χρησιμοποιείται στην σύνθεση του διαχειριστή παραθύρου με ενδιάμεση αποθήκευση απενεργοποιημένης οθόνης (off-screen buffering). Η ενδιάμεση αποθήκευση απενεργοποιημένης οθόνης σημαίνει ότι οι εφαρμογές δεν μπορούν να σχεδιάσουν στην οθόνη άμεσα και πρέπει να σχεδιάσουν στην ενδιάμεση μνήμη της απενεργοποιημένης οθόνης (off-screen buffer). Εκεί συνθέτονται με τα άλλα σχέδια και παίρνουν την τελική μορφή που θα βλέπει ο χρήστης.

SQLite: Το SQLite είναι μια βιβλιοθήκη που χρησιμοποιείται για τις συναλλαγές μικρού μεγέθους αποθήκευσης δεδομένων. Είναι η βιβλιοθήκη που χρησιμοποιήθηκε στην εργασία αυτή, για τον λόγω αυτό θα το αναλύσουμε περισσότερα στο κεφάλαιο 4.

Open GL/ ES: Χρησιμοποιείται στη σύνθεση των διάφορων δισδιάστατων και τρισδιάστατων γραφικών.

Media Framework: Το Media Framework βασίζεται στην πλατφόρμα PacketVideo OpenCORE. Το Media Framework προσφέρει την υποστήριξη όλων των τυπικών codec σε μορφή αρχείου, όπως το MPEG3, MPEG4, H264 και άλλα. Εάν κάποιος θέλει να χρησιμοποιήσει ένα codec που δεν υποστηρίζεται από τον Media Framework αλλά υποστηρίζεται από την συσκευή του μπορεί να κάνει plug-in.

Free Type: Χρησιμοποιείται ως renderer γραμματοσειρών στο Android.

SSL: Παρέχει κλάσεις και διασυνδέσεις που απαιτούνται για την χρήστη του πρωτοκόλλου Secure Sockets Layer (SSL).

Webkit: Το Webkit είναι ένα πρόγραμμα περιήγησης ανοιχτού κώδικα. Είναι επίσης το όνομα του πλαισίου που χρησιμοποιείται από τις εφαρμογές όπως Safari, Dashboard και άλλα. Χρησιμοποιείται επίσης και από το λειτουργικό σύστημα του Android. Υποστηρίζει CSS, Javascript, DOM, AJAX και rendering προσαρμοσμένης εμφάνισης.

1.2.2.3 Native Servers

Οι Native servers εκτελούνται για να πραγματοποιήσουν μερικές από τις βαριές διαδικασίες χειρισμού των συσκευών εισόδου και εξόδου στην πλατφόρμα Android. Οι διαδικασίες αυτές είναι:

SurfaceFlinger: Είναι ένας πολύ δυνατός εξυπηρετητής ο οποίος είναι ένας συνθέτης που λειτουργεί για όλο το σύστημα Android. Παίρνει τα Surfaces που τρέχουν σε διαφορετικές εφαρμογές και διαδικασίες και τα συνδυάζει σε μία ενδιάμεση μνήμη πλαισίου για να τα εξάγει στην οθόνη της

συσκευής. Ο εξυπηρετητής αυτός μπορεί να συνδυάσει τις δισδιάστατες και τρισδιάστατες επιφάνειες.

AudioFlinger: Κάνει την ίδια δουλειά που κάνει και το SurfaceFlinger αλλά σε επίπεδο ήχου. Δηλαδή συνδυάζει όλους τους ήχους όλων εφαρμογών που εκτελούνται στην συσκευή και τα εξάγει στο ηχείο ή στο ακουστικό της συσκευής.

1.2.2.4 Hardware Abstraction Libraries

Οι Hardware Abstraction Libraries(HAL) είναι οι βιβλιοθήκες που δίνουν την δυνατότητα στο σύστημα Android να καλέσει τους οδηγούς των συσκευών χωρίς να γνωρίζει τα χαμηλά επίπεδα υλοποιήσεων των οδηγών και υλικών. Μερικές από τις HAL που παρέχουν καλύτερο Abstraction Layer μεταξύ Hardware και μεγαλύτερων επιπέδων της πλατφόρμας Android είναι:

- Audio
- Camera
- Bluetooth
- GPS
- Radio
- WiFi

1.2.3 Android Runtime

Οι εφαρμογές Android λειτουργούν με την χρήση μιας εικονικής μηχανής, η κάθε εφαρμογή έχει την δικιά της εικονική μηχανή. Οι εφαρμογές Android είναι κώδικας υπό διαχείριση που σημαίνει ότι είναι λιγότερο πιθανό να προκαλέσουν το κρασάρισμα του τηλεφώνου. Το Android χρησιμοποιεί την εικονική μηχανή και τον κώδικα υπό διαχείριση για να τρέξει το «dex-code» (Dalvik Executable), το οποίο συνήθως μεταφράζεται από ένα Java Bytecode. Το 2014, με το Android Lollipop, η εικονική μηχανή Dalvik αντικαταστήθηκε με το Android Runtime (ART), το οποίο είναι ένα περιβάλλον εφαρμογών χρόνου εκτέλεσης. Το ART υπήρχε και στο Android KitKat αλλά ήταν απενεργοποιημένη εξ ορισμού. Αποσκοπεί στη βελτιωμένη απόδοση των εφαρμογών και στη βελτίωση της χρήσης της μπαταρίας. Οι εφαρμογές που έχουν αναπτυχθεί για το DALVIK θα πρέπει να λειτουργούν και με το ART. Ενώ υπάρχουν μερικές παλιές τεχνικές που δεν λειτουργούν στο ART. Μερικές από τις βασικές ιδιότητες του ART είναι:

Ahead-of-time (AOT) compilation: Η έννοια αυτή εισάγεται από το ART και βελτιώνει τις επιδόσεις τις εφαρμογής. Το ART επίσης, κάνει αυστηρότερο έλεγχο από το Dalvik στην ώρα της εγκατάστασης.

Improved garbage collection: Η διαδικασία αυτή μπορεί να μειώσει την απόδοση μιας εφαρμογής με αποτέλεσμα να έχουμε μια ασταθής οθόνη, κακή ανταπόκριση της UI και άλλα προβλήματα. Με το ART βελτιώνετε το garbage collection του Android με διάφορους τρόπους.

Στο επίπεδο αυτό πάνω από την εικονική μηχανή βρίσκονται τα core libraries τα οποία παρέχουν μια δυνατή και απλή πλατφόρμα ανάπτυξης. Οι βιβλιοθήκες αυτές μας παρέχουν:

- Δομές δεδομένων
- Βοηθητικά προγράμματα
- Πρόσβαση στο δίκτυο
- Γραφικά

1.2.4 Application Framework

Το επίπεδο αυτό βρίσκεται πάνω από το Android Runtime και έχει γραφτεί σε γλώσσα Java. Στο επίπεδο αυτό περιλαμβάνονται όλες οι κλάσεις και οι υπηρεσίες που θα χρησιμοποιηθούν από τις εφαρμογές μας. Μπορούμε να πούμε ότι είναι οι βασικές εφαρμογές που έχουν αναπτυχθεί από τον οργανισμό Open Handset Alliance και είναι ανοιχτής πηγής. Το επίπεδο αυτό αποτελείται από τα παρακάτω τμήματα:

Core Platform Services: Είναι οι υπηρεσίες που είναι βασικές για την πλατφόρμα Android, διαχειρίζονται τον κύκλο ζωής της εφαρμογής, τα πακέτα, τους πόρους και τυπικά οι εφαρμογές δεν έχουν άμεση πρόσβαση σε αυτές. Μερικές από τις υπηρεσίες αυτές είναι:

- Activity Manager
- Package Manager
- Window Manager
- Resource Manager
- Content Providers
- View System

Hardware Services: Οι υπηρεσίες αυτές μας δίνουν την δυνατότητα της πρόσβασης στα APIs των χαμηλότερων επιπέδων. Τυπικά η πρόσβαση στις βιβλιοθήκες αυτές γίνεται μέσω του τοπικού αντικειμένου Manager. Μερικές από τις υπηρεσίες που παρέχονται είναι:

- Telephony service

- Location Service
- Bluetooth Service
- WiFi Service
- USB Service
- Sensor Service

1.2.5 Applications

Το επίπεδο αυτό έχει την ψηλότερη θέση στην αρχιτεκτονική του Android και είναι η κοντινότερη στον χρήστη. Περιλαμβάνει όλες τις εφαρμογές που έχουν πρόσβαση σε όλες τις υπηρεσίες και βιβλιοθήκες που εξηγήσαμε παραπάνω. Στο επίπεδο αυτό μπορούμε να δημιουργήσουμε άλλες υπηρεσίες και εφαρμογές μέσω της γλώσσας JAVA.

1.3 Τα Θεμελιώδη των Εφαρμογών

Οι εφαρμογές Android αναπτύσσονται στη γλώσσα προγραμματισμού Java. Το εργαλείο Android SDK μεταγλωττίζει τον κώδικά μας με όλα τα αρχεία που περιέχει και το μεταφράζει σε ένα APK αρχείο που είναι ένα αρχείο Android με κατάληξη .apk. Το αρχείο αυτό περιλαμβάνει όλα τα συστατικά που χρειαζόμαστε για να εγκαταστήσουμε την εφαρμογή στην συσκευή μας.

Εφόσον γίνεται η εγκατάσταση μίας εφαρμογής, η εφαρμογή αυτή θα έχει ένα δικό της χώρο στη συσκευή με τα ειδικά δικαιώματα που έχουν δοθεί από τον χρήστη κατά την διάρκεια της εγκατάστασης. Στην πραγματικότητα, το λειτουργικό σύστημα Android είναι ένα Linux σύστημα πολλών χρηστών στο οποίο η κάθε εφαρμογή είναι ένας χρήστης. Παρέχεται ένα ID χρήστη από το λειτουργικό σύστημα στην κάθε εφαρμογή έτσι ώστε η κάθε εφαρμογή να έχει τα δικά της δικαιώματα. Επίσης η κάθε διαδικασία έχει τη δικιά της εικονική μηχανή και μπορεί να τρέχει ανεξάρτητα από τις υπόλοιπες εφαρμογές της συσκευής. Εξ ορισμού, η κάθε εφαρμογή έχει την δικιά της διαδικασία Linux. Το Android ξεκινά την διαδικασία όταν κάποια από τα συστατικά της εφαρμογής πρέπει να τρέξουν. Μία διαδικασία κλείνει όταν δεν είναι πλέον αναγκαία ή όταν το σύστημα χρειάζεται θέσεις μνήμης για τις υπόλοιπες εφαρμογές.

Με τον τρόπο αυτό, το σύστημα υλοποιεί την "αρχή των ελάχιστων προνομίων"(the principle of least privilege). Με την αρχή αυτή εννοούμε ότι η κάθε εφαρμογή, εξ ορισμού, έχει πρόσβαση μόνο στα στοιχεία που απαιτούνται για να κάνει τη δουλειά του και τίποτα περισσότερο. Αυτό δημιουργεί ένα πολύ ασφαλές περιβάλλον στο οποίο δεν μπορεί μία εφαρμογή να έχει πρόσβαση στα τμήματα του συστήματος που δεν έχει άδεια.

Υπάρχουν διάφοροι τρόποι που μία εφαρμογή μπορεί να μοιράζεται δεδομένα με τις υπόλοιπες εφαρμογές της συσκευής ή να έχει πρόσβαση στις υπηρεσίες του συστήματος.

Ο πρώτος τρόπος είναι να δώσουμε τον ίδιο Linux ID χρήστη στις εφαρμογές που θέλουμε να μοιράζονται τα αρχεία. Στην περίπτωση αυτή η κάθε εφαρμογή έχει πρόσβαση στα αρχεία της άλλης που έχει το ίδιο Linux ID χρήστη. Οι εφαρμογές αυτές μπορούν να μοιράζονται την ίδια εικονική μηχανή και να τρέχουν επίσης στην ίδια διαδικασία.

Ο δεύτερος τρόπος είναι να δώσουμε σε μία εφαρμογή το δικαίωμα να έχει πρόσβαση στα διάφορα δεδομένα της συσκευής όπως τα μηνύματα, οι επαφές, η κάμερα το Wifi και άλλα. Τα δικαιώματα αυτά πρέπει να αναφερθούν από τον προγραμματιστή για να επαληθευθούν από τον χρήστη κατά την διάρκεια της εγκατάστασης.

Τα δικαιώματα αυτά πρέπει να δηλώνονται στο αρχείο Manifest του Android. Στα επόμενα υποκεφάλαια σας παρουσιάζουμε τα βασικά συστατικά ενός πλαισίου, το αρχείο Manifest και τους πόρους μίας εφαρμογής.

1.3.1 Τα Συστατικά μίας Εφαρμογής

Τα συστατικά μίας εφαρμογής είναι τα βασικά τμήματα που αποτελείται μία εφαρμογή Android. Το κάθε συστατικό είναι ένα διαφορετικό σημείο το οποίο δίνει τη δυνατότητα πρόσβασης στο σύστημα. Ορισμένα συστατικά εξαρτώνται από άλλα για να μπορούν να δώσουν την δυνατότητα πρόσβασης στον χρήστη.

Υπάρχουν τέσσερις διαφορετικοί τύποι συστατικών εφαρμογής. Κάθε τύπος εξυπηρετεί έναν συγκεκριμένο σκοπό και έχει τον δικό του κύκλο ζωής το οποίο καθορίζει πως γίνεται η δημιουργία και η καταστροφή του συστατικού .

Παρακάτω βλέπουμε τα τέσσερα συστατικά μίας εφαρμογής Android:

Activity (Δραστηριότητα): Ένα Activity (δραστηριότητα) αντιπροσωπεύει μία μοναδική οθόνη σε μία διεπαφή χρήστη. Οι δραστηριότητες αναλύονται περισσότερα στα επόμενα κεφάλαια.

Service (Υπηρεσία): Μία υπηρεσία είναι ένα συστατικό το οποίο δεν παρέχει διεπαφή χρήστη και τρέχει στο παρασκήνιο για να εκτελέσει μακρόχρονες λειτουργίες ή να εκτελέσει τις εργασίες για απομακρυσμένες διαδικασίες. Οι υπηρεσίες είναι τα άλλα συστατικά που θα αναλυθούν επίσης στα επόμενα κεφάλαια της εργασίας.

Content Provider (Προμηθευτής Περιεχομένου): Ο προμηθευτής περιεχομένου είναι ένα συστατικό το οποίο διαχειρίζεται τα δεδομένα που μοιράζονται από τις εφαρμογές. Μπορούμε να αποθηκεύσουμε τα δεδομένα στο σύστημα αρχείου, σε μία βάση δεδομένων SQLite, στο Web ή σε οποιοδήποτε μέσο αποθήκευσης δεδομένων που έχει πρόσβαση η εφαρμογή μας. Μέσω του προμηθευτή οι

εφαρμογές μπορούν να κάνουν ερωτήματα στα δεδομένα των άλλων εφαρμογών ή ακόμα και να διαμορφώσουν τα δεδομένα των άλλων εφαρμογών. Για παράδειγμα το λειτουργικό σύστημα Android μας παρέχει έναν προμηθευτή για τις επαφές της συσκευής που με αυτόν τον τρόπο μπορεί μία εφαρμογή που έχει τα δικαιώματα να τις διαβάζει ή να τις αλλάξει.

Broadcast Receiver (Δέκτης Μετάδοσης): Ένας δέκτης εκπομπής είναι ένα στοιχείο που ανταποκρίνεται στη μετάδοση προθέσεων (Intent Listeners). Πολλές από τις μεταδόσεις προέρχονται από το σύστημα, για παράδειγμα η οθόνη έχει απενεργοποιηθεί, η μπαταρία είναι χαμηλή ή έχει γίνει ένα Screen-Shot. Οι εφαρμογές ξεκινάνε επίσης μεταδόσεις για να ξέρουν οι υπόλοιπες εφαρμογές ότι έχουν εγκατασταθεί δεδομένα στη συσκευή και είναι διαθέσιμα για χρήση. Αν και οι Broadcast receivers δεν εμφανίζουν μια διεπαφή χρήστη μπορούν να δημιουργήσουν μια ειδοποίηση στη γραμμή κατάστασης που να ειδοποιεί το χρήστη για ένα "συμβάν εκπομπής".

Μία μοναδική πτυχή του σχεδιασμού του συστήματος Android είναι ότι οποιαδήποτε εφαρμογή μπορεί να εκκινήσει συστατικά μίας άλλης εφαρμογής. Για παράδειγμα όταν ο χρήστης θέλει να τραβήξει μία φωτογραφία με την κάμερα της συσκευής, υπάρχει πιθανότατα ήδη μια άλλη εφαρμογή που το κάνει και η εφαρμογή μπορεί να το χρησιμοποιήσει.

Με την εκκίνηση ενός συστατικού από το σύστημα, αρχικοποιούνται οι κατάλληλες κλάσεις για το συστατικό και ενεργοποιείται μία διαδικασία για την εφαρμογή. Επειδή το σύστημα εκτελεί την κάθε εφαρμογή σε ξεχωριστή διαδικασία με άδειες αρχείων που περιορίζουν την πρόσβαση σε άλλες εφαρμογές, η εφαρμογή μας δεν μπορεί να ενεργοποιήσει άμεσα ένα συστατικό από μία άλλη εφαρμογή, το σύστημα Android όμως μπορεί. Έτσι, για να ενεργοποιήσουμε ένα συστατικό σε μία άλλη εφαρμογή πρέπει να μεταφέρουμε ένα μήνυμα στο σύστημα το οποίο καθορίζει την πρόθεση μας να ξεκινήσουμε ένα συγκεκριμένο συστατικό. Στη συνέχεια, το συστατικό ενεργοποιείται από το σύστημα.

1.3.2 Το αρχείο Manifest

Πριν ξεκινήσει ένα συστατικό από το σύστημα Android, το σύστημα πρέπει να ξέρει ότι το συστατικό υπάρχει. Το εάν υπάρχει ένα συστατικό ή όχι καταγράφεται στο αρχείο Manifest μίας εφαρμογής που είναι ένα xml αρχείο (AndroidManifest.xml). Οι εφαρμογές πρέπει να καθορίζουν τα συστατικά τους σε αυτό το αρχείο το οποίο πρέπει να είναι η ρίζα του καταλόγου ενός έργου.

Το αρχείο εκτελεί και άλλες εργασίες εκτός από τον καθορισμό συστατικών μίας εφαρμογής, μερικά από αυτά είναι:

- Η διαπίστωση όλων των δικαιωμάτων χρήστη που απαιτεί η εφαρμογή, όπως η πρόσβαση στο διαδίκτυο ή το δικαίωμα πρόσβασης (read-access) επαφών του χρήστη.

- Η δήλωση του ελάχιστου επιπέδου API που απαιτεί η εφαρμογή, με βάση το ποιο API χρησιμοποιείται από τον χρήστη.

- Η δήλωση των δυνατοτήτων του υλικού ή λογισμικού που χρησιμοποιούνται ή απαιτούνται από την εφαρμογή, όπως η κάμερα ή η υπηρεσία Bluetooth.

Το αρχείο Manifest είναι ένα σημαντικό κομμάτι μίας εφαρμογής Android και θα γίνει αναφορά και περισσότερη ανάλυση του αρχείου αυτού στο δεύτερο κεφάλαιο της πτυχιακής εργασίας.

1.3.3 Οι πόροι των εφαρμογών

Μία εφαρμογή Android δεν αποτελείται μόνο από κώδικα, απαιτεί και πόρους που διαχωρίζονται από τον πηγαίο κώδικα, όπως οι εικόνες, τα αρχεία ήχου και άλλα που έχουν σχέση με την εικονική εμφάνιση της εφαρμογής. Για παράδειγμα, πρέπει να ορίσουμε animations, μενού, χρώματα, και το Layout μίας δραστηριότητας με ένα αρχείο xml. Η χρήση των πόρων διευκολύνουν την διαδικασία των αλλαγών των χαρακτηριστικών της εφαρμογής μας χωρίς να χρειάζεται να τροποποιήσουμε τον κώδικα της εφαρμογής. Επίσης, με διάφορους πόρους που θα χρησιμοποιηθούν σε διαφορετικές καταστάσεις μπορούμε να βελτιώσουμε την απόδοση της εφαρμογής μας, όπως διαφορετικές διαστάσεις οθονών και διαφορετικές γλώσσες.

Για τον κάθε πόρο που εισάγουμε σε ένα έργο Android, το SDK δημιουργεί ένα μοναδικό ακέραιο ID, το οποίο μπορούμε να το χρησιμοποιήσουμε σαν αναφορά στον κώδικα μας ή στους άλλους πόρους που έχουν καθοριστεί στο xml. Για παράδειγμα, όταν η εφαρμογή μας περιέχει ένα αρχείο εικόνας με όνομα picture.png, το οποίο είναι αποθηκευμένο στον κατάλογο res/directory/, το εργαλείο SDK δημιουργεί ένα ID με όνομα R.drawable.picture, το οποίο μπορούμε να το αναφέρουμε στον κώδικά μας και να το εισάγουμε στην διεπαφή χρήστη μας.

Ένα από τα μοναδικά χαρακτηριστικά που μας προσφέρουν οι πόροι είναι η δυνατότητα επιλογής διαφόρων πόρων για διάφορες διαμορφώσεις συσκευών στην ίδια εφαρμογή. Για παράδειγμα, καθορίζοντας συμβολοσειρές διεπαφών χρήστη σε xml, μπορούμε να μεταφράσουμε τις συμβολοσειρές σε μία άλλη γλώσσα και να τις αποθηκεύσουμε σε διαφορετικά αρχεία xml. Στη συνέχεια, όταν εκτελείται η εφαρμογή μας, σύμφωνα με το όνομα που έχουμε δώσει στο αρχείο γλώσσας xml και την γλώσσα που χρησιμοποιείται στην συσκευή, το σύστημα με την χρήση του προσδιοριστή γλώσσας διαλέγει την καταλληλότερη γλώσσα για τον χρήστη. Για παράδειγμα όταν η συσκευή χρησιμοποιείται στα γαλλικά από τον χρήστη και έχουμε ένα αρχείο με όνομα res/values-fr/ στο έργο τότε οι τιμές των

συμβολοσειρών που θα χρησιμοποιηθούν θα είναι αυτές. Εάν δεν υπάρχει το αρχείο αυτό το σύστημα αυτόματα διαλέγει το αρχείο συμβολοσειρών που είναι το res/values/ εξ ορισμού.

Το σύστημα Android υποστηρίζει διάφορους προσδιοριστές για τους εναλλακτικούς πόρους μας. Ένας προσδιοριστής είναι μία μικρή συμβολοσειρά που συμπεριλαμβάνεται στο όνομα του πόρου μας και έχει στόχο να καθορίζει τη διαμόρφωση της συσκευής για να αποφασιστεί ποιους από τους εναλλακτικούς πόρους θα χρησιμοποιηθεί από το σύστημα. Θα μιλήσουμε περισσότερα για τους πόρους στον δεύτερο κεφάλαιο.

1.4 Εργαλεία Ανάπτυξης Εφαρμογών

Οι εφαρμογές Android είναι συνήθως αναπτυγμένες με την γλώσσα Java και το Android SDK. Υπάρχουν δυο γνωστά περιβάλλοντα ανάπτυξης εφαρμογών Android. Το πρώτο είναι το Eclipse. Στο Eclipse πρέπει να γίνει η εισαγωγή του Android SDK για να μπορούμε να αναπτύξουμε εφαρμογές Android. Το δεύτερο και το πιο καινούργιο είναι το Android Studio. Το Android Studio βασίζεται στην IntelliJ IDEA και είναι πλέον το επίσημο IDE στην ανάπτυξη εφαρμογών Android. Το Eclipse με την ενσωμάτωση ADT(Android Development Tool), είναι το εργαλείο που χρησιμοποιήθηκε στην εργασία αυτή. Η επιλογή του Eclipse έγινε για τους παρακάτω λόγους:

Α) Όταν ξεκίνησε η εργασία αυτή το Android Studio ήταν σε δοκιμαστική περίοδο και δεν ήταν το επίσημο IDE του Android.

Β) Εξοικείωση ανάπτυξης εφαρμογών με το Eclipse.

Γ) Το eclipse παρόλο που δεν είναι το επίσημο IDE του Android είναι γρηγορότερο από το Android Studio.

Το μειονέκτημα του Eclipse είναι ότι πρέπει να εισάγουμε το ADT για να αρχίσουμε την ανάπτυξη εφαρμογών Android, ενώ στο Android Studio δεν χρειάζεται να κάνουμε την διαδικασία αυτή αρκεί να εγκαταστήσουμε το IDE[24][25].

Για να τρέξουμε και να εξετάσουμε τις εφαρμογές που γράφουμε, όλα τα IDE ανάπτυξης εφαρμογών Android μας προσφέρουν προσομοιωτές. Οι προσομοιωτές αυτοί είναι αρκετά αργοί και δεν μπορούν να υποστηρίξουν πολλές ιδιότητες όπως το GPS και διάφορους αισθητήρες που υπάρχουν στις συσκευές Android. Εκτός από αυτά, μερικές φορές μπορούν να μην εμφανίζονται σωστά τα στοιχεία της εφαρμογής μας. Για τους λόγους αυτούς χρησιμοποιήσαμε συσκευές Android στην ανάπτυξη και μεταγλώττιση της εφαρμογής μας. Για να εκτελέσουμε και να μεταγλωττίσουμε μία εφαρμογή μέσω του Eclipse χρησιμοποιούμε το USB καλώδιο. Στην περίπτωση που δεν μπορούμε να χρησιμοποιήσουμε πραγματικές συσκευές Android ή θέλουμε να δοκιμάσουμε την εφαρμογή μας με διάφορες συσκευές με διάφορα μεγέθη οθονών μπορούμε να χρησιμοποιήσουμε εξωτερικούς προσομοιωτές που είναι γρηγορότεροι από τους προσομοιωτές των

IDE. Ένας από αυτούς είναι το Genymotion που έχει χρησιμοποιηθεί κατά την διάρκεια της ανάπτυξης της εργασίας μας. Η πρώτη διαφημιστική έκδοση της Genymotion έχει κυκλοφορήσει τον Νοέμβριο του 2013 από την γαλλική εταιρία Genymotion group. Η εγκατάσταση του προσομοιωτή μπορεί να γίνει από την επίσημη ιστοσελίδα [26]

Ένα ακόμα βασικό εργαλείο που χρησιμοποιήθηκε κατά την ανάπτυξη της εργασίας είναι το “DB Browser for SQLite”. Το εργαλείο αυτό είναι ανοιχτού κώδικα και μας δίνει τη δυνατότητα να δημιουργήσουμε, να σχεδιάσουμε και να επεξεργαστούμε τα αρχεία βάσεων δεδομένων SQLite. Κατά την ανάπτυξη της εργασίας το DB Browser for SQLite μας βοήθησε στην καλύτερη κατανόηση των σχημάτων των βάσεων δεδομένων που χρησιμοποιούσαμε και στην επαλήθευση των ερωτημάτων που εκτελέσαμε με την γλώσσα EGQL. Έχει δημιουργηθεί από τον Mauricio Piacetini, ως Arca Database Browser. Τον Αύγουστο του 2014, το project μετονομάστηκε σε “Database Browser for SQLite" μετά από την απαίτηση που είχε ο Richard Hipp(δημιουργός του SQLite), επειδή το προηγούμενο όνομα δημιουργούσε ακούσια προβλήματα υποστήριξης. Τον Σεπτέμβριο του 2014 το project μετονομάστηκε σε “DB Browser for SQLite” για την αποφυγή του μπερδέματος με την υπάρχουσα εφαρμογή “Database Browser”. Μπορούμε να εγκαταστήσουμε το εργαλείο αυτό από την επίσημη ιστοσελίδα του [27].

ΚΕΦΑΛΑΙΟ 2: ΤΑ ΒΑΣΙΚΑ ΤΗΣ ΣΧΕΔΙΑΣΗΣ ΕΦΑΡΜΟΓΩΝ ΜΕ ΤΟ ANDROID

2.1 Τα Βασικά Στοιχεία μίας Εφαρμογής Android

Οι εφαρμογές Android μπορούν να τρέξουν σε διάφορες συσκευές. Πολλές από αυτές έχουν περιορισμένους πόρους και απαιτούν μία καλή κατανόηση του κύκλου ζωής της εφαρμογής και του τρόπου λειτουργίας. Για να κατανοήσουμε τον τρόπο λειτουργίας μίας εφαρμογής Android πρέπει να έχουμε καταλάβει μερικές βασικές έννοιες στοιχείων που χρησιμοποιούνται στην ανάπτυξη εφαρμογών Android. Σε αυτό το κεφάλαιο θα αναλύσουμε τα βασικά στοιχεία μίας εφαρμογής Android. Συγκεκριμένα θα αναλύσουμε τα παρακάτω στοιχεία:

- Context (Περιβάλλον)
- Activity (Δραστηριότητα)
- Intent (Πρόθεση)
- Service (Υπηρεσία)

2.1.1 Context

Ένα context είναι το κεντρικό στοιχείο μίας εφαρμογής που μπορούμε να το χρησιμοποιήσουμε για την διαχείριση των λειτουργιών ρύθμισης παραμέτρων, όπως και για τις λειτουργίες και δεδομένα μίας ολόκληρης εφαρμογής. χρησιμοποιούμε το Context για να μπορούμε να προσπελάσουμε τις ρυθμίσεις και τους πόρους που μοιράζονται σε Activities πολλαπλών στιγμιότυπων[28].

Για την ανάκτηση του Context μίας εφαρμογής μπορούμε να γράψουμε την εντολή:

Context context = getApplicationContext();

Με την ανάκτηση του Context έχουμε την δυνατότητα διαχειρίσεως των χαρακτηριστικών και των πόρων της εφαρμογής μας. Μπορούμε να ανακτήσουμε τους πόρους με την χρήση της εντολής getResources() του Context. Ο πιο άμεσος τρόπος για να ανακτήσουμε έναν πόρο της εφαρμογής είναι να χρησιμοποιήσουμε το μοναδικό ID χρήστη που έχει δοθεί στον κάθε πόρο από το σύστημα. Το ID αυτό παράγεται μέσα στην κλάση R.java και μπορεί να χρησιμοποιηθεί στην ανάκτηση και διαχείριση των πόρων στον κώδικά μας, όπως έχουμε εξηγήσει και στα προηγούμενα κεφάλαια. Για παράδειγμα, για την ανάκτηση στιγμιότυπου ενός String από τους πόρους χρησιμοποιούμε τον παρακάτω κώδικα:

String s greeting = getResources().getString(R.string.hello);

Μπορούμε επίσης να προσπελάσουμε τις προτιμήσεις μας μέσω του Context της εφαρμογής μας. Οι κοινές προτιμήσεις είναι αποθηκευμένες σε ένα ελαφρύ μηχανισμό αποθήκευσης δεδομένων και χρησιμοποιούνται για την αποθήκευση των προτιμήσεων της εφαρμογής και ισχύουν για όλα τα Activities. Με την χρήση της μεθόδου getSharedPreferences() του Context μπορούμε να ανακτήσουμε και να επεξεργαστούμε τις κοινές προτιμήσεις της εφαρμογής μας.

Το context μας δίνει την δυνατότητα πρόσβασης σε αρκετά από τα χαρακτηριστικά του συστήματος του υψηλού επιπέδου. Μερικά από αυτά είναι:

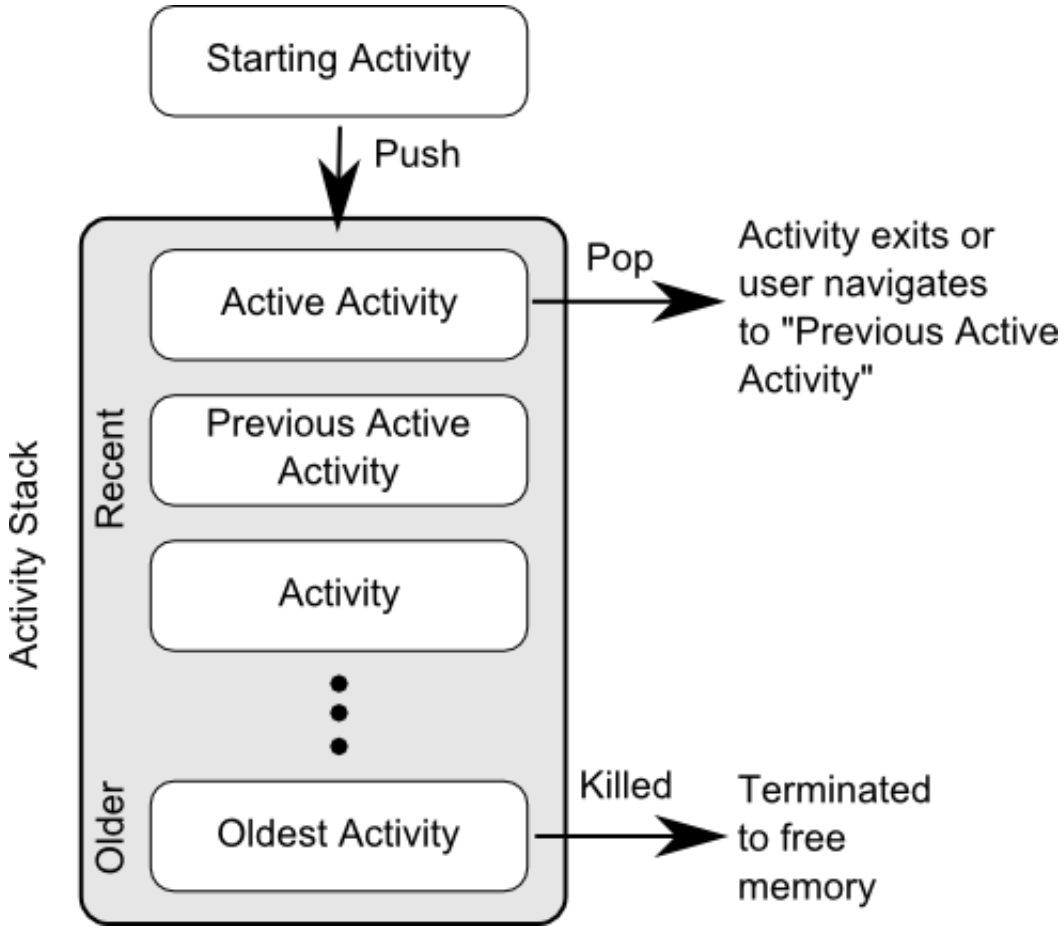
- Εκκίνηση στιγμιότυπων Activity
- Ανάκτηση πόρων που συσκευάζονται μαζί με την εφαρμογή
- Αίτηση υπηρεσίας συστήματος (π.χ. υπηρεσία παραθύρου που έχει χρησιμοποιηθεί και στην εργασία μας)
- Διαχείριση ιδιωτικών αρχείων, καταλόγων και βάσεων δεδομένων
- Επιθεώρηση και επιβολή δικαιωμάτων εφαρμογών

2.1.2 Activity

Ένα activity (android.app.Activity) αποτελεί θεμέλιο για οποιαδήποτε εφαρμογή Android. Τις περισσότερες φορές το κάθε activity είναι και μία οθόνη της εφαρμογής μας που είναι και μία δραστηριότητα που θα πραγματοποιηθεί από τον χρήστη. Για παράδειγμα, μία οθόνη για κύριο μενού είναι ένα activity ή μία οθόνη για υψηλότερες βαθμολογίες είναι ένα άλλο activity.

