ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ»

Εκπαιδευτική Android Εφαρμογή για ημερολογιακή αποθήκευση και οργάνωση εργασιών

Μπάκας Κωνσταντίνος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Γεώργιος Σταμούλης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

Εκπαιδευτική Android Εφαρμογή για ημερολογιακή αποθήκευση και οργάνωση εργασιών

Μπάκας Κωνσταντίνος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Σταμούλης Γεώργιος

Λαμία, 2016

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο Εκπαιδευτική Android Εφαρμογή για ημερολογιακή αποθήκευση και οργάνωση εργασιών αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο ΔΗΛΩΝ

Ημερομηνία

Υπογραφή

Εκπαιδευτική Android Εφαρμογή για ημερολογιακή αποθήκευση και οργάνωση εργασιών

Μπάκας Κωνσταντίνος

Τριμελής Επιτροπή:

Ονοματεπώνυμο, Σταμούλης Γεώργιος

Ονοματεπώνυμο, Αντωνής Κωνσταντίνος

Ονοματεπώνυμο, Λουκόπουλος Αθανάσιος

Επιστημονικός Σύμβουλος:

Οικονόμου Παναγιώτης

ΠΕΡΙΛΗΨΗ

Σκοπός αυτής της εργασίας είναι η υλοποίηση μίας Android εκπαιδευτικής εφαρμογής για ημερολογιακή οργάνωση και διαχείριση εργασιών από εκπαιδευτικούς. Αρχικά, στο πρώτο κεφάλαιο παρουσιάζεται μια ιστορική αναδρομή του λειτουργικού συστήματος. Στη συνέχεια παρουσιάζεται η αρχιτεκτονική του λειτουργικού συστήματος, το λογισμικό δηλαδή στο οποίο στηρίζεται το λειτουργικό σύστημα. Στην ίδια ενότητα γίνεται αναφορά και για την ανατομία των εφαρμογών, από τι δηλαδή αποτελείται μια εφαρμογή. Το τρίτο κεφάλαιο αποτελεί μια παρουσίαση στον τρόπο που αναπτύσσονται οι εφαρμογές για το λειτουργικό σύστημα Android. Τέλος, παρουσιάζεται η εφαρμογή και η ανάλυση του κώδικα της εφαρμογής που υλοποιήθηκε.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1
Εισαγωγή στο λειτουργικό σύστημα και τις εκδόσεις του

1.1 Εισαγωγή

Το Android είναι ένα λειτουργικό σύστημα για κινητά τηλέφωνα που έχει αναπτυχθεί από την Google και βασίζεται στον πυρήνα του Linux. Έχει σχεδιαστεί κυρίως για φορητές συσκευές με οθόνες αφής όπως smartphones και tablets. Εκτός από τις συσκευές με οθόνη αφής, η Google έχει αναπτύξει περαιτέρω το Android TV για τηλεοράσεις, το Android Auto για αυτοκίνητα και το Android Wear για ρολόγια χειρός, το καθένα με ένα εξειδικευμένο περιβάλλον εργασίας χρήσης. Παραλλαγές του λειτουργικού συστήματος χρησιμοποιούνται επίσης για φορητούς υπολογιστές, κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές και άλλα ηλεκτρονικά είδη.

Το λειτουργικό σύστημα αρχικά αναπτύχθηκε από την Android Inc. η οποία εξαγοράστηκε από την Google το 2005[1]. Το Android παρουσιάστηκε το 2007 μαζί με την ίδρυση του Open Handset Alliance - μια κοινοπραξία των hardware, software και εταιρειών τηλεπικοινωνίας.

Ο πηγαίος κώδικας του λειτουργικού συστήματος Android δίνεται από την Google με άδειες χρήσης ανοιχτού κώδικα. Σύμφωνα με τον Brodkin[2] ένα από τα στοιχεία του Android που το κάνει πιο αγαπητό είναι ότι είναι ανοιχτού κώδικα. Αυτό μπορεί να μη σημαίνει πολλά για το μέσο χρήστη, αλλά είναι μια μεγάλη υπόθεση και μια σημαντική αρχή για τη διαφάνεια του λειτουργικού συστήματος.

1.2 Ιστορική Αναδρομή

Η Android Inc. ιδρύθηκε στο Πάλο Άλτο της Καλιφόρνια τον Οκτώβριο του 2003[3] από τον Andy Rubin (συνιδρυτή της Danger)[4], τον Rich Miner (συνιδρυτή της Wildfire Communications) [5], τον Nick Sears (once VP στην T-Mobile) (Vogelstain, 2011), και τον Chris White επικεφαλής του σχεδιασμού και της διασύνδεσης ανάπτυξης στη WebTV. Οι πρώτες προθέσεις της εταιρείας ήταν να αναπτύξει ένα προηγμένο λειτουργικό σύστημα για ψηφιακές φωτογραφικές μηχανές. Όταν έγινε αντιληπτό ότι η αγορά για τις φορητές συσκευές δεν ήταν μεγάλη η εταιρεία έστρεψε τις προθέσεις της για την παραγωγή ενός λειτουργικού συστήματος για smartphones που θα ανταγωνιζόταν το Symbian και το Microsoft Windows Mobile[6]. Παρά την επιτυχημένη αρχή της, η Android Inc. λειτουργούσε κρυφά, αποκαλύπτοντας μόνο ότι δούλευαν για λογισμικό κινητών τηλεφώνων. Τον Ιούλιο του 2005 η Google εξαγόρασε την Android Inc. για πάνω από 50 εκατομμύρια δολάρια και τα βασικά στελέχη έμειναν στην εταιρεία μετά την εξαγορά.

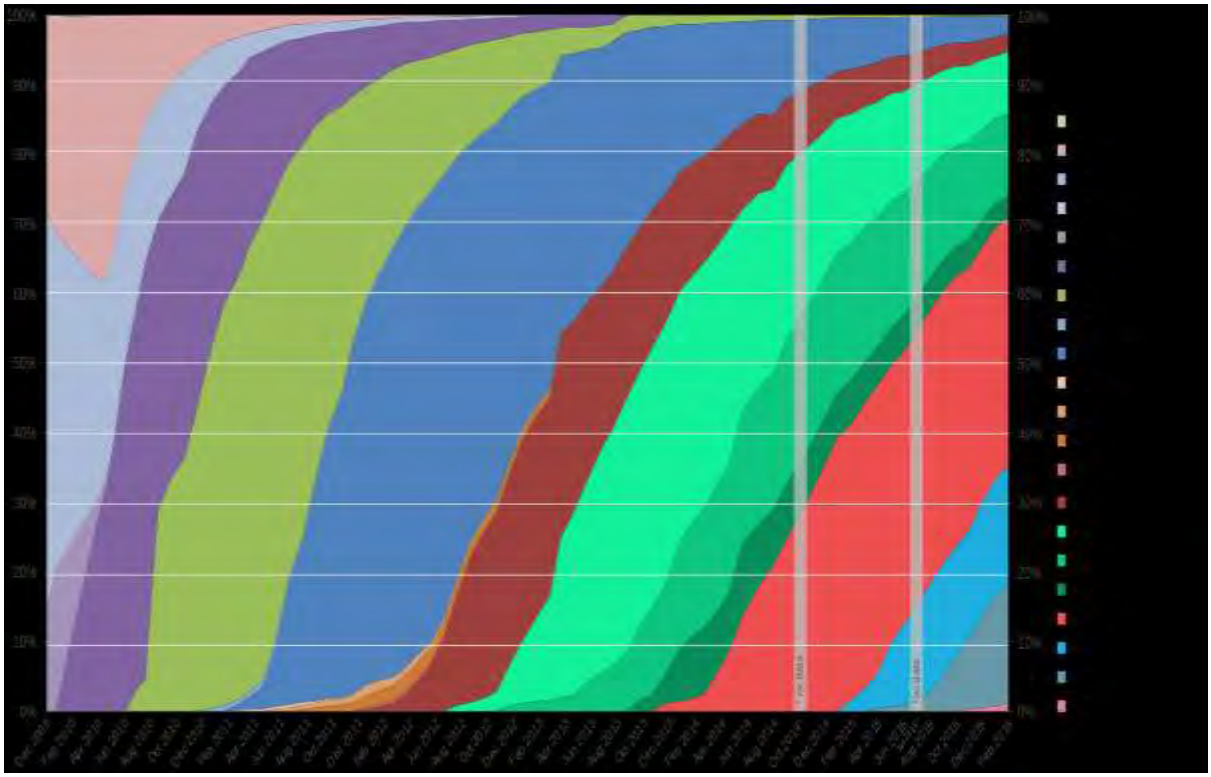
Από το 2008 στο Android έχουν γίνει πολλές ενημερώσεις που έχουν βελτιώσει σταδιακά το λειτουργικό σύστημα, προσθέτοντας νέα χαρακτηριστικά για τον καθαρισμό των σφαλμάτων στις προηγούμενες εκδόσεις. Το 2010 η Google ξεκίνησε την σειρά συσκευών Nexus, μια σειρά από smartphones και tablets που τρέχουν το λειτουργικό σύστημα Android χωρίς τις προσθήκες των κατασκευαστών. Το πρώτο smartphone Nexus[7] κατασκευάστηκε από την HTC. Η Google έκτοτε έχει απελευθερώσει μια σειρά από νεότερες συσκευές, όπως το Nexus 5 από την LG και το tablet Nexus 7 από την Asus.

1.3 Εκδόσεις του λειτουργικού συστήματος

Η πρώτη εμπορική έκδοση, το Android 1.0 κυκλοφόρησε το Σεπτέμβριο του 2008. Από τότε η Google και η Open Handset Alliance (OHA), έχουν κάνει μια σειρά από μεγάλες αλλαγές σε σχέση με την αρχική έκδοση. Η πιο πρόσφατη ενημέρωση του λειτουργικού συστήματος είναι το Android 6.0 το οποίο κυκλοφόρησε τον Οκτώβριο του 2015. Στον πίνακα που ακολουθεί παρουσιάζονται οι εκδόσεις του λειτουργικού συστήματος.

Όνομα	Νούμερο έκδοσης	API level
	1.0	1
	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Eclair	2.0-2.1	5-7
Froyo	2.2-2.2.3	8
Gingerbread	2.3-2.3.7	9-10
Honeycomb	3.0-3.2.6	11-13
Ice Cream Sandwich	4.0-4.0.4	14-15
Jelly Bean	4.1-4.3.1	16-18
KitKat	4.4 - 4.4.4	19-20
Lollipop	5.0-5.1.1	21-22
Marshmallow	6.0-6.0.1	23
N	Developers Preview	

πίνακας 1.1: εκδόσεις του Android



Εικόνα 1.1 Παγκόσμια Android διανομή από τον Δεκέμβριο του 2009. Το Φεβρουάριο του 2016 το Android 4.4 είναι η πιο ευρέως χρησιμοποιούμενη έκδοση του Android.

ΚΕΦΑΛΑΙΟ 2
Αρχιτεκτονική του λειτουργικού συστήματος Android

2.1 Εισαγωγή

Οι εφαρμογές οι οποίες επεκτείνουν το λειτουργικό σύστημα, κατασκευάζονται με τη χρήση του Android software development kit (SDK) και κατά κύριο λόγο με τη γλώσσα προγραμματισμού java στην οποία στηρίζεται και το λειτουργικό σύστημα. Το SDK διαθέτει ένα ολοκληρωμένο σύνολο εργαλείων ανάπτυξης το οποίο περιλαμβάνει πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθήκες λογισμικού, έναν εξομοιωτή συσκευής, δείγματα κώδικα και μαθήματα για την ανάπτυξη εφαρμογών. Αρχικά το περιβάλλον ανάπτυξης κώδικα ήταν το Eclipse το οποίο χρησιμοποιούσε εργαλεία για την εκτέλεση του κώδικα πάνω στο λειτουργικό σύστημα αλλά από τον Δεκέμβριο του 2014 η Google κυκλοφόρησε το Android Studio, το οποίο έχει σαν βάση το IntelliJ IDEA, ως κύριο IDE για ανάπτυξη εφαρμογών Android. Ακόμα είναι διαθέσιμα και άλλα εργαλεία για ανάπτυξη εφαρμογών όπως το NDK για εφαρμογές που στηρίζονται στις γλώσσες C/C++, το Google App Inventor, ένα οπτικό περιβάλλον για αρχάριους προγραμματιστές και διάφορες άλλες πλατφόρμες για web mobile εφαρμογές.

2.2 Αρχιτεκτονική του Android

Στο Android υπάρχει μια ποικιλία στον τρόπο που μπορεί η συσκευή να αποκτήσει και να εγκαταστήσει τις εφαρμογές. Αυτό γίνεται καθώς υπάρχουν διάφορα ηλεκτρονικά καταστήματα που παρέχουν τα εκτελέσιμα αρχεία των εφαρμογών[8]. Το επίσημο κατάστημα για απόκτηση των εφαρμογών είναι το Google Play Store το οποίο βρίσκεται στις συσκευές που συμφωνούν με τις απαιτήσεις συμβατότητας της Google και την άδεια χρήσης της Google για λογισμικά κινητών. Το Google Play Store επιτρέπει στους χρήστες να περιηγηθούν, να κάνουν λήψη και ενημέρωση των εφαρμογών που είναι δημοσιευμένες σε αυτό. Από τον Ιούλιο του 2013, υπάρχουν πάνω από ένα εκατομμύριο εφαρμογές διαθέσιμες στο Play Store.

Το λειτουργικό σύστημα Android είναι μια στοίβα στοιχείων λογισμικού[9] που χονδρικά χωρίζονται σε πέντε ενότητες και τέσσερα κύρια στρώματα όπως φαίνεται και στο παρακάτω διάγραμμα αρχιτεκτονικής (Εικόνα 2.1).

Εικόνα 2.1 Διάγραμμα αρχιτεκτονικής του Android.

Linux kernel

Στο κάτω μέρος των στρωμάτων είναι το Linux - Linux 2.6 με περίπου 115 μονοπάτια. Αυτό παρέχει τις βασικές λειτουργίες στο σύστημα όπως η διαδικασία διαχείρισης, διαχείριση μνήμης, διαχείριση συσκευών όπως κάμερα, πληκτρολόγιο, οθόνη κλπ. Επίσης, ο πυρήνας χειρίζεται όλα τα πράγματα που το Linux είναι πραγματικά καλό, όπως για παράδειγμα τα δίκτυα.

Libraries

Στην κορυφή του Linux Kernel υπάρχει ένα σύνολο βιβλιοθηκών συμπεριλαμβανομένων του ανοικτού κώδικα WebKit, τις γνωστές βιβλιοθήκες libc, την SQLite βάση δεδομένων, η οποία είναι χρήσιμη για αποθηκευτικό χώρο μέσα στις εφαρμογές, βιβλιοθήκες αναπαραγωγής και εγγραφής ήχου και βίντεο, καθώς και τις βιβλιοθήκες SSL οι οποίες είναι υπεύθυνες για την ασφάλεια στο διαδίκτυο.

Android Runtime

Αυτό είναι το τρίτο τμήμα της αρχιτεκτονικής. Αυτή η ενότητα παρέχει ένα βασικό συστατικό που ονομάζεται Dalvik Virtual Machine το οποίο είναι ένα είδος της Java Virtual Machine και έχει ειδικά σχεδιαστεί και βελτιστοποιηθεί για το Android. Η Dalvik VM κάνει χρήση των βασικών χαρακτηριστικών του Linux όπως η διαχείριση της μνήμης και multi-threading, η οποία είναι εγγενής στη γλώσσα Java. Η Dalvik VM επιτρέπει σε κάθε εφαρμογή Android να τρέξει τη δική της διαδικασία, στην εικονική μηχανή Dalvik.

Application Framework

Το στρώμα πλαίσιο εφαρμογής παρέχει πολλές υπηρεσίες υψηλότερου επιπέδου σε εφαρμογές με τη μορφή java κλάσεων. Οι προγραμματιστές εφαρμογών επιτρέπεται να κάνουν χρήση των υπηρεσιών αυτών στις εφαρμογές τους.

Applications

Όλες οι εφαρμογές του λειτουργικού συστήματος βρίσκονται στο υψηλότερο στρώμα. Παράδειγμα εφαρμογών είναι web browsers, παιχνίδια, βιβλία και πολλά άλλα.

2.3 Ανατομία των εφαρμογών Android

Μια εφαρμογή Android αποτελείται από ένα ή περισσότερα συστατικά του πυρήνα (core components). Σε μια εφαρμογή που αποτελείται από πολλά μέρη, η συνεργασία των συστατικών είναι απαραίτητη για την επιτυχία της.

Τα συστατικά του πυρήνα μπορεί να είναι

1. Μια δραστηριότητα (Activity).
2. Μια υπηρεσία (Service).
3. Ένας δέκτης εκπομπής.
4. Ένας πάροχος περιεχομένου.

2.3.1 Δραστηριότητα (Activity)

Μια τυπική εφαρμογή Android αποτελείται από μία ή περισσότερες δραστηριότητες. Ένα Activity δείχνει το περιβάλλον το οποίο βλέπει ο χρήστης της εφαρμογής (GUI). Κατά την ανάπτυξη κάθε εφαρμογής ορίζεται μόνο ένα Activity το οποίο θα εκτελεστεί πρώτο, όταν η εφαρμογή ξεκινήσει. Για την μετάβαση σε άλλο Activity, το οποίο συνεπάγεται τη μετάβαση σε άλλη λειτουργία της εφαρμογής, γίνεται χρήση του πρωτοκόλλου intent το οποίο καλείται μέσα σε κάθε Activity. Στην εικόνα 2.2 παρουσιάζεται ο τρόπος που γίνεται μετάβαση σε διαφορετικά Activities σε μία εφαρμογή καιρού.

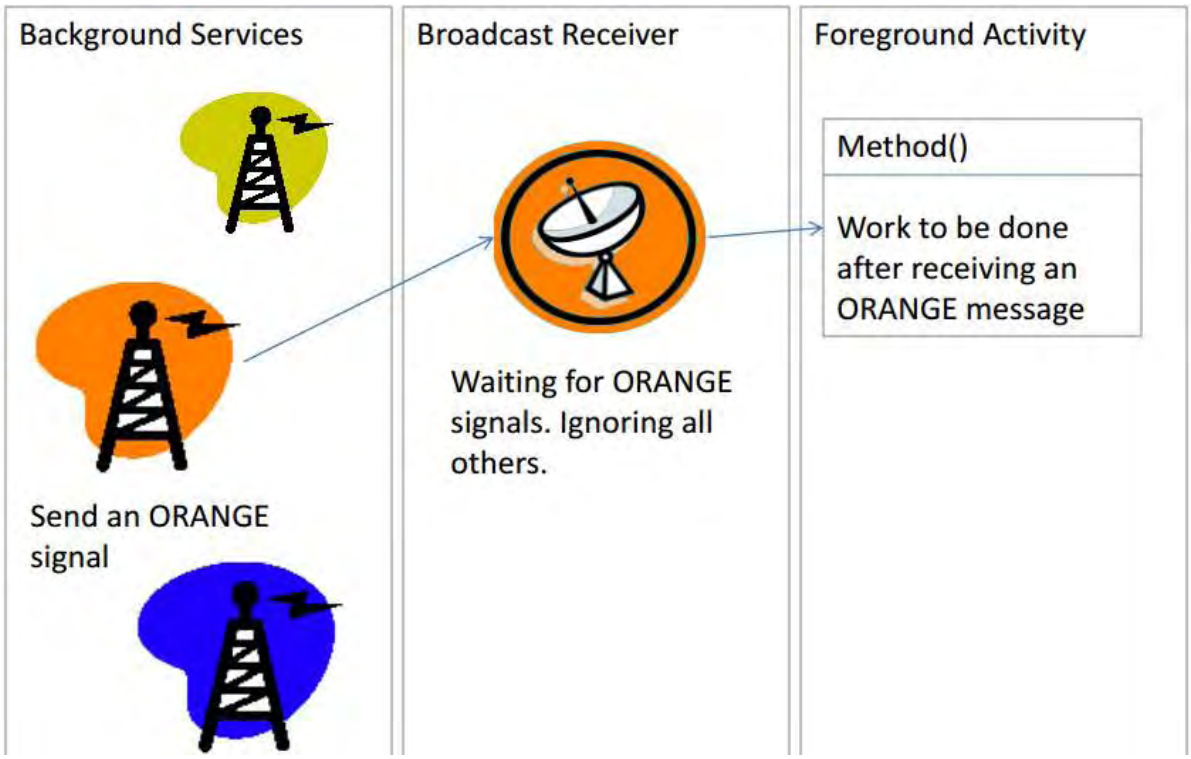
Εικόνα 2.2: Μετάβαση σε 3 διαφορετικά Activities σε μία εφαρμογή.

2.3.2 Υπηρεσία (Service)

Οι υπηρεσίες είναι ειδικοί τύποι δραστηριοτήτων που δεν έχουν οπτική διεπαφή για τον χρήστη. Αυτές είναι υπηρεσίες που εκτελούνται συνήθως στο παρασκήνιο για αόριστο χρονικό διάστημα. Υπηρεσίες όπως το GPS ή το wi-fi τρέχουν στο παρασκήνιο του κινητού ανεξάρτητα από τη χρήση του τηλεφώνου.

2.3.3 Δείκτης εκπομπής (Broadcast Receiver)

Οι δείκτες εκπομπής είναι ειδικοί listeners που περιμένουν ολόκληρο το σύστημα ή τοπικά μεταδιδόμενα μηνύματα και δεν περιέχουν διεπαφή χρήστη όπως και οι υπηρεσίες. Ένας δείκτης διεπαφής θα μπορούσε να απαντήσει είτε εκτελώντας μια συγκεκριμένη δραστηριότητα ή χρησιμοποιώντας το μηχανισμό κοινοποίησης ζητώντας την άδεια του χρήστη. Στην εικόνα 2.3 παρουσιάζεται ένα παράδειγμα για τον τρόπο λειτουργίας του δείκτη εκπομπής.