Συνήθως οι εφαρμογές Android περιλαμβάνουν πολλαπλές διεργασίες και το λειτουργικό σύστημα Android επιτρέπει να εκτελούνται πολλαπλές εφαρμογές ταυτόχρονα εφόσον υπάρχει διαθέσιμη μνήμη και επεξεργαστική ισχύς. Η κάθε εφαρμογή μπορεί να έχει διεργασίες στο παρασκήνιο και μπορεί να διακοπεί ή να παύεται από διάφορα συμβάντα. Συγκεκριμένα, ο χρήστης μπορεί να βλέπει μόνο ένα Activity στην οθόνη κάθε φορά. Το κάθε Activity που εκτελείται παρακολουθείται από το σύστημα Android και τοποθετείται σε μία στοίβα. Όταν ξεκινά ένα νέο Activity τοποθετείται στην κορυφή της στοίβας και το προηγούμενο διακόπτεται και πάει στο ένα κατώτερο επίπεδο της στοίβας μέχρι να σταματήσει η λειτουργία του Activity που εκτελείται. Μπορούμε να πούμε πως ο σωρός δραστηριοτήτων αποτελείται από 4 επίπεδα. Στο πάνω επίπεδο βρίσκεται το Activity που εκτελείται και μπορεί ο χρήστης να αλληλεπιδρά μαζί του. Στο δεύτερο επίπεδο του σωρού βρίσκεται το Activity που μπορεί να φαίνεται όταν το πρώτο Activity δεν γεμίζει όλη την οθόνη ή είναι διαφανή. Το Activity αυτό είναι σε κατάσταση παύσης και θα εκτελεστεί όταν πατήσει το κουμπί Πίσω ο χρήστης ή όταν το παραπάνω Activity καταστραφεί. Στο τρίτο επίπεδο είναι το Activity που δεν μπορεί ο χρήστης να δει ή να αλληλεπιδρά μαζί του. Είναι το Activity που

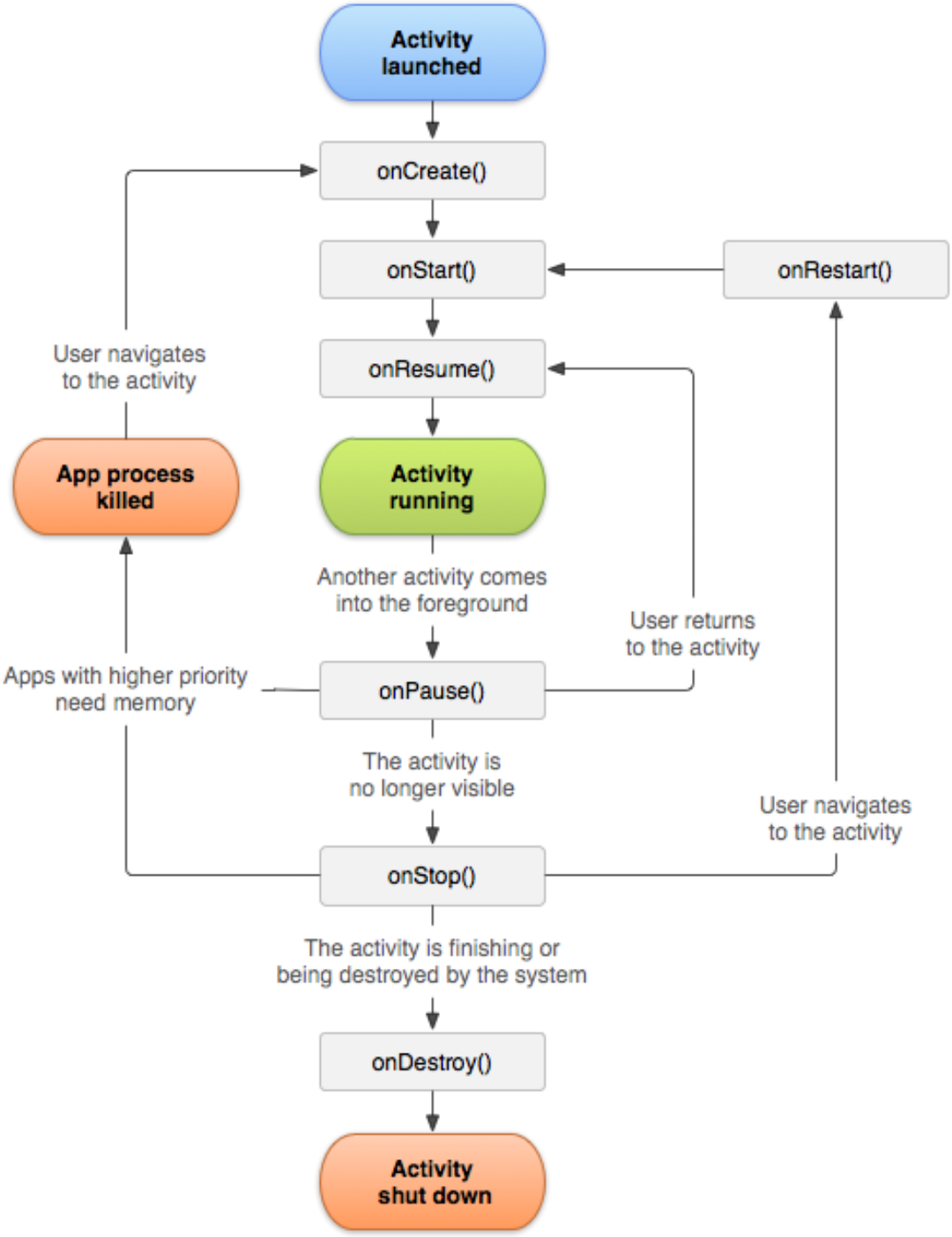
είναι σε κατάσταση διακοπής και για να ξεκινήσει η αλληλεπίδραση πρέπει να καταστραφούν όλοι οι πιο πάνω Activities από αυτόν. Στο τέταρτο επίπεδο βρίσκεται το Activity που είναι στο κατώτερο μέρος της στοίβας και μπορεί να καταστραφεί όταν οι παραπάνω Activities καταναλώνουν πολλούς πόρους. Η εικόνα 3 μας απεικονίζει τον σωρό των δραστηριοτήτων.



Εικόνα 3: Ο σωρός δραστηριοτήτων

Οι εφαρμογές Android είναι υπεύθυνες για τη διαχείριση της κατάστασης της μνήμης, των πόρων και των δεδομένων τους. Πρέπει να κατανοήσουμε τον κύκλο ζωής που έχει το κάθε Activity για να αναπτύξουμε στιβαρές εφαρμογές. Ο κύκλος ζωής ενός στοιχείου αντιπροσωπεύει τις διάφορες αλλαγές που συμβαίνουν στο στοιχείο σε διάφορες χρονικές στιγμές και από διάφορα συμβάντα από την έναρξη εκτέλεσης του στοιχείου μέχρι την καταστροφή του. Μπορούμε να χειριστούμε την κάθε κατάσταση ενός Activity έτσι ώστε να κάνουμε μία σωστή διαχείριση της εφαρμογής μας και των πόρων που χρησιμοποιεί. Οι αλλαγές που γίνονται στις καταστάσεις των κύκλων ζωής των Activities μπορούν να εκφραστούν με μία σειρά από σημαντικές αναστροφές κλήσεων μεθόδων. Οι μέθοδοι αυτές μπορούν να χρησιμοποιηθούν κατά την διάρκεια της ανάπτυξης μίας εφαρμογής Android έτσι ώστε να αναπτύξουμε μία εφαρμογή που εκμεταλλεύεται σωστά τη μνήμη του συστήματος.

Στην εικόνα 4 απεικονίζεται η ροή που ακολουθούν οι καταστάσεις ενός Activity [29].



Εικόνα 4: Η ροή καταστάσεων ενός Activity

Οι βασικοί βρόχοι της παρακολούθησης ενός Activity της εφαρμογής είναι:

Ολόκληρη διάρκεια του κύκλου ζωής: είναι η διάρκεια μεταξύ της πρώτης κλήσης του Activity με τη μέθοδο onCreate() μέχρι τη τελευταία

κλήση του Activity με τη μέθοδο onDestroy(). Ένα Activity θα κάνει όλες τις αρχικοποιήσεις του στη μέθοδο onCreate() και θα απελευθερώσει όλους τους πόρους στη μέθοδο onDestroy(). Για παράδειγμα, αν έχουμε ένα νήμα που τρέχει στο παρασκήνιο για να κατεβάσουμε δεδομένα, δημιουργείται και αρχικοποιείται στη μέθοδο onCreate του Activity και σταματάει τη λειτουργία του στη μέθοδο onDestroy().

Ορατή διάρκεια ενός Activity: Η ορατή διάρκεια ενός Activity είναι μεταξύ της μεθόδου onStart() και της onStop(). Κατά τη διάρκεια αυτή μπορεί ο χρήστης να βλέπει το Activity στην οθόνη, αν και μπορεί να μην είναι στο παρασκήνιο ή να αλληλεπιδρά με τον χρήστη. Μεταξύ αυτών των μεθόδων μπορούμε να διατηρήσουμε τους πόρους που χρειαζόμαστε για να εμφανίσουμε το Activity στον χρήστη. Για παράδειγμα, μπορούμε να καταχωρήσουμε ένα BroadcastReceiver στην onStart() για την παρακολούθηση των αλλαγών που γίνονται στη διεπαφή χρήστη μας, και να το καταργήσουμε στην onStop() όταν ο χρήστης δεν μπορεί να βλέπει πλέον αυτά που προβάλλουμε.

Το Προσκήνιο ενός Activity: Το προσκήνιο ενός Activity συμβαίνει μεταξύ της κλήσης της μεθόδου onResume() και της κλήσης της αντίστοιχης μεθόδου onPause(). Κατά τη διάρκεια αυτή, το Activity είναι μπροστά από όλους τους Activities και αλληλεπιδρά με τον χρήστη. Ένα Activity μπορεί να μπει πολύ συχνά σε αυτές τις καταστάσεις, για παράδειγμα όταν η συσκευή μπαίνει σε κατάσταση ύπνου (sleep mode) ή όταν παραδίδεται το αποτέλεσμα εκτέλεσης ενός άλλου Activity.Για αυτόν τον λόγο ο κώδικας σε αυτές τις μεθόδους πρέπει να είναι πολύ ελαφρύς.

Όλες οι μέθοδοι που μπορούμε να χειριστούμε τον κύκλο ζωής ενός Activity παρουσιάζονται στο παρακάτω κομμάτι κώδικα. Οι μέθοδοι αυτοί μπορούν να παρακάμπτονται (override) έτσι ώστε να κάνουμε τις αλλαγές που θέλουμε στην κατάλληλη κατάσταση του Activity. Όλα τα Activities θα υλοποιήσουν τη μέθοδο onCreate(Bundle) για να κάνουν τις βασικές αρχικοποιήσεις του Activity. Πολλούς από αυτούς θα υλοποιήσουν και την onPause() για να δεσμεύσουμε τις αλλαγές που γίνανε στα δεδομένα ή αλλιώς να σταματήσουμε την αλληλεπίδραση με τον χρήστη. Για να υλοποιήσουμε τις μεθόδους αυτές πρέπει να καλέσουμε την υπέρ-κλάση Activity.

```
Public class Activity extends ApplicationContext{
protected void onCreate(Bundle saved InstanceState);

protected void onStart();
```

```
protected void onRestart();

protected void onResume();

protected void onPause();

protected void onStop();

 protected void onDestroy();
}
```

Παρακάτω εξηγούμε την κάθε μέθοδο με λίγα λόγια:

onCreate(): Καλείται πρώτα όταν δημιουργείται το Activity. Είναι το σημείο που πρέπει να γίνουν όλες οι αρχικοποιήσεις, οι ορισμοί των view, οι συνδέσεις με τις λίστες δεδομένων κτλ. Η μέθοδος αυτή εάν υπάρχει μπορεί να μας παρέχει και την προηγούμενη κατάσταση του Activity με τη χρήση της Bundle. Πάντα ακολουθείται από την onStart().

onRestart(): Καλείται μόνο όταν το Activity μας έχει ήδη σταματήσει και θα μπει ξανά στην πρώτη θέση του σωρού. Ακολουθείται πάντα από την onStart().

onStart(): Καλείται όταν το Activity γίνεται ορατό στον χρήστη. Ακολουθείται από το onResume() εάν το Activity πάει στο προσκήνιο ή από την onStop() εάν πάει να γίνει κρυφό.

onResume():Καλείται όταν το Activity ξεκινάει την αλληλεπίδραση με τον χρήστη. Σε αυτό το σημείο το Activity μας είναι στην πρώτη θέση του σωρού. Ακολουθείται πάντα από την onPause().

onPause():Καλείτε όταν το σύστημα είναι έτοιμο να ξεκινήσει ένα άλλο Activity. Αυτή η μέθοδος χρησιμοποιείται συνήθως για την αποθήκευση των αλλαγών των δεδομένων, σταμάτημα εργασιών και άλλα που μπορούν να καταναλώνουν την CPU. Οι υλοποιήσεις που γίνονται εδώ πρέπει να πραγματοποιηθούν γρήγορα επειδή το επόμενο Activity δεν μπορεί να ξεκινήσει όταν δεν έχει τερματιστεί αυτή η μέθοδος.

onStop(): Καλείται όταν το Activity δεν είναι πλέον ορατό στον χρήστη, επειδή έχει ξεκινήσει μία άλλη λειτουργία να τρέχει και καλύπτει το Activity αυτό. Αυτό μπορεί να συμβαίνει όταν καλείται ένα άλλο Activity ή όταν καταστρέφεται αυτό που τρέχει. Ακολουθείται από την onRestart() εάν αυτό το Activity θα ξανά αλληλεπιδράσει με τον χρήστη ή από την onDestroy() εάν η Activity πάει να καταστραφεί.

onDestroy(): Καλείται πριν την καταστροφή του Activity. Αυτό μπορεί να συμβαίνει όταν το Activity έχει τερματίσει τις εργασίες του ή όταν το σύστημα καταστρέφει προσωρινά το στιγμιότυπο του Activity για να σώσει χώρο. Η διάκριση μεταξύ των σεναρίων αυτών μπορεί να γίνει με την βοήθεια της μεθόδου isFinishing().

2.1.3 Intent

Η πρόθεση(intent) είναι μία αφηρημένη περιγραφή μίας λειτουργίας που θα εκτελεστεί. Μπορεί να χρησιμοποιείται με το startActivity για να ξεκινήσουμε ένα Activity, με το broadcastIntent για να την στείλουμε σε κάθε συστατικό BroadcastReceiver που είναι ενεργοποιημένο και με το bindService(Intent, ServiceConnection, int) για να επικοινωνήσει με μία background υπηρεσία[30].

Οι προθέσεις συνδέουν μεμονωμένα συστατικά μεταξύ τους κατά το χρόνο εκτέλεσης, είτε το στοιχείο ανήκει στην εφαρμογή μας είτε σε κάποια άλλη. Η σημαντικότερη χρήση μίας πρόθεσης μπορεί να θεωρηθεί στην έναρξη των Activities που μπορεί να είναι ο συνδετικός κρίκος μεταξύ των Activities.

Η βασική δομή μίας πρόθεσης αποτελείται από δυο κομμάτια:

Action (Δράση) - Οι γενικές δράσεις που πρέπει να εκτελεστούν, όπως η ACTION_VIEW, η ACTION_EDIT, η ACTION_MAIN κτλ.

Data(Δεδομένα) - Τα δεδομένα που θα επεξεργαστούν, όπως μία εγγραφή στις βάσεις δεδομένων των επαφών.

Εκτός από αυτά τα κύρια χαρακτηριστικά, υπάρχουν και τα δευτερεύοντα χαρακτηριστικά που μπορεί να περιέχει μία πρόθεση:

Category (Κατηγορία): Μας δίνει επιπρόσθετη πληροφορία για την δράση που θα εκτελεστεί.

Type (Τύπος): Καθορίζει έναν συγκεκριμένο τύπο στην πρόθεση. Κανονικά ο τύπος μίας πρόθεσης ορίζεται από τα δεδομένα της. Ρυθμίζοντας αυτήν την ιδιότητα απενεργοποιούμε την ιδιότητα αυτή και επιβάλλουμε να χρησιμοποιηθεί ο συγκεκριμένος τύπος για την πρόθεση.

Component (Συστατικό): Καθορίζει ένα συγκεκριμένο όνομα της κλάσης του συστατικού για την χρήση της πρόθεσης.

Extras (Πρόσθετα): Είναι ένα Bundle που θα περιέχει τις τυχόν πρόσθετες πληροφορίες μίας πρόθεσης. Μπορεί να χρησιμοποιηθεί για να παρέχει εκτεταμένες πληροφορίες στο συστατικό.

Υπάρχουν δύο κύριες μορφές προθέσεων που χρησιμοποιούνται.

Explicit Intents (Ρητές Προθέσεις): Καθορίζουν το συστατικό (δηλώνοντας το όνομα), το οποίο παρέχει την ακριβή κλάση που θα εκτελεστεί. Συχνά,

δεν θα περιλαμβάνουν οποιαδήποτε άλλη πληροφορία. Με αυτόν τον τρόπο μπορεί μία εφαρμογή να ξεκινήσει διάφορες εσωτερικές δραστηριότητες με τις οποίες ο χρήστης αλληλεπιδρά.

Implicit Intents (Έμμεσες Προθέσεις): Αυτές δεν καθορίζουν ένα συγκεκριμένο συστατικό, αντί αυτού πρέπει να περιλαμβάνουν πληροφορίες που να βοηθάει στο σύστημα να μπορεί να καθορίσει ποια από τα διαθέσιμα συστατικά είναι καλύτερο να εκτελεστεί για αυτό το σκοπό.

Παρακάτω παρουσιάζουμε μερικά παραδείγματα των προθέσεων που έχουν χρησιμοποιηθεί και στην εργασία μας:

Ο παρακάτω κώδικας θα εμφανίσει την SettingsActivity.class στην οθόνη, που είναι ένα Activity που εμφανίζει στην οθόνη τις διαθέσιμες ρυθμίσεις

```
Intent intent = new Intent(this, SettingsActivity.class);
startActivity(intent);
```

Ο παρακάτω κώδικας θα εμφανίσει το ExecuteQueryTextActivity.class στην οθόνη, πριν το κάλεσμα της μεθόδου startActivity(intent) εισάγουμε μία συμβολοσειρά που αντιπροσωπεύει το ερώτημα SQL που δημιουργήθηκε στην οθόνη με τις χειρονομίες του χρήστη. Για να τρέξει το Activity και να μας εμφανίσει τα αποτελέσματα έχουμε εισάγει το ερώτημα που δημιουργήθηκε στα πρόσθετα της πρόθεσης μας.

```
Intent intent = new Intent (this, ExecuteQueryTextActivity.class);
intent.putExtra(QUERY_STRING, query);
startActivity(intent);
```

Ενώ, ο παρακάτω κώδικας βρίσκεται στην ExecuteQueryTextActivity.class και διαβάζει τα πρόσθετα της εισερχόμενης πρόθεσης που έχουν όνομα MainActivity.*QUERY_STRING,* δηλαδή την συμβολοσειρά του ερωτήματος που έχει εισαχθεί στην πρόθεση που καλέσαμε την ExecuteQueryTextActivity.class.

Το MainActivity.QUERY_STRING είναι μία public static final συμβολοσειρά που έχουμε δημιουργήσει για την ευκολότερη διαχείρηση των πρόσθετων των προθέσεων.

```
Intent intent = getIntent();
String query=intent.getStringExtra(MainActivity.QUERY_STRING);
```

2.1.4 Service

Μία υπηρεσία(Service) είναι ένα συστατικό εφαρμογής που αντιπροσωπεύει ή την επιθυμία μίας εφαρμογής να εκτελέσει μια λειτουργία μεγάλης διάρκειας ενώ αυτή δεν θα αλληλεπιδρά με τον χρήστη ή να παρέχει λειτουργίες για να χρησιμοποιήσουν άλλες εφαρμογές. Η κάθε κλάση υπηρεσίας πρέπει να έχει μία αντίστοιχη δήλωση <service> στο αρχείο manifest του πακέτου που βρίσκεται. Οι

υπηρεσίες μπορούν να ξεκινήσουν με την εντολή Context.startService() και Context.bindService()[31].

Πρέπει να σημειωθεί ότι οι υπηρεσίες τρέχουν στον κύριο νήμα της διαδικασίας, όπως όλα τα άλλα αντικείμενα μίας εφαρμογής. Αυτό σημαίνει ότι εάν η υπηρεσία μας πρόκειται να κάνει καμία ένταση στη CPU(όπως η αναπαραγωγή MP3) ή μπλοκάρισμα των λειτουργιών(όπως η δικτύωση), πρέπει να δημιουργηθεί ένα ειδικό νήμα για τις υπηρεσίες μας. Πρέπει να ξέρουμε ότι μία υπηρεσία δεν είναι μία ξεχωριστή διεργασία. Η υπηρεσία από μόνη της δεν σημαίνει ότι θα εκτελεστεί σε ξεχωριστή διεργασία. Εάν δεν ορίζουμε την υπηρεσία να τρέχει σε διαφορετική διεργασία, θα τρέχει στην ίδια διεργασία με την εφαρμογή μας. Επίσης μία υπηρεσία δεν είναι ένα νήμα από μόνη της.

Τα δυο βασικά χαρακτηριστικά που μας παρέχει μία υπηρεσία είναι:

- Η δυνατότητα να μπορεί η εφαρμογή να πει στο σύστημα να εκτελέσει διάφορα πράγματα στο background, χωρίς να χρειάζεται να αλληλεπιδρά με τον χρήστη. Αυτό αντιστοιχεί στην κλήση της μεθόδου Context.startService(), το οποίο ρωτάει στο σύστημα να προγραμματίσει εργασία για την υπηρεσία, μέχρι να την σταματήσει μία υπηρεσία ή κάτι άλλο.

- Οι εφαρμογές μπορούν να μοιράζονται τις λειτουργίες τους με τις άλλες εφαρμογές. Αυτό αντιστοιχεί στην κλήση της μεθόδου Context.bindService(), η οποία επιτρέπει μία μακροχρόνια σύνδεση με την υπηρεσία έτσι ώστε να μπορούμε να αλληλεπιδράσουμε με αυτήν.

Στον παρακάτω κώδικα βλέπουμε την δομή μίας δήλωσης υπηρεσίας στο αρχείο Manifest:

```
<application
...
        <service
        android:name="com.me.itapp.services.BulletinService"
        android:icon="@drawable/ic_launcher"
        android:label="BulletinService" >
        </service>
...
</application>
```

Έπειτα επεκτείνουμε την κλάση IntentService για να υλοποιήσουμε τον δομητή ορίζοντας το όνομα της υπηρεσίας και την μέθοδο onHandleIntent() η οποία πραγματοποιεί και την ουσιαστική εργασία.

```
public class DownloadService extends IntentService {
        public final static String URL = "com.me.itapp.services.URL";
        public DownloadService () {
        super("DownloadService");
```

```
}
@Override
protected void onHandleIntent(Intent intent) {
downloadFile();
postNotification();
}
}
```

Στην δραστηριότητα(ή το Fragment) για να εκκινήσουμε την υπηρεσία γράφουμε τον παρακάτω κώδικα:

```
Intent intent = new Intent(getApplicationContext(), DownloadService.class);
        intent.putExtra(DownloadService.URL, urlString);
        context.startService(intent);
```

2.2 Ορισμός Εφαρμογής με το Αρχείο Manifest του Android

Η κάθε εφαρμογή πρέπει να έχει ένα αρχείο manifest στο ριζικό κατάλογο του. Το όνομα του αρχείου έχει μεγάλη σημασία και πρέπει να είναι ακριβώς “AndroidManifest.xml”. Το αρχείο αυτό παρέχει τις βασικές πληροφορίες για την εφαρμογή μας στο σύστημα Android. Το σύστημα πρέπει να γνωρίζει τις πληροφορίες αυτές για να μπορεί να τρέξει τον οποιοδήποτε κώδικα της εφαρμογής μας [32]. Οι πληροφορίες αυτού του αρχείου χρησιμοποιούνται για να γίνουν οι εξής εργασίες:

- Ονομάζει το πακέτο Java μίας εφαρμογής. Το όνομα του πακέτου εξυπηρετεί ως ένα μοναδικό στοιχείο που προσδιορίζει την εφαρμογή.
- Περιγράφει τα στοιχεία της εφαρμογής, τα activities, τις υπηρεσίες, τα broadcast receivers και τα content providers από τα οποία αποτελείται μία εφαρμογή.
- Ονομάζει τις κλάσεις που υλοποιούν το κάθε συστατικό και δημοσιεύει τα χαρακτηριστικά τους (ποιών ειδών προθέσεις μπορούν να χειρίζονται). Με αυτές τις δηλώσεις μπορεί το σύστημα Android να ξέρει ποια είναι τα συστατικά και κάτω από ποιες συνθήκες μπορούν να εκτελεστούν.
- Καθορίζει ποιες διεργασίες θα φιλοξενήσουν τα συστατικά εφαρμογών.
- Δηλώνει ποια δικαιώματα πρέπει να περιέχει η εφαρμογή για να μπορεί να έχει πρόσβαση στα προστατευμένα κομμάτια των API και να αλληλεπιδρά με τις υπόλοιπες εφαρμογές.
- Επίσης δηλώνει τα δικαιώματα που πρέπει να έχουν οι άλλοι για να μπορούν να αλληλεπιδρούν με τα συστατικά της εφαρμογής.
- Αναφέρει τις κλάσεις οργάνων που παρέχουν προφίλ και άλλες πληροφορίες καθώς η εφαρμογή εκτελείται. Οι δηλώσεις αυτές

παρουσιάζονται στο manifest μόνο όταν η εφαρμογή αναπτύσσεται και μεταγλωττίζεται. Αφαιρούνται πριν την δημοσίευση της εφαρμογής.

- Δηλώνει το ελάχιστο επίπεδο Android API που απαιτεί η εφαρμογή για να μπορεί να εκτελεστεί.
- Αναφέρει τις βιβλιοθήκες που απαιτεί μία εφαρμογή.

Στον παρακάτω κώδικα δείχνουμε την γενική δομή του αρχείου manifest και το κάθε στοιχείο που μπορεί να περιέχει. Τα παρακάτω στοιχεία είναι τα μοναδικά στοιχεία που μπορούν χρησιμοποιηθούν στο manifest αρχείο, δεν μπορούμε να προσθέσουμε τα δικά μας.

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

<manifest>
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>
        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>
```

```
        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>
        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>
        <uses-library />

    </application>
</manifest>
```

2.2.1 Συμβάσεις αρχείων

Μερικές συμβάσεις και κανόνες εφαρμόζονται γενικά σε όλα τα στοιχεία και χαρακτηριστικά του αρχείου manifest.

Στοιχεία (Elements)

Μόνο τα στοιχεία <manifest> και <application> πρέπει να χρησιμοποιηθούν και μπορούν να εμφανιστούν μόνο μια φορά στο αρχείο. Τα περισσότερα από τα άλλα μπορούν να εμφανιστούν πολλές φορές ή καθόλου, παρόλο που τουλάχιστον μερικά από αυτά πρέπει να εμφανιστούν για να σχηματιστεί κάτι που έχει νόημα στο αρχείο manifest.

Με την ταξινόμηση των στοιχείων του ίδιου επιπέδου δεν αλλάζει τίποτα στο αρχείο. Για παράδειγμα τα στοιχεία <activity>, <provider> και <service> μπορούν να εμφανιστούν με οποιαδήποτε σειρά μέσα στο αρχείο. Η μόνη εξαίρεση είναι το στοιχείο <activity-alias> που πρέπει να ακολουθήσει το στοιχείο <activity>.

Παράμετροι (Attributes)

Θεωρητικά όλες οι παράμετροι είναι προαιρετικές. Ενώ υπάρχουν μερικές που πρέπει να καθοριστούν για να ολοκληρωθεί ο σκοπός ενός στοιχείου. Εκτός από μερικές παραμέτρους του ριζικού στοιχείου <manifest> , όλα τα ονόματα των παραμέτρων ξεκινάνε με ένα prefix “android”. Ένα παράδειγμα είναι το android:alwaysRetainTaskState.

Καθορισμός ονομάτων κλάσεων

Πολλά στοιχεία αντιστοιχούν σε Java αντικείμενα, περιλαμβανομένων και των στοιχείων για την εφαρμογή (το στοιχείο <application>) και των βασικών στοιχείων του – activities (<activity>), υπηρεσίες (<service>), broadcast receivers (<receivers>) και content providers (<provider>).

Εάν δηλώνουμε μία υποκλάση, όπως θα κάναμε για όλες τις κλάσεις συστατικών (Activity, Service, BroadcastReceiver και ContentProvider), η υποκλάση δηλώνεται μέσω της παραμέτρου name. Το name πρέπει να περιέχει τον σχεδιασμό όλου του πακέτου. Για παράδειγμα, μία υποκλάση υπηρεσίας θα δηλωνόταν ως εξής:

```
<manifest . . . >
   <application . . . >
      <service android:name="com.example.project.SecretService" . . . >
         . . .
      </service>
      . . .
   </application>
</manifest>
```

Ωστόσο, ως συντομογραφία, εάν ο πρώτος χαρακτήρας της συμβολοσειράς είναι μια περίοδος, η συμβολοσειρά θα προστεθεί στο όνομα του πακέτου της εφαρμογής:

```
<manifest package="com.example.project" . . . >
   <application . . . >
      <service android:name=".SecretService" . . . >
         . . .
      </service>
      . . .
   </application>
</manifest>
```

Όταν ένα συστατικό ξεκινάει, το Android δημιουργεί ένα στιγμιότυπο της υποκλάσης με ίδιο όνομα. Αν η υποκλάση δεν έχει καθοριστεί, δημιουργεί ένα στιγμιότυπο της βασικής κλάσης.

Πολλαπλές τιμές

Για να καθορίσουμε πολλαπλές τιμές στο ίδιο στοιχείο πρέπει να επαναλαμβάνουμε το στοιχείο για την κάθε τιμή. Για παράδειγμα, ένα φίλτρο πρόθεσης μπορεί να καταγράφει πολλές κινήσεις:

```
<intent-filter . . . >
   <action android:name="android.intent.action.EDIT" />
   <action android:name="android.intent.action.INSERT" />
   <action android:name="android.intent.action.DELETE" />
   . . .
</intent-filter>
```

Οι τιμές των πόρων

Μερικές ιδιότητες έχουν τιμές οι οποίες μπορούν να εμφανιστούν στον χρήστη, για παράδειγμα, μία ετικέτα και μία εικόνα σε ένα activity. Οι τιμές των ιδιοτήτων αυτών πρέπει να εντοπιστούν και για αυτό πρέπει να θέτονται από έναν πόρο ή θέμα. Οι τιμές των πόρων εκφράζονται με τον παρακάτω τρόπο:

@[*package*:]*type*:*name*

Το όνομα του πακέτου μπορεί να παραλειφθεί εάν ο πόρος είναι στον ίδιο πακέτο με την εφαρμογή, το type είναι ο τύπος του πόρου, όπως ένα “string” ή “drawable” και το name είναι το όνομα το οποίο προσδιορίζει τον συγκεκριμένο πόρο. Για παράδειγμα:

```
<activity android:icon="@drawable/smallPic" . . . >
```

Τιμές από ένα θέμα (theme) εκφράζονται με τον ίδιο τρόπο αλλά με ένα ‘?’ αντί για ‘@’

?[*package*:]*type*:*name*

Τιμές συμβολοσειρών

Όταν μία ιδιότητα είναι μία συμβολοσειρά πρέπει να χρησιμοποιήσουμε τα διπλά backslash (‘ \\ ‘) πριν την χρήση των συμβολοσειρών, για παράδειγμα το ‘\\n’ για καινούργια γραμμή ή το ‘\\uxxxxx’ για Unicode χαρακτήρες.

2.2.2 Τα στοιχεία του Αρχείου Manifest

Παρακάτω θα εξηγήσουμε μερικά βασικά στοιχεία του αρχείου Manifest που έχουν χρησιμοποιηθεί και στην εργασία μας

<manifest>

Το στοιχείο <manifest> είναι το ριζικό στοιχείο του αρχείου AndroidManifest.xml. Πρέπει να περιέχει ένα στοιχείο <application> και να προσδιορίσει τις ιδιότητες xmlns:android και package.

<uses-sdk>

Με το στοιχείο <uses-sdk> μπορούμε να εκφράσουμε την συμβατότητα της εφαρμογής μας με μία ή πολλές εκδόσεις της πλατφόρμας Android, με την χρήση ενός αριθμού που δηλώνει το API Level.

<uses-permission>

Το στοιχείο <uses-permission> απαιτεί τα δικαιώματα που χρειάζεται η εφαρμογή μας για να μπορεί να λειτουργεί σωστά. Τα δικαιώματα χορηγούνται από τον χρήστη κατά την διάρκεια της εγκατάστασης.

<application>

Με το στοιχείο <application> κάνουμε την δήλωση της εφαρμογής μας. Το στοιχείο αυτό περιέχει υπό-στοιχεία που δηλώνουν όλα τα συστατικά της εφαρμογής και έχει τις ιδιότητες που μπορούν να επηρεάσουν όλα τα συστατικά.

<activity>

Το <activity> δηλώνει ένα activity που υλοποιεί κομμάτι της εικονικής διεπαφής χρήστη της εφαρμογής. Όλα τα activities πρέπει να δηλώνονται με τα στοιχεία <activity> μέσα στο αρχείο manifest. Τα activities που δεν δηλώνονται δεν θα είναι ορατά από το σύστημα και δεν θα εκτελεστούν ποτέ.

<intent-filer>

Το <intent-filer> δηλώνει τους τύπους των προθέσεων που μπορεί ένα activity, μία υπηρεσία ή ένα broadcast receiver να απαντήσει. Περιέχει υπό-στοιχεία όπως <activity> <category> και<data>.

2.3 Διαχείριση πόρων εφαρμογών

Θα πρέπει πάντα να αποφεύγουμε την χρήση των πόρων στον κώδικα της εφαρμογής μας, όπως οι εικόνες και συμβολοσειρές, έτσι ώστε να μπορούμε να τους διατηρήσουμε ανεξάρτητα. Η εξωτερίκευση των πόρων από τον κώδικα μας επιτρέπει επίσης να παρέχουμε εναλλακτικούς πόρους που υποστηρίζουν ειδικές διαμορφώσεις, όπως διαφορετικές γλώσσες και διαφορετικά μεγέθη οθονών. Αυτή η ιδιότητα είναι ιδιαίτερα σημαντική επειδή όλες οι συσκευές Android είναι διαθέσιμες με διαφορετικές διαμορφώσεις. Προκειμένου να παρέχουμε συμβατότητα με διάφορες διαμορφώσεις πρέπει να οργανώσουμε τους πόρους μας στον κατάλογο του project res/, με την χρήση των διάφορων υποκαταλόγων που ομαδοποιούν τους πόρους με βάση το όνομα και τη διαμόρφωση [33].

Για οποιοδήποτε πόρο μπορούμε να δηλώσουμε πολλαπλούς εναλλακτικούς πόρους για την εφαρμογή μας.

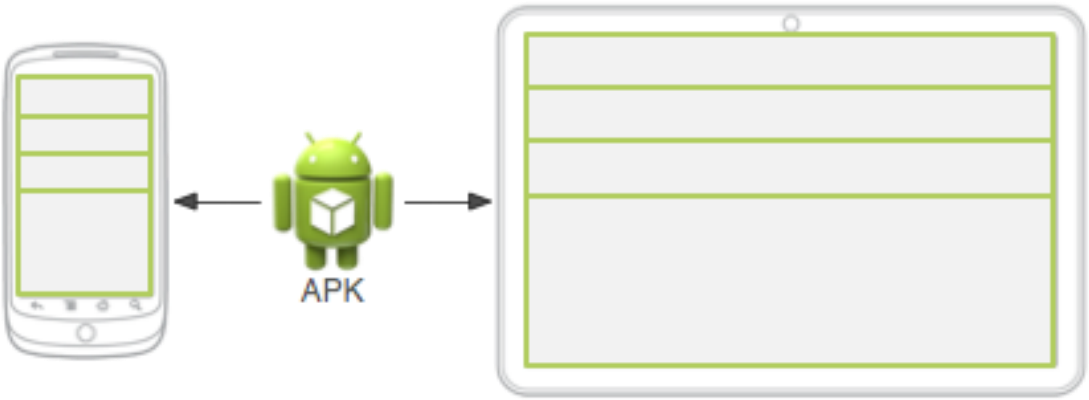
Οι default πόροι είναι εκείνοι που χρησιμοποιούνται ανεξάρτητα από την διαμόρφωση της συσκευής ή όταν δεν υπάρχουν εναλλακτικοί πόροι που ταιριάζουν με την τρέχουσα διαμόρφωση.

Οι εναλλακτικοί πόροι είναι εκείνοι που έχουμε σχεδιάσει για συγκεκριμένες διαμορφώσεις. Για να δηλώσουμε ότι μία ομάδα πόρων είναι για μία

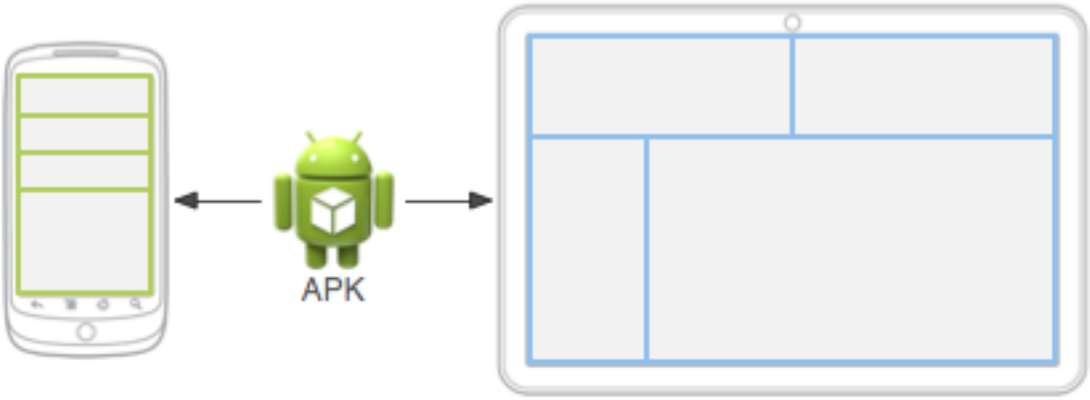
συγκεκριμένη διαμόρφωση, προσθέτουμε τους κατάλληλους προσδιοριστικούς χαρακτήρες στο όνομα του πόρου.

Για παράδειγμα εφόσον το default layout διεπαφής χρήστη είναι αποθηκευμένο στον κατάλογο res/layout/, θα μπορούσαμε να δηλώσουμε ένα διαφορετικό layout για να χρησιμοποιηθεί όταν η οθόνη είναι σε κατάσταση landscape orientation, αποθηκεύοντας απλά το layout στον κατάλογο res/layout-land/. Το σύστημα Android εφαρμόζει αυτόματα τους κατάλληλους πόρους που ταιριάζουν με την τρέχουσα διαμόρφωση της συσκευής και με τα ονόματα των καταλόγων των πόρων μας.

Η εικόνα 5 απεικονίζει το πώς το σύστημα εφαρμόζει την ίδια διάταξη για δυο διαφορετικές συσκευές όταν δεν υπάρχουν διαθέσιμες εναλλακτικές πηγές. Ενώ στην εικόνα 6 βλέπουμε τις ίδιες διατάξεις αλλά όταν έχουμε εναλλακτικούς πόρους για τις μεγάλες οθόνες.



Εικόνα 5: Δυο διαφορετικές συσκευές που χρησιμοποιούν τις default διατάξεις



Εικόνα 6: Δυο διαφορετικές συσκευές που η καθεμία χρησιμοποιεί τους παρεχόμενους πόρους για διαφορετικές διαστάσεις οθονών

Υπάρχουν διάφοροι τύποι πόρων που βασίζονται οι εφαρμογές Android, όπως συμβολοσειρές κειμένου, γραφικά και συνδυασμούς χρωμάτων, για την σχεδίαση μίας διεπαφής χρήστη.

Οι πόροι αποθηκεύονται στον κατάλογο /res του έργου μας σε ένα αυστηρό οργανωμένο σύνολο καταλόγων και αρχείων. Όλα τα ονόματα των αρχείων των πόρων του έργου αποτελούνται από γράμματα, αριθμούς και από χαρακτήρες υπογράμμισης underscore μόνο.

Παρακάτω βλέπουμε τον πίνακα 1 [33] που μας παρουσιάζει τους πόρους που υποστηρίζονται από το Android SDK και τους καταλόγους αποθήκευσης τους μέσα στο έργο.

Πίνακας 1

Κατάλογος	Τύπος πόρου
animator/	XML αρχεία που καθορίζουν τα property animations.
anim/	XML αρχεία που καθορίζουν τα tween animations. (Τα Property animations μπορούν να αποθηκεύονται και σε αυτόν τον κατάλογο αλλά το /animator προτιμάται για property animations για να μπορούν να ξεχωρίζονται.)
color/	XML αρχεία που καθορίζουν μία λίστα καταστάσεων των χρωμάτων.
drawable/	Bitmap αρχεία (.png, .9.png, .jpg, .gif) ή XML αρχεία που έχουν μεταγλωττιστεί στους παρακάτω υπό-τύπους των σχεδιάσιμων (drawable) πόρων: • Bitmap αρχεία • Nine-Patches • Λίστες καταστάσεων • Σχήματα κ.α.
mipmap/	Drawable αρχεία για διάφορες πυκνότητες εικόνες εκκίνησης.
layout/	XML αρχεία που καθορίζουν την διάταξη της διεπαφής του χρήστη.

menu/	XML αρχεία που καθορίζουν τα μενού της εφαρμογής, όπως το Options Menu, Context Menu ή το Sub Menu.
raw/	Αυθαίρετα αρχεία για την αποθήκευση των ακατέργαστων μορφών τους.
values/	XML αρχεία που περιέχουν απλές τιμές, όπως συμβολοσειρές, ακέραιους αριθμούς και χρώματα. Μερικά ονόματα xml αρχείων που περιέχει ο κατάλογος αυτός είναι: • arrays.xml για πίνακες πόρων • colors.xml για τιμές χρωμάτων • dimens.xml για τιμές διαστάσεων • strings.xml για τιμές συμβολοσειρών • styles.xml για μορφές
xml/	Αυθαίρετα XML αρχεία που μπορούν να διαβαστούν κατά την εκτέλεση της εφαρμογής με την κλήση της μεθόδου Resources.getXML(). Διάφορα αρχεία XML διαμορφώσεων μπορούν να αποθηκεύονται εδώ.

ΚΕΦΑΛΑΙΟ 3: ΔΗΜΙΟΥΡΓΙΑ ΔΙΕΠΑΦΩΝ ΧΡΗΣΤΗ ΚΑΙ ΓΡΑΦΙΚΑ

3.1 Η Αρχιτεκτονική των Γραφικών του Android

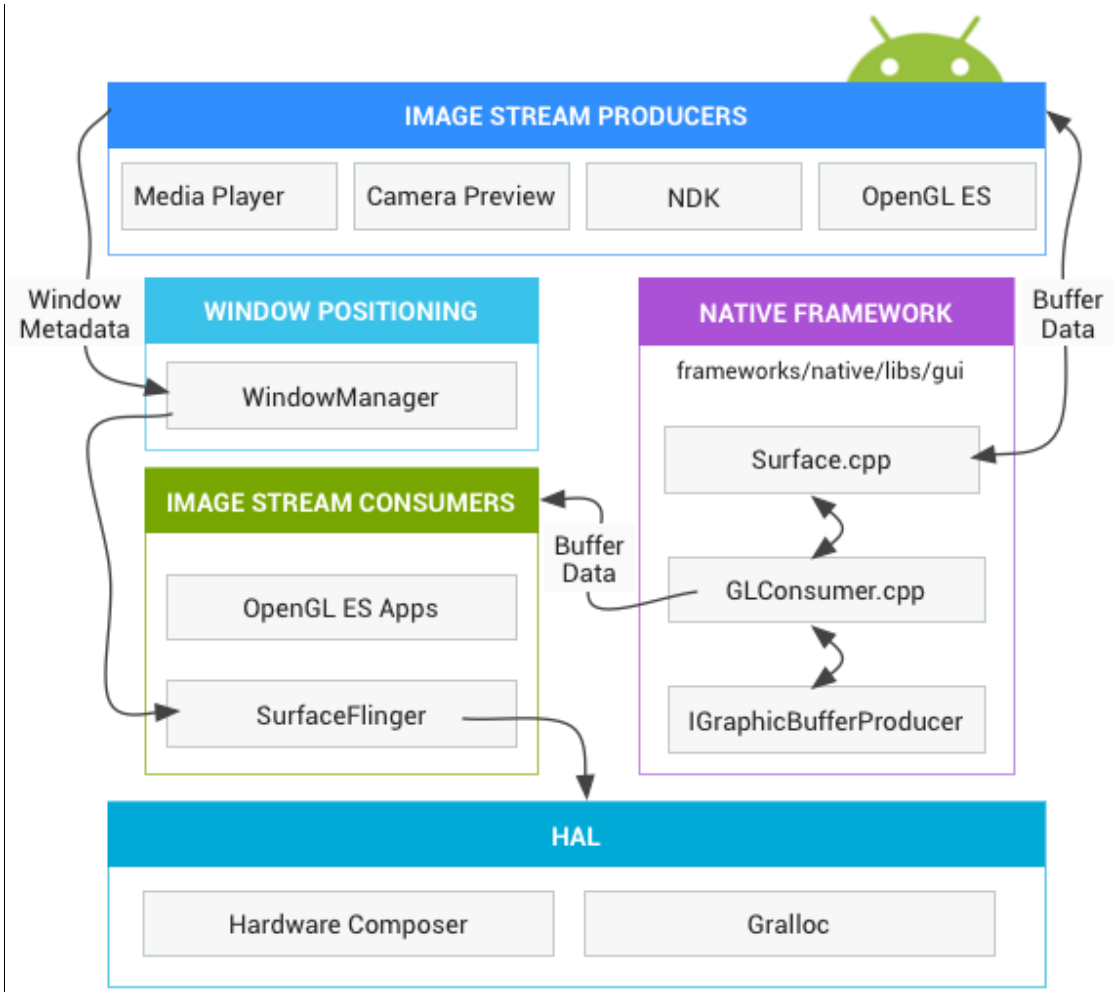
Το πλαίσιο Android προσφέρει διάφορα APIs για την δισδιάστατη και τρισδιάστατη σχεδίαση των γραφικών που αλληλεπιδρούν με τους οδηγούς των γραφικών. Είναι σημαντικό να κατανοήσει κανείς το πως δουλεύουν αυτά τα APIs στα ψηλότερα επίπεδα.

Οι προγραμματιστές σχεδιάζουν σχήματα στην οθόνη με δυο τρόπους, με την χρήση του Canvas ή με την χρήση του OpenGL. Το android.graphics.Canvas είναι ένα API που χρησιμοποιείται στην δισδιάστατη σχεδίαση. Οι λειτουργίες του Canvas σχεδιάζονται πάνω σε όλα τα τυποποιημένα και τα προσαρμοσμένα View (Custom View) του Android(android.view.Views). Στο Android, η επιτάχυνση του υλικού για τους Canvas APIs επιτυγχάνεται με την βοήθεια της βιβλιοθήκης OpenGLRenderer η οποία μεταφράζει τις λειτουργίες του Canvas σε λειτουργίες OpenGL, έτσι ώστε να μπορούν να εκτελεστούν στην κάρτα γραφικών[34].

Από το Android 4.0 και μετά, το Canvas με επιτάχυνση υλικού είναι ενεργοποιημένο ως προεπιλογή. Κατά συνέπεια, μια κάρτα γραφικών που υποστηρίζει το OpenGL ES 2.0 χρησιμοποιείται μόνο στις συσκευές Android 4.0 και μετά. Εκτός από το Canvas, ο άλλος κύριος τρόπος που χρησιμοποιείται για την σχεδίαση γραφικών είναι η άμεση χρήση του OpenGL ES για την σχεδίαση στην οθόνη.

Δεν έχει σημασία ο τρόπος που χρησιμοποιείται από τους προγραμματιστές, ο σχεδιασμός εμφανίζεται στην ίδια επιφάνεια(Surface). Η επιφάνεια αντιπροσωπεύει την παραγόμενη πλευρά μιας ουράς ενδιάμεσης μνήμης που συχνά καταναλώνεται από μια SurfaceFlinger. Κάθε παράθυρο που έχει δημιουργηθεί για την πλατφόρμα Android υποστηρίζεται από μία επιφάνεια. Όλες οι ορατές επιφάνειες συνδέονται με την οθόνη μέσω του SurfaceFlinger.

Η εικόνα 7 δείχνει πως τα βασικά συστατικά των γραφικών του Android συνεργάζονται μεταξύ τους:



Εικόνα 7: Η αρχιτεκτονική των γραφικών του Android

Μερικά από τα βασικά συστατικά αναλύονται παρακάτω:

Image Stream Producers

Μια image stream producer μπορεί να είναι οτιδήποτε που παράγει γραφική ενδιάμεση μνήμη για κατανάλωση. Για παράδειγμα το OpenGL ES ή Canvas 2D.

Image Stream Consumers

Ο πιο κοινός Image Stream Consumer είναι το SurfaceFlinger. Μία υπηρεσία του συστήματος η οποία καταναλώνει την ορατή επιφάνεια και τα συστατικά πάνω σε αυτήν, χρησιμοποιώντας τις πληροφορίες που παρέχονται από το Window Manager. Το SurfaceFlinger είναι η μοναδική υπηρεσία η οποία μπορεί να τροποποιήσει το περιεχόμενο της επιφάνειας. Το SurfaceFlinger χρησιμοποιεί το OpenGL και το Hardware Composer για να συνθέσει μια ομάδα επιφανειών.

Άλλες OpenGL ES εφαρμογές μπορούν επίσης να καταναλώσουν τα image streams, όπως για παράδειγμα η εφαρμογή της κάμερας καταναλώνει μια camera

preview image stream. Οι εφαρμογές που δεν χρησιμοποιούν GL μπορούν επίσης να γίνουν καταναλωτές, για παράδειγμα η κλάση ImageReader.

Window Manager

Η Window Manager είναι η υπηρεσία του συστήματος Android που ελέγχει ένα παράθυρο, το οποίο είναι ένα δοχείο για τα views. Ένα παράθυρο υποστηρίζεται πάντα από μια επιφάνεια. Η υπηρεσία αυτή επιβλέπει τον κύκλο ζωής, τον προσανατολισμό της οθόνης, τις μεταβάσεις, τα κινούμενα σχέδια και πολλές άλλες όψεις ενός παραθύρου. Το Window Manager στέλνει όλα τα μεταδεδομένα του παραθύρου στο SurfaceFlinger, το οποίο μπορεί να συνθέσει τις επιφάνειες στην οθόνη.

Hardware Composer

Το Hardware Composer είναι η αφαίρεση υλικού (hardware abstraction) για το υποσύστημα οθόνης. Το SurfaceFlinger μπορεί να αναθέτει ορισμένες σύνθετες εργασίες στο Hardware Composer για να ξεφορτώσει κάποιες εργασίες του OpenGL και της κάρτας γραφικών. Το SurfaceFlinger ενεργεί σαν να είναι ένας OpenGL ES πελάτης. Έτσι όταν το SurfaceFlinger συνθέτει ενεργά μία ή δυο ενδιάμεσες μνήμες μέσα στην τρίτη, για στιγμιότυπο, χρησιμοποιείται το OpenGL ES. Με την διαδικασία αυτή, η σύνθεση επιτυγχάνεται με χαμηλότερη κατανάλωση ενέργειας από ότι την καθοδήγηση της GPU (graphics processing unit – κάρτας γραφικών) όλων των υπολογισμών.

Το Hardware Composer HAL πραγματοποιεί το άλλο μισό της δουλειάς. Αυτό το HAL είναι το κεντρικό σημείο για όλες τις γραφικές αποδόσεις(rendering) του Android. Μερικά από τα γεγονότα που πρέπει να υποστηρίζει το Hardware Composer είναι το VSYNC και το hotplug για plug-and-play HDMI υποστήριξη.

Gralloc

Ο κατανεμητής μνήμης γραφικών (graphics memory allocator) χρησιμοποιείται στην εκχώρηση της μνήμης που απαιτείται από τους παραγωγούς εικόνας (image producers).

3.2 Τα βασικά της σχεδίασης των διεπαφών χρήστη

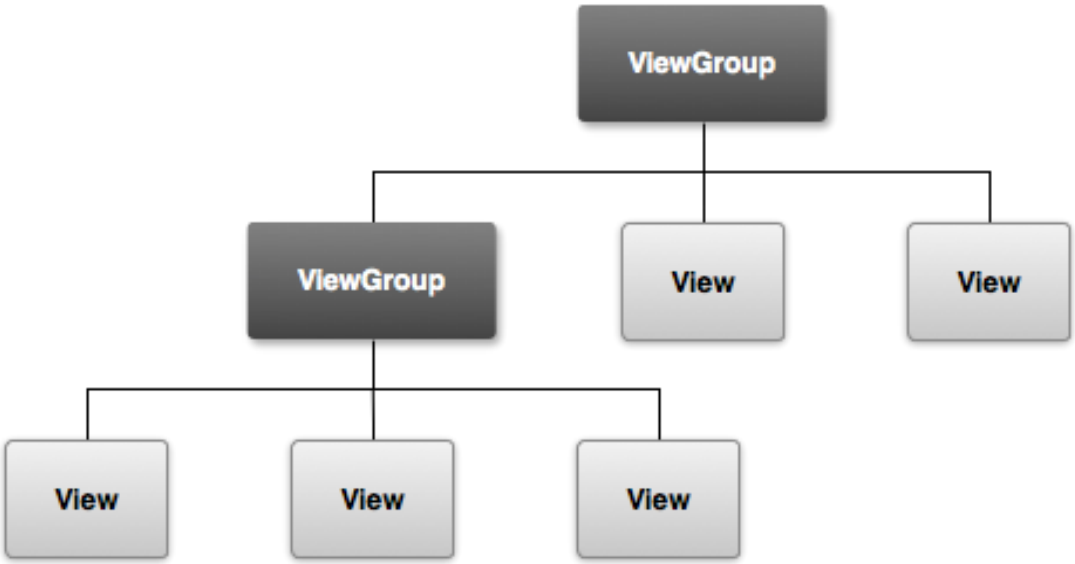
3.2.1 Επισκόπηση Διεπαφής Χρήστη

Όλα τα στοιχεία των διεπαφών χρήστη σε μία εφαρμογή Android χτίζονται με την χρήση των αντικειμένων View και ViewGroup. Ένα View είναι ένα αντικείμενο που σχεδιάζει ένα σχήμα στην οθόνη με το οποίο μπορεί ο χρήστης να αλληλεπιδρά. Ένα ViewGroup είναι ένα αντικείμενο που μπορεί να περιέχει άλλα View ή ViewGroup αντικείμενα προκειμένου να καθορίζουν το layout της διεπαφής[35].

Το Android παρέχει υποκλάσεις των View και ViewGroup αντικειμένων που μας προσφέρουν μερικά κοινά στοιχεία εισόδου (όπως buttons και text fields) και διάφορα μοντέλα διατάξεων (όπως τα linear layout και relative layout).

User Interface Layout (Διάταξη Διεπαφής χρήστη)

Η διεπαφή χρήστη του κάθε συστατικού της εφαρμογής μας, καθορίζεται με την χρήση μίας ιεραρχίας των View και ViewGroup αντικειμένων, όπως μπορούμε να δούμε και στην εικόνα 8. Το κάθε view group είναι ένα αόρατο δοχείο που οργανώνει child views, εφόσον τα child views μπορούν να είναι input controls ή άλλα widgets που σχηματίζουν τα άλλα κομμάτια της διεπαφής χρήστη. Το δέντρο ιεραρχίας μπορεί να είναι πολύ απλό ή και πολύπλοκο σύμφωνα με τις ανάγκες που έχουμε.



Εικόνα 8: Απεικόνιση ενός δέντρου ιεραρχίας των View και ViewGroup που αποτελείται ένα layout.

Για να δηλώσουμε μία διάταξη αρχικοποιούμε τα αντικείμενα View στον κώδικά μας και αρχίζουμε να χτίζουμε το δέντρο, αλλά ο ποιο εύκολος και ποιο αποτελεσματικός τρόπος για να δηλώσουμε μία διάταξη είναι να χρησιμοποιήσουμε τα xml αρχεία. Τα xml αρχεία μας προσφέρουν μία δομή διάταξης που μπορεί να αναγνωριστεί ποιο εύκολα και είναι αυτά που έχουμε χρησιμοποιήσει στην ανάπτυξη της εφαρμογής μας. Θα αναφέρουμε τις διατάξεις στα επόμενα υποκεφάλαια.

Η κλάση View αποτελεί το βασικό στοιχείο μίας διεπαφής χρήστη Android και αναπαριστά ένα ορθογώνιο τμήμα της οθόνης. Η κλάση View αποτελεί σχεδόν όλα τα στοιχεία ελέγχου της διεπαφής χρήστη και τις διατάξεις μέσα στο Android. Στα επόμενα υποκεφάλαια θα αναφέρουμε τα στοιχεία ελέγχου και τις διατάξεις διεπαφών χρήστη του Android.

3.2.2 Layouts (Διατάξεις)

Μία διάταξη καθορίζει την εικονική δομή για μία διεπαφή χρήστη, όπως η διεπαφή χρήστη για μία δραστηριότητα ή app widget. Μπορούμε να δηλώσουμε μία διάταξη με δυο τρόπους[36]:

> **Δήλωση στοιχείων διεπαφής χρήστη σε XML**: Το Android παρέχει ένα απλό λεξιλόγιο XML που αντιστοιχεί με τις κλάσεις και υποκλάσεις του View.
>
> **Αρχικοποίηση των στοιχείων διάταξης σε χρόνο εκτέλεσης**: Η εφαρμογή μας μπορεί να δημιουργήσει View και ViewGroup αντικείμενα προγραμματιστικά.

Το πλαίσιο του Android μας δίνει την ευελιξία να μπορούμε να χρησιμοποιήσουμε και τις δυο τεχνικές ταυτόχρονα στη δήλωση και διαχείριση των συστατικών διεπαφής χρήστη της εφαρμογής μας. Για παράδειγμα, θα μπορούσαμε να δηλώσουμε τις διατάξεις της διεπαφής χρήστη σε XML, συμπεριλαμβανομένων και των στοιχείων που εμφανίζονται μέσα σε αυτά μαζί με τις ιδιότητες τους. Έπειτα, μπορούμε να εισάγουμε κώδικα στην εφαρμογή μας, που θα τροποποιήσει την κατάσταση των αντικειμένων της οθόνης σε χρόνο εκτέλεσης.

Το πλεονέκτημα της δήλωσης της διεπαφής χρήστη σε XML είναι ότι μας δίνει την δυνατότητα του διαχωρισμού της εμφάνισης της εφαρμογής από τον κώδικα που χειρίζεται την συμπεριφορά της εφαρμογής. Οι περιγραφές είναι εξωτερικές στον κώδικα της εφαρμογής, το οποίο σημαίνει ότι μπορούμε να τροποποιήσουμε ή να προσαρμόσουμε χωρίς να τροποποιήσουμε τον πηγαίο κώδικα ή να τον μεταγλωττίσουμε ξανά. Για παράδειγμα μπορούμε να δημιουργήσουμε XML διατάξεις για διάφορες διαστάσεις οθονών, διάφορους προσανατολισμούς ή διάφορες γλώσσες. Επιπλέον η δήλωση της διάταξης σε XML καθιστά την ευκολότερη απεικόνιση της δομής της διεπαφής χρήστη μας, έτσι και ο εντοπισμός σφαλμάτων είναι πιο εύκολος.

Σε γενικές γραμμές, το XML λεξιλόγιο όπου χρησιμοποιείται για την δήλωση των στοιχείων διεπαφής χρήστη ακολουθεί την δομή και ονομασία των κλάσεων και μεθόδων, όπου το όνομα της κατηγορίας αντιστοιχεί στο όνομα της κλάσης και το όνομα της ιδιότητας αντιστοιχεί στο όνομα της μεθόδου. Η αντιστοιχία είναι τόσο άμεση που μπορεί κανείς να μαντέψει το όνομα της ιδιότητας από την μέθοδο ή το όνομα της κλάσης από το όνομα της κατηγορίας. Ενώ σε ορισμένες περιπτώσεις υπάρχουν μερικές μικρές διαφορές στις ονομασίες. Για παράδειγμα, το στοιχείο EditText έχει μία ιδιότητα κειμένου που αντιστοιχεί στην μέθοδο EditText.setText().

Το όνομα ενός στοιχείου xml για ένα view αντιστοιχεί με την κλάση Android που αντιπροσωπεύει. Έτσι ένα στοιχείο <TextView> δημιουργεί ένα TextView widget στην διεπαφή χρήστη και το στοιχείο <LinearLayout> δημιουργεί ένα LinearLayout view group.

Παρακάτω βλέπουμε ένα παράδειγμα μίας απλής κάθετης διάταξης με ένα text view και ένα button σε xml :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical" >
    <TextView android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a TextView" />
    <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="I am a Button" />
</LinearLayout>
```

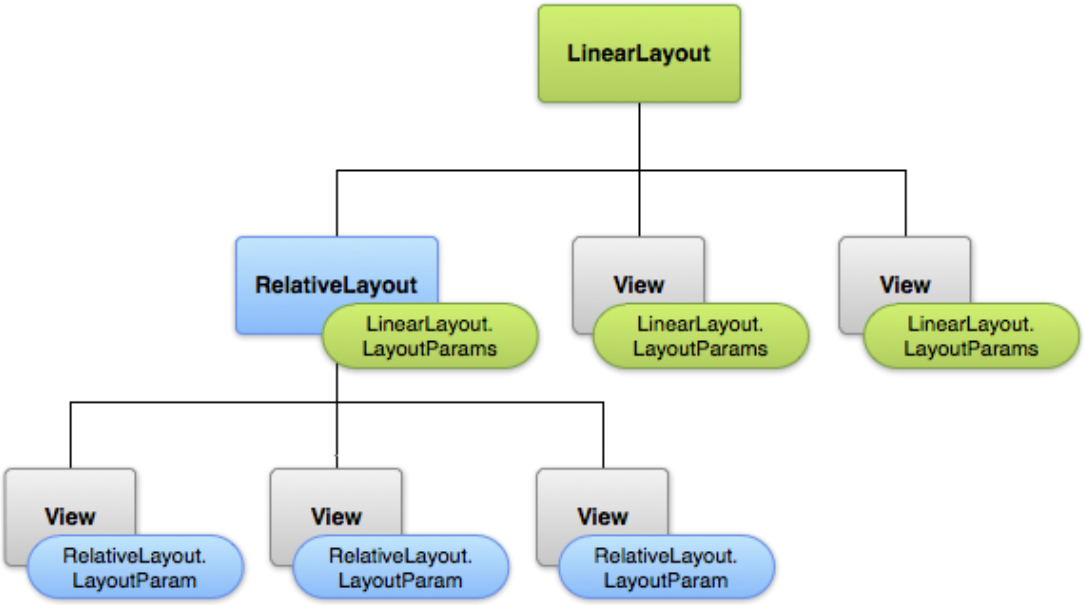
Μετά την δήλωση της διάταξης μας σε XML, αποθηκεύουμε το αρχείο με κατάληξη .xml, στο κατάλογο res/layout/ του έργου μας για να μπορεί να μεταγλωττιστεί.

Κατά την μεταγλώττιση της εφαρμογής, το κάθε αρχείο διάταξης XML μεταγλωττίζεται σε έναν πόρο View. Πρέπει να φορτώσουμε τον πόρο διάταξης από τον κώδικα της εφαρμογής μας, στο Activity.onCreate που υλοποιείται σε κάθε Activity. Για να γίνει αυτό, πρέπει να καλέσουμε τον setContentView και να περάσουμε τον πόρο διάταξης μας για αναφορά σε μορφή: R.layout.layout_file_name. Για παράδειγμα, εάν το όνομα της XML διάταξης μας είναι main_layout.xml, θα το φορτώναμε στο Activity με τον παρακάτω κώδικα:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

Τα ονόματα των ιδιοτήτων των διατάξεων που ξεκινάνε με την λέξη layout καθορίζουν τις παραμέτρους των διατάξεων που είναι κατάλληλες για το ViewGroup που ανήκουν.

Η κάθε κλάση ViewGroup υλοποιεί μία εμφωλευμένη κλάση η οποία επεκτείνει το ViewGroup.LayoutParams. Η υποκλάση αυτή περιέχει τύπους ιδιοτήτων που καθορίζουν την διάσταση και την θέση του κάθε child view, Στην εικόνα 9, το parent view καθορίζει τις παραμέτρους της διάταξης για κάθε child view (συμπεριλαμβανομένου και του child view group).

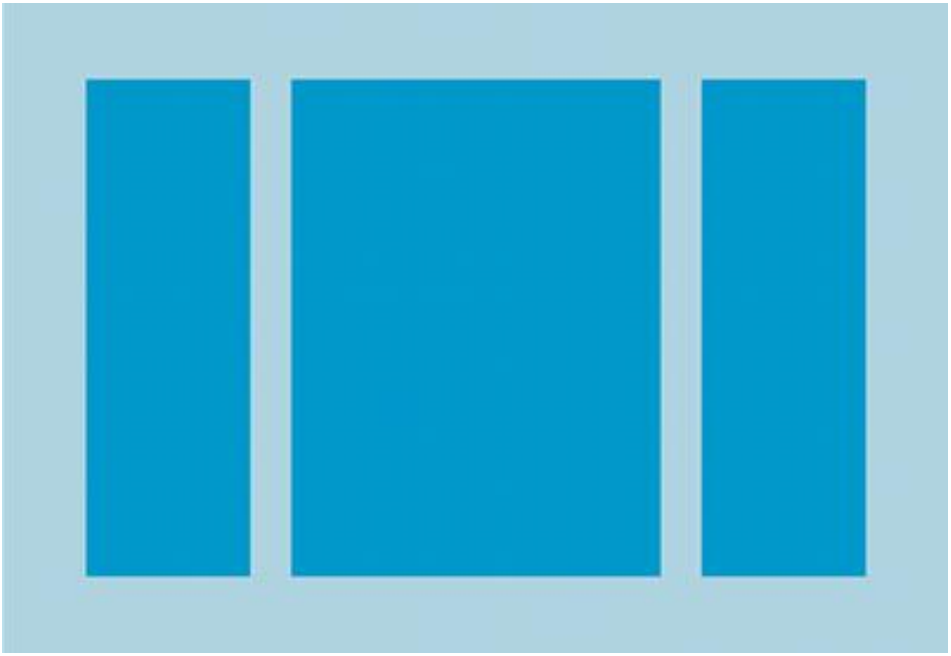


Εικόνα 9: Ιεραρχία παραμέτρων Layout

Η κάθε υποκλάση LayoutParams έχει την δικιά της σύνταξη που μπορούμε να ρυθμίσουμε τις τιμές. Το κάθε στοιχείο παιδί πρέπει να καθορίζει LayoutParams που θα ισχύουν για τους γονείς του, έτσι μπορεί να ορίσει διαφορετικά LayoutParams για τα παιδιά του.

Η κάθε υποκλάση έχει τον δικό της τρόπο για να παρουσιάσει τα View που περιέχει. Παρακάτω βλέπουμε μερικούς από τους ποιο κοινούς τρόπους των παρουσιάσεων των διατάξεων:

Linear Layout: Η διάταξη αυτή οργανώνει τα στοιχεία που περιέχει σε μία οριζόντια ή κάθετη σειρά. Δημιουργεί scrollbar όταν η διάσταση του παραθύρου ξεπερνάει την διάσταση της οθόνης. Η διάταξη αυτή απεικονίζεται στην εικόνα 10.



Εικόνα 10: LinearLayout

Relative Layout: Με την διάταξη αυτή μπορούμε να καθορίσουμε τις θέσεις των αντικειμένων παιδιών που περιέχει, σε σχέση με το που βρίσκεται το κάθε αντικείμενο (Το παιδί Α βρίσκεται στα αριστερά του παιδιού Β). Στην εικόνα 11 απεικονίζεται μία οθόνη με διάταξη Relative.

Εικόνα 11: RelativeLayout

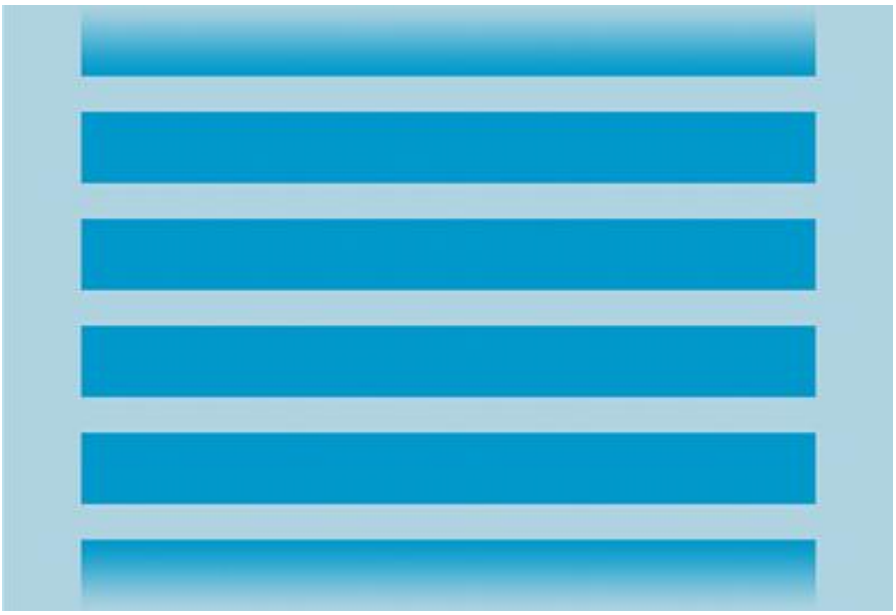
WebView: Με την διάταξη WebView μπορούμε να εμφανίσουμε ιστοσελίδες στην οθόνη της συσκευής μας.

Όταν το περιεχόμενο της διάταξης μας είναι δυναμική ή μη προκαθορισμένη μπορούμε να χρησιμοποιήσουμε μία διάταξη που είναι υποκλάση μίας κλάσης AdapterView για να συμπληρώσουμε την διάταξη με View σε χρόνο εκτέλεσης. Η

υποκλάση μίας κλάσης AdapterView χρησιμοποιεί ένα Adapter για να δεσμεύσει τα δεδομένα στην διάταξη τους. Ένα Adapter είναι ο μεσάζων μεταξύ των πηγαίων δεδομένων και του AdapterView.

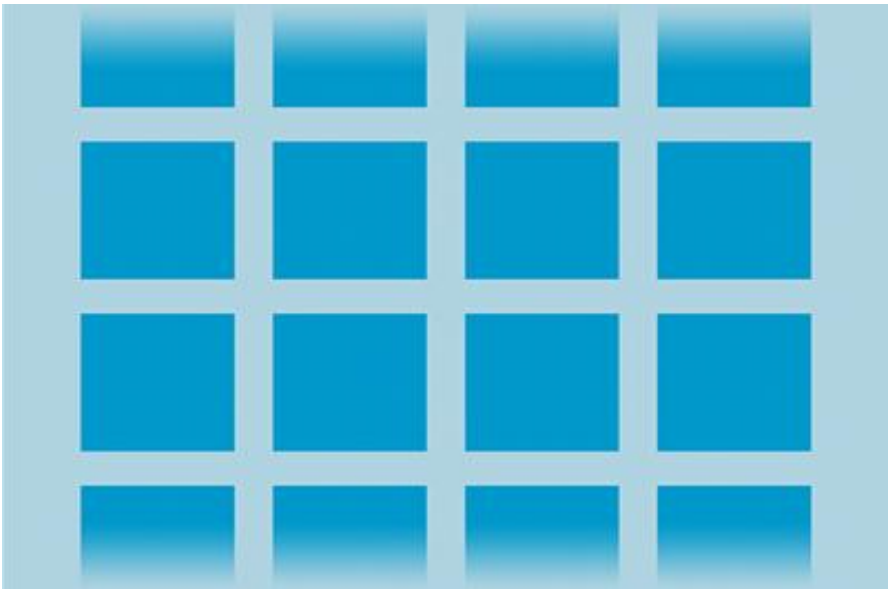
Μερικές από τις κοινές διατάξεις που υποστηρίζονται από ένα Adapter είναι:

ListView: Το ListView είναι ένα ViewGroup που παρουσιάζει μία λίστα των κυλιόμενων στοιχειών (scrollable items). Τα στοιχεία εισάγονται αυτόματα στην λίστα με την χρήση ενός Adapter που αποκτά τα δεδομένα από μία πηγή όπως έναν πίνακα ή ένα ερώτημα βάσεων δεδομένων και μετατρέπει το κάθε στοιχείο σε ένα View που θα προστεθεί στη λίστα (Εικόνα 12).



Εικόνα 12: ListView

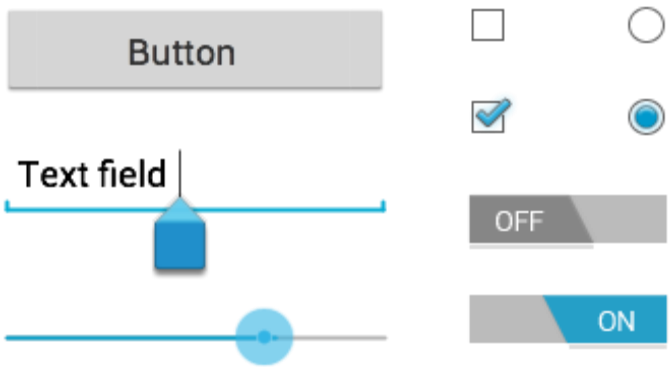
GridView: Το GridView είναι ένα ViewGroup που παρουσιάζει τα στοιχεία σε κυλιόμενο πλέγμα (scrollable grid) των δυο διαστάσεων (Εικόνα 13).



Εικόνα 13: GridView

3.2.3 Input Controls (Στοιχεία Ελέγχου)

Τα στοιχεία ελέγχου είναι αλληλεπιδραστικά συστατικά μέσα στην διεπαφή χρήστη της εφαρμογής μας. Το Android παρέχει διάφορους ελέγχους που μπορούμε να χρησιμοποιήσουμε στη διεπαφή χρήστη της εφαρμογής μας, όπως τα κουμπιά, τα πεδία κειμένου, πλαίσια ελέγχου και πολλά άλλα που βλέπουμε στην εικόνα 14.