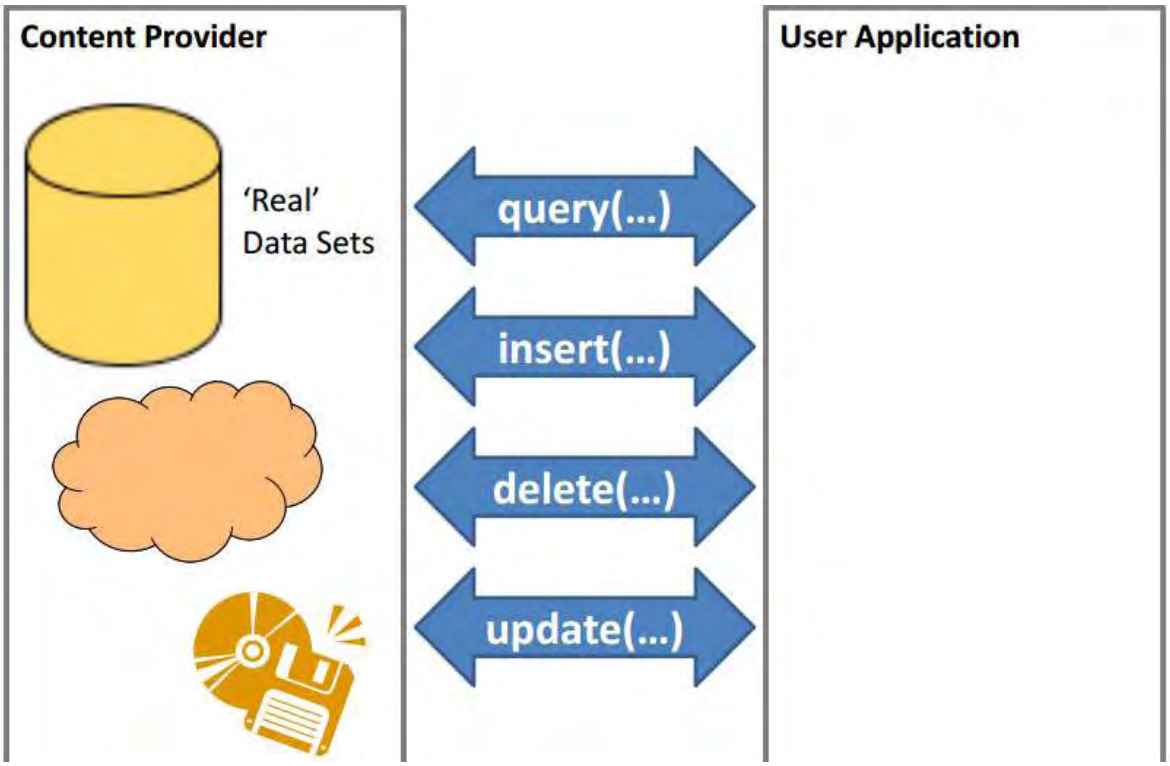


Εικόνα 2.3: Παράδειγμα λειτουργίας του δείκτη εκπομπής.

2.3.4 Πάροχος περιεχομένου (Content Provider)

Η κλάση πάροχος περιεχομένου προσφέρει ένα τυποποιημένο σύνολο μεθόδων που μοιάζει με βάση δεδομένων και επιτρέπει σε άλλες εφαρμογές την ανάκτηση, διαγραφή, ενημέρωση, και τοποθέτηση τα δεδομένων.

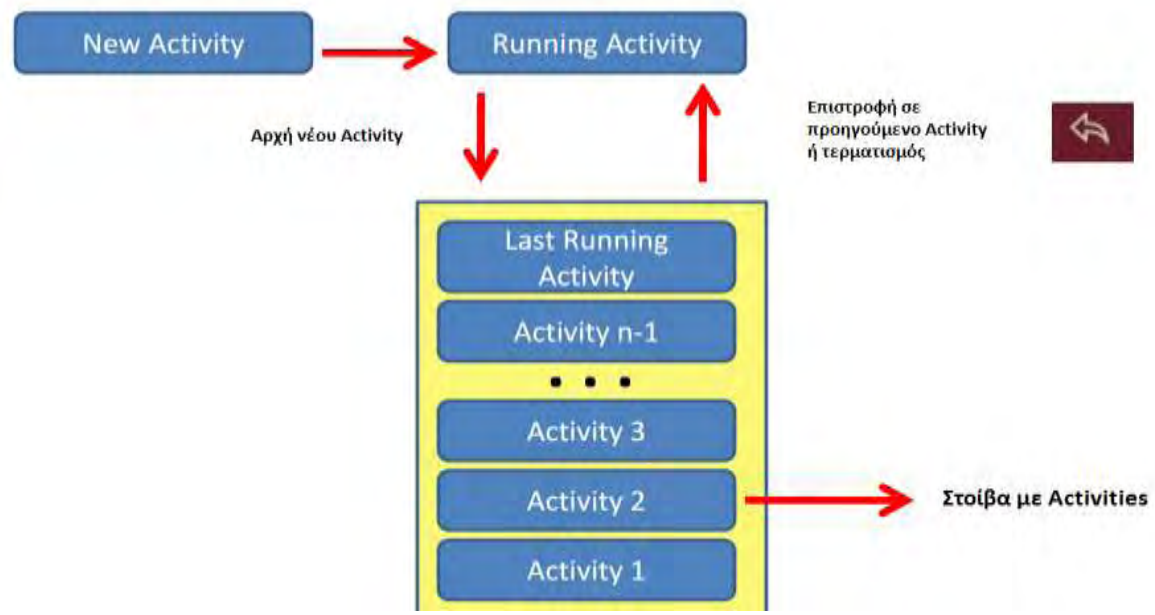
Κοινή πάροχοι δεδομένων περιλαμβάνουν: επαφές, φωτογραφίες, μηνύματα, αρχεία ήχου και μηνύματα ηλεκτρονικού ταχυδρομείου (Εικόνα 2.4).



Εικόνα 2.4: Ο Content Provider είναι ένα περιτύλιγμα που κρύβει τα πραγματικά φυσικά στοιχεία. οι χρήστες αλληλεπιδρούν με τα δεδομένα τους μέσω μιας κοινής διεπαφής αντικειμένου.

2.4 Κύκλος ζωής των εφαρμογών

Κατά τη διάρκεια εκτέλεσης μίας εφαρμογής γίνεται χρήση μιας στοίβας από δραστηριότητες. Όταν ξεκινάει μια νέα δραστηριότητα τοποθετείται στην κορυφή της στοίβας ώστε να λειτουργήσει, ενώ η προηγούμενη δραστηριότητα ωθείται προς τα κάτω επίπεδα της στοίβας και μπορεί να επανέλθει στο προσκήνιο μετά την ολοκλήρωση της δραστηριότητας που εκτελείται. Οι χρήστες μπορούν να διακόψουν τις δραστηριότητες που εκτελούνται και να μεταβούν σε κάποια άλλη μέσω του γραφικού περιβάλλοντος της εφαρμογής ή να μεταβούν στην προηγούμενη δραστηριότητα πατώντας το κουμπί πίσω που διαθέτουν οι Android συσκευές. Στην εικόνα 2.5 παρουσιάζεται μια σχηματική παρουσίαση της στοίβας δραστηριοτήτων.



Εικόνα 2.5: Εκτέλεση δραστηριοτήτων

Όταν υπάρχει αλλαγή σε μία κατάσταση μέσα στην εφαρμογή, όπως για παράδειγμα τερματισμός της εφαρμογής, το λειτουργικό σύστημα ειδοποιεί την εφαρμογή κάνοντας χρήση τον ακόλουθων μεθόδων οι οποίες θα αναλυθούν εκτενέστερα στο επόμενο καφάλαιο.

- **void onCreate(Bundle savedInstanceState)**
- **void onStart()**
- **void onRestart()**
- **void onResume()**
- **void onPause()**
- **void onStop()**
- **void onDestroy()**

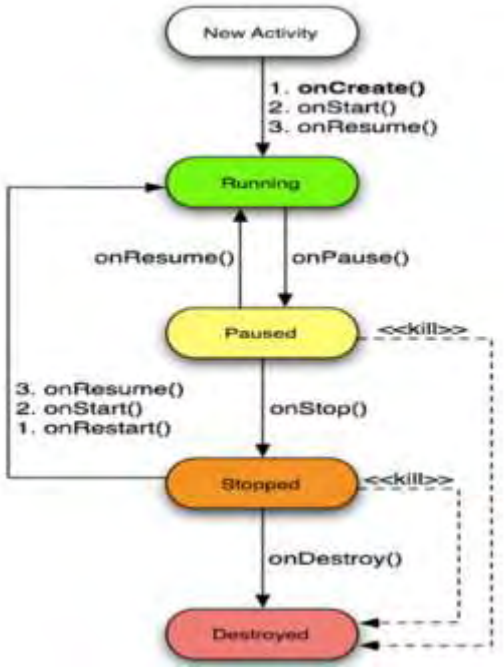
Στο επόμενο παράδειγμα παρουσιάζεται κώδικας με χρήση των μεθόδων αυτών.

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

Σε αυτό το παράδειγμα βλέπουμε την χρήση των μεθόδων. Ο κώδικας που δημιουργεί την εφαρμογή και καλείται το γραφικό περιβάλλον διεπαφής τοποθετείται στην μέθοδο onCreate. Σε περίπτωση που η εφαρμογή θα πρέπει να σώσει κάποια δεδομένα αυτό θα γίνει στην μέθοδο onPause.

Μια δραστηριότητα έχει ουσιαστικά τρεις καταστάσεις (Εικόνα 2.6).

1. Είναι ενεργή ή τρέχει.
2. Είναι σε παύση.
3. Έχει σταματήσει.



Εικόνα 2.6

Όταν η εφαρμογή είναι ενεργή τότε αυτή η δραστηριότητα που τρέχει βρίσκεται στο επάνω μέρος της στοίβας δραστηριοτήτων. Επίσης κατά τη διάρκεια εκτέλεσης αυτής της δραστηριότητας ο χρήστης βλέπει το γραφικό περιβάλλον που αντιστοιχεί σε αυτή ώστε να γίνει η διεπαφή.

Όταν η εφαρμογή βρίσκεται σε κατάσταση παύσης, τότε εξακολουθεί να τρέχει αλλά δεν είναι ορατή ή δεν καλύπτει την πλήρη οθόνη της συσκευής. Μια δραστηριότητα που είναι σε κατάσταση παύσης μπορεί να σταματήσει να δουλεύει σε περίπτωση που για παράδειγμα, η διαθέσιμη μνήμη είναι εξαιρετικά χαμηλή.

Τέλος, μια δραστηριότητα που σταματά παύει να λειτουργεί και τη θέση της παίρνουν άλλες δραστηριότητες Η δραστηριότητα εξακολουθεί να κρατάει τις πληροφορίες που διαθέτει αλλά δεν είναι ορατή από τον χρήστη.

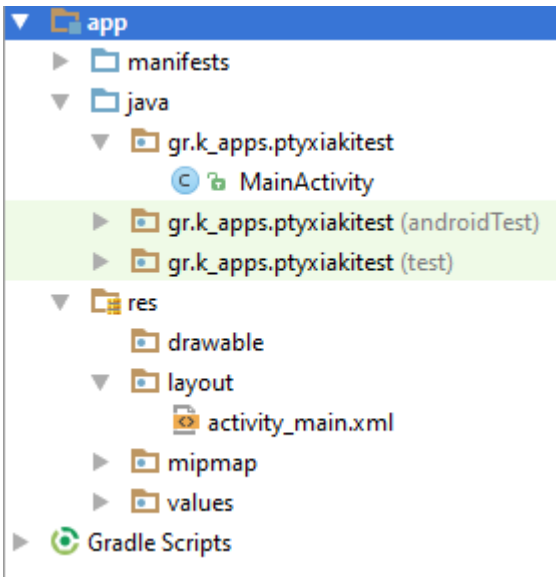
ΚΕΦΑΛΑΙΟ 3
ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ

3.1Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει αναφορά για το πως κατασκευάζουμε εφαρμογές για το λειτουργικό σύστημα Android. Για την ανάπτυξη εφαρμογών απαιτείται η λήψη των απαιτούμενων εργαλείων και το SDK. Τα εργαλεία που απαιτούνται μπορούν να χρησιμοποιηθούν σε οποιοδήποτε λειτουργικό σύστημα (Windows, Mac OS, Linux) και προσφέρονται δωρεάν στο Διαδίκτυο από την Google. Τα παραδείγματα που εκτελούνται σε αυτή την εργασία έχουν δημιουργηθεί σε λειτουργικό σύστημα Windows 8.1. Επιπρόσθετα, το IDE που χρησιμοποιήθηκε είναι το επίσημο IDE που παρέχει η Google για το Android, το Android Studio στην έκδοση 2.0. Τέλος, για την χρήση του Android SDK θα πρέπει να υπάρχει στον υπολογιστή εγκατεστημένο το Java SE Development Kit (JDK). Το πρώτο βήμα για την ανάπτυξη των εφαρμογών είναι η απόκτηση του Integrated development environment (IDE). Το Android Studio IDE είναι το περιβάλλον το οποίο έχει όλα τα εργαλεία που απαιτούνται από το Android και στο οποίο γράφεται ο κώδικας για την κατασκευή των εφαρμογών. Το Android SDK παρέχεται μαζί με το Android Studio και περιέχει ένα πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθήκες, έναν εξομοιωτή, δείγματα κώδικα και παραδείγματα.

3.2 Hello World

Με τη δημιουργία μιας νέας εφαρμογής στο περιβάλλον ανάπτυξης εφαρμογών δημιουργούνται ταυτόχρονα όλα τα αρχεία και οι φάκελοι που είναι απαραίτητα για την εκτέλεση της εφαρμογής. Ωστόσο, εκτός από αρχεία που είναι απαραίτητα για την εκτέλεση, βρίσκονται και ο φάκελος στον οποίο γράφεται ο κώδικας της εφαρμογής, τα αρχεία τα οποία καθορίζεται η έκδοσή του λειτουργικού συστήματος, ο φάκελος που θα τοποθετηθούν τα αρχεία XML για το γραφικό περιβάλλον της εφαρμογής και το αρχείο manifest όπου τοποθετούνται οι άδειες και τα activities που τρέχουν στην εφαρμογή (εικόνα 3.1.1).



Εικόνα 3.1.1: Οι φάκελοι και τα αρχεία τις εφαρμογής.

Το πρώτο βήμα είναι ο σχεδιασμός του γραφικού περιβάλλοντος, η οθόνη δηλαδή που θα βλέπει ο χρήστης καθώς εκτελείται η εφαρμογή. Η σχεδίαση αυτή γίνεται με τη χρήση της τεχνολογίας XML, όπως έχει προαναφερθεί. Ο κώδικας που ακολουθεί δημιουργεί την οθόνη στην οποία θα εμφανίζεται το μήνυμα “Hello World”.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
```

Επόμενο βήμα θα είναι η δημιουργία του Activity. Δηλαδή της java κλάσης που θα εκτελείται.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
   }
}
```

Στον κώδικα παρουσιάζεται ο τρόπος με τον οποίο αρχικοποιούμε την κλάση. Το αρχείο activity_main είναι το XML αρχείο στο οποίο τοποθετήσαμε πριν το γραφικό περιβάλλον. Τέλος, θα πρέπει να ορίσουμε στο αρχείο manifest το Activity όπως παρουσιάζεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="gr.k_apps.ptyxiakitest">

   <application
       android:allowBackup="true"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:supportsRtl="true"
       android:theme="@style/AppTheme">
       <activity android:name=".MainActivity">
       <intent-filter>
       <action android:name="android.intent.action.MAIN" />

  <category android:name="android.intent.category.LAUNCHER" />
       </intent-filter>
       </activity>
   </application>

</manifest>
```

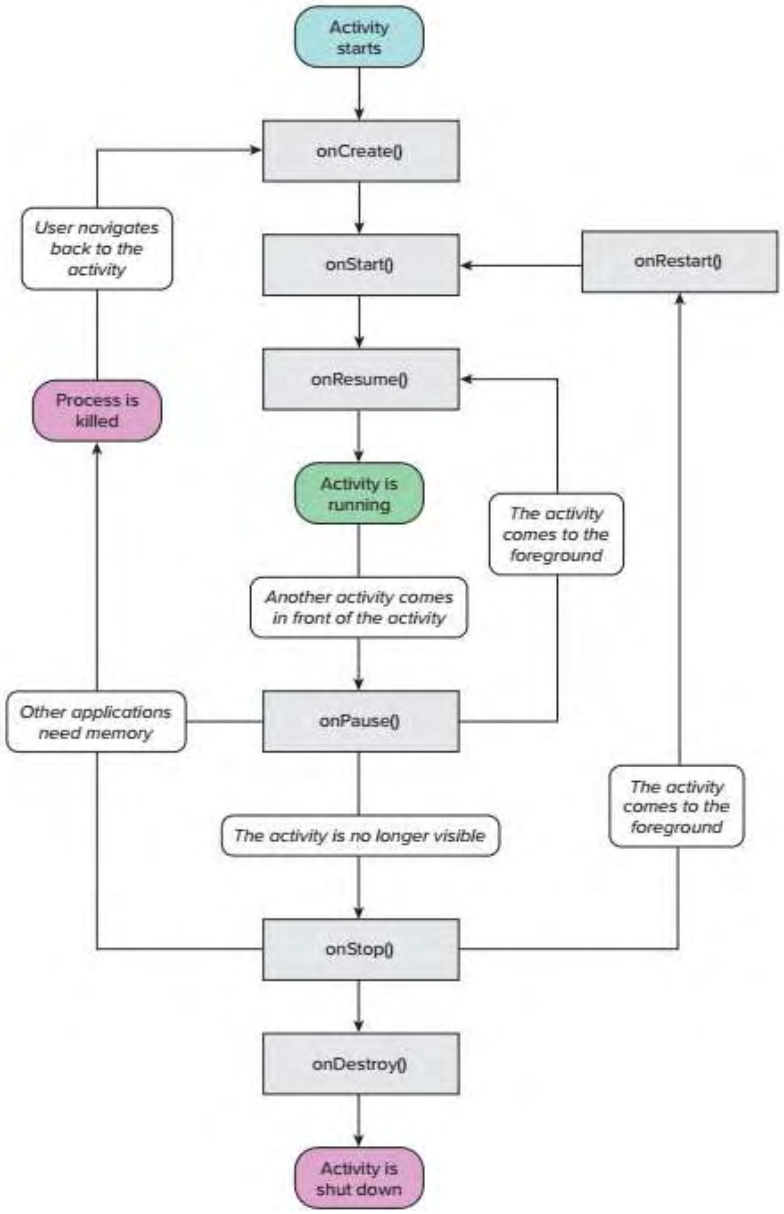
Εκτελώντας το παραπάνω παράδειγμα θα εμφανιστεί στην συσκευή μας η οθόνη της εικόνας 3.1.2. Επίσης, στην συσκευή μας θα εγκατασταθεί η εφαρμογή και θα μπορούμε να την ανοίξουμε οποιαδήποτε στιγμή.

εικόνα 3.1.2: Εκτελώντας την πρώτη εφαρμογή

Όπως αναφέρθηκε και στο Κεφάλαιο 2, η βασική κλάση ορίζει μια σειρά από γεγονότα που διέπουν τον κύκλο ζωής ενός activity. Αυτά τα γεγονότα είναι τα ακόλουθα:

- ➔ onCreate() - Καλείται όταν η δραστηριότητα δημιουργείται πρώτη φορά.
- ➔ onStart() - Καλείται όταν γίνεται ορατή η δραστηριότητα στον χρήστη.
- ➔ onResume() - Καλείται όταν ξεκινά η δραστηριότητα που αλληλεπιδρά με το χρήστη.
- ➔ onPause() - Καλείται όταν η τρέχουσα δραστηριότητα θα διακοπεί για να ξεκινήσει η επόμενη.
- ➔ onStop() - Καλείται όταν η δραστηριότητα δεν είναι πλέον ορατή στον χρήστη.
- ➔ onDestroy() - Καλείται πριν την διακοπή της δραστηριότητας.
- ➔ onRestart() - Καλείται όταν η δραστηριότητα έχει σταματήσει και επανεκκινείται πάλι.

Από προεπιλογή, η δραστηριότητα που δημιουργήθηκε πρώτη περιέχει το συμβάν onCreate() όπως φαίνεται και στο πρώτο παράδειγμα. Το σχήμα της εικόνας 3.1.3 δείχνει τον κύκλο ζωής μιας δραστηριότητας και τα διάφορα στάδια που περνά από τη στιγμή που ξεκινάει να εκτελείται μέχρι τη στιγμή που τελειώνει.



Εικόνα 3.1.3: Κύκλος ζωής μιας δραστηριότητας.

3.2 Intents και Intent Filters

Ένα Intent είναι ένα μήνυμα αντικειμένου που μπορεί να χρησιμοποιηθεί για να ζητηθεί μια ενέργεια από μια άλλη εφαρμογή. Υπάρχουν τρεις θεμελιώδεις τρόποι χρήσης:

- Για να ξεκινήσει μια δραστηριότητα:

Μια δραστηριότητα αντιπροσωπεύει μια ενιαία οθόνη σε μια εφαρμογή. Μπορεί να ξεκινήσει μια νέα δραστηριότητα μέσα από μια άλλη προσθέτοντας το startActivity(). Για να λάβουμε ένα αποτέλεσμα από τη δραστηριότητα, όταν τελειώνει, καλούμε την startActivityForResult(). Η δραστηριότητα λαμβάνει το αποτέλεσμα ως ξεχωριστό αντικείμενο.

- Για να ξεκινήσει μια υπηρεσία:

Μια υπηρεσία είναι ένα στοιχείο που εκτελεί εργασίες στο παρασκήνιο χωρίς περιβάλλον εργασίας χρήστη. Μπορούμε να ξεκινήσουμε μια υπηρεσία για να εκτελέσει μια λειτουργία μια φορά, όπως για παράδειγμα η λήψη ενός αρχείου με την πρόθεση startService(). Εάν η υπηρεσία έχει σχεδιαστεί για μια διεπαφή πελάτη-εξυπηρετητή, μπορεί να γίνει χρήση της πρόθεσης bindService().

- Για παράδοση ενός broadcast:

Ένα broadcast είναι ένα μήνυμα που μπορεί να λάβει οποιαδήποτε εφαρμογή. Το σύστημα παρέχει διάφορα broadcast και συμβάντα συστήματος. Μπορεί να παραδώσει ένα broadcast σε άλλες εφαρμογές περνώντας την πρόθεση sendBroadcast(), sendOrderedBroadcast() ή sendStickyBroadcast().

Τύποι Intent

Υπάρχουν δύο τύποι προθέσεων (Intents)

Explicit intents: Καθορίζει το στοιχείο για να ξεκινήσει με βάση το όνομα. Μπορούμε να κάνουμε χρήση μιας Explicit Intent συνήθως στη δική μας εφαρμογή, γιατί ξέρουμε το όνομα της κλάσης ή της υπηρεσίας η οποία θέλουμε να ξεκινήσει.

Implicit intents: Δεν αναφέρουμε συγκεκριμένο στοιχείο, αλλά αντίθετα, ορίζουμε μια γενική δράση για την εκτέλεση η οποία επιτρέπει σε στοιχεία από άλλη εφαρμογή να το χειριστεί. Για παράδειγμα αν θέλουμε να δείξουμε στον χρήστη ένα στοιχείο στον χάρτη, μπορούμε να κάνουμε χρήση μιας Implicit Intent για να ζητηθεί από άλλη εφαρμογή να δείξει το συγκεκριμένο σημείο.

Παράδειγμα Explicit intent

Μια Explicit intent χρησιμοποιείται για να ξεκινήσει μια ειδική συνιστώσα εφαρμογής, όπως μια συγκεκριμένη δραστηριότητα ή υπηρεσία. Για την δημιουργία μιας Explicit intent καθορίζουμε το όνομα του στοιχείου για την πρόθεση αντικειμένου.
Για παράδειγμα, αν έχουμε δημιουργήσει μια υπηρεσία για την εφαρμογή μας με όνομα DownloadService, η οποία έχει σχεδιαστεί για να κατεβάσει αρχεία από το Διαδίκτυο μπορούμε να ξεκινήσουμε με τον ακόλουθο κώδικα:

```
// Executed in an Activity, so 'this' is the Context

// The fileUrl is a string URL, such as "http://www.example.com/image.png"
```

```
Intent downloadIntent = new Intent(this, DownloadService.class);

downloadIntent.setData(Uri.parse(fileUrl));

startService(downloadIntent);
```

Ο κατασκευαστής Intent (Context, Class) παρέχει το πλαίσιο εφαρμογής και ένα αντικείμενο κλάσης. Ως εκ τούτου, αυτή η πρόθεση ξεκινά ρητά την κλάση DownloadService.

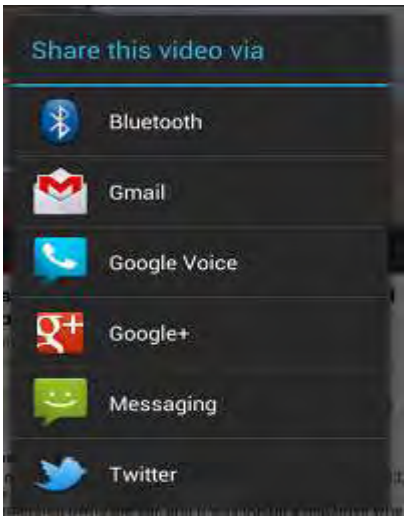
Παράδειγμα Implicit intent

Μια Implicit intent καθορίζει μια ενέργεια που μπορεί να επικαλεστεί οποιαδήποτε εφαρμογή στη συσκευή. Χρησιμοποιώντας μια Implicit intent είναι χρήσιμο για την εφαρμογή όταν δεν μπορεί να εκτελέσει μια ενέργεια να καλεί εφαρμογές που μπορούν να την εκτελέσουν.

Για παράδειγμα αν έχουμε περιεχόμενο και θέλουμε να το μοιραστούμε με άλλους χρήστες θα δημιουργήσουμε μια πρόθεση με τη δράση ACTION_SEND. Όταν καλείται η startActivity() με την εν λόγω πρόθεση, ο χρήστης μπορεί να επιλέξει μια εφαρμογή μέσω της οποίας να μοιραστεί το περιεχόμενο.

```
// Create the text message with a string

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION_SEND);

sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);

sendIntent.setType("text/plain");


// Verify that the intent will resolve to an activity

if (sendIntent.resolveActivity(getPackageManager()) != null) {

    startActivity(sendIntent);

}
```

Όταν καλείται η startActivity το σύστημα εξετάζει όλες τις εγκατεστημένες εφαρμογές για να καθορίσει ποια από αυτές μπορεί να χειριστεί αυτό το είδος των προθέσεων. Εάν υπάρχει μια μόνο εφαρμογή που μπορεί να το χειριστεί, τότε η εφαρμογή ανοίγει αμέσως. Εάν είναι πολλές οι εφαρμογές, τότε το σύστημα εμφανίζει ένα παράθυρο διαλόγου ώστε να επιλέξει ο χρήστης την εφαρμογή που θα εκτελέσει την πρόθεση (Εικόνα 3.2.1).



Εικόνα 3.2.1: Παράθυρο επιλογής εφαρμογής για την εκτέλεση πρόθεσης

3.3 Κουμπιά (Buttons)

Ένα κουμπί αποτελείται από κείμενο, ένα εικονίδιο ή και τα δύο, και επικοινωνεί την δράση του χρήστη με την εφαρμογή.

Ανάλογα με το αν θέλουμε το κουμπί με κείμενο, εικόνα ή και τα δύο μπορούμε να το δημιουργήσουμε με τους ακόλουθους τρόπους:

- Με κείμενο:

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/button_text"
   ... />
```

- Με εικονίδιο:

```
<ImageButton
   android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

- Με κείμενο και εικονίδιο:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:drawableLeft="@drawable/button_icon"
    ... />
```

Απαντώντας σε Click Events

Όταν ο χρήστης πατάει ένα κουμπί το αντικείμενο Button λαμβάνει μια εκδήλωση για κλικ. Για να ορίσουμε το πρόγραμμα χειρισμού συμβάντων κλικ για ένα κουμπί προσθέτουμε το android:onClick στην διάταξη XML. Η τιμή για αυτό το χαρακτηριστικό θα πρέπει να είναι το όνομα της μεθόδου που θα εκτελεστεί στον κώδικα java.

Για παράδειγμα εδώ είναι ένα layout που κάνει χρήση του android:onClick:

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Με την δραστηριότητα του κάνει χρήση της μεθόδου:

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

3.4 Πεδία Κειμένου

Ένα πεδίο κειμένου επιτρέπει στον χρήστη να πληκτρολογήσει κείμενο στην εφαρμογή. Αυτό μπορεί να είναι είτε σε μία είτε σε πολλές γραμμές. Στο Android αγγίζοντας ένα πεδίο κειμένου τοποθετείται ο κέρσορας και εμφανίζεται αυτόματα το πληκτρολόγιο. Επιπλέον, εκτός από το πληκτρολόγιο, τα πεδία κειμένου επιτρέπουν μια ποικιλία από άλλες δραστηριότητες, όπως η επιλογή κειμένου (αποκοπή, αντιγραφή, επικόλληση) και τα δεδομένα look-up μέσω αυτόματης συμπλήρωσης. Για να προστεθεί ένα πεδίο κειμένου πρέπει να γίνει προσθήκη στη διάταξη XML το στοιχείο <EditText>.

3.5 Ειδικοί τύποι πληκτρολογίων

Τα πεδία κειμένου μπορούν να δέχονται διαφορετικού τύπου εισόδους όπως για παράδειγμα έναν αριθμό, μια ημερομηνία, έναν κωδικό πρόσβασης ή μία διεύθυνση ηλεκτρονικού ταχυδρομείου. Ο τύπος καθορίζει το είδος των χαρακτήρων που επιτρέπονται στο εσωτερικό του τομέα και μπορεί να ζητήσει από το εικονικό πληκτρολόγιο για τη βελτιστοποίηση της διάταξης του για συχνά χρησιμοποιούμενους χαρακτήρες.

Μπορούμε να καθορίσουμε τον τύπο του πληκτρολογίου χρηστημοποιώντας το inputType μέσα στο στοιχείο EditText. Για παράδειγμα αν θέλουμε να εισάγουμε μια διεύθυνση ηλεκτρονικού ταχυδρομείου θα πρέπει να χρησιμοποιήσουμε το textEmailAddress.

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:hint="@string/email_hint"

    android:inputType="textEmailAddress" />
```

Υπάρχουν αρκετοί διαφορετικοί τύποι εισόδων για διαφορετικές καταστάσεις.

- Για απλό κείμενο: "text"
- Για διευθύνσεις ηλεκτρονικού ταχυδρομείου: "textEmailAddress"
- Απλό κείμενο με τον χαρακτήρα /: "textUri"
- Αριθμούς: "number"
- Τηλέφωνα: "phone"

3.6 Ελέγχοντας άλλες συμπεριφορές

Το Android: inputType επιτρέπει επίσης να καθορίσουμε ορισμένες συμπεριφορές πληκτρολογίου όπως το αν θα αξιοποιήσει όλες τις νέες λέξεις ή λειτουργίες χρήσης, αυτόματη συμπλήρωση και έλεγχο ορθογραφίας.

Εδώ είναι ορισμένες από αυτούς τους τύπους που καθορίζουν την συμπεριφορά του πληκτρολογίου:

- Απλό πληκτρολόγιο που βάζει στην αρχή της φράσης κεφαλαίο γράμμα: "textCapSentences"
- Απλό πληκτρολόγιο που βάζει όλα τα γράμματα κεφαλαία: "textCapWords"
- Απλό πληκτρολόγιο που διορθώνει τα ορθογραφικά λάθη: "textAutoCorrect"
- Μετατρέπει τους χαρακτήρες σε τελείες: "textPassword"
- Επιτρέπει την αλλαγή γραμμής σε μεγάλα κείμενα: "textMultiLine"

Ακολουθεί ένα παράδειγμα χρήσης των παραπάνω:

```
<EditText

    android:id="@+id/postal_address"

    android:layout_width="fill_parent"

    android:layout_height="wrap_content"

    android:hint="@string/postal_address_hint"

    android:inputType="textPostalAddress|
```

```
                textCapWords|

                textNoSuggestions" />
```

Απαντώντας σε δράση από γεγονός κουμπιού

Μπορούμε να καθορίσουμε την ενέργεια που πρέπει να γίνει όταν οι χρήστες έχουν ολοκληρώσει τη συμβολή τους. Η δράση καθορίζει τα μέτρα που πρέπει να γίνουν όπως αποστολή ή αναζήτηση.

Μπορούμε να καθορίσουμε τη δράση θέτοντας το Android:imeOptions. Στο παράδειγμα που ακολουθεί καθορίζεται η δράση αποστολή:

```
<EditText
   android:id="@+id/search"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:hint="@string/search_hint"
   android:inputType="text"
   android:imeOptions="actionSend" />
```

Για να "ακούσει" η κλάση μας για τη δράση αυτή θα πρέπει να γίνει χρήση της TextView.OnEditorActionListener. Αυτή η διεπαφή παρέχει μια μέθοδο επανάκλησης που ονομάζεται onEditAction() που δηλώνει το είδος δράσης όπως για παράδειγμα IME_ACTION_SEND ή IME_ACTION_SEARCH.
Εδώ παρουσιάζεται ο κώδικας για το πως μπορεί να ακούσει όταν ο χρήστης κάνει κλικ στο κουμπί αποστολή στο πληκτρολόγιο:

```
EditText editText = (EditText) findViewById(R.id.search);
editText.setOnEditorActionListener(new OnEditorActionListener() {
   @Override
```

```
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEND) {
            sendMessage();
            handled = true;
        }
        return handled;
    }
});
```

3.7 Η βιβλιοθήκη Volley

Η βιβλιοθήκη Volley είναι μία HTTP βιβλιοθήκη που κάνει τις δικτυακές κλήσεις ευκολότερες αλλά κυρίως πιο γρήγορες.

Η βιβλιοθήκη Volley προσφέρει τα ακόλουθα πλεονεκτήματα:

- Αυτόματος προγραμματισμός των αιτήσεων δικτύου.
- Πολλαπλές ταυτόχρονες συνδέσεις δικτύου.
- Δυνατότητα προσωρινής αποθήκευσης με το πρότυπο HTTP cache.
- Ευκολία προσαρμογής.
- Αποσφαλμάτωση και εργαλεία εντοπισμού λάθους.
- Υποστηρίζει ιεραρχημένα αιτήματα.

Η βιβλιοθήκη Volley δεν εξυπηρετεί μεγάλες λήψεις και ροές εργασιών. Για αυτούς τους σκοπούς υπάρχει η εναλλακτική χρησιμοποίησης του DownloadManager.

Στέλνοντας ένα απλό ερώτημα

Σε υψηλό επίπεδο μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη δημιουργώντας ένα RequestQueue για να περάσουμε το αντικείμενο. Η RequestQueue διαχειρίζεται θέματα των εργασιών του δικτύου, την ανάγνωση,όπως και την εγγραφή στη μνήμη cache.

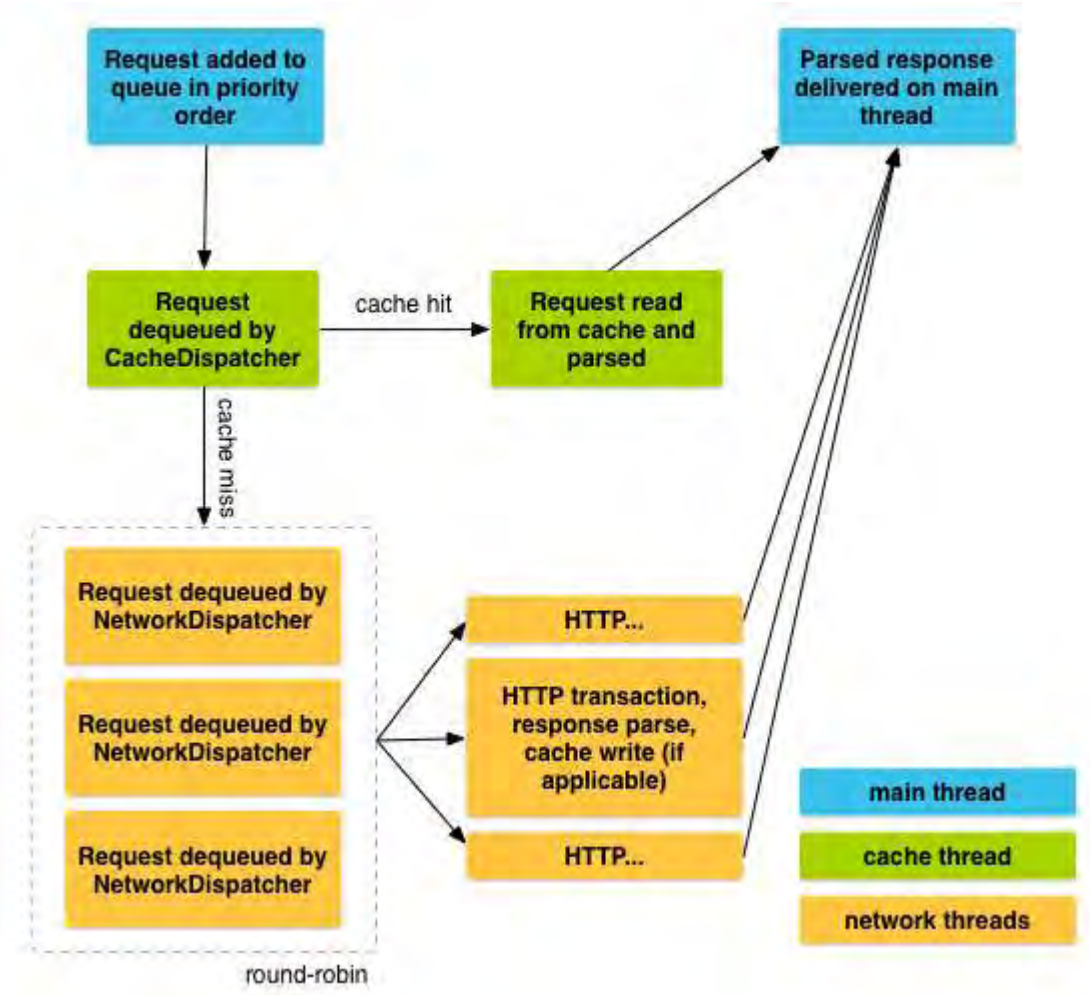
Για να κάνουμε χρήση της βιβλιοθήκης θα πρέπει να προσθέσουμε στο αρχείο manifest τα δικαιώματα για τη χρήση του δικτύου. Αυτό το κάνουμε προσθέτοντας την γραμμή android.permission.INTERNET.

Ο τρόπος που η βιβλιοθήκη δημιουργεί μια νέα μέθοδο παρουσιάζεται στον κώδικα που ακολουθεί:

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...
// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";
// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        mTextView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
        public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Για να στείλουμε ένα αίτημα, μπορούμε απλά να κατασκευάσουμε ένα και να το προσθέσουμε στο RequestQueue με add (), όπως φαίνεται στον παραπάνω κώδικα.

Στο σχήμα που ακολουθεί απεικονίζεται η ζωή μιας αίτησης:



ΚΕΦΑΛΑΙΟ 4
ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ (Client)

4.1 Εισαγωγή

Η εφαρμογή που δημιουργήθηκε αποτελεί μια εκπαιδευτική εφαρμογή στην οποία θα οργανώνονται ημερολογιακά οι εργασίες των μαθητών. Η εφαρμογή αναπτύχθηκε με τη χρήση του Android Studio (έκδοση 2.0.0) και είναι εξολοκλήρου γραμμένη σε γλώσσα προγραμματισμού java, ενώ το γραφικό περιβάλλον είναι σχεδιασμένο σε XML. Τέλος, ο κώδικας στην πλευρά του server αναπτύχθηκε με php και χρησιμοποιήθηκε βάση MySQL. Σε αυτή την ενότητα αρχικά θα γίνει μια παρουσίαση της εφαρμογής στην πλευρά του client και στην συνέχεια θα αναλυθούν τα σημαντικότερα κομμάτια του κώδικα της εφαρμογής.

4.2 Περιγραφή εφαρμογής

Η εφαρμογή που αναπτύχθηκε είναι μια εφαρμογή η οποία θα χρησιμοποιείται από εκπαιδευτικούς για την οργάνωση και διαχείριση εργασιών.

Ο κάθε εκπαιδευτικός πριν ξεκινήσει την καταχώρηση των εργασιών θα πρέπει να πραγματοποιήσει την εγγραφή του, να δημιουργήσει δηλαδή λογαριασμό. Τα στοιχεία που απαιτούνται για την εγγραφή του εκπαιδευτικού είναι το όνομά του, το email του και ένας κωδικός πρόσβασης (Εικόνα 4.2.1). Έτσι ο εκπαιδευτικός έχει τη δυνατότητα να επιλέγει ημερομηνία παράδοσής της εργασίας, να δίνει τον τίτλο της εργασίας, την διάρκεια παράδοσής, να επιλέγει αν η εργασία είναι ατομική, συλλογική ή σε ομάδες και τέλος, να επιλέγει τον τύπο της εργασίας. Οι τύποι εργασίας χωρίζονται σε αυθόρμητες, σχέδιο, γλώσσα, περιβάλλον, μαθηματικά, υπολογιστές και άλλα .

Εικόνα 4.1.2

Οι καταχωρήσεις των εκπαιδευτικών αποθηκεύονται σε βάση δεδομένων και έτσι ο εκπαιδευτικός μπορεί να χρησιμοποιήσει την εφαρμογή από οποιαδήποτε συσκευή Android επιθυμεί, αρκεί να έχει εγκαταστήσει την εφαρμογή σε αυτή τη συσκευή. Τέλος, ο εκπαιδευτικός μπορεί να επεξεργαστεί τις καταχωρήσεις του και να τις αποθηκεύσει

ανανεωμένες. Όλες οι καταχωρήσεις παρουσιάζονται σε λίστα μέσα από την εφαρμογή (Εικόνα 4.1.3).

Εικόνα 4.1.3

4.2 Κατασκευή της εγγραφής και εισόδου στο σύστημα

Το πρώτο πράγμα που απαιτείται για την χρησιμοποίηση της εφαρμογής είναι η εγγραφή του εκπαιδευτικού στο σύστημα. Σε αυτή την υποενότητα θα παρουσιαστεί αναλυτικά ο κώδικας για τον σχεδιασμό αυτών των λειτουργιών.

Αρχικά σχεδιάστηκε σε XML το γραφικό περιβάλλον όπου ο χρήστης-εκπαιδευτικός θα κάνει εγγραφή. Τα στοιχεία που χρειάζονται είναι το ονοματεπώνυμο του χρήστη, το email του και ο κωδικός πρόσβασης. Παρακάτω παρουσιάζεται ο τρόπος που δημιουργήθηκαν αυτά με την τεχνολογία XML.

```
<EditText
   android:id="@+id/name"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="10dp"
   android:hint="Ονοματεπώνυμο"
   android:padding="10dp"
   android:singleLine="true"
   android:inputType="textCapWords" />

<EditText
   android:id="@+id/email"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="10dp"
   android:hint="Email"
   android:inputType="textEmailAddress"
   android:padding="10dp"
   android:singleLine="true" />

<EditText
   android:id="@+id/password"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="10dp"
   android:hint="Κωδικός"
   android:inputType="textPassword"
   android:padding="10dp"
   android:singleLine="true" />
```

Τα τρία EditTexts αντιστοιχούν στο όνομα, το email και τον κωδικό του χρήστη.

Όταν ο χρήστης πληκτρολογήσει τα απαραίτητα για την εγγραφή του στοιχεία και πατήσει το κουμπί για την εγγραφή του, τότε εκτελείται ένα event και καλείται η συνάρτηση η οποία θα στείλει τα δεδομένα στην βάση δεδομένων.