Εικόνα 14: Διάφορα Input Controls

Η πρόσθεση ενός στοιχείου ελέγχου στην διεπαφή χρήστη είναι τόσο απλή όσο η προσθήκη ενός στοιχείου στην XML διάταξη μας. Για παράδειγμα στον παρακάτω κώδικα βλέπουμε μία διάταξη που περιέχει ένα πεδίο κειμένου και ένα κουμπί[37]:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />
</LinearLayout>
```

Το κάθε στοιχείο ελέγχου υποστηρίζει ένα σετ των συμβάντων εισόδου και μπορούν να χειρίζονται τα συμβάντα όπως όταν ο χρήστης εισάγει ένα κείμενο ή κάνει κλικ σε ένα κουμπί.

Παρακάτω βλέπουμε μία λίστα των μερικών στοιχειών ελέγχου που μπορούμε να χρησιμοποιήσουμε στην εφαρμογή μας:

Button (κουμπί): Είναι ένα push-button που μπορεί να πατηθεί από τον χρήστη και να εκτελέσει μία ενέργεια.

Text Fields (πεδίο κειμένου): Είναι ένα επεξεργάσιμο πεδίο κειμένου. Μπορούμε να χρησιμοποιήσουμε το AutoCompleteTextView Widget για να δημιουργήσουμε ένα widget εισαγωγής κειμένου που μπορεί να μας κάνει προτάσεις αυτόματης συμπλήρωσης κειμένου. Οι κλάσεις EditText και AutoCompleteTextView είναι μερικά παραδείγματα των στοιχειών αυτών.

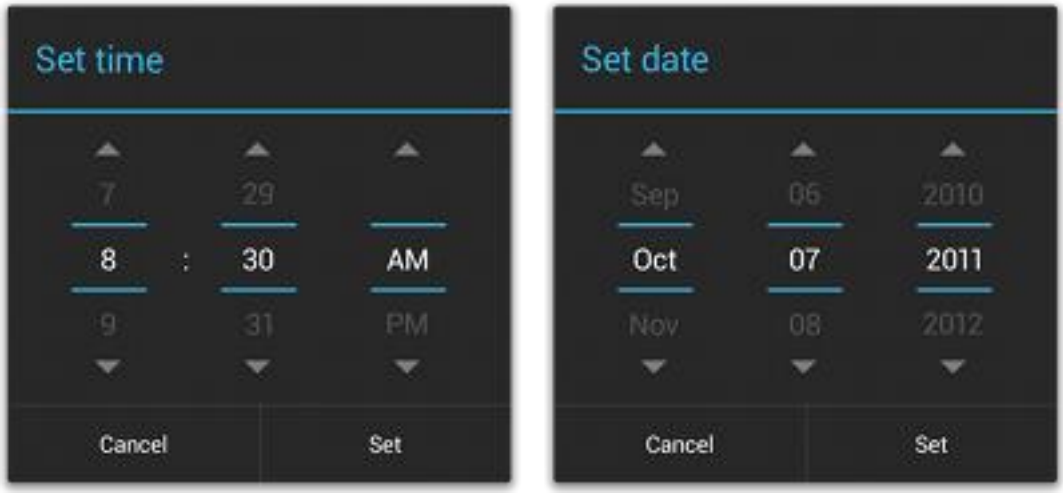
CheckBox: Είναι ένα στοιχείο διακόπτης που μπορεί να ενεργοποιηθεί από τον χρήστη. Πρέπει να χρησιμοποιείται κατά την παρουσίαση των διαθέσιμων επιλογών που δεν αλληλοαποκλείονται.

RadioButton: Είναι παρόμοια με τα CheckBoxes μόνο που στα στοιχεία αυτά μπορεί ο χρήστης να επιλέξει μόνο ένα στοιχείο από την ομάδα που ανήκουν. Μερικές από τις κλάσεις που ανήκουν σε αυτήν την κατηγορία είναι η RadioGroup και η RadioButton.

Toggle Button: Είναι ένα κουμπί on/off με ενδεικτική λυχνία.

Spinner: Μία αναπτυσσόμενη λίστα που επιτρέπει στους χρήστες να επιλέξουν μία τιμή από ένα σύνολο.

Pickers: Είναι ένα παράθυρο που μπορούν οι χρήστες να επιλέξουν μία τιμή με την χρήση των up/down κουμπιών ή διάφορων χειρονομιών. Για παράδειγμα με το DataPicker μπορεί ο χρήστης να επιλέξει μία ημερομηνία ή με το TimePicker να επιλέξει ένα χρονικό διάστημα. Στην εικόνα 15 βλέπουμε δυο παραδείγματα απεικονίσεων των κλάσεων αυτών σε μία εφαρμογή.



Εικόνα 15: TimePicker και DatePicker

3.3 Τα Βασικά Συστατικά Σχεδίασης Διεπαφής Χρήστη της EGQL

Παρακάτω θα αναλύσουμε τα βασικά συστατικά Android που χρησιμοποιήθηκαν κατά την ανάπτυξη της EGQL και δεν έχουν αναφερθεί στα προηγούμενα κεφάλαια.

3.3.1 Canvas

Το Android framework API παρέχει ένα σύνολο δισδιάστατων σχεδιάσεων που μας επιτρέπουν να καταστήσουμε(render) τα δικά μας προσαρμοσμένα γραφικά σε έναν καμβά ή να τροποποιήσουμε τα υπάρχοντα View για να προσαρμοστούν στην εμφάνιση τους. Η κλάση Canvas περιέχει τις “draw” κλήσεις. Για να σχεδιάσουμε κάτι θα πρέπει να έχουμε τα 4 βασικά συστατικά: ένα Bitmap που θα περιέχει τα pixels, ένα Canvas που θα φιλοξενήσει τις κλήσεις “draw”, ένα βασικό γραφικό σχέδιο όπως το Rect, το Path, το Text ή το Bitmap και ένα Paint αντικείμενο για την περιγραφή των χρωμάτων και στυλ που θα σχεδιαστούν [38]. Η σύνταξη δισδιάστατων γραφικών μπορούν να πραγματοποιηθούν με δυο τρόπους:

1. Σχεδίαση των γραφικών ή των κινούμενων σχεδίων μέσα σε ένα View αντικείμενο από την διάταξη μας. Με αυτόν τον τρόπο, το σχέδιο των γραφικών μας, χειρίζεται από την κανονική διαδικασία σχεδίασης ιεραρχίας View, αρκεί να καθορίσουμε το View να περιέχει το γραφικό μας.

2. Σχεδίαση των γραφικών κατευθείαν σε έναν καμβά (Canvas). Με τον τρόπο αυτό καλούμε την κατάλληλη μέθοδο onDraw() της κλάσης ή μία από τις μεθόδους σχεδίασης του Canvas, όπως η drawPicture(). Κάνοντας αυτό, έχουμε τον έλεγχο του κάθε κινούμενου σχεδίου.

Η πρώτη επιλογή, σχεδίαση σε ένα View, είναι η καλύτερη επιλογή όταν θέλουμε να σχεδιάσουμε απλά γραφικά που δεν χρειάζεται να αλλάζουν δυναμικά και δεν είναι κομμάτια απόδοσης ενός παιχνιδιού. Για παράδειγμα, πρέπει να

σχεδιάσουμε τα γραφικά μας μέσα σε ένα View όταν θέλουμε να εμφανίσουμε ένα στατικό γραφικό ή ένα προκαθορισμένο κινούμενο σχέδιο.

Η δεύτερη επιλογή, σχεδίαση σε ένα Canvas, είναι η καλύτερη όταν η εφαρμογή μας θα πρέπει να ξανασχεδιαστεί τακτικά. Οι εφαρμογές, όπως τα βίντεο-παιχνίδια, πρέπει να σχεδιάζονται σε καμβά από μόνες τους. Ωστόσο, υπάρχουν παραπάνω από ένα τρόπους για να το κάνουμε αυτό.

1. Στο ίδιο νήμα με την διεπαφή χρήστη του Activity, όπου μπορούμε να σχεδιάσουμε ένα συστατικό custom View στη διάταξη μας, καλούμε τη μέθοδο invalidate() και χειριζόμαστε την onDraw() callback.

2. Ή σε ξεχωριστό νήμα, όπου μπορούμε να διαχειριστούμε ένα SurfaceView και να εκτελέσουμε σχεδιάσεις στον καμβά όσο πιο γρήγορα μπορεί το νήμα μας, στην περίπτωση αυτή δεν χρειάζεται να καλέσουμε το invalidate().

3.3.1.1 Σχεδίαση με Canvas

Όταν γράφουμε μία εφαρμογή στην οποία θα εκτελέσουμε εξειδικευμένα σχέδια και θα χειριστούμε τις κινήσεις των γραφικών, πρέπει οι σχεδιασμοί των γραφικών να γίνουν μέσω του αντικειμένου Canvas. Ένα Canvas χρησιμοποιείται ως διασύνδεση στην πραγματική επιφάνεια πάνω στην οποία θα σχεδιαστούν τα γραφικά μας και περιέχει όλες τις κλήσεις “draw”. Στην πραγματικότητα, μέσω του Canvas ο σχεδιασμός μας θα εκτελείται σε ένα Bitmap, το οποίο τοποθετείται στο παράθυρο.

Στην περίπτωση που σχεδιάζουμε με την μέθοδο onDraw(), το Canvas παρέχεται για εμάς και πρέπει απλά να πραγματοποιήσουμε τις κλήσεις μας πάνω σε αυτό. Μπορούμε επίσης να αποκτήσουμε ένα Canvas από το SurfaceHolder.lockCanvas(), όταν χρησιμοποιούμε το SurfaceView αντικείμενο. Στην δημιουργία ενός νέου Canvas χρησιμοποιούμε πάντα ένα Bitmap. Παρακάτω βλέπουμε ένα παράδειγμα ορισμού ενός Canvas.

```
Bitmap b = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB_8888);
Canvas c = new Canvas(b);
```

Τώρα το Canvas θα σχεδιαστεί στο καθορισμένο Bitmap. Μετά τον σχεδιασμό του με το Canvas, μπορούμε να μεταφέρουμε τα Bitmap σε άλλο Canvas με μία από τις μεθόδους Canvas.drawBitmap(Bitmap,…). Συνιστάται να σχεδιάσουμε το τελικό σχέδιο με μία από τις μεθόδους που παρέχονται από το Canvas, το View.onDraw() ή το SurfaceHolder.lockCanvas().

Το Canvas έχει τον δικό του σύνολο μεθόδων σχεδιασμού που μπορούμε να χρησιμοποιήσουμε, όπως η drawBitmap(…), η drawRect(…), η drawText(…) και πολλά άλλα. Οι υπόλοιπες κλάσεις που θα μπορούσαμε να χρησιμοποιήσουμε

έχουν επίσης draw() μεθόδους. Για παράδειγμα, πιθανότατα να έχουμε Drawable αντικείμενα που θα προσθέσουμε στο Canvas. Ένα Drawable έχει την δικιά του draw() μέθοδο που παίρνει το Canvas ως μία παράμετρο.

3.3.1.2 Σε ένα View

Εάν η εφαρμογή μας δεν απαιτεί μία σημαντική ταχύτητα επεξεργασίας ή της ταχύτητας ρυθμού πλαισίου τότε πρέπει να εξεταστεί το ενδεχόμενο της δημιουργίας των προσαρμοσμένων συστατικών View και η σχεδίαση τους μέσω του Canvas στην View.onDraw(). Στην περίπτωση αυτή, το πλαίσιο Android θα μας παρέχει ένα προκαθορισμένο Canvas το οποίο μπορούμε να τοποθετήσουμε τις κλήσεις των σχεδίων μας.

Για να ξεκινήσουμε πρέπει να πραγματοποιηθεί η επέκταση της κλάσης View και ο καθορισμός της μεθόδου onDraw(). Η μέθοδος αυτή θα καλείται από το πλαίσιο Android, για να ζητήσει να σχεδιαστεί το View. Αυτό είναι το μέρος που θα εκτελεστούν όλες οι κλήσεις μας για να σχεδιάσουμε μέσω του Canvas.

Το πλαίσιο Android θα καλέσει την onDraw() μόνο όταν απαιτηθεί. Κάθε φορά που η εφαρμογή μας είναι έτοιμη να σχεδιάζει, πρέπει να ζητήσουμε το View μας να ακυρώνεται με την κλήση της invalidate(). Αυτό σημαίνει ότι θέλουμε το View μας να σχεδιάζεται και έτσι καλείται η μέθοδος onDraw() κάθε φορά.

Μέσα στην onDraw() του συστατικού View χρησιμοποιούμε το Canvas που μας δίνεται για να σχεδιάσουμε διάφορα με την χρήση των διάφορων Canvas.draw…() μεθόδων ή των draw() των υπόλοιπων κλάσεων που δέχονται το Canvas ως μία παράμετρο. Μόλις η μέθοδος onDraw() ολοκληρωθεί, το πλαίσιο Android θα χρησιμοποιήσει το Canvas για να σχεδιάσει ένα Bitmap που έχει χειριστεί από το σύστημα.

3.3.1.3 Σε ένα SurfaceView

Το SurfaceView είναι μία ειδική υποκλάση του View που μας προσφέρει μία εξειδικευμένη επιφάνεια στην ιεραρχία του View. Ο σκοπός του είναι να μας προσφέρει την επιφάνεια αυτή σε ένα δευτερεύον νήμα της εφαρμογής, έτσι ώστε η εφαρμογή να μην χρειάζεται να περιμένει μέχρι να είναι έτοιμη για σχεδίαση η ιεραρχία View του συστήματος. Ένα δευτερεύον νήμα που αναφέρεται σε ένα SurfaceView σχεδιάζει τον δικό του Canvas με τον δικό του ρυθμό.

Για να ξεκινήσουμε πρέπει να δημιουργήσουμε μία νέα κλάση που επεκτείνει το SurfaceView. Η κλάση πρέπει επίσης να υλοποιήσει το SurfaceHolder.Callback. Η υποκλάση αυτή είναι μία διεπαφή που θα μας ενημερώσει με τις πληροφορίες για τον υποκείμενο Surface, όπως όταν δημιουργείται, αλλάζει ή καταστρέφεται. Τα γεγονότα αυτά είναι σημαντικά έτσι ώστε να ξέρουμε πότε αρχίζει η σχεδίαση, αν πρέπει να προσαρμόσουμε τις ιδιότητες με βάση την καινούργια επιφάνεια, πότε να σταματήσει η σχεδίαση και πιθανότατα ο τερματισμός μερικών διεργασιών. Μπορούμε επίσης να καθορίσουμε το δευτερεύον νήμα μέσα στην κλάση SurfaceView που θα εκτελέσει όλες τις διαδικασίες σχεδίασης στο Canvas.

Αντί να χειριζόμαστε απευθείας τα Surface αντικείμενα, πρέπει να χειριζόμαστε με την χρήση του SurfaceHolder. Όταν αρχικοποιείται το SurfaceView πρέπει να πάρουμε το SurfaceHolder καλώντας την μέθοδο getHolder(). Έπειτα πρέπει να ενημερώσουμε το SurfaceHolder ότι θέλουμε να λάβουμε SurfaceHolder callbacks. Αυτό το κάνουμε με την χρήση της μεθόδου addCallback(). Στη συνέχεια παρακάμπτονται (override) όλες οι μέθοδοι της SurfaceHolder.Callback μέσα στην κλάση SurfaceView.

Για να σχεδιάσουμε στο Surface Canvas από το δευτερεύον νήμα, πρέπει να περάσουμε το νήμα στο SurfaceHolder και να ανακτήσουμε το Canvas με την lockCanvas(). Μπορούμε τώρα να πάρουμε το Canvas που μας δίνεται από το SurfaceHolder και να κάνουμε τα απαραίτητα σχέδια πάνω σε αυτό. Μόλις τελειώνουμε με την σχεδίαση του Canvas καλούμε το unlockCanvasAndPost(). Το Surface τώρα θα σχεδιάσει το Canvas όπως τον έχουμε αφήσει. Εκτελούμε αυτήν την διαδικασία κλειδώματος και ξεκλειδώματος κάθε φορά που θέλουμε να σχεδιάσουμε.

3.3.2 Drawables

Το Android μας προσφέρει μία βιβλιοθήκη δισδιάστατων γραφικών για σχεδιασμούς διάφορων σχημάτων και εικόνων. Στο πακέτο android.graphics.drawable μπορούμε να βρούμε τις κοινές κλάσεις που χρησιμοποιούμε κατά την δισδιάστατη σχεδίαση.

Ένα Drawable είναι η μία γενική αφαίρεση που χρησιμοποιείται για κάτι που μπορεί να σχεδιαστεί. Τις περισσότερες φορές βλέπουμε τα Drawables ως πόρους που ανακτήσαμε για να σχεδιάσουμε στην οθόνη της συσκευής. Η κλάση Drawable παρέχει ένα γενικό API για να αντιμετωπίσουμε έναν εικονικό πόρο που μπορεί να είναι σε διάφορες μορφές. Σε αντίθεση με το View, ένα Drawable δεν έχει καμία δυνατότητα να λάβει γεγονότα ή να αλληλεπιδρά με τον χρήστη.

Εκτός από ένα απλό σχέδιο, το Drawable παρέχει μία σειρά γενικών μηχανισμών που το επιτρέπουν να αλληλεπιδρά με τον πελάτη του:

Η μέθοδος setBounds() πρέπει να καλείται για να πει στο Drawable σε ποιο σημείο να σχεδιαστεί και τις διαστάσεις που μπορεί να έχει. Όλα τα Drawables πρέπει να σέβονται το προτιμώμενο μέγεθος, συχνά απλώς και μόνο με την κλιμάκωση των εικόνων τους. Μπορούμε να βρούμε τους προτιμώμενους μεγέθους με την χρήση των μεθόδων getIntrinsicHeight() και getIntirinsicWidth.

Με την μέθοδο getPadding(Rect) μπορούμε να πάρουμε πληροφορίες για το πώς μπορούμε να σχεδιάσουμε το περιεχόμενο των Drawables.

Η μέθοδος setState(int[]) μας επιτρέπει να πούμε σε ποια κατάσταση θέλουμε να σχεδιαστεί το Drawable, όπως “selected”, “focused” κτλ.

Μερικά Drawables μπορούν να τροποποιήσουν την εμφάνιση τους με βάση την επιλεγμένη κατάσταση.

Η μέθοδος setLevel(int) μας επιτρέπει να μπορούμε να τροποποιήσουμε το Drawable με συνεχούς ελέγχους, όπως για παράδειγμα, το επίπεδο της μπαταρίας ή επίπεδο προόδου (progress level). Μερικά Drawables μπορούν να τροποποιήσουν τις εικόνες με βάση το τρέχον επίπεδο τους.

Ένα Drawable μπορεί να εκτελέσει κινήσεις με κλήση της διασύνδεσης Drawable.Callback.

Αν και συνήθως δεν είναι ορατές σε εφαρμογές, τα Drawables μπορούν να πάρουν διάφορες μορφές:

Bitmap: Το απλούστερο Drawable, μία εικόνα JPEG ή PNG.

Nine Patch: Μία επέκταση στην PNG μορφή που επιτρέπει τον καθορισμό των πληροφοριών για το πώς να τροποποιηθεί.

Shape: Περιέχει απλές εντολές σχεδιασμού και όχι ακατέργαστα Bitmap, που μας δίνει την δυνατότητα να αλλάζουμε τα μεγέθη καλύτερα σε ορισμένες περιπτώσεις.

Layers: Ένα σύνθετο Drawable, που σχεδιάζει πολλαπλά υποκείμενα Drawables το ένα πάνω στο άλλο.

States: Ένα σύνθετο Drawable που επιλέγει ένα Drawable από ένα σύνολο με βάση την κατάσταση του.

Levels: Ένα σύνθετο Drawable που επιλέγει ένα από ένα σύνολο των Drawables με βάση το επίπεδο του.

Scale: Ένα σύνθετο Drawable με ένα μοναδικό Drawable παιδί, του οποίου το συνολικό μέγεθος του έχει καθοριστεί με βάση το τρέχον επίπεδο του.

3.3.3 SurfaceView

Η κλάση SurfaceView παρέχει μία ειδική επιφάνεια ενσωματωμένη στο εσωτερικό μίας ιεραρχίας View. Μπορούμε να ελέγχουμε την μορφή της επιφάνειας αυτής και των διαστάσεων της. Το SurfaceView φροντίζει για την τοποθέτηση της επιφάνειας στην σωστή θέση στην οθόνη.

Η επιφάνεια είναι πίσω από το παράθυρο που περιέχει το SurfaceView. Η ιεραρχία view θα φροντίσει να γίνει μία σωστή σύνθεση των Surface και των τυχόν συνδεόμενων συστατικών σε αυτό για να εμφανίζονται από πάνω. Αυτό μπορεί να χρησιμοποιηθεί για να τοποθετήσει τα συστατικά πάνω στην επιφάνεια, όπως τα κουμπιά, ενώ πρέπει να ξέρουμε ότι αυτή η κίνηση μπορεί να επιδράσει στην απόδοση της εφαρμογής μας εφόσον κάθε φορά που το Surface αλλάζει θα εκτελείται μία πλήρες σύνθεση των συστατικών.

Η πρόσβαση στην υποκείμενη επιφάνεια παρέχεται από την διασύνδεση SurfaceHolder, που μπορεί να ανακτηθεί με την κλήση της getHolder().

Το Surface θα δημιουργείται, εφόσον είναι ορατό το παράθυρο του SurfaceView. Πρέπει να υλοποιήσουμε το surfaceCreated(SurfaceHolder) και surfaceDestroyed(SurfaceHolder) για να ανακαλύψουμε πότε μία Surface δημιουργείται και πότε καταστρέφεται όσο το παράθυρο εμφανίζεται ή κρύβεται.

Ένας από τους σκοπούς αυτής της κλάσης είναι να παρέχει μία επιφάνεια που θα μπορεί να τρέχει στο δευτερεύον νήμα της οθόνης. Κάποιος που θέλει να χρησιμοποιήσει την κλάση SurfaceView για τον σκοπό αυτό πρέπει να γνωρίζει τα παρακάτω:

> Όλες οι SurfaceView και SurfaceHolder.Callback μέθοδοι θα καλούνται από ένα νήμα που τρέχει το παράθυρο του SurfaceView, τυπικά είναι το βασικό νήμα μίας εφαρμογής. Πρέπει απλά να συγχρονίζονται σωστά με την κάθε κατάσταση που θα θίγεται από το σχεδιασμένο νήμα.
>
> Πρέπει να βεβαιωθούμε ότι όταν το Surface είναι έγκυρο θα αγγίζει μόνο το υποκείμενο Surface, μεταξύ SurfaceHolder.Callback.surfaceCreated() και SurfaceHolder.Callback.surfaceDestroyed().

3.3.4 CustomView

Το Android μας προσφέρει ένα ισχυρό μοντέλο συνθέσεων συστατικών για να χτίσουμε την διεπαφή χρήστη μας, με βάση τις βασικές κλάσεις διατάξεων που έχουμε: View και ViewGroup. Η πλατφόρμα στην αρχή περιέχει μία ποικιλία προκατασκευασμένων View και ViewGroup υποκλάσεων, που ονομάζονται widgets ή layouts και μπορούμε να χτίσουμε την διεπαφή χρήστη μας βασίζοντας σε αυτά.

Για τα widgets αυτά έχουμε μιλήσει και στα προηγούμενα κεφάλαια, μερικά από αυτά είναι τα Button, TextView, EditText, ListView, CheckBox, RadioButton, Gallery, Spinner, και τα ποιο ειδικού σκοπού όπως τα AutoCompleteTextView, ImageSwitcher και TextSwitcher.

Εάν κανένα από τα προκαθορισμένα widgets ή διατάξεις δεν ικανοποιούν τις ανάγκες μας, μπορούμε να δημιουργήσουμε την δικιά μας υποκατηγορία View. Όταν το μόνο που χρειαζόμαστε είναι να κάνουμε μικρές τροποποιήσεις στα υπάρχοντα widgets και διατάξεις μπορούμε απλά να υλοποιήσουμε την υποκλάση της widget ή διάταξης και να παρακάμψουμε τις μεθόδους της.

Η δημιουργία της δικιάς μας View υποκλάσης μας δίνει την δυνατότητα του ελέγχου της εμφάνισης της View και της λειτουργίας της. Στην ανάπτυξη της EGQL έχει χρησιμοποιηθεί ένα Custom SurfaceView. Παρακάτω βλέπουμε μερικά παραδείγματα για να καταλάβουμε τις δυνατότητες που έχουμε με προσαρμοσμένα View:

Θα μπορούσαμε να δημιουργήσουμε έναν εντελώς καινούργιο τύπο View, για παράδειγμα ένα “volume control” που μπορούμε να το χρησιμοποιήσουμε για να αλλάξουμε την ένταση του ήχου με την χρήση των δυο διαστάσεων γραφικών.

Μπορούμε να συνδυάσουμε μία ομάδα View συστατικών σε ένα μοναδικό συστατικό.

Θα μπορούσαμε, για παράδειγμα, να αλλάξουμε τον τρόπο που ένα EditText συστατικό αποδίδεται στην οθόνη.

Μπορούμε να προσθέσουμε άλλα γεγονότα όπως πατήματα πλήκτρων και χειρισμός αυτών με προσαρμοσμένους τρόπους.

ΚΕΦΑΛΑΙΟ 4: SQLite

4.1 Εισαγωγή στο SQLite

Το SQLite είναι ένας συμπαγής, ελαφρύς και ισχυρός μηχανισμός σχεσιακών βάσεων δεδομένων το οποίο έχει εισαχθεί στο Android και περιέχεται σε μία προγραμματιστική βιβλιοθήκη C. Το SQLite διατίθεται ελεύθερα και υποστηρίζει τα τυπικά χαρακτηριστικά των σχεσιακών βάσεων δεδομένων, όπως η σύνταξη SQL, συναλλαγές και έτοιμες καταστάσεις. Επιπλέον, απαιτεί μόνο λίγη μνήμη κατά τον χρόνο εκτέλεσης[23]. Το SQLite είναι η βάση δεδομένων που χρησιμοποιήθηκε κατά την ανάπτυξη της γλώσσας EGQL. Στην εικόνα 16 βλέπουμε το λογότυπο του SQLite.

Εικόνα 16: Λογότυπο SQLite

Το SQLite υποστηρίζει τύπους δεδομένων όπως TEXT(παρόμοια με τα String του Java), INTEGER (παρόμοια με την long της Java) και REAL(παρόμοια με την double σε Java). Όλοι οι υπόλοιποι τύποι πρέπει να μετατρέπονται σε έναν από αυτούς τους τύπους πριν αποθηκευτούν στην βάση δεδομένων. Το SQLite από μόνο του δεν εγκρίνει τους τύπους εάν οι τύποι που έχουν γραφτεί στις στήλες είναι των καθορισμένων τύπων, π.χ. μπορούμε να γράψουμε ένα integer σε μία στήλη string και αντιστρόφως.

Σε αντίθεση με άλλα συστήματα διαχείρισης βάσεων δεδομένων το SQLite δεν είναι ένας μηχανισμός πελάτη/εξυπηρετητή. Αντί αυτού ενσωματώνεται στην τελική εφαρμογή. Το SQLite είναι συμβατό με ACID και υλοποιεί το μεγαλύτερο μέρος του προτύπου SQL, χρησιμοποιώντας μια δυναμική και εβδομαδιαία τυπωμένη SQL σύνταξη που δεν εγγυάται την ακεραιότητα του τομέα.

Το SQLite είναι μια δημοφιλής επιλογή ως ενσωματωμένο λογισμικό βάσεων δεδομένων για τοπική αποθήκευση ή αποθήκευση πελάτη σε λογισμικό εφαρμογής όπως οι φυλλομετρητές. Είναι η πιο γρήγορα αναπτυσσόμενη μηχανή βάσης δεδομένων και χρησιμοποιείται σήμερα από πολλούς φυλλομετρητές

ευρείας χρήσης , από λειτουργικά συστήματα και από ενσωματωμένα συστήματα και άλλα[39].

Η σχεδίαση του SQLite έγινε από τον D. Richard Hipp. Ο D. Richard Hipp, ενώ δούλευε στη General Dynamics σε συμβόλαιο με το United States Navy, σχεδίασε το SQLite την άνοιξη του 2000 [40]. Ο Hipp σχεδίαζε λογισμικό για ενσωματωμένη χρήση σε καθοδηγούμενους καταστροφείς πυραύλων, που αρχικά βασίστηκε σε HP-UX με μια βάση δεδομένων IBM Informix. Ο σκοπός της σχεδίασης του SQLite ήταν να επιτρέπει στο πρόγραμμα να λειτουργήσει χωρίς εγκατάσταση ενός συστήματος διαχείρισης βάσης δεδομένων ή χωρίς να απαιτείται ένας διαχειριστής βάσης δεδομένων. Τον Αύγουστο του 2000, εκδόθηκε το SQLite 1.0, με βάση το gdbm (Διαχειριστής βάσης δεδομένων GNU). Το SQLite 2.0 αντικατέστησε το gdbm με μια προσαρμοσμένη εφαρμογή B-tree, προσθέτοντας υποστήριξη για συναλλαγές. Στο SQLite 3.0, που χρηματοδοτήθηκε μερικώς από την America Online, προστέθηκε διεθνοποίηση και τοπικοποίηση, manifest typing και άλλες σημαντικές βελτιώσεις.

Το 2011, ο Hipp ανακοίνωσε τα σχέδια του να προσθέσει μια διεπαφή UnQL στις βάσεις δεδομένων SQLite και να αναπτύξει το *UnQLite*, μια αντικειμενοστραφή βάση δεδομένων που θα μπορεί να ενσωματώνεται[41].

Το SQLite στις μέρες μας συνδέεται με ένα μεγάλο αριθμό γλωσσών προγραμματισμού, μερικές από αυτές είναι η Basic, η Delphi, η C, η C#, η C++, η Java, η JavaScript, η Julia, η Perl , η PHP, η Python, η Ruby, η Swift και η Visual Basic.

Το SQLite περιλαμβάνεται ως προεπιλογή στο λειτουργικό σύστημα Android. Άλλα παραδείγματα τέτοιων λειτουργικών συστημάτων είναι το BlackBerry 10 OS, το Windows Phone 8, το Symbian OS, το OpenBSD, το Oracle Solaris 10 και το iOS.

4.2 SQLite σε Android

Λόγω του μικρού μεγέθους το SQLite προσαρμόζεται σε πολλά ενσωματωμένα συστήματα, ένα από τα ποιο βασικά από αυτά τα συστήματα είναι το Android. Το SQLite είναι διαθέσιμο σε όλες τις συσκευές Android. Η χρήση μίας βάσης δεδομένων SQLite σε Android δεν απαιτεί καμία εγκατάσταση ή διαχείριση βάσης δεδομένων.

Το μόνο που πρέπει να κάνουμε είναι να καθορίσουμε τις SQL προτάσεις για την δημιουργία και την ενημέρωση της βάσης δεδομένων. Στη συνέχεια η βάση δεδομένων μπορεί να διαχειρίζεται από την πλατφόρμα Android.

Η πρόσβαση σε μία βάση δεδομένων SQLite περιλαμβάνει και την πρόσβαση στο σύστημα αρχείων, που μπορεί να είναι και αργή. Για αυτό συνιστάται να εκτελέσουμε ασύγχρονα τις λειτουργίες βάσης δεδομένων, για παράδειγμα με τη χρήση της κλάσης AsyncTask.

Όταν η εφαρμογή μας δημιουργεί μία βάση δεδομένων, αυτή η βάση αποθηκεύεται στον κατάλογο DATA/data/APP_NAME/databases/FILENAME.

Τα κομμάτια του παραπάνω καταλόγου αποτελούνται από τους ακόλουθους κανόνες. Το DATA είναι η διαδρομή η οποία μας επιστρέφει η μέθοδος Enviroment.getDataDirectory(). Το APP_NAME είναι το όνομα της εφαρμογής μας. Το FILE_NAME είναι το όνομα της βάσης δεδομένων που ορίζουμε στην εφαρμογή μας. Η μορφή των αρχείων βάσης δεδομένων είναι τυποποιημένη και μπορεί να μεταφερθεί σε άλλες πλατφόρμες. Μπορούμε να χρησιμοποιήσουμε το πρόγραμμα εξερεύνησης File Explorer αρχείων του DDMS (Dalvik Debug Monitor Service) ώστε να εξάγουμε το αρχείο βάσης δεδομένων και να το εξετάσουμε με εργαλεία τρίτων, αν θέλουμε.

Το πακέτο android.database.sqlite περιέχει τις κλάσεις διαχειρίσεων των βάσεων δεδομένων που θα χρησιμοποιούσε μία εφαρμογή για να διαχειριστεί τις ιδιωτικές βάσεις δεδομένων της. Οι εφαρμογές χρησιμοποιούν τις κλάσεις αυτές για την διαχείριση των ιδιωτικών βάσεων δεδομένων. Εάν δημιουργούμε ένα content provider πιθανότατα θα πρέπει να χρησιμοποιήσουμε τις κλάσεις αυτές για την δημιουργία και διαχείριση της βάσης δεδομένων μας για την αποθήκευση των περιεχομένων. Το Android λειτουργεί με την έκδοση sqlite 3.4.0 και με το εργαλείο sqlite3 που βρίσκεται στο αρχείο /folder. Μπορούμε να χρησιμοποιήσουμε το εργαλείο αυτό για την εξερεύνηση και εκτέλεση των εντολών SQL στην συσκευή. Η εκτέλεση γίνεται με την εντολή sqlite3 στο παράθυρο shell.

Το σύστημα Android μας παρέχει βοηθητικές κλάσεις για την δημιουργία και διαχείριση των SQLite βάσεων δεδομένων. Οι δυο κλάσεις που μπορούμε να διαχειριστούμε τις βάσεις δεδομένων SQLite είναι:

SQLiteOpenHelper

SQLiteAssetHelper

4.2.1 SQLiteOpenHelper

Το SQLiteOpenHelper είναι μία βοηθητική κλάση για την διαχείριση δημιουργίας βάσεων δεδομένων και διαχείριση εκδόσεως. Δημιουργούμε μία υποκλάση που υλοποιεί τις μεθόδους onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) και προαιρετικά την onOpen(SQLiteDatabase)[42]. Η κλάση αυτή φροντίζει να ανοίγει τη βάση δεδομένων εφόσον υπάρχει, τη δημιουργία της βάσης όταν δεν υπάρχει και την ενημέρωση της όταν είναι απαραίτητο. Οι συναλλαγές χρησιμοποιούνται για να βεβαιωθούμε ότι η βάση δεδομένων είναι πάντα σε μία λογική κατάσταση.

Η κλάση αυτή διευκολύνει τις υλοποιήσεις των ContentProvider να αναβάλλουν το άνοιγμα και την ενημέρωση της βάσης πριν την πρώτη χρήση, για την αποφυγή της εγκατάστασης εφαρμογής με μακροχρόνιες αναβαθμίσεις βάσεων δεδομένων.

Οι δημόσιοι δομητές τις κλάσεις αυτής είναι:

SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) : Δημιουργεί ένα βοηθητικό αντικείμενο για την δημιουργία, άνοιγμα ή και διαχείριση της βάσης δεδομένων.

SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler) : Δημιουργεί ένα βοηθητικό αντικείμενο για την δημιουργία, άνοιγμα ή και διαχείριση της βάσης δεδομένων.

4.2.2 SQLiteAssetHelper

Το SQLiteAssetHelper είναι μία κλάση που μας βοηθάει στην διαχείριση της δημιουργίας βάσεων δεδομένων και διαχείριση εκδόσεως με την χρήση των raw asset αρχείων της εφαρμογής[43].

Η κλάση αυτή παρέχει στους προγραμματιστές με έναν απλό τρόπο να αναπτύσσουν την εφαρμογή τους με μία ήδη υπάρχουσα βάση δεδομένων (που μπορεί να συμπληρωθούν τα δεδομένα της) και να διαχειρίζονται την αρχική δημιουργία και τις τυχόν αναβαθμίσεις που μπορούν αν προκύψουν με τις επόμενες εκδόσεις που θα κυκλοφορήσουν.

Υλοποιείται ως μία επέκταση της κλάσης SQLiteOpenHelper, παρέχοντας έναν αποτελεσματικό τρόπο για υλοποιήσεις ContentProvider να αναβάλλουν το άνοιγμα και την ενημέρωση της βάσης μέχρι την πρώτη χρήση.

Αντί να χρησιμοποιήσουμε τις μεθόδους onCreate() και onUpgrade() για να εκτελέσουμε SQL προτάσεις, μπορούμε απλά να περιλαμβάνουμε την βάση στα αρχεία asset της εφαρμογής μας με τον κατάλληλο όνομα και την κατάλληλη θέση στον κατάλογο. Αυτό θα αρκεί για την αρχικοποίηση του αρχείου SQLite βάσης δεδομένων για την δημιουργία και προαιρετικά για τις πιθανές αναβαθμίσεις SQL.

Η βοηθητική κλάση αυτή έχει χρησιμοποιηθεί και στην εργασία μας κατά την διαχείριση της βάσης μας στην EGQL. Για την χρήση της κλάσης αυτής, την επεκτείνουμε και την αρχικοποιούμε στον δομητή της. Για να μπορούμε να χρησιμοποιήσουμε την κλάση αυτή πρέπει η βάση δεδομένων μας να βρίσκεται στον κατάλογο του έργου. Στον παρακάτω κώδικα Java βλέπουμε την αρχικοποίηση της κλάσης SQLiteAssetHelper:

```
public class MyDatabase extends SQLiteAssetHelper {
        private static final String DATABASE_NAME = "Chinook.db";
        private static final int DATABASE_VERSION = 1;

        public MyDatabase(Context context) {
```

```
            super(context, DATABASE_NAME, null,DATABASE_VERSION);
        }
    }
```

4.3 Sqlite3

Εκτός από την προγραμματιστική πρόσβαση στη διαδικασία δημιουργίας και χρήσης βάσεων δεδομένων SQLite μέσα από τις εφαρμογές μας, μπορούμε επίσης να χρησιμοποιήσουμε το εργαλείο γραμμής εντολών sqlite3, εκτίθεται αν χρησιμοποιήσουμε το εργαλείο ADB (Android Debug Bridge), που είναι το εργαλείο γραμμής εντολών του Android[23].

Το Sqlite3 περιλαμβάνει πολλές χρήσιμες εντολές, όπως η .dump για την εμφάνιση των περιεχομένων του πίνακα ή η .schema για την εμφάνιση των SQL CREATE προτάσεων για το υπάρχον πίνακα. Το εργαλείο επίσης μας δίνει την δυνατότητα εκτέλεσης των SQLite εντολών.

Για την χρήση της sqlite3 από έναν απομακρυσμένο κέλυφος:

1. Εισαγωγή στον απομακρυσμένο κέλυφος με την παρακάτω εντολή:

```
adb [-d|-e|-s {<serialNumber>}] shell
```

2. Το ξεκίνημα του εργαλείου sqlite3 με την εισαγωγή της παρακάτω εντολής από τον απομακρυσμένο κέλυφος:

```
sqlite3
```

Μπορούμε επίσης να ορίσουμε προαιρετικά μία διαδρομή στην βάση δεδομένων που θέλουμε να εξερευνήσουμε. Στην περίπτωση αυτή ο προσομοιωτής ή η συσκευή θα αποθηκεύσει SQLite3 βάσεις δεδομένων στον κατάλογο /data/data/<package_name>/databases/.

3. Εφόσον έχει γίνει η κλήση της sqlite3 μπορούμε να εκτελέσουμε τις εντολές της sqlite3 στο κέλυφος. Για να βγούμε ή να επιστρέψουμε στον απομακρυσμένο κέλυφος adb , εισάγουμε exit ή πατάμε CTRL+D στο πληκτρολόγιο.

Παρακάτω βλέπουμε ένα παράδειγμα:

```
$ adb -s emulator-5554 shell


# sqlite3
/data/data/com.example.google.rss.rssexample/databases/rssitems.db
```

```
SQLite version 3.3.12

Enter ".help" for instructions

.... enter commands, then quit...