```
// Register Button Click event
btnRegister.setOnClickListener((view) → {
        String name = inputFullName.getText().toString().trim();
        String email = inputEmail.getText().toString().trim();
        String password = inputPassword.getText().toString().trim();

        if (!name.isEmpty() && !email.isEmpty() && !password.isEmpty()) {
            registerUser(name, email, password);
        } else {
            Toast.makeText(getApplicationContext(),
                    "Please enter your details!", Toast.LENGTH_LONG)
                    .show();
        }
});
```

Αρχικά με την χρήση της getText τοποθετεί τους χαρακτήρες που πληκτρολόγησε ο χρήστης σε String μεταβλητές. Στην συνέχεια γίνεται έλεγχος για το αν δεν έχει συμπληρωθεί κάποιο από τα στοιχεία και καλεί την συνάρτηση registerUser όπου θα γίνει η καταχώρηση των στοιχείων στη βάση.

Όταν καλείται η συνάρτηση registerUser, η διαδικασία που εκτελείται είναι η αποστολή των δεδομένων, όπως έχει ήδη αναφερθεί και παρουσιάζεται στον κώδικα που ακολουθεί:

```
    protected Map<String, String> getParams() {
        // Posting params to register url
        Map<String, String> params = new HashMap<~>();
        params.put("name", name);
        params.put("email", email);
        params.put("password", password);

        return params;
    }

};

// Adding request to request queue
AppController.getInstance().addToRequestQueue(strReq, tag_string_req);
```

Παράλληλα όμως, θα δεχθεί και μία απάντηση ότι αποθηκεύτηκαν τα δεδομένα στη βάση, και θα μας επιστρέψει αυτά τα δεδομένα. Αυτά τα δεδομένα θα αποθηκευτούν και σε μια τοπική βάση SQLite. Ο λόγος είναι να παραμένει ο χρήστης συνδεδεμένος και να μην χρειάζεται μετά τον τερματισμό της εφαρμογής να κάνει νέα σύνδεση.
Ο κώδικας που ακολουθεί δείχνει την διαδικασία που επιστρέφει τα αποτελέσματα και αποθηκεύονται στην τοπική βάση της εφαρμογής.

```
    @Override
    public void onResponse(String response) {
        Log.d(TAG, "Register Response: " + response.toString());
        hideDialog();

        try {
            JSONObject jObj = new JSONObject(response);
            boolean error = jObj.getBoolean("error");
            if (!error) {
                // User successfully stored in MySQL
                // Now store the user in sqlite
                String uid = jObj.getString("uid");

                JSONObject user = jObj.getJSONObject("user");
                String name = user.getString("name");
                String email = user.getString("email");
                String created_at = user
                        .getString("created_at");

                // Inserting row in users table
                db.addUser(name, email, uid, created_at);

                Toast.makeText(getApplicationContext(),
                        "User successfully registered. Try login now!",
                        Toast.LENGTH_LONG).show();

                // Launch login activity
                Intent intent = new Intent(
                        RegisterActivity.this,
                        LoginActivity.class);
                startActivity(intent);
                finish();
            } else {

                // Error occurred in registration. Get the error
                // message
                String errorMsg = jObj.getString("error_msg");
                Toast.makeText(getApplicationContext(),
                        errorMsg, Toast.LENGTH_LONG).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }

    }
}, new Response.ErrorListener() {
```

Παρόμοια διαδικασία πραγματοποιείται και για την δημιουργία της εισόδου του χρήστη στην εφαρμογή. Για την πραγματοποίηση των HTTP κλήσεων δημιουργήθηκε μία κλάση η οποία εκτελεί αυτές τις διαδικασίες. Παρακάτω παρουσιάζεται ο κώδικας αυτής της κλάσης:

```
import ...

public class AppController extends Application {

    public static final String TAG = AppController.class.getSimpleName();

    private RequestQueue mRequestQueue;

    private static AppController mInstance;

    @Override
    public void onCreate() {
        super.onCreate();
        mInstance = this;
    }

    public static synchronized AppController getInstance() { return mInstance; }

    public RequestQueue getRequestQueue() {
        if (mRequestQueue == null) {
            mRequestQueue = Volley.newRequestQueue(getApplicationContext());
        }

        return mRequestQueue;
    }

    public <T> void addToRequestQueue(Request<T> req, String tag) {
        req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
        getRequestQueue().add(req);
    }

    public <T> void addToRequestQueue(Request<T> req) {
        req.setTag(TAG);
        getRequestQueue().add(req);
    }

    public void cancelPendingRequests(Object tag) {
        if (mRequestQueue != null) {
            mRequestQueue.cancelAll(tag);
        }
    }
}
```

4.3 Τοπική βάση δεδομένων στην εφαρμογή

Όπως έχει ήδη αναφερθεί, για να παραμένει ο χρήστης συνδεδεμένος στην εφαρμογή και να μην χρειάζεται να κάνει κάθε φορά νέα σύνδεση, χρησιμοποιήθηκε μια τοπική βάση μέσα στην εφαρμογή.

Με τον κώδικα που ακολουθεί δημιουργείται ο πίνακας της τοπικής βάσης:

```
// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_LOGIN_TABLE = "CREATE TABLE " + TABLE_USER + "("
            + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
            + KEY_EMAIL + " TEXT UNIQUE," + KEY_UID + " TEXT,"
            + KEY_CREATED_AT + " TEXT" + ")";
    db.execSQL(CREATE_LOGIN_TABLE);

    Log.d(TAG, "Database tables created");
```

Επόμενο βήμα είναι η καταχώρηση του χρήστη στην βάση δεδομένων:

```
public void addUser(String name, String email, String uid, String created_at) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, name); // Name
    values.put(KEY_EMAIL, email); // Email
    values.put(KEY_UID, uid); // Email
    values.put(KEY_CREATED_AT, created_at); // Created At

    // Inserting Row
    long id = db.insert(TABLE_USER, null, values);
    db.close(); // Closing database connection

    Log.d(TAG, "New user inserted into sqlite: " + id);
}
```

Τέλος, κάθε φορά που ο χρήστης κάνει αποσύνδεση ταυτοχρόνως διαγράφεται και από την τοπική βάση:

```
public void deleteUsers() {
    SQLiteDatabase db = this.getWritableDatabase();
    // Delete All Rows
    db.delete(TABLE_USER, null, null);
    db.close();

    Log.d(TAG, "Deleted all user info from sqlite");
}
```

4.4 Αποστολή δεδομένων

Για την εισαγωγή των στοιχείων στον server χρησιμοποιήθηκε ξανά η βιβλιοθήκη Volley. Αφού ο χρήστης επιλέξει ημερομηνία και γράψει στα απαιτούμενα πεδία τα δεδομένα ,μπορεί να αποθηκεύσει τα δεδομένα του στον server πατώντας το κουμπί αποθήκευση. Με το πάτημα του κουμπιού καλείται η συνάρτηση insert() η οποία θα στείλει τα δεδομένα αλλά και θα της επιστραφεί μια απάντηση από τον server για το αν τα στοιχεία αποθηκεύτηκαν ή όχι.

Στον κώδικα η πραγματοποίηση αυτής της διαδικασίας απαιτεί πρώτα το πέρασμα των δεδομένων που έγραψε ο χρήστης σε μεταβλητές String:

```
item_date = item_etDate.getText().toString();
item_title = item_etTitle.getText().toString();
item_duration = item_etDuration.getText().toString();
item_project = item_etProject.getText().toString();
type = item_tvType.getText().toString();
item_name = item_etName.getText().toString();
```

Και στη συνέχεια την αποστολή αυτών των παραμέτρων:

```
protected Map<String, String> getParams() {
    Map<String, String> params = new HashMap<~>();
    params.put("item_date", item_date);
    params.put("item_title", item_title);
    params.put("item_duration", item_duration);
    params.put("item_project", item_project);
    params.put("item_name", item_name);
    params.put("type", type);

    return params;
}
```

Για την αποστολή των δεδομένων χρησιμοποιείται η μέθοδος POST:

```
StringRequest postRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
```

Η διαδικασία της αποστολής ολοκληρώνεται όταν γίνει λήψη της απάντησης για επιτυχημένη αποθήκευση. Όταν η απάντηση είναι επιτυχημένη η εφαρμογή θα εμφανίσει μήνυμα επιτυχημένης αποθήκευσης και θα καθαρίσει τα πεδία κειμένου:

```
new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        Log.d(TAG, "Insert Response: " + response.toString());
        PD.dismiss();
        //item_etDate.setText("");
        item_etTitle.setText("");
        item_etDuration.setText("");
        item_etProject.setText("");
        //item_etName.setText("");
        Toast.makeText(getApplicationContext(),
                "Αποθηκεύτηκε με επιτυχία",
                Toast.LENGTH_SHORT).show();

    }
}, new Response.ErrorListener() {
```

4.4 Εμφάνιση των αποθηκευμένων δεδομένων

Οι χρήστες της εφαρμογής μπορούν να δουν τα αποθηκευμένα αποτελέσματα σε λίστα με τις βασικές πληροφορίες όπως το όνομα της εργασίας, την ημερομηνία και τον τύπος της εργασίας. Όταν ο χρήστης επιλέγει να δει όλα τα αποθηκευμένα δεδομένα τότε καλείται η συνάρτηση ReadDataFromDB().
Η διαδικασία που εκτελείται σε αυτή τη συνάρτηση είναι να επιστρέφει με τη μέθοδο GET τα δεδομένα τα οποία στη συνέχεια τοποθετούνται σε πεδία κειμένου ώστε να εμφανιστούν στον χρήστη.

```
try {
    int success = response.getInt("success");

    if (success == 1) {
        JSONArray ja = response.getJSONArray("orders");

        for (int i = 0; i < ja.length(); i++) {

            JSONObject jobj = ja.getJSONObject(i);
            HashMap<String, String> item = new HashMap<~>();
            item.put(ITEM_ID, jobj.getString(ITEM_ID));
            item.put(ITEM_DATE, jobj.getString(ITEM_DATE));
            item.put(ITEM_TITLE, jobj.getString(ITEM_TITLE));
            item.put(ITEM_TYPE, jobj.getString(ITEM_TYPE));
            item.put(ITEM_DURATION, jobj.getString(ITEM_DURATION));
            item.put(ITEM_PROJECT, jobj.getString(ITEM_PROJECT));
            item.put(ITEM_NAME, jobj.getString(ITEM_NAME));


            Item_List.add(item);

        } // for loop ends

        String[] from = { ITEM_DATE, ITEM_TITLE, ITEM_TYPE };
        int[] to = { R.id.item_date, R.id.item_title, R.id.type };

        adapter = new SimpleAdapter(
                getApplicationContext(), Item_List,
                R.layout.list_items, from, to);

        setListAdapter(adapter);
```

Σε περίπτωση που η βάση δεδομένων είναι άδεια τότε καλείται ένα intent το οποίο μας επιστρέφει στο Activity της προηγούμενης κλάσης:

```
    } // if ends
    //if data base is empty
    if (success == 0){
        Intent i = new Intent (ReadActivity.this, MainActivity.class);
        startActivity(i);
    }
```

4.5 Επεξεργασία και διαγραφή

Τέλος, ο εκπαιδευτικός έχει τη δυνατότητα να επεξεργαστεί ή να διαγράψει τις καταχωρήσεις που έχει κάνει. Η επεξεργασία μπορεί να γίνει στα δεδομένα της ημερομηνίας, του τίτλου της εργασίας, τη διάρκειά της καθώς και τον τύπο της.

Κατά τη διαδικασία της επεξεργασίας των στοιχείων στέλνονται οι αλλαγές που έγιναν στα πεδία κειμένων,

```
String update_url = "http://k-apps.gr/android/update_item.php";

Map<String, String> params = new HashMap<~>();
params.put("id", id);
params.put("item_date", date);
params.put("item_title",title);
params.put("item_duration", duration);
params.put("item_project", project);
params.put("item_name", name);

CustomRequest update_request = new CustomRequest(update_url,
        params, new Response.Listener<JSONObject>() {
```

και στη συνέχεια η εφαρμογή αναμένει απάντηση από τον διακομιστή. Αν η απάντηση που θα λάβει είναι θετική τότε εμφανίζει μήνυμα επιτυχημένης αποθήκευσης, διαφορετικά εμφανίζει μήνυμα αποτυχίας. Η παραπάνω περιγραφή επιτυγχάνεται με τον ακόλουθο κώδικα:

```
@Override
public void onResponse(JSONObject response) {

    try {
        int success = response.getInt("success");

        if (success == 1) {
            PD.dismiss();
            Toast.makeText(getApplicationContext(),
                    "Αποθηκεύτηκε με επιτυχία",
                    Toast.LENGTH_SHORT).show();
            // redirect to readdata
            MoveToReadData();

        } else {
            PD.dismiss();
            Toast.makeText(getApplicationContext(),
                    "Σφάλμα κατα την αποθήκευση, προσπαθήστε ξανά", Toast.LENGTH_SHORT)
                    .show();
        }
```

Στη διαγραφή η διαδικασία είναι πιο απλή καθώς γίνεται διαγραφή όλων των στοιχείων που αντιστοιχούν στα δεδομένα που βλέπει ο χρήστης.