# sqlite> .exit
```

Για την τοπική χρήση της sqlite3, αντί να χρησιμοποιήσουμε ένα κέλυφος, αποκτούμε το αρχείο της βάσης δεδομένων από την συσκευή και ξεκινάμε το sqlite3.

1. Αντιγραφή του αρχείου βάσης δεδομένων από την συσκευή στον υπολογιστή μας:

```
adb pull <database-file-on-device>
```

2. Ξεκίνημα του εργαλείου sqlite3 από τον κατάλογο /tools, καθορισμός του αρχείου βάσης δεδομένων:

```
sqlite3 <database-file-on-host>
```

Μετά την σύνδεση του εργαλείου μπορούμε να αλληλεπιδράσουμε με την βάση δεδομένων μας, και να εκτελέσουμε τις διάφορες εντολές που υποστηρίζει. Με τις εντολές αυτές μπορούμε να κάνουμε διάφορες πράξεις όπως:

Εξερεύνηση της βάσης δεδομένων, με τις εντολές όπως .databases, .tables, .indices, .schema.

Εισαγωγή και εξαγωγή της βάσης δεδομένων και των δεδομένων της με τις εντολές όπως .output.

Εκτέλεση εντολών SQLite στη γραμμή εντολών

Μπορούμε να παρατηρήσουμε ότι οι εντολές sqlite3 ξεκινάνε με μία τελεία από μπροστά για να μπορεί το σύστημα να τα ξεχωρίζει από τις εντολές SQL. Για να δούμε μία πλήρη λίστα με τις εντολές της sqlite3 πληκτρολογούμε την παρακάτω εντολή:

```
sqlite> .help
```

4.4 Η σωστή χρήση του SQLite

Το SQLite είναι αρκετά ισχυρό αλλά έχει και σημαντικούς περιορισμούς σε σχέση με παραδοσιακές υλοποιήσεις του SQL server, όπως η MySQL, η Oracle ή η PostgreSQL. Οι μηχανές βάσεων δεδομένων SQL πελάτη/εξυπηρετητή προσπαθούν να εφαρμόσουν ένα κοινό αρχείο καταγραφής δεδομένων των δεδομένων των επιχειρήσεων. Δίνουν έμφαση στην επεκτασιμότητα, στον ταυτοχρονισμό, στην συγκέντρωση και έλεγχο. Το SQLite προσπαθεί να παρέχει έναν τοπικό χώρο αποθήκευσης δεδομένων για μεμονωμένες εφαρμογές και συσκευές. Το SQLite δίνει έμφαση στην οικονομία, στην αποδοτικότητα, στην αξιοπιστία, στην ανεξαρτησία και στην απλότητα[44].

Οι περιπτώσεις που μία SQLite βάση δεδομένων θα δούλευε χωρίς προβλήματα είναι:

- Ενσωματωμένες συσκευές και το διαδίκτυο των πραγμάτων
- Σε μορφή αρχείου εφαρμογής
- Ιστοσελίδες απλές
- Ανάλυση δεδομένων
- Μνήμη cache για δεδομένα επιχειρήσεων
- Δεδομένα διακομιστή
- Αρχειοθέτηση
- Αντικατάσταση ad-hoc αρχείων στο δίσκο
- Εσωτερική ή προσωρινή βάση δεδομένων
- Stand-in για μία βάση δεδομένων της επιχείρησης κατά την διάρκεια των επιδείξεων ή δοκιμών
- Εκπαίδευση και εξάσκηση
- Πειραματικές εξετάσεις της γλώσσας SQL

Οι περιπτώσεις που θα πρέπει καλύτερα να χρησιμοποιήσουμε ένα σύστημα διαχείρισης βάσεων δεδομένων πελάτη/εξυπηρετητή είναι όταν έχουμε:

- Οι εφαρμογές πελάτη/εξυπηρετητή
- Ιστοσελίδες μεγάλου όγκου
- Πολύ μεγάλα σύνολα δεδομένων
- Μεγάλες ταχύτητες ταυτοχρονισμού

Αναλύοντας τις παραπάνω περιπτώσεις μπορεί να καταλάβει κανείς γιατί επιλέχθηκε το SQLite κατά την ανάπτυξη της EGQL.

4.5 Παραδείγματα SQLite

Για την καλύτερη κατανόηση της SQLite θα εξηγήσουμε τις τυπικές εντολές SQL για τη δημιουργία και επεξεργασία μίας βάσης δεδομένων. Ξεκινάμε με τον σχεδιασμό μίας βάσης δεδομένων, πρέπει να ξέρουμε ότι η sqlite3 υποστηρίζει του παρακάτω κοινούς τύπους[23]:

INTEGER (προσημασμένοι ακέραιοι)

REAL (δεκαδικοί αριθμοί)

TEXT (συμβολοσειρές UTF-8 ή UTF-16, κωδικοποιημένες με την κωδικοποίηση της βάσης δεδομένων)

BLOB (κομμάτι δεδομένων)

Μία μικρή συμβουλή θα μπορούσε να είναι να μην αποθηκεύουμε τα αρχεία, όπως οι εικόνες, κατευθείαν στη βάση δεδομένων. Αντίθετα, να αποθηκεύουμε τις εικόνες σαν αρχεία στον κατάλογο του αρχείου εφαρμογής και να αποθηκεύουμε το όνομα του αρχείου ή τη διαδρομή URI στη βάση δεδομένων.

Ακολουθεί ένα παράδειγμα πρότασης SQL για την δημιουργία απλών πινάκων:

CREATE TABLE Students (

id INTEGER PRIMARY KEY AUTOINCREMENT,

fname TEXT NOT NULL,

lname TEXT NOT NULL);

Ο πίνακας θα έχει ένα id που θα είναι το κύριο κλειδί και θα αυξάνεται αυτόματα με την κάθε εισαγωγή. Ένα fname που αντιπροσωπεύει το όνομα ενός φοιτητή και το lname για το επίθετο του. Οι τιμές αυτές πρέπει να περιέχουν τα ονόματα των φοιτητών, δεν μπορούν να έχουν null τιμές.

Για την εισαγωγή στον πίνακα αυτό μπορούμε να πληκτρολογήσουμε την παρακάτω πρόταση:

INSERT into Students (fname, lname)

VALUES ('Harry', 'Potter');

Μετά την εκτέλεση της πρότασης θα εισαχθεί στον πίνακα ο φοιτητής με όνομα Harry και επίθετο Potter.Το id της εγγραφής θα δοθεί αυτόματα από το σύστημα ανάλογα με την σειρά εγγραφής του φοιτητή, δηλαδή εάν είναι ο δέκατος φοιτητής που γράφτηκε θα έχει id = 10.

Μπορούμε να κάνουμε ερωτήματα σε αυτούς τους πίνακες για αποτελέσματα με την εντολή *SELECT*. Για παράδειγμα για να εμφανίσουμε όλες τις εγγραφές των φοιτητών στον πίνακα Students με όλα του τα πεδία πρέπει να εκτελέσουμε την παρακάτω πρόταση:

*SELECT * FROM Students;*

Η εκτέλεση του ερωτήματος θα μας επιστρέψει όλους τους φοιτητές που έχουν αποθηκευτεί στον πίνακα μας με όλα τους τα πεδία.

Για να επιστρέψουμε το ονοματεπώνυμο σαν ένα πεδίο για κάθε φοιτητή, γράφουμε την παρακάτω πρόταση

SELECT fname || ' ' || lname AS Fullname, id

FROM Students

Αν υποθέσουμε ότι έχουμε 3 φοιτητές στον πίνακα μας η εκτέλεση της πρότασης αυτής μπορεί να μας δώσει το παρακάτω αποτέλεσμα:

Fullname	*id*
Harry Poter	*1*
Stan Cartman	*2*
Aren Erkan	*3*

Στη παρακάτω πρόταση θα προσθέσουμε έναν πίνακα που θα έχει δύο ξένα κλειδιά από τα κύρια κλειδιά των δυο πινάκων και θα τα χρησιμοποιεί ως κύριο κλειδί, και το αποτέλεσμα του συγκεκριμένου τεστ για τον συγκεκριμένο φοιτητή που ο βαθμός θα πρέπει να είναι μεταξύ 0 και 100. Ας υποθέσουμε ότι έχουμε έναν δεύτερο πίνακα στη βάση με όνομα Tests και με κύριο κλειδί το id και δυο άλλα πεδία με ονόματα testname που δίνει ένα φιλικό όνομα σε κάθε τεστ και weight που είναι μία ποσοστιαία τιμή η οποία αντανακλά το μετρητή διαγωνισμάτων για τον τελικό βαθμό του φοιτητή. Τώρα μπορούμε να δημιουργήσουμε τον πίνακα που θα περιέχει τα αποτελέσματα των εξετάσεων:

CREATE TABLE TestResults (

studentid INTEGER REFERENCES Students(id),

testid REFERENCES Tests(id),

score INTEGER CHECK (score<=100 AND score>=0)

PRIMARY KEY (studentid, testid));

Μπορούμε επίσης να κάνουμε αλλαγές ή ενημερώσεις στους πίνακες. Για παράδειγμα, υποθέτουμε ότι θέλουμε να αλλάξουμε τον βαθμό που πήρε ο φοιτητής με id 2 στο τεστ με id 1. Ο βαθμός ήταν 55 και θέλουμε να το κάνουμε 70:

UPDATE TestResults

SET score=60

WHERE studentid=2 AND testid=1;

Για παράδειγμα όταν θέλουμε να διαγράψουμε το αποτέλεσμα του τεστ αυτού θα πρέπει να εκτελέσουμε την παρακάτω πρόταση:

DELETE FROM TestResults

WHERE studentid=2 AND testid=1;

Για να διαγράψουμε όλες τις γραμμές του πίνακα αυτού εκτελούμε την παρακάτω πρόταση:

DELETE FROM TestResults;

Για την διαγραφή του πίνακα από την βάση εκτελούμε την παρακάτω πρόταση:

DROP TABLE TestResults;

Με το SQLite μπορούν να υλοποιηθούν οι περισσότερες ιδιότητες των κοινών τυπικών ιδιοτήτων του SQL. Παρακάτω είναι η λίστα των ιδιοτήτων που δεν υποστηρίζονται από το SQLite ενώ υποστηρίζονται από SQL:

RIGHT και FULL OUTER JOIN: η LEFT OUTER JOIN υλοποιείται από το SQLite αλλά όχι το RIGHT OUTER JOIN και FULL OUTER JOIN.

Πλήρης υποστήριξη ALTER TABLE: Υποστηρίζονται μόνο οι παραλλαγές RENAME TABLE και ADD COLUMN της εντολής ALTER TABLE. Οι άλλες λειτουργίες του ALTER TABLE δεν υποστηρίζονται, όπως το DROP COLUMN, ALTER COLUMN, ADD CONSTRANT και ούτω καθεξής

Πλήρης υποστήριξη των trigger: Οι triggers FOR EACH ROW υποστηρίζονται αλλά όχι τα FOR EACH STATEMENT triggers.

Εγγραφή στα VIEWs: Τα VIEWs στο SQLite είναι μόνο για ανάγνωση. Δεν μπορούμε να εκτελέσουμε μία δήλωση DELETE, INSERT ή UPDATE σε ένα view και να κάνουμε αυτό που θέλουμε στο σώμα του trigger.

GRANT και REVOKE: Εφόσον το SQLite κάνει την ανάγνωση και εγγραφή σε ένα τακτικό αρχείο στο δίσκο, τα μόνα δικαιώματα μπορούν να εφαρμοστούν είναι αυτά του υποκείμενου λειτουργικού συστήματος. Οι εντολές GRANT και REVOKE συνήθως χρησιμοποιούνται σε βάσεις δεδομένων πελάτη/εξυπηρετητή και δεν υλοποιούνται στο SQLite λόγω του

ότι δεν θα είχαν νόημα για ένα ενσωματωμένο σύστημα διαχείρισης βάσης δεδομένων.

ΚΕΦΑΛΑΙΟ 5: GRAPHICAL QUERY LANGUAGE

5.1 Εισαγωγή στο GQL

Τα graphic query languages είναι γλώσσες που θέτουν ερωτήματα στις βάσεις δεδομένων με την χρήση των οπτικών αναπαραστάσεων για να απεικονίσουν την περιοχή ενδιαφέροντος και να εκφράσουν τις σχετικές αιτήσεις. Τα GQL μας παρέχουν μία γλώσσα για να εκφράσουμε τα ερωτήματά μας σε οπτική μορφή και είναι προσανατολισμένα σε ένα ευρύ φάσμα χρηστών, ιδιαίτερα των αρχαρίων που έχουν περιορισμένη εμπειρία υπολογιστών και γενικά αγνοούν την εσωτερική δομή μίας βάσης δεδομένων. Η πλειοψηφία των χρηστών ανήκουν σε αυτήν την κατηγορία των αρχαρίων, ειδικά των χρηστών των συσκευών μικρών διαστάσεων όπως τα smartphones και τα tablets που το μόνο που χρειάζονται είναι να μάθουν να ολοκληρώνουν τις απλές εργασίες Τα προβλήματα που καλούνται για να λύσουν συνήθως μπορούν να εκφράζονται με μη υπολογιστικούς όρους.

Για κάθε είδους αλληλεπιδραστικού συστήματος ο σημαντικότερος σκοπός είναι η αλληλεπίδραση με τον τελικό χρήστη που θα χειρίζεται το σύστημα και όχι η αλληλεπίδραση με το σύστημα. Το σύστημα πρέπει να προσαρμόζεται για να διευκολύνει τους χρήστες στην εκτέλεση των διεργασιών τους. Ως εκ τούτου, τα χαρακτηριστικά των κλάσεων που θα λειτουργούν με ένα συγκεκριμένο περιβάλλον και οι εργασίες που θα εκτελέσει ο χρήστης πρέπει να είναι κατανοητές.

Οι εργασίες των βασικών χρηστών είναι η κατανόηση του περιεχομένου της βάσης δεδομένων, εστίαση στα σημαντικά στοιχεία, εύρεση προτύπων ερωτημάτων και συλλογή αποτελεσμάτων των ερωτημάτων. Οι εργασίες αυτές απαιτούν ειδικές τεχνικές για να επιτυγχάνονται με αποτελεσματικότητα και τέτοιες τεχνικές περιλαμβάνουν τυπικές δραστηριότητες όπως την περιήγηση, την εστίαση, το φιλτράρισμα και ζουμ. Μπορούμε να πούμε ότι οι γραφικές διεπαφές χρηστών είναι οι καταλληλότερες για την εκτέλεση των παραπάνω εργασιών.

Η κοινότητα των βάσεων δεδομένων επίσης αναγνωρίζει την σημασία της γραφικής αναπαράστασης για την εννοιολογική σχεδίαση και η έρευνα σε αυτόν τον τομέα μας οδήγησε στην ανάπτυξη αρκετών εννοιολογικών μοντέλων όπως το Entity Relationship Model[45]. Υπήρχε επίσης μεγάλο ενδιαφέρον στην ανάπτυξη των Graphical Query Languages(GQL) που είναι οι γραφικές διεπαφές που επιτρέπουν την εκτέλεση ad-hoc ερωτημάτων στην βάση δεδομένων.

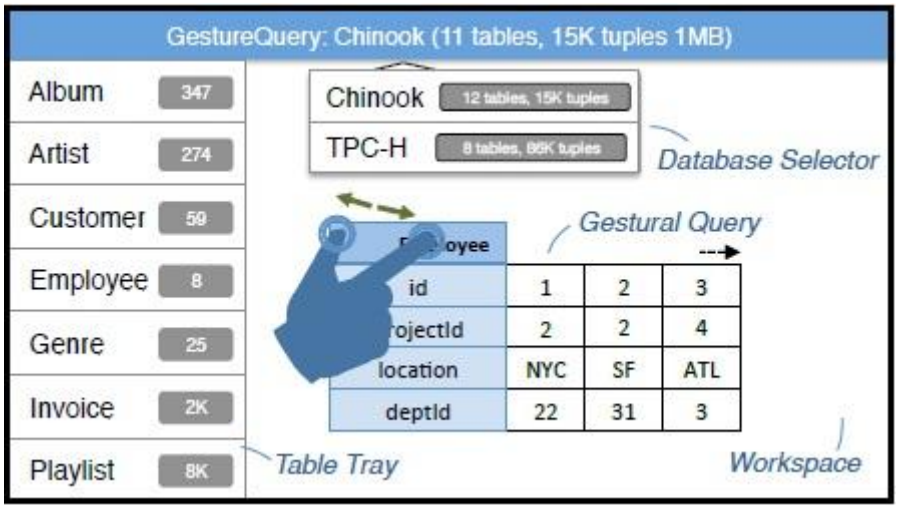
5.2 Ανάλυση και Σύγκριση των Γραφικών Γλωσσών Ερωτημάτων

Έχουν γίνει αρκετές προσεγγίσεις στο θέμα των γραφικών ερωτημάτων. Μία καλή ανάλυση και σύγκριση των γραφικών γλωσσών βάσεων δεδομένων γίνεται στο

άρθρο[46]. Οι γλώσσες που αναλύονται στο άρθρο[46] δεν χρησιμοποιούν χειρονομίες στην εκτέλεση των ερωτημάτων, η μοναδική γλώσσα που χρησιμοποιείται η οθόνη αφής και οι χειρονομίες είναι η GestureQuery[47] και βασίζεται σε iOS.

Το μοντέλο GestureQuery

Ένα Gesture Query Language είναι μία γλώσσα γραφικών ερωτημάτων που τα ερωτήματα δημιουργούνται με την χρήση των χειρονομιών που πραγματοποιούνται στην οθόνη. Οι διάφορες χειρονομίες μεταφράζονται σε κομμάτια κώδικα SQL και εκτελούνται πάλι με την χρήση των χειρονομιών. Στο άρθρο [47] έχουν μελετηθεί διάφορες γλώσσες και προσεγγίσεις στο θέμα της γραφικής αναπαράστασης των σχεσιακών βάσεων δεδομένων. Στο ίδιο άρθρο αναλύεται και η γλώσσα GestureQuery. Η γλώσσα αυτή βασίζεται σε χειρονομίες που γίνονται σε μία συσκευή με οθόνη αφής και βασίζεται σε λειτουργικό σύστημα iOS. Όπως βλέπουμε και στην εικόνα 17, η διεπαφή του μοντέλου αυτού χωρίζεται σε τρία κομμάτια. Το πρώτο που περιέχει το όνομα του επιλεγμένου πίνακα, το δεύτερο που περιέχει τους διαθέσιμους πίνακες και το τρίτο που είναι το κομμάτι που δημιουργούνται τα ερωτήματα του χρήστη. Το μοντέλο αυτό απευθύνεται σε οθόνες μεσαίων διαστάσεων όπως τα tablets και όχι σε smartphones και θα ήταν δύσκολη η χρήση του σε οθόνες μικρών διαστάσεων με πολλούς πίνακες πολλών ιδιοτήτων.



Εικόνα 17: Το μοντέλο GestureQuery

Το μοναδικό παράδειγμα στις γλώσσες χειρονομιών οθόνης αφής είναι η GestureQuery που έχουμε αναφέρει παραπάνω. Στην EGQL επίσης γίνονται χρήση των χειρονομιών σε μία multi-touch οθόνη. Χρησιμοποιούνται επίσης οι χειρονομίες για το σωστό μέγεθος και θέση των πινάκων και ολόκληρου του καμβά. Τα χρώματα που χρησιμοποιούνται στην EGQL για την αναπαράσταση των πληροφοριών είναι επίσης μία σημαντική δυνατότητα εκμετάλλευσης χώρου των μικρών οθονών για την αναπαράσταση του χώρου. Στο άρθρο [48] γίνεται η

ανάλυση των διάφορων τύπων των χειρονομιών που πραγματοποιούνται στις οθόνες αφής.

ΚΕΦΑΛΑΙΟ 6: EGQL

6.1 Η Διεπαφή Χρήστη του EGQL

Το κύριο περιβάλλον που δημιουργούνται όλα τα ερωτήματα της γλώσσας EGQL βασίζεται στα Surfaces του λειτουργικού συστήματος Android, και τα ερωτήματα βασίζονται στο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων SqLite. Οι multi-touch χειρονομίες που πραγματοποιούνται στην οθόνη της συσκευής μεταφράζονται σε ερωτήματα sql, εκτελούνται σε μία βάση δεδομένων SqLite και τροποποιούνται τα αποτελέσματα έτσι ώστε να εμφανίζονται στον τελικό χρήστη. Το πρώτο βήμα για την δημιουργία ενός ερωτήματος είναι να επιλέγουμε τους πίνακες που σχετίζονται με το ερώτημα που θέλουμε να δημιουργήσουμε. Στην EGQL, οι πίνακες επιλέγονται με δυο τρόπους. Ο πρώτος είναι να επιλέγουμε το “Select Tables” από το μενού της εφαρμογής και επιλογή των κατάλληλων πινάκων που διαθέτονται από την τρέχουσα βάση. Ο δεύτερος τρόπος είναι να κάνουμε διπλό κλικ σε μία περιοχή του καμβά που είναι άδεια και μετά να επιλέγουμε τους πίνακες.

Ο κάθε πίνακας της βάσης δεδομένων παριστάνεται γραφικά με έναν πίνακα EGQL που περιέχει δυο στήλες και γραμμές όσες είναι τα πεδία του επιλεγμένου πίνακα της βάσης δεδομένων. Η πρώτη στήλη ενός πίνακα EGQL είναι η στήλη που αντιπροσωπεύει την συνιστώσα SELECT ενός ερωτήματος SQL και με τις αλλαγές που κάνουμε στην στήλη αυτή επιλέγουμε τα πεδία που θα εμφανιστούν στο τελικό αποτέλεσμα. Η πρώτη γραμμή της στήλης αυτής είναι πράσινη και περιέχει ένα μάτι μέσα της, με ένα κλικ στο μάτι αυτό επιλέγονται όλα τα κελιά της πρώτης στήλης του πίνακα και γίνονται και αυτά πράσινα που σημαίνει ότι θα εμφανιστούν όλα αυτά τα πεδία στο τελικό αποτέλεσμα του ερωτήματος. Η μετάφραση της χειρονομίας αυτής σε SQL κώδικα είναι:

*SELECT TableName.**

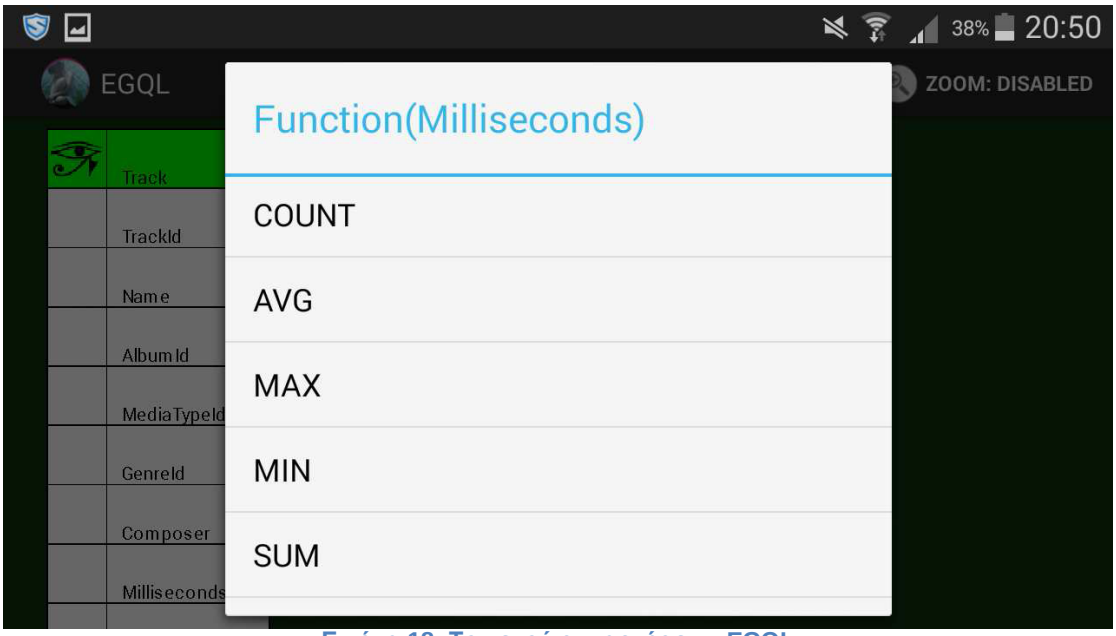
Με διπλό κλικ στο μάτι του πίνακα μπορούμε να επιλέξουμε μία συνάρτηση που θα ισχύει για όλο τον πίνακα και θα εμφανιστεί στο τελικό αποτέλεσμα, επίσης το κελί με το μάτι θα γίνει σκούρο πράσινο. Η μετάφραση της χειρονομίας αυτής σε SQL κώδικα είναι:

SELECT FUNCTION(TableName.)*

Ενώ, με διπλό κλικ στα υπόλοιπα στοιχεία της πρώτης στήλης μπορούμε να επιλέξουμε συναρτήσεις για τα πεδία του πίνακα της επιλογής μας και όχι για όλο το πίνακα. Μετά την επιλογή της συνάρτησης, το κελί του πίνακα EGQL παίρνει μπλε χρώμα. Η μετάφραση της χειρονομίας αυτής είναι:

SELECT FUNCTION(TableName.SelectedField)

Το μενού επιλογών μίας συνάρτησης απεικονίζεται στην εικόνα 18.



Εικόνα 18: Το μενού συναρτήσεων EGQL

Η δεύτερη στήλη και πρώτη γραμμή των πινάκων EGQL είναι πράσινη και περιλαμβάνει το όνομα του πίνακα. Κάνοντας κλικ στο όνομα του πίνακα με απενεργοποιημένο ζουμ μπορούμε να αλλάξουμε την θέση του πίνακα στον καμβά για να το επεξεργαστούμε ευκολότερα. Οι επόμενες γραμμές της δεύτερης στήλης του πίνακα EGQL υποδηλώνουν τα ονόματα των πεδίων του πίνακα της βάσης, και οι αλλαγές που γίνονται σε αυτά τα κελιά θα προσθέτονται στην συνιστώσα WHERE του τελικού ερωτήματος.

Για να εισάγουμε μια συνθήκη στην συνιστώσα WHERE κάνουμε κλικ στο όνομα του πεδίου και επιλέγουμε το στοιχείο που θέλουμε να αναζητήσουμε από τα διαθέσιμα στοιχεία που περιλαμβάνει το συγκεκριμένο πεδίο του πίνακα στην βάση δεδομένων. Μετά την επιλογή του στοιχείου το κελί γίνεται κίτρινο. Εάν δεν είναι ενεργοποιημένη η αυτόματη εύρεση ξένων κλειδιών και δημιουργία συζεύξεων μπορούμε να συνδέσουμε πίνακες κάνοντας διπλό κλικ στα πεδία των πινάκων που θέλουμε να συνδέσουμε, συνήθως είναι τα κύρια κλειδιά των πινάκων. Μετά την επιλογή δημιουργίας των συζεύξεων τα κελιά των πινάκων γίνονται ματζέντα, εάν ήδη είχε γίνει προσθήκη μιας συνθήκης στον ίδιο κελί, το κελί θα γίνει κυανό.

Με την EGQL μπορούμε να κάνουμε zoom in και zoom out για να προσαρμόσουμε τις διαστάσεις του καμβά, έτσι ώστε να μπορούμε να κάνουμε ευκολότερα τις πράξεις που επιθυμούμε να κάνουμε ανεξάρτητα από την οθόνη της συσκευής ή τον αριθμό και τις διαστάσεις των πινάκων. Μετά την σμίκρυνση ή μεγέθυνση του πίνακα στις επιθυμητές διαστάσεις μπορούμε να προσαρμόσουμε και την θέση του καμβά μέσα στην οθόνη χωρίς να χρειαστεί να αλλάξουμε τις θέσεις των πινάκων ένα-ένα, εφόσον η λειτουργία ζουμ είναι ενεργοποιημένη. Για την

ενεργοποίηση της λειτουργίας ζουμ κάνουμε κλικ στο κουμπί που βλέπουμε στην εικόνα 19.

Εικόνα 19: Το κουμπί zoom

Για να κάνουμε zoom in ή zoom out ακουμπάμε με δυο δάχτυλα στην οθόνη και αλλάζουμε την απόσταση μεταξύ δαχτύλων που είναι πάνω στην οθόνη έτσι ώστε να βρούμε την σωστή διάσταση καμβά που ικανοποιεί τις ανάγκες μας για το συγκεκριμένο αριθμό πινάκων που έχουν επιλεχθεί για το ερώτημα που επιθυμούμε να εκτελέσουμε. Μετά την επιλογή της διάστασης, βρίσκουμε την καταλληλότερη θέση του καμβά μέσα στην οθόνη της συσκευής κάνοντας μονό κλικ οπουδήποτε μέσα στον καμβά και σύροντας τον καμβά στην οθόνη, εφόσον είναι ενεργοποιημένη η λειτουργία ζουμ. Όταν είναι ενεργοποιημένη η λειτουργία ζουμ δεν μπορούμε να κάνουμε αλλαγές στο ερώτημα, τα κλικ που κάνουμε στον καμβά θα μεταφράζονται ως αλλαγή θέσης του καμβά στην οθόνη της συσκευής. Για να επιστρέψουμε στην επεξεργάσιμη μορφή του καμβά του ερωτήματος πρέπει να απενεργοποιήσουμε την λειτουργία ζουμ κάνοντας κλικ στο κουμπί zoom:enabled.

Όταν ανοίγουμε την εφαρμογή εμφανίζεται ένας καμβάς που δεν περιέχει πίνακες, για να εισάγουμε πίνακες στον καμβά κάνουμε διπλό κλικ και επιλέγουμε τους πίνακες από την τρέχουσα βάση δεδομένων. Εφόσον έχουμε επιλέξει τους πίνακες που θέλουμε να εκτελέσουμε το ερώτημα και οι πίνακες αυτοί σχετίζονται μεταξύ τους και εμφανίζεται μια γραμμή που αντιπροσωπεύει την σχέση που έχουν οι δυο πίνακες. Επίσης η γραμμή αυτή συμβολίζει ότι θα υπάρχει σύζευξη μεταξύ πινάκων κατά την εκτέλεση του ερωτήματος. Για να δούμε ποιο στοιχείο του πίνακα περιέχει το ξένο κλειδί κάνουμε κλικ στο μικρό πράσινο τετράγωνο που βρίσκεται δίπλα στον πίνακα κάτω από την γραμμή που αντιπροσωπεύει την σχέση μεταξύ πινάκων εφόσον αυτοί συσχετίζονται. Η αντιστοίχιση αυτή εκτελείται στο ερώτημα ως σύζευξη. Για να εκτελεστούν οι επιλεγμένοι πίνακες με καρτεσιανό γινόμενο πρέπει ο χρήστης να αλλάξει την ρύθμιση στην εφαρμογή που απενεργοποιεί την δημιουργία συζεύξεων μεταξύ πινάκων που συσχετίζονται.

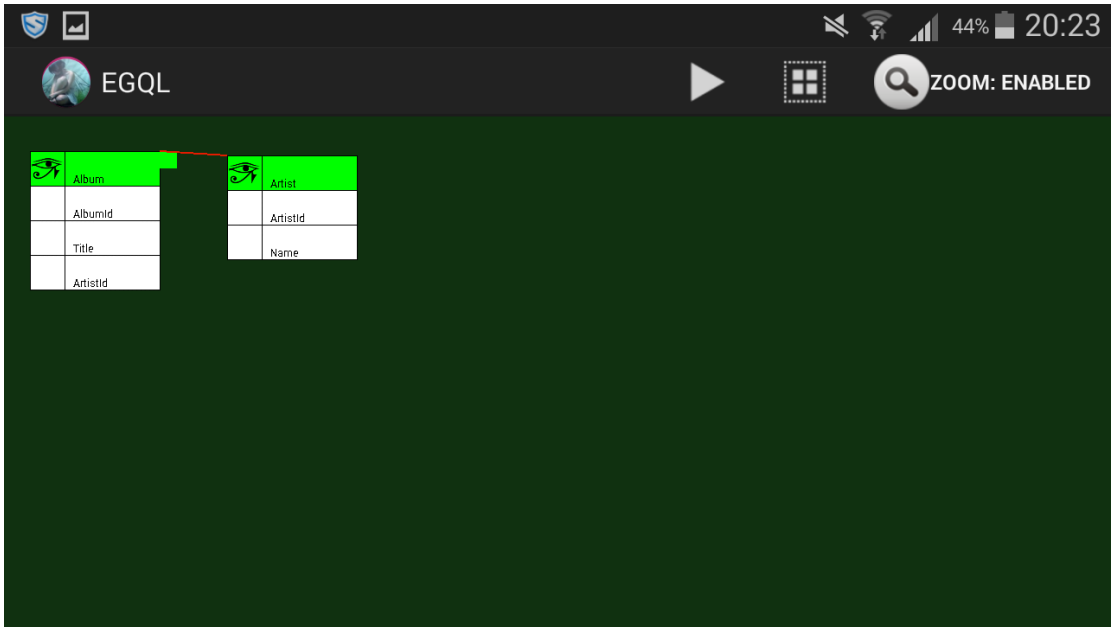
Πριν εκτελεστούν τα ερωτήματα σε μορφή SQL, η EGQL ελέγχει εάν υπάρχουν σφάλματα λογικής και σύνταξης στο ερώτημα. Για παράδειγμα το ερώτημα της εικόνας 20 δεν μπορεί να εκτελεστεί επειδή δεν έχουμε βάλει τι θέλουμε να εμφανίσουμε στον τελικό αποτέλεσμα, δηλαδή στο πεδίο ΕΠΙΛΟΓΗ του ερωτήματος είναι άδειο. Εάν εκτελέσουμε το ερώτημα αυτό θα μας εμφανιστεί ένα

μήνυμα που θα μας λέει να συμπληρώσουμε τα πεδία εμφανίσεων των πινάκων. το ερώτημα αυτό σε SQL κώδικα μεταφράζεται σε:

SELECT

FROM Album, Artist

WHERE (Album.ArtistId=Artist.ArtistId)



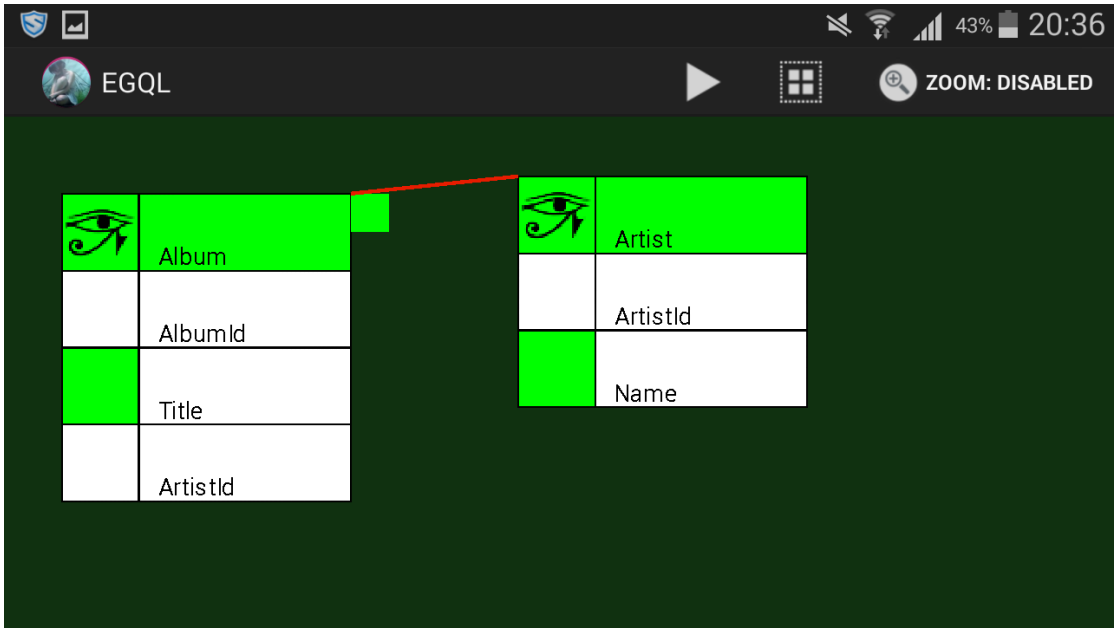
Εικόνα 20: Μη εκτελέσιμο ερώτημα EGQL

Για να επιλέξουμε τα πεδία που θέλουμε να εμφανιστούνε στο τελικό αποτέλεσμα αρκεί να κάνουμε ένα κλικ στο κελί που βρίσκεται στην πρώτη στήλη με το μάτι και είναι στην ίδια γραμμή με αυτό που θέλουμε να εμφανίσουμε στο τελικό αποτέλεσμα. Το πράσινο χρώμα στην στήλη με το μάτι δείχνει ότι έχει γίνει επιλογή της συγκεκριμένης γραμμής να εμφανιστεί στο πεδίο ΕΠΙΛΟΓΗ του ερωτήματος που θα εκτελεστεί, ενώ το φόντο του κελιού με το μάτι και του κελιού που περιέχει το όνομα του πίνακα είναι πράσινα εξ ορισμού. Η εκτέλεση του ερωτήματος που εμφανίζεται στην εικόνα 21 θα μεταφραζόταν στον παρακάτω SQL κώδικα:

SELECT Album.Title, Artist.Name

FROM Album, Artist

WHERE (Album.ArtistId=Artist.ArtistID)



Εικόνα 21: Εκτελέσιμο ερώτημα EGQL

Τα αποτελέσματα του ερωτήματος αυτού θα εμφανιστούν με την μορφή που βλέπουμε στην εικόνα 22, που είναι ένα πλέγμα δεδομένων. Μπορούμε να κάνουμε αλλαγές πριν εμφανιστούν τα αποτελέσματα στην πηγή του κώδικα SQL που δημιουργείται μέσω του ερωτήματος EGQL. Επίσης, μπορούμε να ταξινομήσουμε τα αποτελέσματα κάνοντας κλικ στα αρχικά πεδία που περιέχουν τα ονόματα των στοιχειών που έχουν επιλεχθεί για εμφάνιση. Με Scroll Down και Scroll Up μπορούμε να βλέπουμε όλα τα αποτελέσματα του ερωτήματος που εκτελέστηκε. Για να αλλάξουμε τις διαστάσεις των στηλών των αποτελεσμάτων για την καλύτερη εμφάνιση των αποτελεσμάτων πρέπει να κάνουμε κλικ στο τρίγωνο κουμπί που είναι ανάμεσα σε δυο στήλες και να την τραβήξουμε με το δάχτυλο όσο χρειάζεται. Στην εικόνα 22 έχει γίνει κλικ στο Name έτσι ώστε να γίνει ταξινόμηση με βάση το όνομα του καλλιτέχνη και όχι τον τίτλο του τραγουδιού που εμφανιζόταν εξ ορισμού έτσι ο πίνακας πλέγματος δεδομένων.

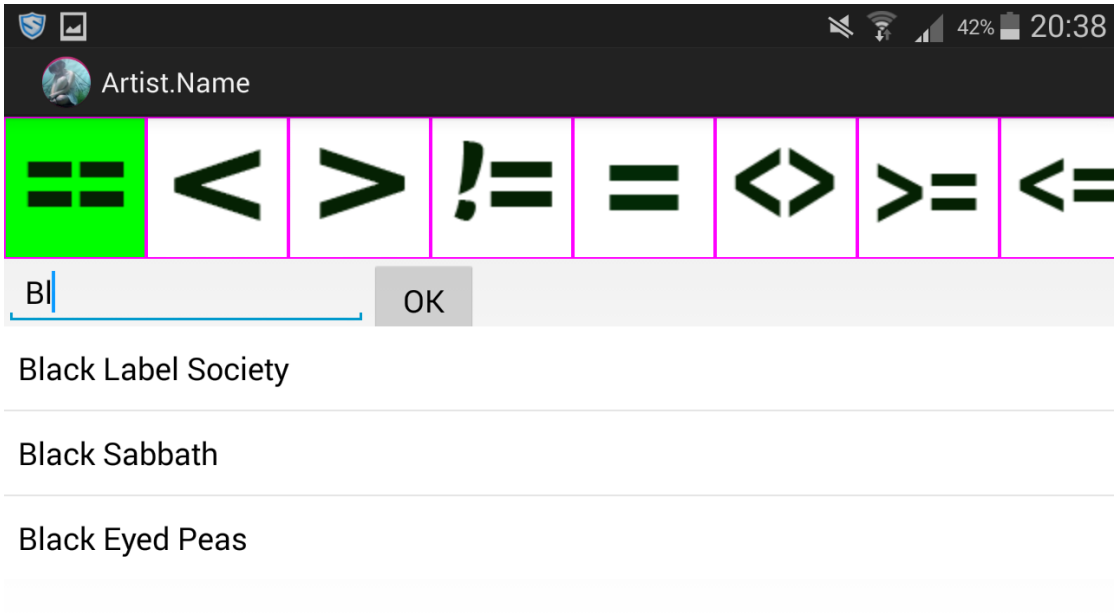
Εικόνα 22: Αποτελέσματα ερωτήματος EGQL

Εφόσον έχουμε επιλέξει τους πίνακες μπορούμε να δημιουργήσουμε πιο πολύπλοκα ερωτήματα. Για να εισάγουμε μια συνθήκη στο πεδίο WHERE του ερωτήματος κάνουμε κλικ στο όνομα του πεδίου που θέλουμε να εισάγουμε την συνθήκη. Αφού γίνει το κλικ εμφανίζονται όλα τα στοιχεία που περιέχει το πεδίο του πίνακα που έχει επιλεχθεί. Μπορεί ο χρήστης να επιλέξει το στοιχείο και τον συντελεστή που θέλει να προσθέσει στην συνθήκη του WHERE. Μπορεί επίσης ο χρήστης να πληκτρολογήσει αυτό που θέλει να ψάξει μέσα στον πίνακα. Κατά την πληκτρολόγηση μπορούμε να δούμε εάν υπάρχει αυτό που θέλουμε να ψάξουμε μέσα στον πίνακα (Εικόνα 23). Επιλέγοντας το στοιχείο και τον συντελεστή που επιλέγουμε από το οριζόντιο scrollView, πατάμε το OK και το στοιχείο εισάγεται στο ερώτημα. Αυτή η διαδικασία εμφανίζεται με κίτρινο χρώμα στο ερώτημα του EGQL. Το χρώμα αυτό εμφανίζεται στο πεδίο που γράφει το όνομα του στοιχείου του πίνακα που περιέχει την συνθήκη που επιλέξαμε, όπως βλέπουμε και στην εικόνα 24. Για να δούμε τι περιέχει ένα στοιχείο κάνουμε LONG CLICK πάνω στο στοιχείο και βλέπουμε τις συνθήκες που περιέχονται στο στοιχείο, σε αυτό το μενού μπορούμε να επιλέξουμε τις συνθήκες που θέλουμε να αφαιρέσουμε από το ερώτημα(Εικόνα 25). Σύμφωνα με τον EGQL ερώτημα που βλέπουμε στην εικόνα 24, θα δημιουργηθεί ο παρακάτω SQL κώδικας, ο οποίος θα μας εμφανίσει όλα τα άλμπουμ του Black Sabbath που περιέχει η βάση δεδομένων:

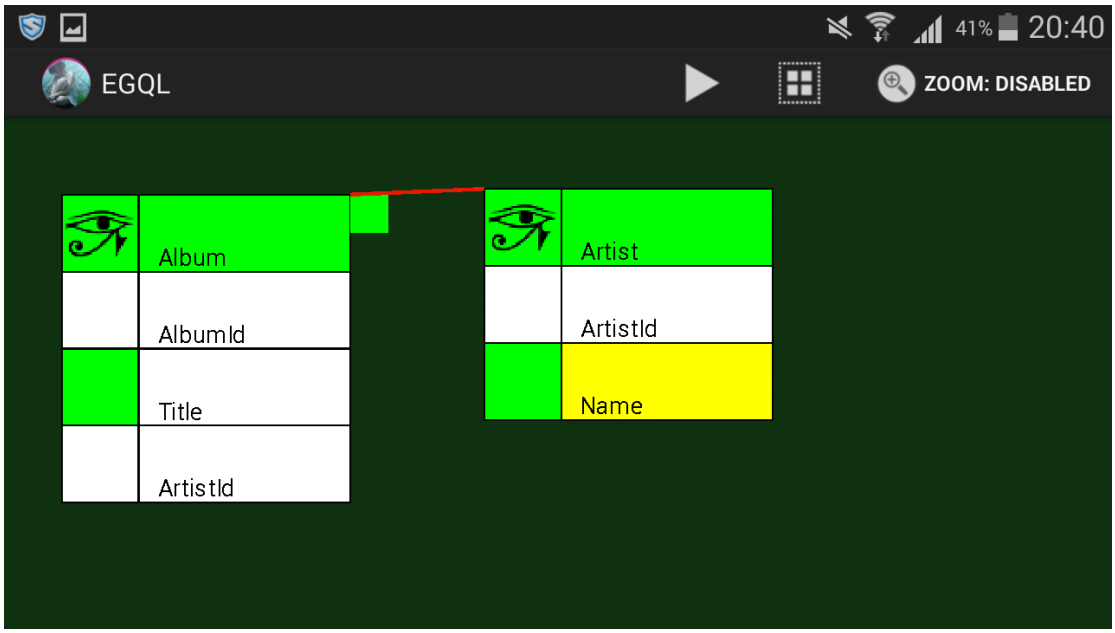
SELECT Album.Title, Artist.Name

FROM Album, Artist

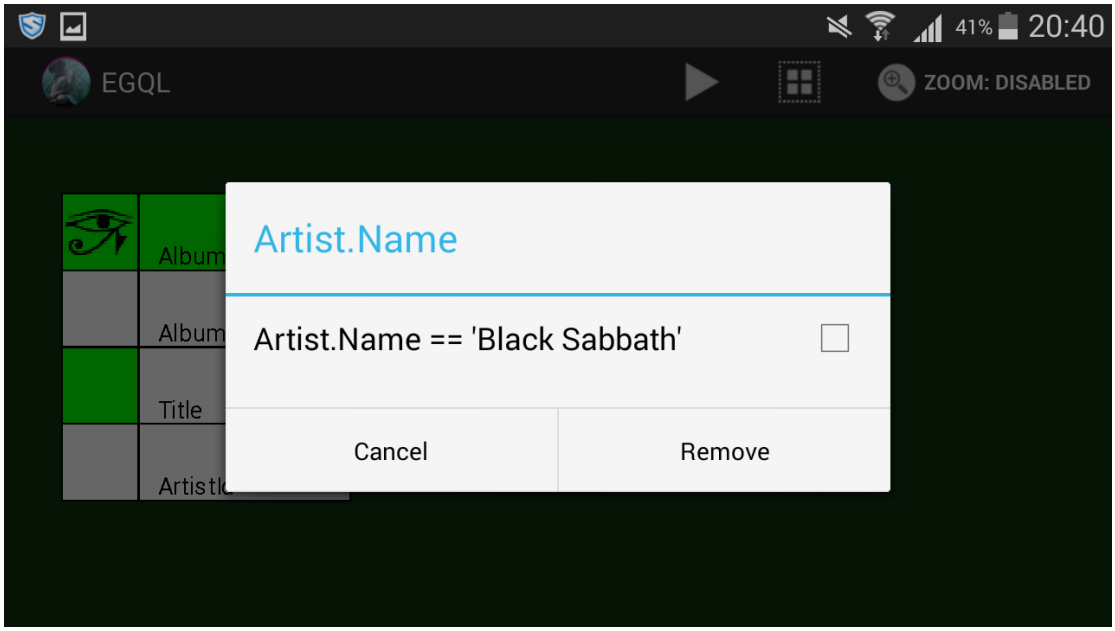
WHERE (Artist.Name=='Black Sabbath') AND
(Album.ArtistId=Artist.ArtistId)



Εικόνα 23: Προσθήκη της συνθήκης WHERE στο ερώτημα

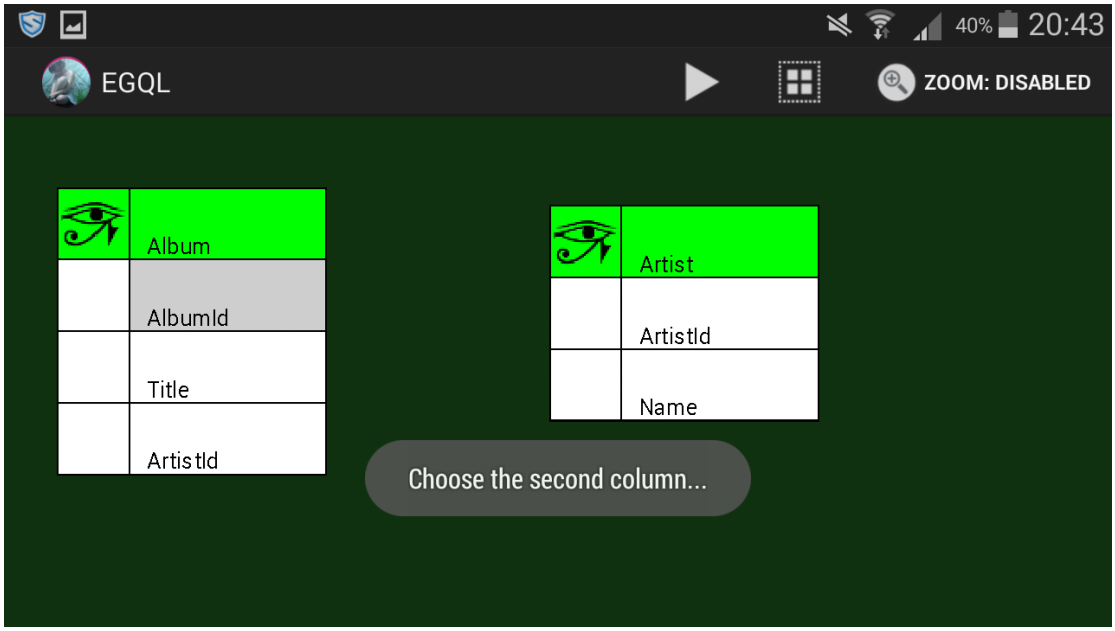


Εικόνα 24: Το ερώτημα με την συνθήκη WHERE

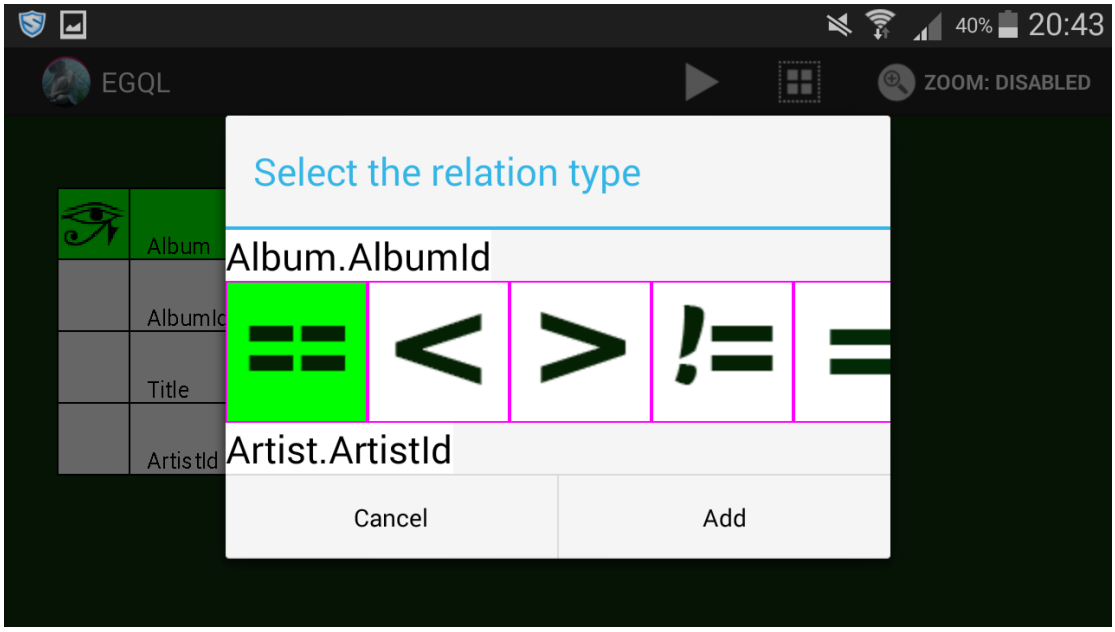


Εικόνα 25: Αφαίρεση συνθήκης με LONG CLICK

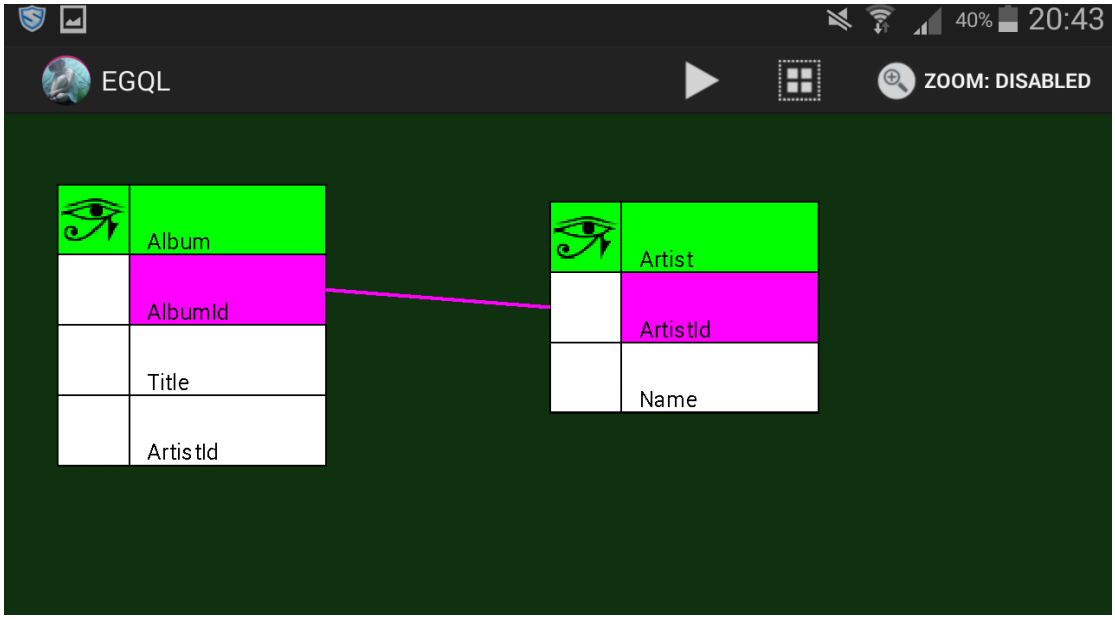
Μπορούμε να απενεργοποιήσουμε από τις ρυθμίσεις την αυτόματη δημιουργία των συζεύξεων των πινάκων του ερωτήματος, που δημιουργεί αυτόματες συζεύξεις στην περίπτωση που σχετίζονται μεταξύ τους οι πίνακες. Σε αυτήν την περίπτωση, ο χρήστης μπορεί να δημιουργήσει τις σχέσεις που επιθυμεί ο ίδιος να έχουν οι πίνακες μεταξύ τους, χωρίς να είναι αναγκαία οι πίνακες να είναι συσχετιζόμενες. Για να γίνει αυτό κάνουμε διπλό κλικ σε ένα στοιχείο ενός πίνακα, εμφανίζεται το μήνυμα για να γίνει επιλογή του δεύτερου στοιχείου και το πρώτο επιλεγμένο στοιχείο παίρνει το γκρι χρώμα μέχρι να επιλεχθεί το δεύτερο στοιχείο. Η κατάσταση αυτή απεικονίζεται στην εικόνα 26. Για την επιλογή του δεύτερου στοιχείου κάνουμε διπλό κλικ στο στοιχείο που θέλουμε να επιλέξουμε, και εμφανίζεται το μενού για να επιλέξουμε την σχέση που θα έχουν τα δυο στοιχεία μεταξύ τους. Το μενού αυτό μας εμφανίζει τα δυο στοιχεία που έχουν επιλεχθεί και ανάμεσα τους βρίσκεται ένα οριζόντιο ScrollView που μπορούμε να επιλέξουμε τον συντελεστή της σχέσης(Εικόνα 27). Επιλέγουμε τον συντελεστή σχέσης και πατάμε το OK για να καταχωρηθεί στο EGQL ερώτημα. Αφού γίνει η εισαγωγή της σχέσης των στοιχείων, τα στοιχεία που επιλέχθηκαν θα έχουν ως φόντο το χρώμα ματζέντα και θα υπάρχει μια γραμμή που θα βρίσκεται στην ίδια ευθεία με τα επιλεγμένα στοιχεία. Η κατάσταση αυτή απεικονίζεται στην εικόνα 28.



Εικόνα 26: Επιλογή του πρώτου στοιχείου στην δημιουργία συζεύξεων



Εικόνα 27: Επιλογή συντελεστή σχέσεων



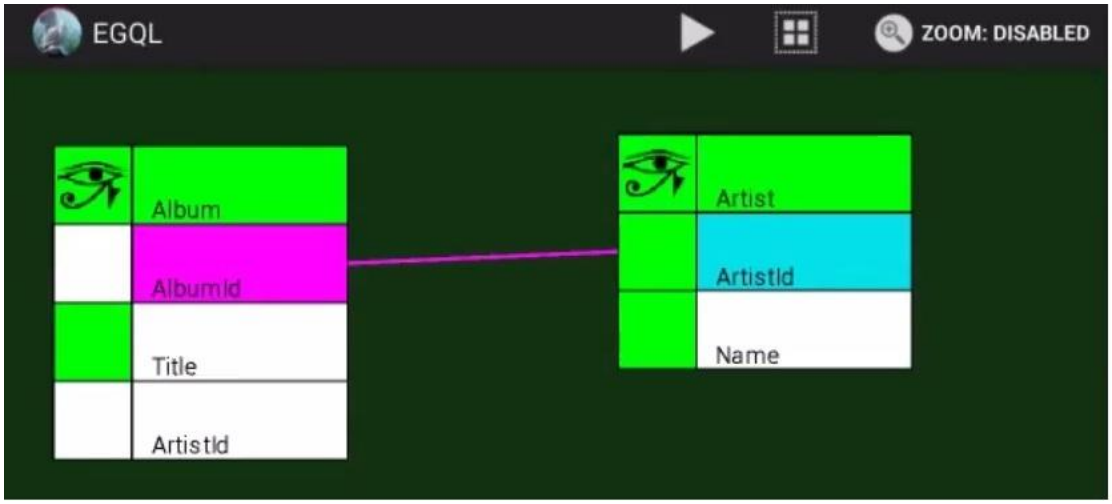
Εικόνα 28: Ερώτημα EGQL με μη αυτόματη σύζευξη

Εφόσον έχουμε επιλέξει την σχέση που θα έχουν οι πίνακες μπορούμε να προσθέσουμε και άλλες συνθήκες. Μπορεί για παράδειγμα, ο χρήστης να θέλει να αναζητήσει ένα συγκεκριμένο ID του πίνακα που ήδη έχει ένα χρώμα στον πίνακα EGQL ερωτήματος, στην περίπτωση μας την ματζέντα. Υπάρχει ένα πρόβλημα στην περίπτωση αυτή, όταν κάνουμε κλικ στο στοιχείο που θέλουμε να αναζητήσουμε, αλλά το στοιχείο έχει ήδη ένα χρώμα και άμα γινόταν κίτρινο θα χανόταν το χρώμα ματζέντα που δείχνει ότι το στοιχείο συνδέεται με ένα άλλο στοιχείο. Για να λύσουμε το πρόβλημα αυτό, το στοιχείο παίρνει ένα χρώμα που είναι κοντά στον κυανό το οποίο είναι ο μέσος όρος στο δεκαεξαδικό σύστημα του κίτρινου και της ματζέντας που χρησιμοποιούνται στην συγκεκριμένη περίπτωση. Η κατάσταση αυτή απεικονίζεται στην εικόνα 29. Εάν κάνουμε LONG CLICK στο στοιχείο που περιέχει αυτό το χρώμα μπορούμε να βλέπουμε τις συνθήκες που περιέχει και να τις αφαιρούμε αν θέλουμε (Εικόνα 30). Όταν αφαιρούμε μια συνθήκη αλλάζει και το χρώμα του στοιχείου ανάλογα με το τι έχει αφαιρεθεί. Το ερώτημα EGQL που εμφανίζεται στην εικόνα 29 θα δημιουργήσει τον παρακάτω SQL κώδικα:

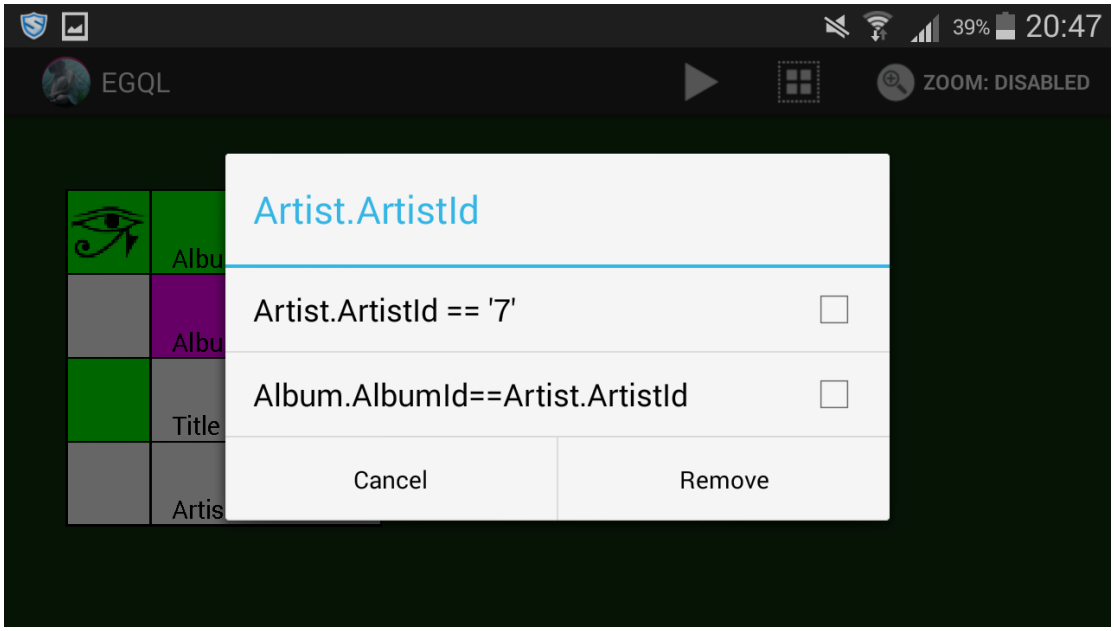
SELECT Album.AlbumId, Artist.ArtistId, Artist.Name

FROM Album, Artist

WHERE (Album.AlbumId=Artist.ArtistId) AND (Artist.ArtistId=='7')



Εικόνα 29: Ερώτημα EGQL με συνθήκη και σύζευξη

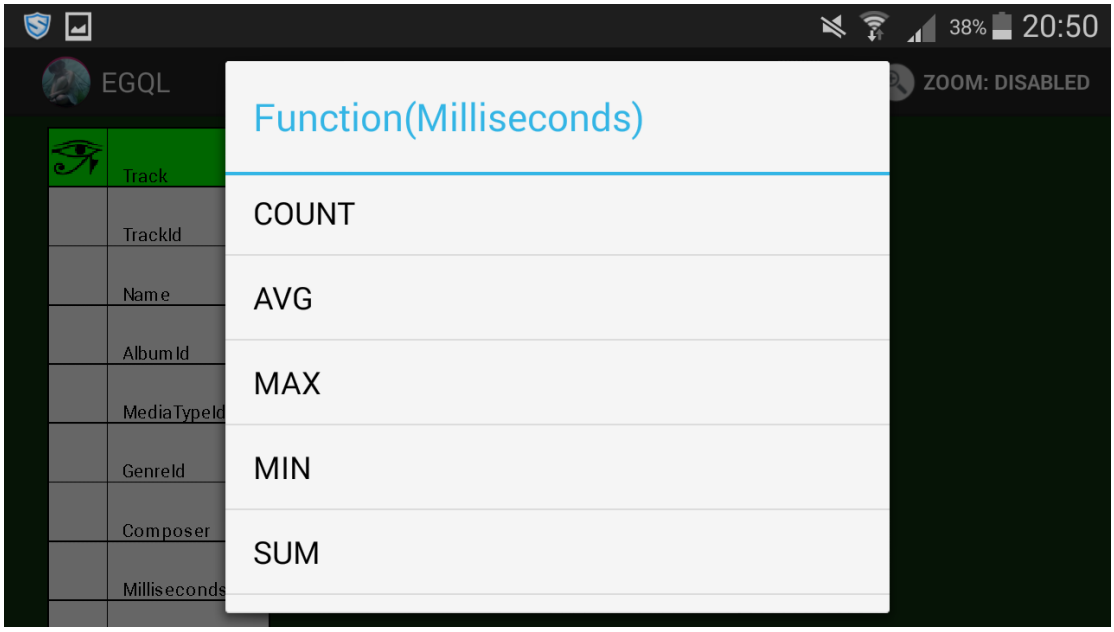


Εικόνα 30: Αφαίρεση συνθηκών και συζεύξεων

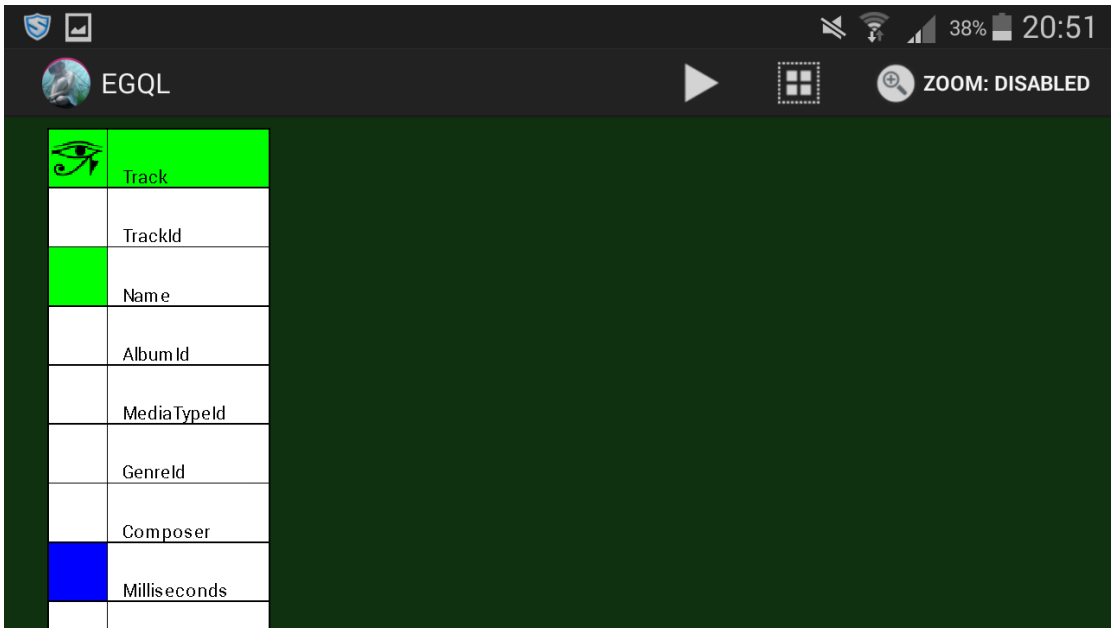
Μπορούμε να προσθέσουμε και συναρτήσεις στο ερώτημα. Για να γίνει αυτό πρέπει να κάνουμε διπλό κλικ στην στήλη που βρίσκεται το μάτι και στην γραμμή του πίνακα EGQL που βρίσκεται το στοιχείο του πίνακα που θέλουμε να προσθέσουμε μια συνάρτηση. Στο μενού που εμφανίζεται μπορούμε να επιλέξουμε μία συνάρτηση από τις βασικές συναρτήσεις που περιέχονται στο SqlLite3 με ένα απλό κλικ πάνω τους(Εικόνα 31). Μετά την επιλογή της συναρτήσεως, η πρώτη στήλη του στοιχείου παίρνει το μπλε χρώμα. Η εκτέλεση του ερωτήματος που εμφανίζεται στην εικόνα 32 θα μας δώσει τον παρακάτω SQL κώδικα, και θα μας εμφανίσει το όνομα του τραγουδιού με την μεγαλύτερη διάρκεια στον πίνακα Track και την διάρκεια που έχει το συγκεκριμένο τραγούδι:

SELECT Track.Name, MAX(Track.Milliseconds)

FROM Track

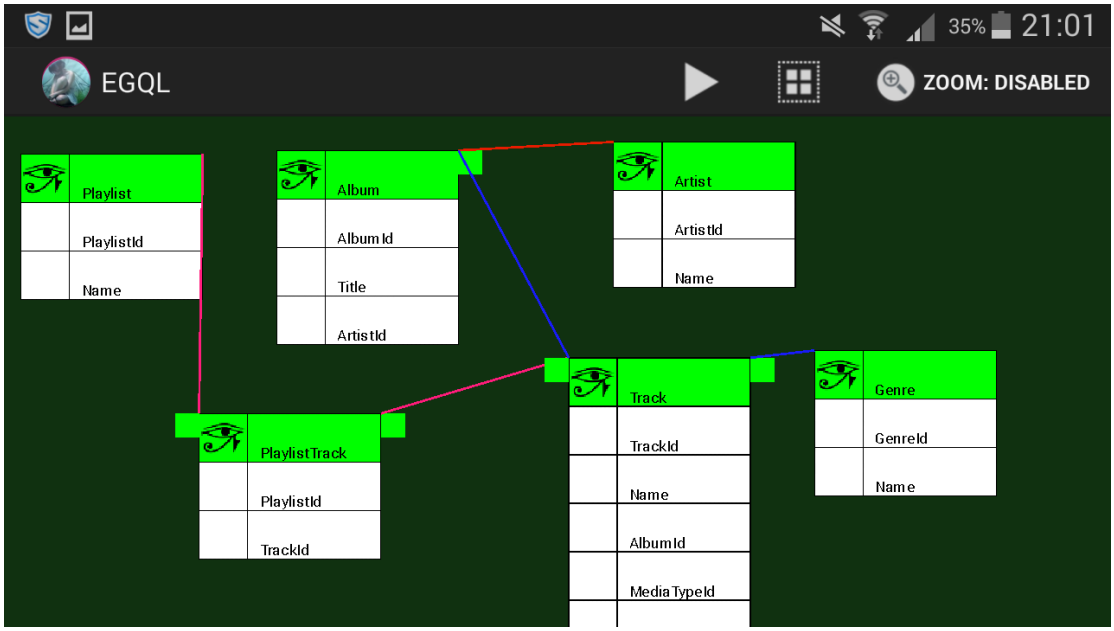


Εικόνα 31: Επιλογή συναρτήσεων



Εικόνα 32: Το ερώτημα EGQL που περιέχει συναρτήσεις

Με την γλώσσα EGQL μπορούμε να βλέπουμε την σχέση που έχουν οι πίνακες μεταξύ τους απλά βάζοντας τους πίνακες που συσχετίζονται μέσα στον καμβά. Ένα τέτοιο παράδειγμα βλέπουμε στην εικόνα 33. Για να δούμε ποιο στοιχείο είναι το ξένο κλειδί του κάθε πίνακα κάνουμε ένα απλό κλικ στο πράσινο τετράγωνο που βρίσκεται έξω από το σχήμα του πίνακα EGQL και δίπλα στην γραμμή που δείχνει με ποιους πίνακες συνδέεται ο συγκεκριμένος πίνακας. Αυτή η ιδιότητα που δείχνει τις σχέσεις και δεν χρειάζεται να τις καθιερώσει χρήστης μπορεί να απενεργοποιηθεί και έτσι η εφαρμογή δεν θα δείχνει τις γραμμές αυτές, επίσης δεν θα εκτελεστούν οι συζεύξεις στα ερωτήματα.



Εικόνα 33: Αυτόματος συσχετισμός πινάκων EGQL

Όταν ο χρήστης κάνει απλό κλικ στα κελιά που περιέχουν τα μάτια επιλέγονται όλα τα στοιχεία του συγκεκριμένου πίνακα, για παράδειγμα όταν ο χρήστης κάνει απλό κλικ στον πίνακα Artist, επιλέγονται όλα τα στοιχεία του πίνακα Artist για εμφάνιση στο Select, δηλαδή δημιουργείται το ερώτημα:

*SELECT Artist.**

FROM Artist

Ενώ, όταν κάνουμε διπλό κλικ στο κελί με το μάτι μπορούμε να εισάγουμε και συναρτήσεις για όλο τον πίνακα. Για παράδειγμα όταν κάνουμε διπλό κλικ στο κελί με το μάτι του πίνακα Artist και επιλέγουμε την συνάρτηση Count δημιουργείται ο παρακάτω SQL κώδικας:

SELECT COUNT(Artist.)*

FROM Artist

Μετά την επιλογή της συνάρτησης αυτής το κελί με το μάτι παίρνει το σκούρο πράσινο χρώμα που σημαίνει ότι περιέχει μια συνάρτηση. Όπως σε όλα τα κελιά μπορούμε να αφαιρέσουμε τις συναρτήσεις από το κελί με το μάτι με το LONG CLICK και επιλογή του στοιχείου που θέλουμε να αφαιρέσουμε, μετά την διαδικασία της αφαίρεσης το κελί ξαναγυρνάει στον βασικό χρώμα του που είναι το πράσινο. Για να αφαιρέσουμε έναν πίνακα από ένα ερώτημα EGQL κάνουμε long click στο κελί που περιέχει το όνομα του πίνακα. Επίσης για να αλλάξουμε την θέση ενός πίνακα πρέπει να κάνουμε κλικ στο κελί που περιέχει το όνομα του πίνακα και να το σύρουμε εκεί που θέλουμε.

Στην EGQL υπάρχουν τρία κουμπιά που βρίσκονται πάνω δεξιά της οθόνης και είναι αρκετά χρήσιμα. Το πρώτο κουμπί από αριστερά που βλέπουμε και στην εικόνα 34 και έχει το σχήμα του κλασικού play κουμπιού, εκτελεί το ερώτημα που έχει δημιουργηθεί. Όταν ο χρήστης πατάει στο δεύτερο κουμπί με τον αριθμό 2 στην ίδια εικόνα, επιλέγονται όλα τα στοιχεία των πινάκων που βρίσκονται μέσα στον καμβά, είναι το ίδιο με το αστεράκι σε SQL κώδικα. Το τρίτο κουμπί ενεργοποιεί ή απενεργοποιεί το ζουμ. Όταν το ζουμ είναι ενεργοποιημένο δεν μπορούμε να κάνουμε αλλαγές στο ερώτημα, για να κάνουμε αλλαγές στο ερώτημα ή στις θέσεις που βρίσκονται οι πίνακες πρέπει να απενεργοποιήσουμε το ζουμ.

Εικόνα 34: Το ActionBar μενού

Μπορεί επίσης ο χρήστης να χρησιμοποιήσει το μενού που εμφανίζεται κάνοντας κλικ στο κουμπί μενού που η θέση της επιλογής αυτής διαφέρει από συσκευή σε συσκευή. Στην EGQL, στο μενού αυτό έχουμε τις ρυθμίσεις που μπορεί ο χρήστης να επιλέξει εάν θέλει να έχει την σύζευξη και σύνδεση των πινάκων μέσω της γραμμής ή όχι. Επίσης έχουμε τις επιλογή πινάκων, τον καθαρισμό του καμβά και την εκτέλεση του ερωτήματος. Η επιλογή πινάκων και εκτέλεση ερωτήματος μπορούν να γίνουν και με άλλους τρόπους όπως έχουμε εξηγήσει παραπάνω και δεν έχουνε καμία διαφορά.

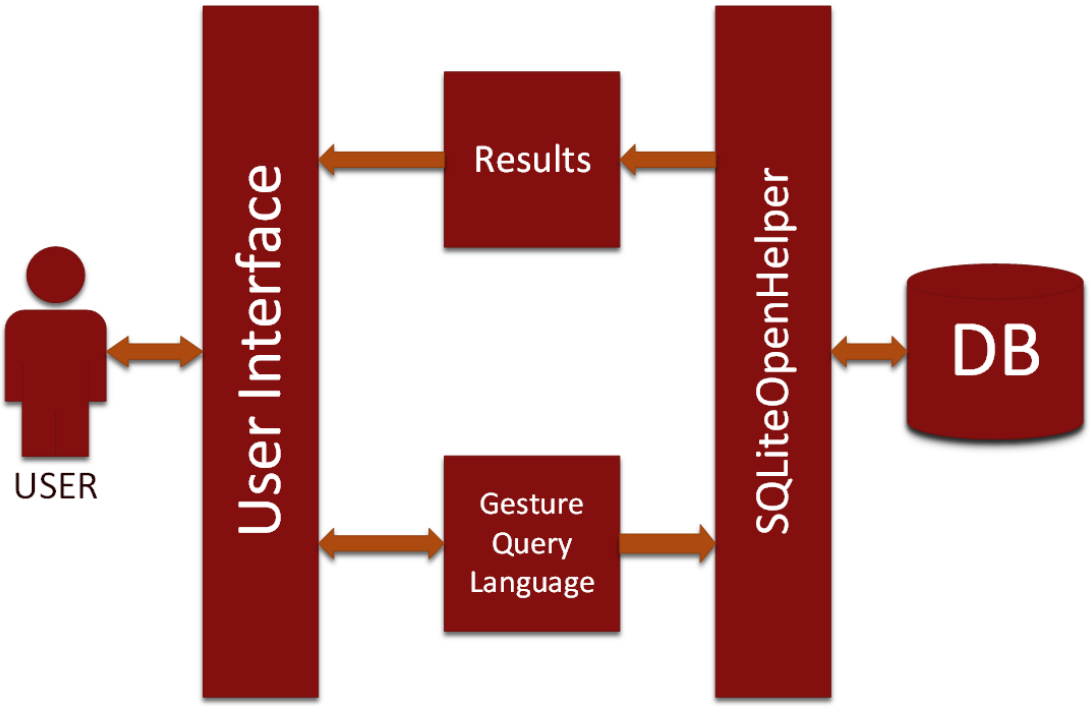
Ο παρακάτω πίνακας δείχνει την σχέση που έχουν τα χρώματα και οι χειρονομίες που έχουν πραγματοποιηθεί στο ερώτημα:

Πίνακας 2

QUERY SITUATION	COLOR NAME	HEX COLOR	COLOR
SELECT VIEW, DEFAULT COLOR OF FIRST ROW	GREEN	#00FF00	
CONDITION IN “WHERE” CLAUSE	YELLOW	#FFFF00	
COUPLING IN “WHERE” CLAUSE (JOIN)	MAGENTA	#FF00FF	
CONDITION & COUPLING IN “WHERE” CLAUSE	CYAN	#00DDE1	
FUNCTION	BLUE	#0000FF	
FUNCTION IN THE EYE CELL	DARK GREEN	#088800	
COUPLING SELECTION	LTGREY	#CCCCCC	
DEFALUT COLOR OF THE CELL	WHITE	#FFFFFF	

6.2 Η Αρχιτεκτονική του EGQL

Στην εικόνα 35 βλέπουμε την αρχιτεκτονική της ροής εκτέλεσης ενός ερωτήματος στην EGQL. Ο χρήστης αλληλεπιδρά με την διεπαφή χρήστη της γλώσσας που είναι ένα SurfaceView. Η κάθε χειρονομία που γίνεται στην οθόνη της συσκευής επιδρά στον καμβά και στα χρώματα που έχει ο κάθε πίνακας για να κατανοηθεί η επίδραση της χειρονομίας στο ερώτημα που θα εκτελεστεί και μεταφράζεται στο επίπεδο Gesture Query Language σε ένα κομμάτι κώδικα του ερωτήματος SQL. Το ερώτημα που δημιουργείται μετά από τις χειρονομίες του χρήστη εκτελείται με την βοήθεια της κλάσης SQLiteOpenHelper του λειτουργικού συστήματος Android για να αποκτήσουμε τα επιθυμητά αποτελέσματα από την επιλεγμένη βάση δεδομένων SQLite. Τα αποτελέσματα της εκτέλεσης θα εμφανίζονται σε μορφή πλέγματος δεδομένων, έτσι ώστε να μπορεί ο χρήστης να τα ταξινομήσει εύκολα ή να μετακινηθεί στον πίνακα αποτελεσμάτων με Scroll Down και Scroll Up. Οι διαστάσεις των στηλών και των γραμμών των πινάκων αυτών μπορούν να αλλάξουν ανάλογα με τις ανάγκες του χρήστη.



Εικόνα 35: Η αρχιτεκτονική της EGQL

Η διεπαφή η οποία αλληλεπιδρά ο χρήστης περιέχει ένα SurfaceView και ένα μενού που βρίσκεται πάνω από αυτό. Οι πίνακες και τα άλλα σχήματα της γλώσσας σχεδιάζονται σε έναν καμβά σε μορφή τετραγώνων και άλλων σχημάτων. Ο καμβάς αυτός βρίσκεται μέσα στο SurfaceView της οθόνης μας.

Το πρώτο στάδιο αναγνώρισης της χειρονομίας σε μια διεπαφή χρήστη SurfaceView είναι να αναγνωρίσουμε τις συντεταγμένες χ και ψ που δηλώνουν το που έχει κάνει κλικ ο χρήστης μέσα στον καμβά που περιέχει το SurfaceView. Αυτή η διαδικασία γίνεται με την βοήθεια της μεθόδου onTouch() της κλάσης SurfaceView του Android, που καλείται από το σύστημα όταν πραγματοποιείται ένα touch event σε ένα View. Μετά την αναγνώριση των παραμέτρων χ και ψ ελέγχουμε εάν είναι ενεργοποιημένη η λειτουργία ζουμ ή όχι . Εάν είναι απενεργοποιημένη η λειτουργία ζουμ γίνεται αναγνώριση του είδους της χειρονομίας που γίνεται μέσα στον καμβά της SurfaceView. Έπειτα ελέγχουμε σε ποιο στοιχείο των επιλεγμένων πινάκων EGQL έγινε η συγκεκριμένη χειρονομία και κάνουμε τις αντίστοιχες αλλαγές για να αλλάξει το χρώμα του κελιού και να καταλάβει ο χρήστης ότι έχει καταχωρηθεί η χειρονομία του στο ερώτημα EGQL. Παράλληλα μεταφράζεται η χειρονομία σε κώδικα SQL και εισάγεται στον τελικό κώδικα που θα εκτελεστεί.

Αφού δημιουργούμε τον κώδικα μας σε μορφή EGQL εκτελούμε για να δούμε τα αποτελέσματα σε μορφή πινάκων πλέγματος δεδομένων. Παράλληλα με τις χειρονομίες του χρήστη δημιουργείται ένας κώδικας SQL που αντιστοιχεί στις χειρονομίες που πράξαμε σύμφωνα με την Gesture Query Language EGQL, αυτός είναι ο κώδικας που θα εκτελεστεί στην επιλεγμένη βάση δεδομένων SQLite με την βοήθεια της κλάσης SQLiteOpenHelper του Android, που μας βοηθά στην

διαχείριση των βάσεων δεδομένων στο λειτουργικό σύστημα Android. Στην περίπτωση που έχουμε ενεργοποιημένη την λειτουργία ζουμ, δεν μπορούμε να κάνουμε οποιαδήποτε αλλαγή στο ερώτημα επειδή τα κλικ που κάνουμε μεταφράζονται σε κινήσεις ολίσθησης του καμβά στην οθόνη και τα κλικ με δυο δάχτυλα σε κινήσεις zoom in και zoom out του καμβά.

Εκτός από το SurfaceView που εμφανίζονται οι πίνακες μέσα του με την βοήθεια του καμβά, η διεπαφή χρήστη της EGQL περιέχει δυο διαφορετικά μενού για να διευκολύνει τις αλληλεπιδράσεις των χρηστών. Το ένα μενού βρίσκεται στο Action Bar της εφαρμογής. Το Action Bar μίας εφαρμογής είναι ένα παράθυρο που βρίσκεται πάνω στην εφαρμογή και προσδιορίζει την θέση του χρήστη, ενέργειες του χρήστη και τους τρόπους πλοήγησης. Μία άλλη ιδιότητα του Action Bar είναι ότι μπορεί να περιέχει και επιλογές για μενού στην εφαρμογή όπως στην περίπτωσή μας που εκτός από το όνομα της γλώσσας και την θέση του χρήστη μας δίνει την δυνατότητα μερικών επιλογών που μπορούμε να κάνουμε μέσω του μενού αυτού με εύκολο τρόπο. Το μενού αυτό παρουσιάζεται στην εικόνα 36. Το πρώτο στοιχείο του μενού είναι μία συντόμευση για την εκτέλεση του ερωτήματος που έχει δημιουργηθεί. Όταν ο χρήστης κάνει κλικ στο δεύτερο στοιχείο του μενού επιλέγονται όλα τα πεδία όλων των πινάκων για εμφάνιση και προσθέτονται στην συνιστώσα SELECT του τελικού ερωτήματος. Ο αντίστοιχος κώδικας σε SQL είναι

*SELECT **

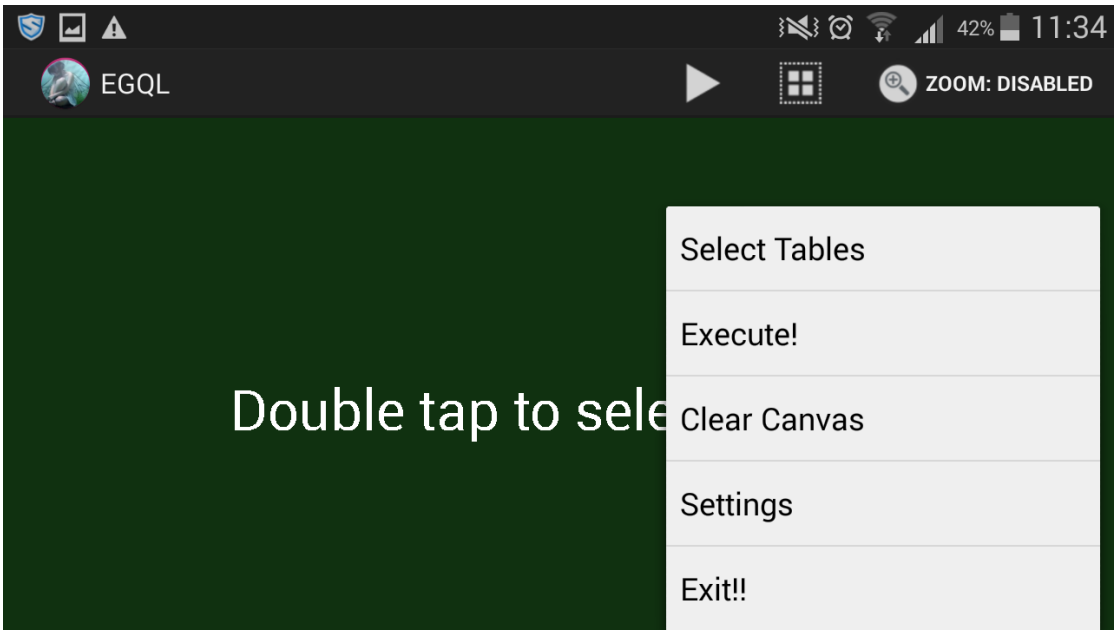
FROM SelectedTable1

Ενώ στην EGQL όλα τα κελιά της πρώτης στήλης όλων των επιλεγμένων πινάκων θα γίνουν πράσινα που σημαίνει ότι θα εμφανιστούν στο τελικό αποτέλεσμα όλα τα πεδία των επιλεγμένων πινάκων. Το τρίτο στοιχείο είναι το κουμπί ζουμ. Κάνοντας κλικ σε αυτό ο χρήστης μπορεί να ενεργοποιήσει ή να απενεργοποιήσει την λειτουργία ζουμ. Μετά από κάθε κλικ στην επιλογή αυτή αλλάζει η εικόνα του κουμπιού ανάλογα με το εάν είναι ενεργοποιημένη ή απενεργοποιημένη η λειτουργία, επίσης αλλάζει και το κείμενο που δηλώνει την κατάσταση της λειτουργίας ζουμ.

Εικόνα 36: ActionBar της EGQL

Το δεύτερο μενού που μπορεί ο χρήστης να έχει διάφορες επιλογές είναι το μενού επιλογών του Android. Για να εμφανιστεί το μενού αυτό στον χρήστη πρέπει ο χρήστης να κάνει κλικ στο κουμπί μενού που βρίσκεται σε κάθε συσκευή και όχι απαραίτητα στο ίδιο σημείο μίας συσκευής. Με το πρώτο στοιχείο του μενού επιλογών της εφαρμογής μας μπορούμε να επιλέξουμε πίνακες για το ερώτημα

μας και είναι το στοιχείο “Select Tables”. Το δεύτερο στοιχείο του μενού είναι το “Execute!”, και με ένα κλικ στο στοιχείο αυτό μπορούμε να εκτελέσουμε το ερώτημα που έχουμε διαμορφώσει. Το στοιχείο αυτό δεν έχει καμία διαφορά από την συντόμευση εκτέλεσης ερωτημάτων που έχουμε στο Action Bar. Το τρίτο στοιχείο αυτού του μενού είναι το “Clear Canvas” και είναι το στοιχείο που αφαιρεί όλους τους πίνακες από τον καμβά και καθαρίζει επίσης όλες τις κινήσεις που έχουν καταχωρηθεί. Στο τέταρτο στοιχείο του μενού βρίσκονται τα “Settings” και μας δίνουν την δυνατότητα να κάνουμε αλλαγές στην εφαρμογή σύμφωνα με τις προτιμήσεις μας. Προς το παρόν υπάρχουν δύο επιλογές στις ρυθμίσεις, το πρώτο είναι εάν θέλουμε να δουλεύει ο αυτόματος προσδιοριστής σχέσεων και δημιουργίας συζεύξεων των συσχετισμένων πινάκων, ενώ ο δεύτερος μας ρωτάει εάν θέλουμε να μας εμφανίσει και να μας δίνει την δυνατότητα αλλαγής του τελικού κώδικα SQL που έχει δημιουργηθεί μέσω των χειρονομιών μας. Η Πέμπτη και η τελευταία επιλογή του μενού είναι το “Exit” και είναι η επιλογή για να βγούμε από την εφαρμογή. Παρακάτω, στην εικόνα 37, βλέπουμε το μενού επιλογών. Μην ξεχάσουμε ότι η θέση του κουμπιού και η εμφάνιση του μενού μπορεί να αλλάξει από συσκευή σε συσκευή, αλλά όχι τα περιεχόμενα του.



Εικόνα 37: Μενού επιλογών EGQL

6.3 Το Αρχείο Manifest της EGQL

Το αρχείο manifest είναι το πρώτο αρχείο που διαβάζεται από το σύστημα κατά την εκτέλεση μίας εφαρμογής έτσι ώστε να μπορεί να ελέγξει εάν η εφαρμογή μπορεί να τρέξει στη συγκεκριμένη συσκευή χωρίς προβλήματα. Στο υποκεφάλαιο αυτό θα δούμε μερικά κομμάτια του αρχείου Manifest της EGQL και θα τα εξηγήσουμε.

Με το στοιχείο <uses-sdk> μπορούμε να εκφράσουμε την συμβατότητα της εφαρμογής μας με μία οι πολλές εκδόσεις της πλατφόρμας Android, με την χρήση ενός αριθμού που δηλώνει το API Level. Το μικρότερο API που μπορεί να τρέξει η

εφαρμογή μας είναι το 14, ενώ οι δοκιμές μας έχουν γίνει στο API 19 που υποτίθεται ότι θα δουλεύει καλύτερα η εφαρμογή μας. Παρακάτω βλέπουμε τις δηλώσεις των API στο αρχείο manifest της EGQL:

```
<uses-sdk
        android:minSdkVersion="14"
         android:targetSdkVersion="19" />
```

Το στοιχείο <uses-permission> απαιτεί τα δικαιώματα που χρειάζεται η εφαρμογή μας για να μπορεί να λειτουργεί σωστά. Τα δικαιώματα χορηγούνται από τον χρήστη κατά την διάρκεια της εγκατάστασης. Τα δικαιώματα που χρειαζόμαστε για την EGQL είναι δυο. Το WRITE.EXTERNAL.STORAGE που δίνει το δικαίωμα εγγραφής δεδομένων σε εξωτερικούς χώρους αποθήκευσης και το READ.EXTERNAL.STORAGE που δίνει δικαίωμα αναγνώρισης των εξωτερικών χωρών αποθήκευσης. Τα δικαιώματα αυτά δεν δηλώνονται από το API 19 και μετά. Παρακάτω βλέπουμε τις δηλώσεις των δικαιωμάτων που θα ζητηθούν από τον χρήστη.

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

Μέσα στο στοιχείο <application> δηλώνουμε τις δραστηριότητες και διάφορες άλλες επιλογές. Οι δραστηριότητες πρέπει να δηλωθούν και αυτό γίνεται με το στοιχείο <activity>. Το <activity> δηλώνει ένα activity (μία υποκλάση Activity) που υλοποιεί κομμάτι της εικονικής διεπαφής χρήστη της εφαρμογής. Όλα τα activities πρέπει να αντιπροσωπεύονται με τα στοιχεία <activity> μέσα στο αρχείο manifest. Τα activities που δεν δηλώνονται δεν θα είναι ορατά από το σύστημα και δεν θα εκτελεστούν ποτέ. Παρακάτω βλέπουμε την δήλωση της αρχικής δραστηριότητας της EGQL, που δηλώνει τον προσανατολισμό της δραστηριότητας που είναι οριζόντια και διάφορες άλλες ρυθμίσεις:

```
<activity
        android:name="com.me.myapp.MainActivity"
        android:label="@string/app_name"
        android:screenOrientation="landscape"
```

```
        android:configChanges="keyboardHidden|orientation|screenSize" >
         <intent-filter>
                 <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
         </intent-filter>
</activity>
```

Στον παραπάνω κώδικα βλέπουμε επίσης το στοιχείο <intent-filter> μέσα στην δραστηριότητα. Το <intent-filer> δηλώνει τους τύπους των προθέσεων που μπορεί μία activity, υπηρεσία ή broadcast receiver να απαντήσει. Το συγκεκριμένο δηλώνει ότι η δραστηριότητα MainActivity θα είναι η βασική δραστηριότητα της εφαρμογής και θα είναι η πρώτη που θα ανοίξει όταν ο χρήστης ξεκινάει την εφαρμογή. Εκτός από την κύρια δραστηριότητα που περιέχει το προσαρμοσμένο SurfaceView, η EGQL περιέχει και άλλες δραστηριότητες που πρέπει να δηλωθούν στο αρχείο manifest και δηλώνονται με τον παρακάτω τρόπο η καθεμία. Συγκεκριμένα στον παρακάτω κώδικα δηλώνουμε το ResultsActivity που εμφανίζονται τα τελικά αποτελέσματα του ερωτήματος μας.

```
<activity
        android:name="com.me.myapp.ResultsActivity"
        android:label="@string/title_activity_results" >
</activity>
```

6.4 Η Χρήση της SQLite στην EGQL

Για να μπορούμε να χρησιμοποιήσουμε τα συστήματα διαχειρίσεων βάσεων δεδομένων SQLite στην εργασία μας εκμεταλλευόμαστε την βοηθητική κλάση SQLiteAssetHelper. Η κλάση αυτή είναι μία επέκταση της κλάσης SQLiteOpenHelper. Για την χρήση της πρέπει να εισάγουμε το βασικό πακέτο της κλάσης στο έργο μας και να δημιουργήσουμε μία κλάση που υλοποιεί την SQLiteAssetHelper.

```
import com.erkan.sqliteasset.SQLiteAssetHelper;

public class DataBaseHelper extends SQLiteAssetHelper{
```

Με την κλάση αυτή μπορούμε να διαχειριζόμαστε τις βάσεις δεδομένων που βρίσκονται στους πόρους τις εφαρμογής μας. Εισάγουμε μία βάση στον κατάλογο assets/databases/ της εφαρμογής και το ονομάζουμε ChinookDB.sqlite. Προσέχουμε να χρησιμοποιήσουμε το ίδιο όνομα ακριβώς στον κώδικά μας. Παρακάτω δηλώνουμε την βάση και μετά την δήλωση αυτή μπορούμε να την χειριστούμε με την χρήση της myDataBase.

```
private SQLiteDatabase myDataBase;

public DataBaseHelper(Context context) {
```

```
        super(context, "ChinookDB.sqlite", null, 1);
        myDataBase=getReadableDatabase();
    }
```

6.5 Η Χρήση του SurfaceView στην EGQL

Για να σχεδιάσουμε διάφορα σχήματα με διάφορα χρώματα με αποδοτικότερο τρόπο χρησιμοποιούμε το SurfaceView του Android. Ο χειρισμός των σχημάτων έτσι ώστε να σχηματίσουμε το ερώτημα που θέλουμε, γίνονται με χειρονομίες που πραγματοποιούνται στην οθόνη της συσκευής. Η κλάση SurfaceView που χρησιμοποιούμε στην εργασία μας υλοποιεί και επεκτείνει τις κλάσεις που μπορούμε να δούμε στον παρακάτω κώδικα:

```
public class MySurfaceView extends SurfaceView implements Runnable,
Callback, OnTouchListener{
```

Το SurfaceView της εργασίας μας εμφανίζεται μέσα στην κύρια δραστηριότητα της οθόνης. Οι βασικότερες μέθοδοι που υλοποιούνται από την κλάση είναι η run() της κλάσης Runnable και η onTouch() της OnTouchListener.

Στην μέθοδο run() καλείται η onDraw μέθοδος της SurfaceView για να σχεδιάσει τα καινούργια σχήματα σε ένα διαφορετικό νήμα από την βασική εφαρμογή. Τα καινούργια σχήματα σχεδιάζονται κάθε φορά με τις καινούργιες παραμέτρους που αλλάζουν σύμφωνα με τις χειρονομίες του χρήστη. Το κλείδωμα και ξεκλείδωμα του Canvas παρουσιάζεται στην μέθοδο αυτή και μεταξύ αυτών καλείται η onDraw μέθοδος. Παρακάτω βλέπουμε τα βασικά τμήματα που αποτελείται η μέθοδος run() της εφαρμογής μας:

```
public void run() {

        c = holder.lockCanvas();

        onDraw(c);

        holder.unlockCanvasAndPost(c);


        }
```

Η μέθοδος onTouch() είναι μέθοδος της κλάσης OnTouchListener. Χρησιμοποιούμε την κλάση αυτή μέσα στην SurfaceView για να χειριζόμαστε τις χειρονομίες που πραγματοποιούνται στην SurfaceView. Για να δούμε σε ποιο αντικείμενο απευθύνεται η χειρονομία χρησιμοποιούμε τα χ και ψ συντεταγμένες του καμβά. Εδώ έχουμε ένα σημαντικό πρόβλημα μετά από ζουμ του καμβά η αλλαγή θέσης μέσα στην οθόνη. Για να πάρουμε κάθε φορά τις ανεξάρτητες από το ζουμ συντεταγμένες του καμβά γράφουμε τον παρακάτω κώδικα.

```
realX=(event.getX()-(translateX))/scaleFactor;
realY=(event.getY()-(translateY))/scaleFactor;
```

Οι event.getX() και event.getY() μας επιστρέφει το που στην οθόνη έχει γίνει η χειρονομία. Τα translateX και translateY είναι τιμές που αντιπροσωπεύουν το κατά πόσο έχει αλλάξει η θέση του καμβά μέσα στην οθόνη. Το scaleFactor υπολογίζεται κάθε φορά που κάνουμε ζουμ και περιέχει μία τιμή που μας δείχνει το κατά πόσο έχουμε κάνει ζουμ.

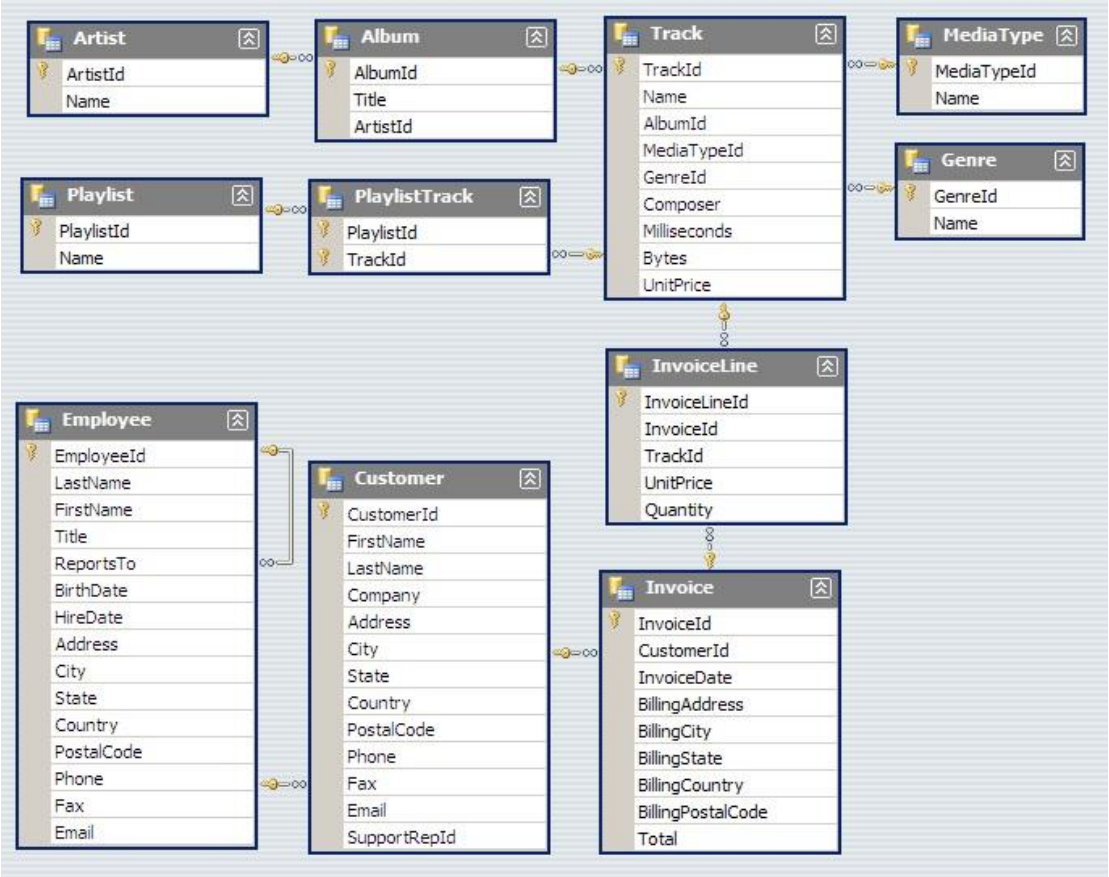
Για τον καλύτερο διαχωρισμό του διπλού κλικ από το μονό δημιουργούμε την GestureListener κλάση μέσα στην SurfaceView για να χειριστούμε διάφορες χειρονομίες που πραγματοποιούνται και είναι αδύνατο να ανιχνευτούν από την OnTouchListener. Παρακάτω βλέπουμε την δήλωση της κλάσης αυτής.

```
private class GestureListener implements
GestureDetector.OnDoubleTapListener, OnGestureListener {
```

Με την βοήθεια των παραπάνω κλάσεων ανιχνεύουμε την συντεταγμένη της κάθε διαφορετικής χειρονομίας και υπολογίζουμε τις σωστές συντεταγμένες της χειρονομίας. Το δεύτερο στάδιο είναι να βλέπουμε εάν υπάρχουν αντικείμενα σε αυτές τις συντεταγμένες που πραγματοποιήθηκε η χειρονομία και να κάνουμε τις κατάλληλες αλλαγές στα σχήματα και στο ερώτημα που προσαρμόζεται ανάλογα.

6.6 Παραδείγματα EGQL

Στα ερωτήματα μας χρησιμοποιούμε την SQLite βάση δεδομένων Chinook 1.3 από το [49]. Το μοντέλο δεδομένων Chinook παριστάνει ένα κατάστημα ψηφιακών μέσων, συμπεριλαμβανομένων των πινάκων για τους καλλιτέχνες, άλμπουμ, τραγούδια, τα τιμολόγια και τους πελάτες. Στην εικόνα 38 βλέπουμε το σχήμα της βάσης δεδομένων που χρησιμοποιούμαι στα παραδείγματα των ερωτημάτων μας.[49]



Εικόνα 38: Το σχήμα του μοντέλου δεδομένων Chinook

Παρακάτω θα δημιουργήσουμε μερικά παραδείγματα ερωτημάτων EGQL και θα τα γράψουμε και με κώδικα SQL για την καλύτερη κατανόηση της γλώσσας EGQL.

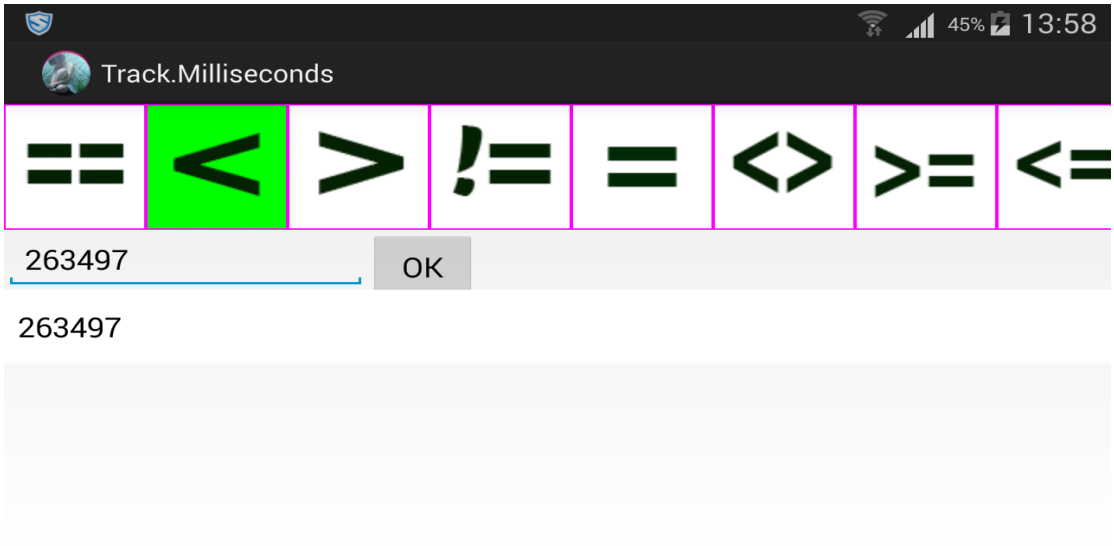
Παράδειγμα 1

Στο πρώτο παράδειγμα ερωτήματος ζητάμε από το EGQL να μας εμφανίσει τα ονόματα και τις διάρκειες των τραγουδιών που έχουν διάρκεια μικρότερη από ‘263497’ milliseconds. Στην εικόνα 39 βλέπουμε την δραστηριότητα που επιλέγουμε την συνθήκη αυτή. Για να γίνει αυτό εισάγουμε τον πίνακα Track και επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε στα αποτελέσματα και εισάγουμε την συνθήκη που περιορίζει τα αποτελέσματα που θα εμφανιστούν, στην εικόνα 41 βλέπουμε τα αποτελέσματα που θα εμφανιστούν μετά την εκτέλεση του ερωτήματος. Στην εικόνα 40 βλέπουμε πως θα είναι το ερώτημα στην EGQL. Ο κώδικας SQL που δημιουργείται από το ερώτημα αυτό είναι:

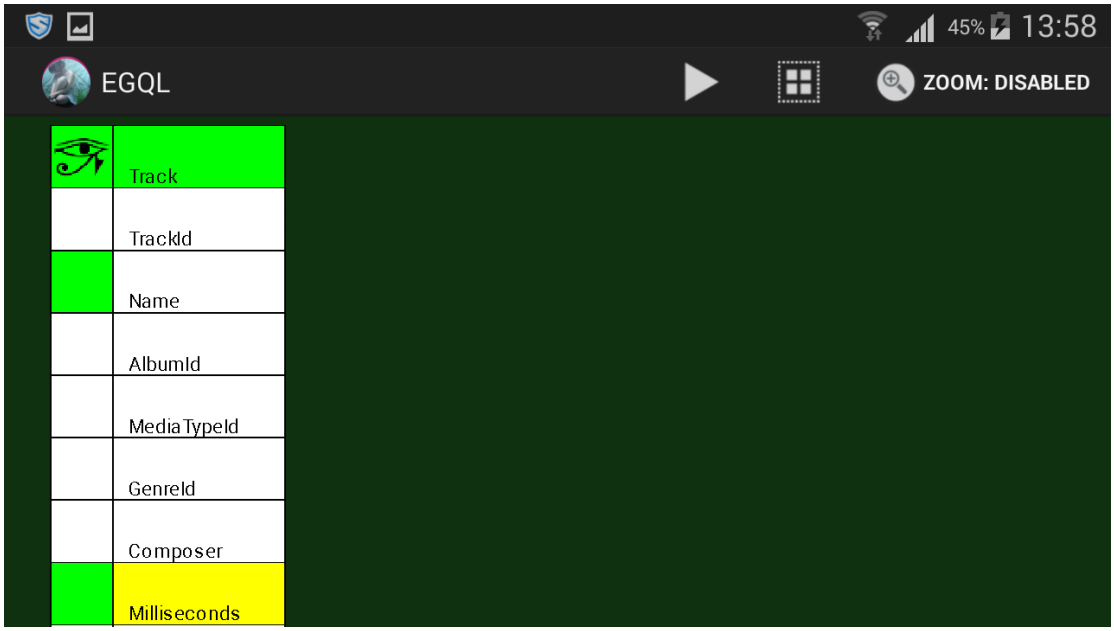
SELECT Track.Name, Track.Milliseconds

FROM Track

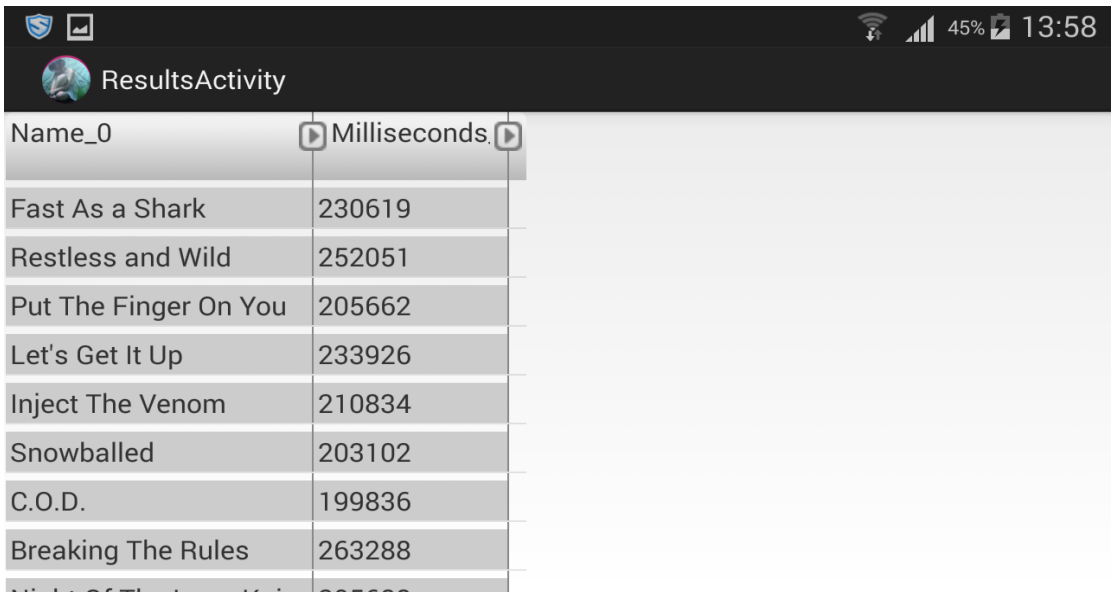
WHERE ((Track.Milliseconds < ‘263497’))



Εικόνα 39: Επιλογή συνθήκης (1ο Παράδειγμα)



Εικόνα 40: Το ερώτημα EGQL (1ο Παράδειγμα)



Εικόνα 41: Τα αποτελέσματα (1ο Παράδειγμα)

Παράδειγμα 2

Ας υποθέσουμε ότι θέλουμε να εμφανίσουμε τα ονόματα των καλλιτεχνών, τα ID των καλλιτεχνών, τους τίτλους και τα ID των άλμπουμ που έχουν AlbumId μικρότερα από 10. Για να πραγματοποιηθεί το συγκεκριμένο ερώτημα πρέπει να εισάγουμε δυο πίνακες, το Artist και το Album, η σύζευξη των δυο πινάκων δημιουργείται χωρίς να χρειαστεί να κάνουμε επιπλέον κινήσεις εφόσον έχουμε ενεργοποιημένη την συγκεκριμένη ρύθμιση που έχουμε εξηγήσει παραπάνω. Στην εικόνα 42 βλέπουμε το ερώτημα στο EGQL, κάνοντας LONG CLICK στο κίτρινο κελί μπορούμε να δούμε τους περιορισμούς που περιέχει, που στην συγκεκριμένη περίπτωση είναι το AlbumId<10. Στην εικόνα 43 βλέπουμε τα αποτελέσματα της εκτέλεσης του ερωτήματος. Ο κώδικας SQL που δημιουργείται με το ερώτημα αυτό είναι:

SELECT Album.Title, Artist.Name, Album.AlbumId, Artist.ArtistId

FROM Album, Artist

WHERE (Album.ArtistId=Artist.ArtistId) AND (Album.AlbumId<'10')

Εικόνα 42: Το ερώτημα EGQL (2ο Παράδειγμα)

Title_0	Name_1	AlbumId_2	ArtistId_
For Those About To Rock We Salute You	AC/DC	1	1
Balls to the Wall	Accept	2	2
Restless and Wild	Accept	3	2
Let There Be Rock	AC/DC	4	1
Big Ones	Aerosmith	5	3
Jagged Little Pill	Alanis Morissette	6	4
Facelift	Alice In Chains	7	5
Warner 25 Anos	Antônio Carlos Jobim	8	6
Plays Metallica By Four Cellos	Apocalyptica	9	7

Εικόνα 43: Τα αποτελέσματα (2ο Παράδειγμα)

Παράδειγμα 3

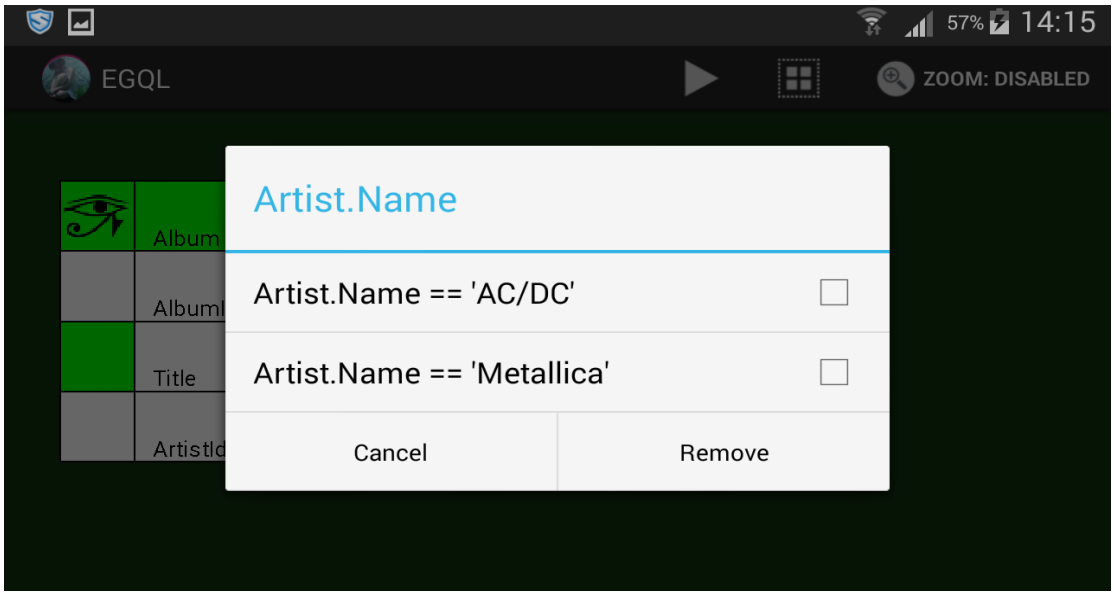
Στον επόμενο παράδειγμα θα εμφανίσουμε τα ονόματα των άλμπουμ των Metallica και AC/DC. Όταν βάζουμε δυο περιορισμούς στο ίδιο στοιχείο η EGQL τα συνδυάζει με OR, εφόσον δεν μπορεί ένα στοιχείο να περιέχει δυο διαφορετικές εγγραφές και ο συνδυασμός αυτών με AND δεν θα μας εμφάνιζε τίποτα. Επιλέγουμε τους περιορισμούς που θέλουμε να προσθέσουμε στο ερώτημα κάνοντας κλικ στο όνομα του καλλιτέχνη για κάθε καλλιτέχνη που θέλουμε να προσθέσουμε στο ερώτημα. Κάνοντας LONG CLICK στο όνομα του καλλιτέχνη βλέπουμε τους περιορισμούς που θα εισαχθούν στο ερώτημα, εικόνα 44. Το ερώτημα EGQL που βλέπουμε στην εικόνα 45 θα μας εμφανίσει το όνομα του καλλιτέχνη και τον τίτλο του άλμπουμ των καλλιτεχνών Metallica και AC/DC, τα

αποτελέσματα απεικονίζονται στην εικόνα 46. Ο κώδικας SQL που θα δημιουργηθεί είναι:

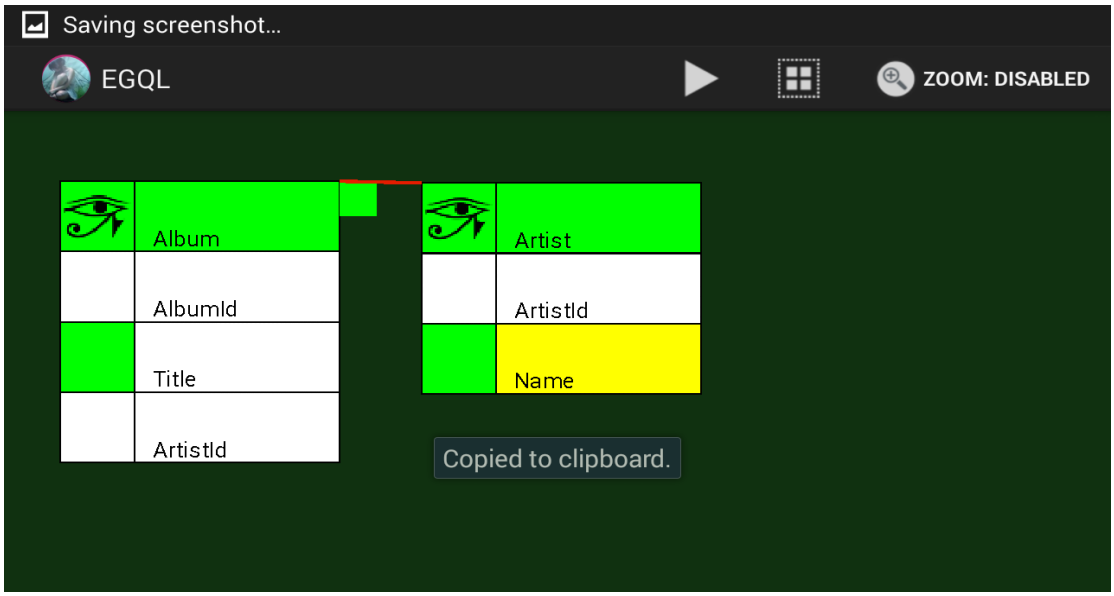
SELECT Album.Title, Artist.Name

FROM Album, Artist

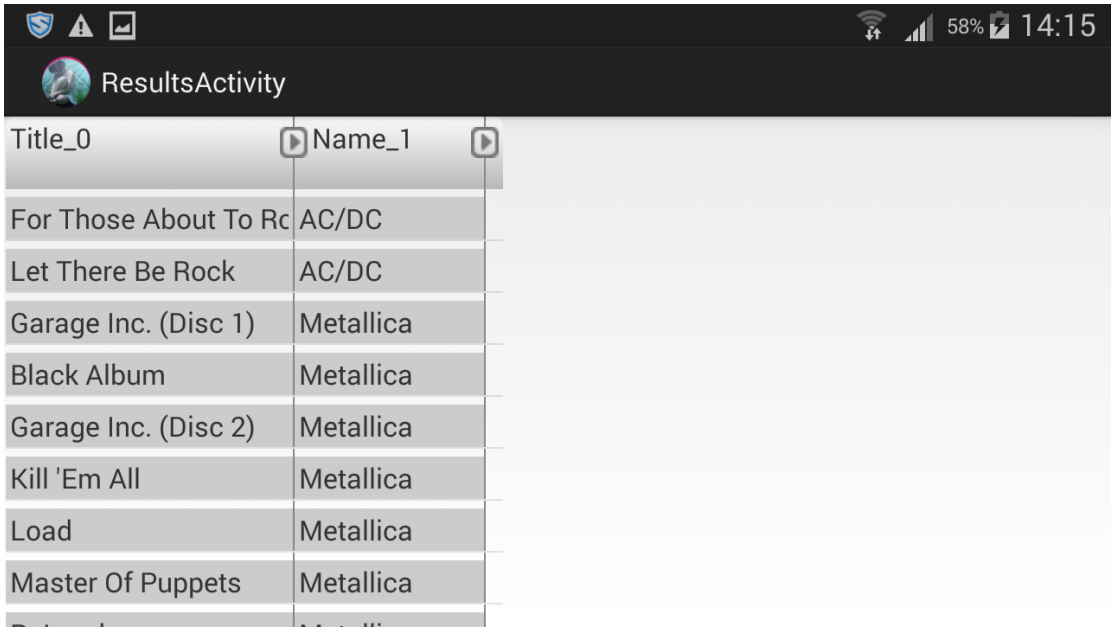
WHERE ((Artist.Name == 'AC/DC') OR (Artist.Name == 'Metallica')) AND (Album.ArtistId=Artist.ArtistId)



Εικόνα 44: Εμφάνιση επιλεγμένων συνθηκών (3ο Παράδειγμα)



Εικόνα 45: Το ερώτημα EGQL (3ο Παράδειγμα)



Εικόνα 46: Τα αποτελέσματα (3ο Παράδειγμα)

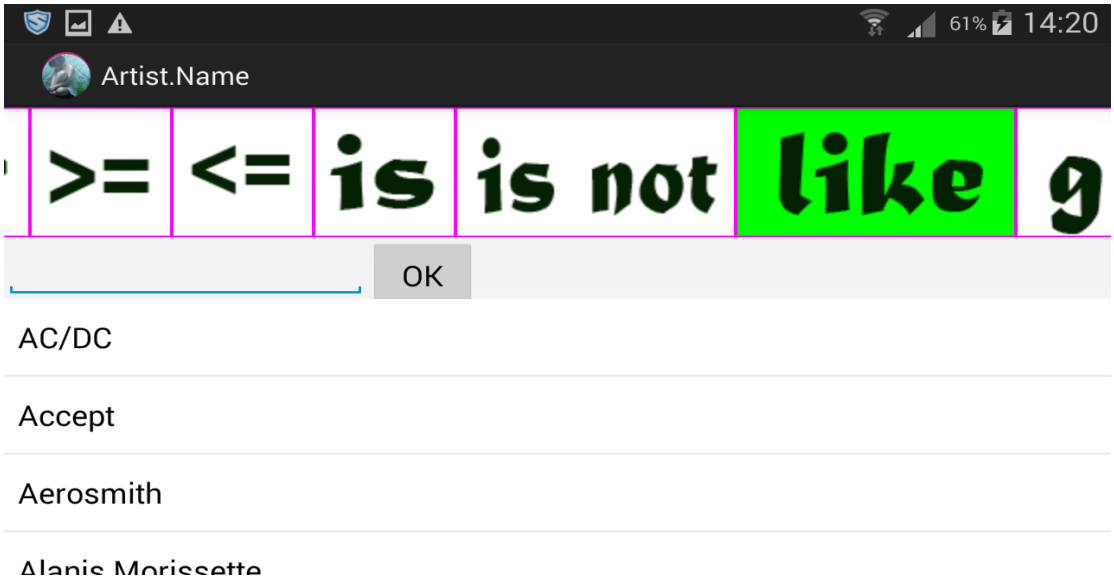
Παράδειγμα 4

Ας υποθέσουμε ότι θέλουμε να εμφανίσουμε τα ονόματα των άλμπουμ και καλλιτεχνών που τα ονόματα των καλλιτεχνών θα αρχίσουν με το γράμμα 'M' και θα τελειώνουν με το γράμμα 'a'. Για να γίνει αυτό κάνουμε κλικ στο όνομα του καλλιτέχνη και επιλέγουμε τον τελεστή 'LIKE' από τους τελεστές του EGQL, όπως βλέπουμε στην εικόνα 47. Γράφουμε 'M_%a' στο πεδίο αναζήτησης και πατάμε OK. Στην εικόνα 48 βλέπουμε το ερώτημα που έχει δημιουργηθεί στο περιβάλλον του EGQL, παρατηρούμε ότι μοιάζει με το προηγούμενο ερώτημα, εφόσον μοιάζει η δομή του ερωτήματος, για να δούμε τις διαφορές πρέπει να κάνουμε LONG CLICK στο κίτρινο κελί που περιέχει τους περιορισμούς, που θα περιέχει το (Artist.Name LIKE 'M_%a'). Ο κώδικας SQL που δημιουργείται θα είναι το:

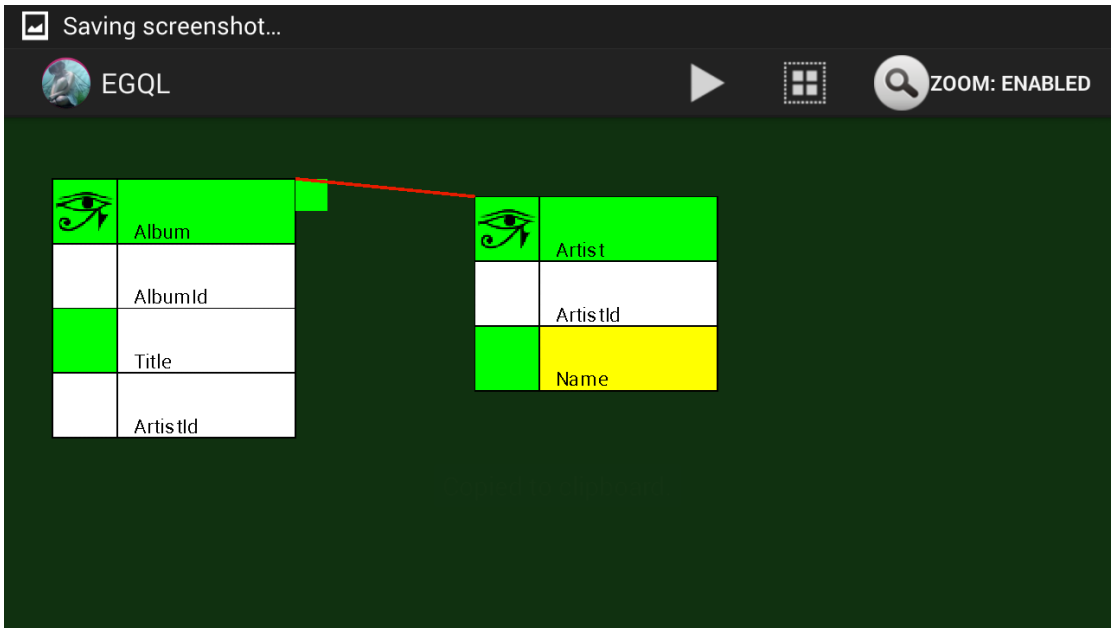
```
SELECT Album.Title, Artist.Name