```
String delete_url = "http://k-apps.gr/android/delete_item.php?id="
        + id;

JsonObjectRequest delete_request = new JsonObjectRequest(delete_url,
        null, new Response.Listener<JSONObject>() {
```

Όπως και στις προηγούμενες περιπτώσεις έτσι και εδώ η εφαρμογή περιμένει απάντηση από την πλευρά του διακομιστή ότι έγινε η διαγραφή των δεδομένων και εμφανίζει αντίστοιχο μήνυμα.

```
@Override
public void onResponse(JSONObject response) {

    try {
        int success = response.getInt("success");

        if (success == 1) {
            PD.dismiss();
            Toast.makeText(getApplicationContext(),
                    "Επιτυχής Διαγραφή",
                    Toast.LENGTH_SHORT).show();
            // redirect to readdata
            MoveToReadData();
        } else {
            PD.dismiss();
            Toast.makeText(getApplicationContext(),
                    "Σφάλμα κατά τη διαγραφή", Toast.LENGTH_SHORT)
                    .show();
        }
```

ΚΕΦΑΛΑΙΟ 5
ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ (Server)

5.1 Εισαγωγή

Η εφαρμογή όπως έχει προαναφερθεί επιτρέπει την διαχείριση εργασιών και την αποθήκευση αυτών σε βάση δεδομένων. Για να επιτευχθεί η επικοινωνία μεταξύ της εφαρμογής και της βάσης που γίνεται η αποθήκευση των δεδομένων αναπτύχθηκε κώδικας στην πλευρά του server. Σε αυτό το κεφάλαιο θα αναλυθεί ο κώδικας που αναπτύχθηκε για την υλοποίηση της εφαρμογής στην πλευρά του server. Στη βάση δεδομένων έχουν δημιουργηθεί δύο tables. Το πρώτο εξυπηρετεί την αποθήκευση των χρηστών που κάνουν την εγγραφή, ενώ το δεύτερο αποθηκεύει τις εργασίες, την ημερομηνία και τις υπόλοιπες πληροφορίες.

5.2 Εγγραφή των χρηστών

Για να επιτραπεί στο χρήστη να κάνει χρήση της εφαρμογής θα πρέπει πρώτα να κάνει εγγραφή. Στο προηγούμενο κεφάλαιο είδαμε τη διαδικασία που ακολουθείται στον κώδικα από πλευράς client. Σε αυτό το κεφάλαιο θα δούμε τι γίνεται στην πλευρά του server κατά τη διάρκεια της εγγραφής των χρηστών.

Αρχικά με τη μέθοδο POST παίρνει ο server τις παραμέτρους που στέλνει ο χρήστης όπως παρουσιάζεται στον κώδικα που ακολουθεί.

```
if (isset($_POST['name']) && isset($_POST['email']) && isset($_POST['password'])) {

    // receiving the post params
    $name = $_POST['name'];
    $email = $_POST['email'];
    $password = $_POST['password'];
```

Στη συνέχεια γίνεται έλεγχος για το αν υπάρχει ήδη ο χρήστης που προσπάθησε να κάνει νέα εγγραφή. Αν ο χρήστης υπάρχει ήδη καταχωρημένος στη βάση, η εφαρμογή επιστρέφει μήνυμα σφάλματος, διαφορετικά γίνεται η εγγραφή του χρήστη.

```
// check if user is already existed with the same email
if (isUserExisted($email)) {
    // user already existed
    $response["error"] = TRUE;
    $response["error_msg"] = "User already existed with " . $email;
    echo json_encode($response);
} else {
    // create a new user
    $user = storeUser($name, $email, $password);
    if ($user) {
        // user stored successfully
        $response["error"] = FALSE;
        $response["uid"] = $user["unique_id"];
        $response["user"]["name"] = $user["name"];
        $response["user"]["email"] = $user["email"];
        $response["user"]["created_at"] = $user["created_at"];
        $response["user"]["updated_at"] = $user["updated_at"];
        echo json_encode($response);
```

Η συνάρτηση που κάνει την καταχώρηση του χρήστη:

```php
/* FUNCTIONS SECTION */
   function isUserExisted($email) {
               $email = mysql_real_escape_string($email);
               $sql = "SELECT id from users WHERE email = '$email'";
               $result = mysql_query($sql);
               if (mysql_num_rows($result) > 0)
           return true;
               else
           return false;
   }

   function storeUser($name, $email, $password) {
               $email = mysql_real_escape_string($email);
               $name = mysql_real_escape_string($name);
               $password = mysql_real_escape_string($password);

       $uuid = uniqid('', true);
       $hash = hashSSHA($password);
       $encrypted_password = $hash["encrypted"]; // encrypted password
       $salt = $hash["salt"]; // salt
               $sql = sprintf("INSERT INTO users(unique_id, name, email, encrypted_password, salt, created_at)
                               $uuid,
                               mysql_real_escape_string($name),
                               mysql_real_escape_string($email),
                               $encrypted_password,
                               $salt);
               $result = mysql_query($sql);
```

5.3 Σύνδεση του χρήστη

Παρόμοια διαδικασία εκτελείται και κατά τη διάρκεια σύνδεσης του χρήστη στην εφαρμογή. Ο χρήστης στέλνει το email και τον κωδικό του και αν υπάρχει εγγεγραμμένος με αυτά τα στοιχεία του επιτρέπεται η είσοδος στην εφαρμογή.

```php
// json response array
$response = array("error" => FALSE);

if (isset($_POST['email']) && isset($_POST['password'])) {

    // receiving the post params
    $email = $_POST['email'];
    $password = $_POST['password'];

    // get the user by email and password
    $user = getUserByEmailAndPassword($email, $password);
```

```
    if ($user != false) {
        // use is found
        $response["error"] = FALSE;
        $response["uid"] = $user["unique_id"];
        $response["user"]["name"] = $user["name"];
        $response["user"]["email"] = $user["email"];
        $response["user"]["created_at"] = $user["created_at"];
        $response["user"]["updated_at"] = $user["updated_at"];
        echo json_encode($response);
    } else {
        // user is not found with the credentials
        $response["error"] = TRUE;
        $response["error_msg"] = "Login credentials are wrong. Please try again!";
        echo json_encode($response);
    }
} else {
    // required post params is missing
    $response["error"] = TRUE;
    $response["error_msg"] = "Required parameters email or password is missing!";
    echo json_encode($response);
}
```

5.4 Εισαγωγή πληροφοριών στη βάση δεδομένων

Αφού οι χρήστες εγγραφούν και συνδεθούν στην εφαρμογή, επόμενο βήμα είναι να καταχωρήσουν τις πληροφορίες των εργασιών στη βάση. Οι πληροφορίες που καταχωρούνται είναι η ημερομηνία, ο τίτλος της εργασίας, η διάρκεια εκτέλεσης από τους μαθητές και το είδος της εργασίας. Ο κώδικας που ακολουθεί αποθηκεύει αυτά τα στοιχεία στη βάση.

```
<?php
error_reporting(0);
include("db_config.php");

// array for JSON response
$response = array();

if( !(empty($_POST['item_date'])))
{
    $item_date=$_POST['item_date'];
  $item_title=$_POST['item_title'];
  $item_duration=$_POST['item_duration'];
  $item_project=$_POST['item_project'];

  $type=$_POST['type'];
  $item_name=$_POST['item_name'];
```

```
    $result = mysql_query("INSERT INTO myorder(id,item_date, item_title,
     item_duration,
     item_project, item_type, item_name) VALUES('','$item_date','$item_title',
     '$item_duration',
     '$item_project','$type', '$item_name')");

    if($result>0){
          $response["success"] = 1;
        }
     else{
          $response["success"] = 0;
        }
     // echoing JSON response
     echo json_encode($response);
}

?>
```

5.5 Διαγραφή Στοιχείων

Ο χρήστης μπορεί να διαγράψει τα στοιχεία του. Εδώ θα δούμε τον κώδικα που εκτελείται στην πλευρά του server για την διαγραφή των στοιχείων.

```
<?php
 error_reporting(0);
 include("db_config.php");

// array for JSON response
$response = array();
if( isset($_GET['id'] ) ) {

    $id=$_GET['id'];
    $item=$_GET['item_title'];

    $result = mysql_query("delete from myorder where id='$id' ");

    $row_count = mysql_affected_rows();

    if($row_count>0){
        $response["success"] = 1;
        $response["message"] = "Deleted Sucessfully.";
       }
    else{
        $response["success"] = 0;
        $response["message"] = "Failed To Delete";
     }
  // echoing JSON response
  echo json_encode($response);

 }
?>
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Elgin, Ben (August 17, 2005). "Google Buys Android for Its Mobile Arsenal".*Bloomberg Businessweek*. Bloomberg. Archived from the original on February 24, 2011. Retrieved February 20, 2012. In what could be a key move in its nascent wireless strategy, Google (GOOG) has quietly acquired startup Android.

[2] Brodkin, Jon (November 5, 2012). "On its 5th birthday, 5 things we love about Android"

[3] "Industry Leaders Announce Open Platform for Mobile Devices" (Press release).Open Handset Alliance. November 5, 2007. Retrieved February 17, 2012

[4] Markoff, John (November 4, 2007). "I, Robot: The Man Behind the Google Phone".*The New York Times*. Retrieved February 15, 2012.

[5] Kirsner, Scott (September 2, 2007). "Introducing the Google Phone". *The Boston Globe*. Archived from the original on January 4, 2010. Retrieved February 15, 2012

[6] Chris Welch (April 16, 2013). "Before it took over smartphones, Android was originally destined for cameras". The Verge. Retrieved May 1, 2013.

[7] Richard Wray (March 14, 2010). "Google forced to delay British launch of Nexus phone". London: guardian.co.uk. Retrieved February 17, 2012.

[8] "Tools Overview". *Android Developers*. July 21, 2009.

[9] Stone, Brad (April 27, 2010). "Google's Andy Rubin on Everything Android". *New York Times*. Archived from the original on April 30, 2010. Retrieved May 20, 2010.