FROM Album, Artist

WHERE ((Artist.Name LIKE 'M_%a')) AND (Album.ArtistId=Artist.ArtistId)
```



Εικόνα 47: Επιλογή συντελεστή συνθήκης (4ο Παράδειγμα)



Εικόνα 48: Το ερώτημα EGQL (4ο Παράδειγμα)

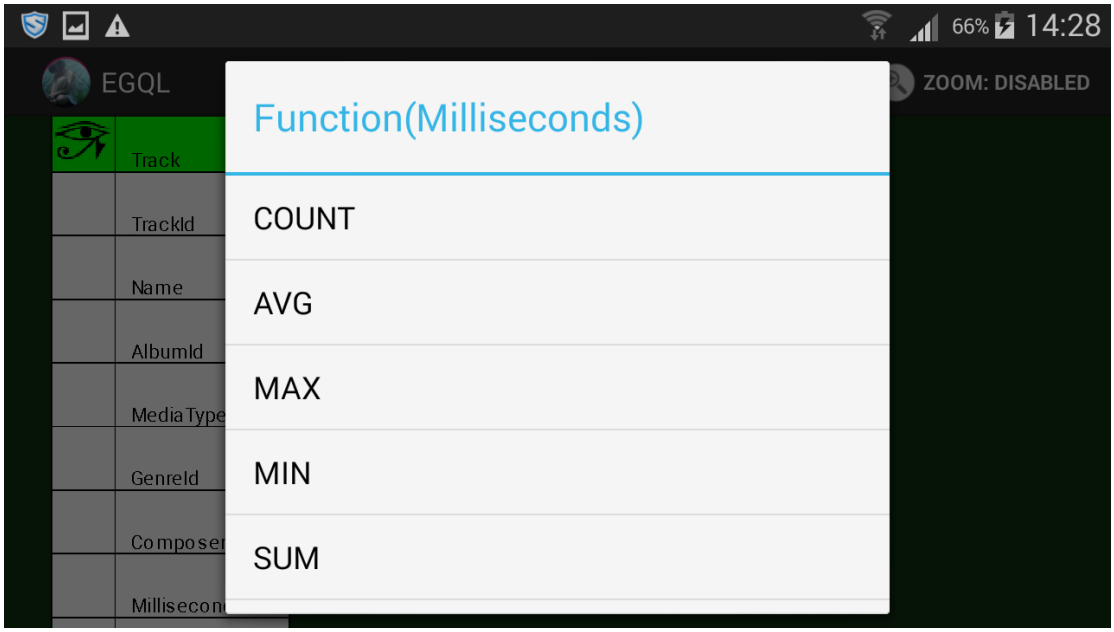
Παράδειγμα 5

Στο περιβάλλον της EGQL μπορούμε να προσθέσουμε και συναρτήσεις στα ερωτήματά μας. Ας υποθέσουμε ότι θέλουμε να βρούμε τον μέσο όρο διάρκειας των τραγουδιών της βάσης δεδομένων Chinook. Κάνοντας διπλό κλικ στην πρώτη στήλη και την γραμμή Milliseconds του πίνακα Track που περιέχει την διάρκεια των τραγουδιών σε χιλιοστά του δευτερολέπτου. Μπορούμε να επιλέξουμε την συνάρτηση που θέλουμε, την AVG στην συγκεκριμένη περίπτωση (Εικόνα 49). Το κελί που επιλέξαμε θα γίνει μπλε που σημαίνει ότι περιέχει μια συνάρτηση, για να δούμε την συνάρτηση που περιέχει ένα κελί κάνουμε LONG CLICK στο κελί και μας εμφανίζει τις συναρτήσεις που περιέχει και θα τις εμφανίσει στο τελικό αποτέλεσμα (Εικόνα 50). Το τελικό αποτέλεσμα θα περιέχει μόνο μια γραμμή και

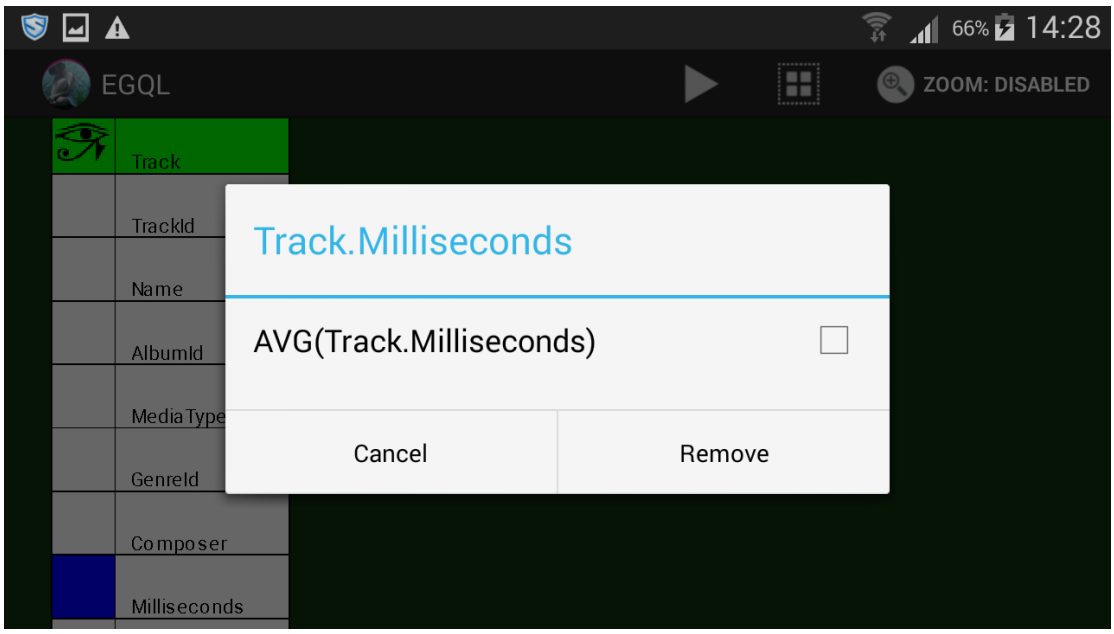
μια στήλη που θα μας λέει πόσα milliseconds είναι ο μέσος όρος διάρκειας των τραγουδιών στην βάση δεδομένων. Ο κώδικας SQL που θα δημιουργείται θα είναι:

SELECT AVG(Track.Milliseconds)

FROM Track



Εικόνα 49: Επιλογή συνάρτησης (5ο Παράδειγμα)



Εικόνα 50: Εμφάνιση επιλεγμένης συνάρτησης (5ο Παράδειγμα)

Παράδειγμα 6

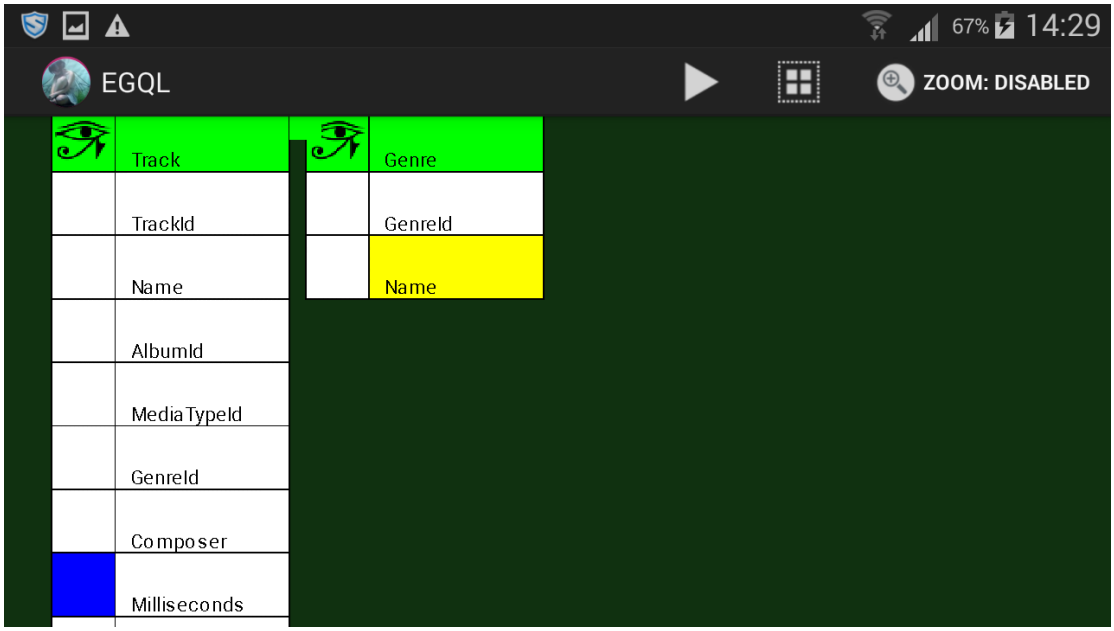
Αν υποθέσουμε ότι θέλουμε να δούμε τον μέσο όρο διάρκειας των reggae τραγουδιών πρέπει να εισάγουμε 2 πίνακες στο καμβά, τον πίνακα Track που περιέχει τις διάρκειες των τραγουδιών σε χιλιοστά του δευτερολέπτου και τον

πίνακα Genre που μπορούμε να δούμε τα ονόματα των ειδών. Επιλέγουμε το reggae για είδος από τον πίνακα Genre και εισάγουμε μια συνάρτηση AVG στην πρώτη στήλη της γραμμής Milliseconds του πίνακα. Το ερώτημα το βλέπουμε στην εικόνα 51 σε μορφή EGQL. Το αποτέλεσμα της εκτέλεσης του ερωτήματος θα μας δώσει τον αριθμό 247178, που είναι ο μέσος όρος των reggae τραγουδιών σε milliseconds. Ο κώδικας SQL που θα πάρουμε από αυτό το ερώτημα είναι:

SELECT AVG(Track.Milliseconds)

FROM Track, Genre

WHERE ((Genre.Name == 'Reggae')) AND (Track.GenreId=Genre.GenreId)



Εικόνα 51: Το ερώτημα EGQL (6ο Παράδειγμα)

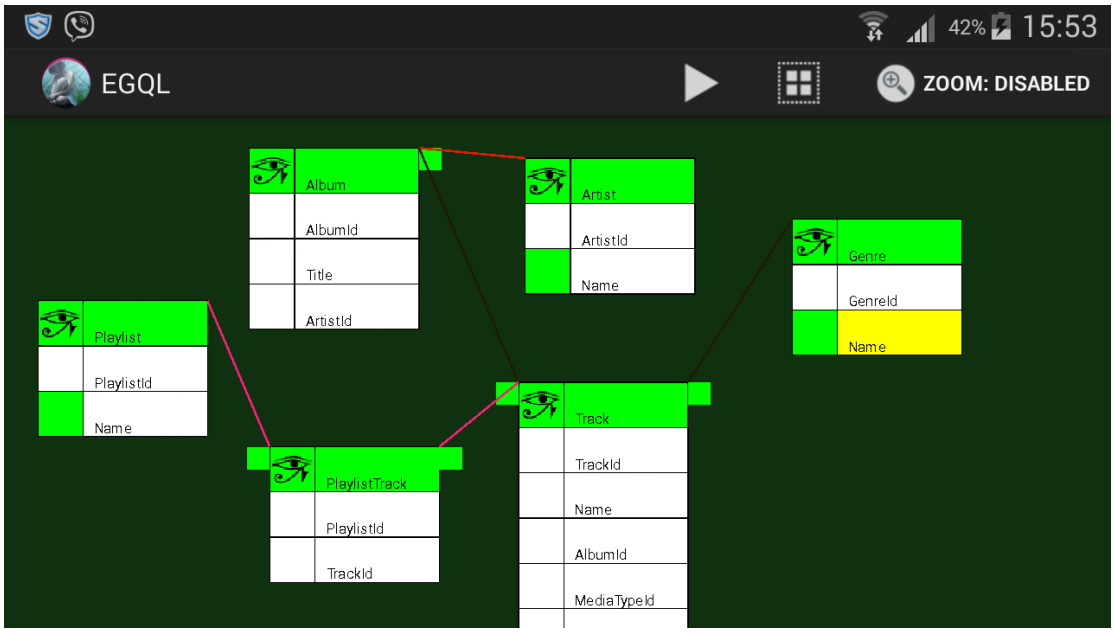
Παράδειγμα 7

Στο παρακάτω παράδειγμα εμφανίζουμε τα ονόματα των καλλιτεχνών που έχουν HIP HOP τραγούδια και τα ονόματα των playlist που περιέχουν αυτά τα τραγούδια. Για να δημιουργηθεί αυτό το ερώτημα στην EGQL πρέπει απλά να επιλέξουμε τους κατάλληλους πίνακες και να κάνουμε απλό κλικ στα κελιά που θέλουμε να εμφανίσουμε, δηλαδή στο όνομα του καλλιτέχνη, το είδος που τραγουδάει και το όνομα του playlist, και να προσθέσουμε μια συνθήκη στο είδος κάνοντας κλικ στο κελί Name του πίνακα Genre και επιλέγοντας το "Hip Hop/Rap" από τις διαθέσιμες επιλογές που περιέχει ο συγκεκριμένος πίνακας. Στην εικόνα 52 βλέπουμε το ερώτημα σε EGQL, για να δούμε το είδος που έχει επιλεχθεί κάνουμε LONG CLICK στο κίτρινο κελί και βλέπουμε ότι έχει επιλεχθεί το στοιχείο που θέλουμε να ψάξουμε δηλαδή το "Hip Hop/Rap" (Εικόνα 53). Ο κώδικας SQL που δημιουργείται μετά την εκτέλεση του ερωτήματος είναι:

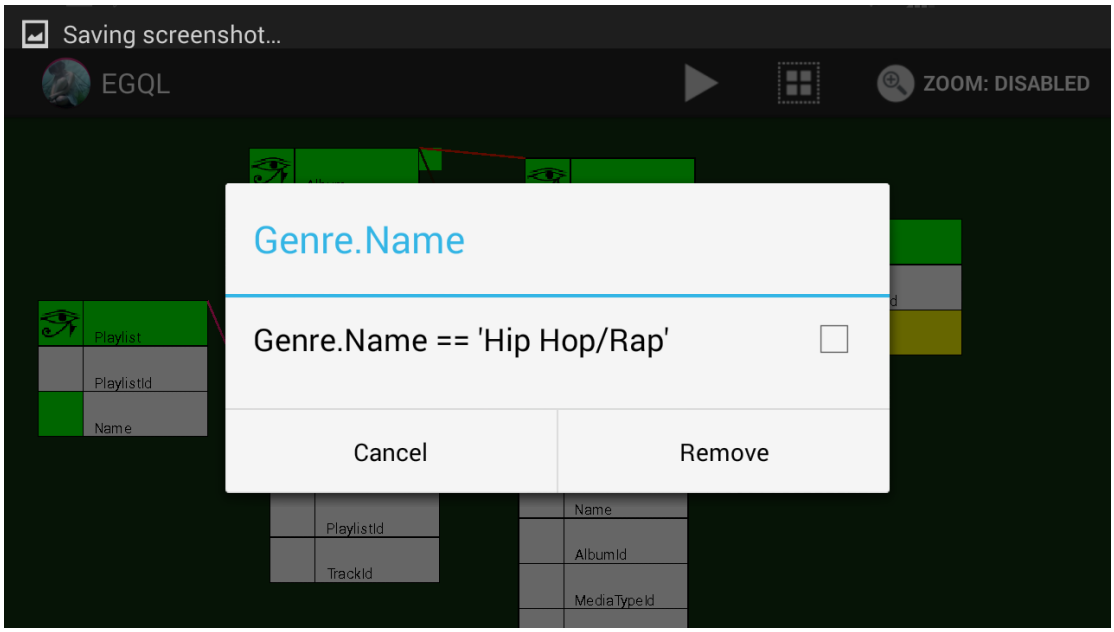
SELECT Genre.Name, Playlist.Name, Artist.Name

FROM Album, Artist, Genre, Track, PlaylistTrack, Playlist

WHERE (Genre.Name=='Hip Hop/Rap') AND (Album.ArtistId=Artist.ArtistId) AND (Track.GenreId=Genre.GenreId) AND (Track.AlbumId=Album.AlbumId) AND (PlaylistTrack.TrackId=Track.TrackId) AND (PlaylistTrack.PlaylistId=Playlist.PlaylistId)



Εικόνα 52: Το ερώτημα EGQL (7ο Παράδειγμα)



Εικόνα 53: Εμφάνιση επιλεγμένης συνθήκης (7ο Παράδειγμα)

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην εργασία μας έχουμε παρουσιάσει τα βασικά ενός συστήματος Android, τα βασικά της διεπαφής του και των γραφικών του συστήματος. Επίσης έχουμε παρουσιάσει τα βασικά των συστημάτων διαχειρίσεων βάσεων δεδομένων SQLite και μερικές γραφικές γλώσσες ερωτημάτων. Στο τέλος παρουσιάσαμε την γραφική γλώσσα ερωτημάτων χειρονομιών EGQL που έχει αναπτυχθεί μέσω του Android και βασίζεται σε SQLite. Είδαμε ότι η EGQL είναι μία Gesture Query Language που απευθύνεται σε συσκευές μικρών διαστάσεων όπως τα tablets και smartphones. Υποστηρίζει multi-touch χειρονομίες και μία προσέγγιση δομής ερωτημάτων που απευθύνεται σε μη έμπειρους χρήστες στα θέματα των βάσεων δεδομένων. Έχουν χρησιμοποιηθεί διάφορες μεταφορές, χρώματα και άλλα ανθρωπογενείς παράγοντες για την ενίσχυση της ευχρηστίας της EGQL. Στο μέλλον σκοπεύουμε να εξελίξουμε την ευχρηστία και την κατανοησιμότητα της EGQL. Επίσης διερευνούμε διαφορετικούς τύπους διεπαφών χρηστών για συσκευές μικρών διαστάσεων που εφαρμόζονται σε σχεσιακές βάσεις αλλά και σε άλλα μοντέλα δεδομένων όπως τα αντικειμενοστραφή, τα αντικειμενο-σχεσιακά, τα ημι-δομημένα και τα NoSQL.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. "Taneli Armanto: Snake Creator Receives Special Recognition". Dexigner. Retrieved from http://www.dexigner.com/news/4785 , 13 June 2015.

2. Estes, Adam Clark (September 9, 2014). "iPhone 6: A Little Bit Bigger, A Whole Lot Better". Gizmodo. Gawker Media. Retrieved from http://gizmodo.com/iphone-6-a-little-bit-bigger-a-whole-lot-better-1632406881 , 13 June 2015.

3. [Alliance FAQ | Open Handset Alliance] Retrieved from http://www.openhandsetalliance.com/oha_faq.html , 13 June 2015.

4. "Industry Leaders Announce Open Platform for Mobile Devices". Open Handset Alliance. Retrieved from http://www.openhandsetalliance.com/press_110507.html , 13 June 2015.

5. "Open Handset Alliance members page". Open Handset Alliance. Retrieved from http://www.openhandsetalliance.com/oha_members.html , 13 June 2015.

6. Alibaba: Google just plain wrong about our OS".CNET News. Retrieved from http://www.cnet.com/news/alibaba-google-just-plain-wrong-about-our-os/ , 13 June 2015.

7. Amadeo, Ron (21 October 2013). "Google's iron grip on Android: Controlling open source by any means necessary". Ars Technica (p.3). Retrieved from http://arstechnica.com/gadgets/2013/10/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/ , 13 June 2015.

8. "Industry Leaders Announce Open Platform for Mobile Devices" Open Handset Alliance. Retrieved from http://www.openhandsetalliance.com/press_110507.html , 13 June 2015.

9. "FCC Approved HTC Dream". Engadget. Retrieved from http://www.engadget.com/2008/08/18/htc-dream-fcc-approved-android-clear-for-launch/ , 13 June 2015.

10. JOSHUA BROCKMAN Google Is Calling. Will You Answer? Retrieved from http://www.npr.org/templates/story/story.php?storyId=94982690 , 13 June 2015.

11 . Mahapatra, Lisa (November 11, 2013). "Android Vs. iOS: What's The Most Popular Mobile Operating System In Your Country?". Retrieved from http://www.ibtimes.com/android-vs-ios-whats-most-popular-mobile-operating-system-your-country-1464892 13 June 2015.

12. Elmer-DeWitt, Philip (January 10, 2014). "Don't mistake Apple's market share for its installed base". CNN. Retrieved from http://fortune.com/2014/01/10/dont-mistake-apples-market-share-for-its-installed-base/ 13 June 2015.

13. Yarow, Jay (March 28, 2014). "This Chart Shows Google's Incredible Domination Of The World's Computing Platforms". Retrieved from http://www.businessinsider.com/androids-share-of-the-computing-market-2014-3 13 June 2015.

14. "Samsung sells more smartphones than all major manufacturers combined in Q1". Retrieved from http://www.sammobile.com/2014/05/01/samsung-sells-more-smartphones-than-all-major-manufacturers-combined-in-q1/ , 13 June 2015.

15. "Android's Google Play beats App Store with over 1 billion apps, now officially largest". Phonearena.com. Retrieved from http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680 , 13 June 2015.

16. Developer Economics Q3 2013 analyst report – http://www.visionmobile.com/DevEcon3Q13 – retrieved in 13 June 2015.

17. Brady, P. (2008, May). Anatomy & physiology of an android. In *Google I/O Developer Conference*.

18. "Android 4.4.2 KitKat running Kernel 3.10 on the Samsung Galaxy Ace Style". gsmarena.com. Retrieved from http://cdn.gsmarena.com/vv/newsimg/14/04/galaxy-ace-style/gsmarena_003.jpg , 13 June 2015.

19. "Android 4.4.2 KitKat running Kernel 3.10 on the Exynos variant of the Samsung Galaxy S5 (SM-G900H)". gsmkhmer.com. Retrieved from http://www.gsmkhmer.com/upload/data/image/n2f8uv-1283e1.png , 13 June 2015.

20. Ryan Whitwam . "HTC Posts Android 4.4 Kernel Source And Framework Files For One Google Play Edition, OTA Update Can't Be Far Off". androidpolice.com. Retrieved from http://www.androidpolice.com/2013/11/25/htc-posts-android-4-4-kernel-source-and-framework-files-for-one-google-play-edition-ota-update-cant-be-far-off/ , 13 June 2015.

21. "Android 4.4.2 on a Nexus 5 (screenshot)". mobilesyrup.com. Retrieved from http://mobilesyrup.com/wp-content/uploads/2013/12/android442nexus5.jpg , 13 June 2015.

22 "Android 4.4.2 on a Galaxy Note 3 (screenshot)". mobilesyrup.com. Retrieved from http://www.sammobile.com/wp-content/uploads/2014/01/Screenshot_2014-01-12-20-03-13.png , 13 June 2015.

23. Darcey, L., & Conder, S. Android application development. Sams Teach Yourself in, 24.

24. https://eclipse.org/ Retrieved in , 13 June 2015.

25. https://developer.android.com/sdk/index.html Retrieved in , 13 June 2015.

26. https://www.genymotion.com/ Retrieved in , 13 June 2015.

27. http://sqlitebrowser.org/ Retrieved in , 13 June 2015.

28. http://developer.android.com/reference/android/content/Context.html Retrieved in , 13 June 2015.

29. http://developer.android.com/reference/android/app/Activity.html Retrieved in , 13 June 2015.

30. http://developer.android.com/reference/android/content/Intent.html Retrieved in , 13 June 2015.

31. http://developer.android.com/reference/android/app/Service.html Retrieved in , 13 June 2015.

32. http://developer.android.com/guide/topics/manifest/manifest-intro.html Retrieved in , 13 June 2015.

33. http://developer.android.com/reference/android/content/res/Resources.html Retrieved in , 13 June 2015.

34. https://source.android.com/devices/graphics.html Retrieved in , 13 June 2015.

35. https://developer.android.com/guide/topics/ui/overview.html Retrieved in , 13 June 2015.

36. http://developer.android.com/guide/topics/ui/declaring-layout.html Retrieved in , 13 June 2015.

37. http://developer.android.com/guide/topics/ui/controls.html Retrieved in , 13 June 2015.

38. http://developer.android.com/guide/topics/graphics/2d-graphics.html Retrieved in , 13 June 2015.

39. "Most Widely Deployed SQL Database Estimates". Sqlite.org. Retrieved from http://sqlite.org/mostdeployed.html, 13 June 2015.

40. Owens, M. (2006). The Definitive Guide to SQLite, Apress.

41. «Interview: Richard Hipp on UnQL, a New Query Language for Document Databases». InfoQ. Retrieved from http://www.infoq.com/news/2011/08/UnQL , 13 June 2015.

42. «Interview: Richard Hipp on UnQL, a New Query Language for Document Databases». InfoQ. Retrieved from http://www.infoq.com/news/2011/08/UnQL , 13 June 2015.

43. https://github.com/jgilfelt/android-sqlite-asset-helper Retrieved in , 13 June 2015.

44. https://www.sqlite.org/whentouse.html Retrieved in , 13 June 2015.

45. Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS), 1*(1), 9-36.

46. Keramopoulos, E. U. C. L. I. D., Pouyioutas, P. H. I. L. I. P. P. O. S., & Ptohos, T. A. S. O. S. (2002). A Comparison Analysis of Graphical Models of Object-Oriented Databases and the GOQL Model. In Proc. 6th International Conference on Computers (pp. 43-49).

47. Jiang, L., Mandel, M., & Nandi, A. (2013). GestureQuery: a multitouch database query interface. Proceedings of the VLDB Endowment, 6(12), 1342-1345.

48. Bragdon, A., Nelson, E., Li, Y., & Hinckley, K. (2011, May). Experimental analysis of touch-screen gesture designs in mobile environments. InProceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 403-412). ACM.

49. https://chinookdatabase.codeplex.com Retrieved in , 13 June 2015.