Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος : Οδηγός χώρων με τη χρήση συστημάτων εντοπισμού θέσης

Μαρκαντωνάκη Ευαγγελία (Α.Μ:3310)

Επιβλέπων καθηγητής : Μαλάμος Αθανάσιος

Επιτροπή Αξιολόγησης : Μαλάμος Αθανάσιος

Παπαδάκης Νικόλαος

Παχουλάκης Ιωάννης

Ημερομηνία Παρουσίασης : 8/7/2015

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής εργασίας , θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή **Κ. Μαλάμο Αθανάσιο** για την καθοδήγηση και την επίβλεψη της εργασίας μου. Καθώς και ένα μεγάλο ευχαριστώ στην **οικογένεια μου** για την υποστήριξη τους καθ' όλη την διάρκεια των σπουδών μου.

Abstract

Often there is a need to describe and browse interior. This project enables the user to scan objects that exist in space area using QR_Codes. As the user done with the scanning, the system indicates to the user the shortest route between objects and himself, through space distance calculation and implementation of the corresponding algorithm.

Σύνοψη

Συχνά παρουσιάζεται η ανάγκη για περιγραφή και περιήγηση εσωτερικού χώρου. Η παρούσα πτυχιακή εργασία δίνει την δυνατότητα στον χρήστη να σαρώνει αντικείμενα που υπάρχουν στον χώρο με την χρήση QR_codes. Κατά την ολοκλήρωση της σάρωσης το σύστημα υποδεικνύει στον χρήστη την συντομότερη διαδρομή μεταξύ των αντικειμένων και τον ίδιο μέσω υπολογισμών αποστάσεων χώρων και υλοποίησης του αντίστοιχου αλγορίθμου.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

1 Εισαγωγή

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Είναι συχνά δύσκολο έως ακατόρθωτο σε κλειστούς χώρους η πλοήγηση και ο εντοπισμός της θέσης χωρίς τη χρήση GPS. Με το ίδιο σκεπτικό είναι πιο δύσκολος ο εντοπισμός της θέσης αντικειμένων και του χρήστη ταυτόχρονα. Αυτός ήταν λόγος που δημιουργήθηκε η επιθυμία μου να υλοποιήσω ένα σύστημα όπου θα μπορεί να βρει την θέση του χρήστη, μέσα σε κλειστό χώρο με την βοήθεια σκαναρίσματος qr_codes.

1.3 Σκοπός και Στόχοι Εργασίας

Ο στόχος της εργασίας μου ήταν η υλοποίηση ενός τέτοιου συστήματος χωρίς τη χρήστη GPS. Ο εντοπισμός των θέσεων των αντικειμένων που βρίσκονται στον ίδιο χώρο με τον χρήστη και ως αποτέλεσμα να έχουμε την εύρεση της θέσης του χρήστη αλλά και των αποστάσεων των αντικειμένων αυτών με τον ίδιο. Ο απώτερος σκοπός είναι εύρεση της κοντινότερης απόστασης μέσω του αλγορίθμου Dijkstra.

1.4 Δομή Εργασίας

Η παρούσα εργασία είναι χωρισμένη σε τρία μέρη. Στην δημιουργία της βάσης δεδομένων με τους απαραίτητους πίνακες και τα ανάλογα πεδία, στην δημιουργία των αντίστοιχων servlets στη μεριά του server οι οποίοι επικοινωνούν με την βάση δεδομένων. Η επικοινωνία του συστήματος με τα αρχεία αυτά επιτυγχάνεται μέσω HTTPRequest(client) και τέλος στη δημιουργία της android εφαρμογή.

2 Τεχνολογίες Υλοποίησης

2.1 Google

Εικόνα 1 Λογότυπο Google

Η παγκοσμίως γνωστή εταιρεία που ονομάζεται Google ξεκίνησε την λειτουργία της στις 27 Σεπτεμβρίου του 1998. Ο στόχος της εταιρείας είναι να οργανώσει όλες τις πληροφορίες του κόσμου καθιστώντας τις παγκοσμίως διαθέσιμες. Η Google ξεκίνησε σαν ένα project κολεγιακού επιπέδου από τον Λάρρυ Πέιτζ (Larry Page) και τον Σεργκέι Μπριν (Sergey Brin) το 1996 για μια μηχανή αναζήτησης, και σήμερα αποτελεί μια από τις μεγαλύτερες εταιρείες διαδικτυακών υπηρεσιών και έναν κολοσσό στον χώρο της ψηφιακής αγοράς εφαρμογών.

Επιπλέον η Google, έχει επιδείξει ένα ενδιαφέρον τόσο στην εξάπλωση και διάδοση τόσο της κεντρικής ιδέας της, του εμπορικού ονόματός (brand name) της, της πλατφόρμας αναζήτησης όσο και στην διαφήμιση του πακέτου εργαλείων της στην αγορά ασύρματης επικοινωνίας και συνεπώς της κινητής τηλεφωνίας. Το επιχειρηματικό μοντέλο της εταιρείας έχει τρομακτική επιτυχία στο Internet και συνεχίζει με αύξουσα πορεία. Ένας από τους στόχους της Google ήταν να μεταφέρει τις υπηρεσίες που απολάμβαναν οι χρήστες του internet στους συνδρομητές κινητής τηλεφωνίας. Οι αρχικές απόπειρες της εταιρείας στην αγορά των κινητών τηλεφώνων στιγματίστηκαν από πολλά προβλήματα. Με τη φιλοσοφία της να επικεντρώνεται στο χρήστη και τη σχεδίαση της, η Google ηγείται ενός κινήματος που επιχειρεί να μετατρέψει την υπάρχουσα και ιδιαίτερα κλειστή και φυλασσόμενη ασύρματη αγορά σε μία αγορά όπου οι χρήστες τηλεφώνων μπορούν να μετακινούνται από τον ένα φορέα στον άλλο και να έχουν ελεύθερη πρόσβαση τόσο σε εφαρμογές όσο και σε υπηρεσίες. Επίσης, η Google ακολουθεί μία ευρεία προσέγγιση εξετάζοντας την ασύρματη υποδομή απ' την οπτική της Αμερικανικής επιτροπής επικοινωνιών (FCC) για τις απαιτήσεις των κατασκευαστών τηλεφωνικών συσκευών, τις ανάγκες των προγραμματιστών εφαρμογών (developers) και τις επιθυμίες των πάροχων κινητής τηλεφωνίας. Στη συνέχεια, η Google στάθηκε στο πλευρό άλλων ομοϊδεατών μελών της ασύρματης κοινότητας θέτωντας το εξής ερώτημα: Τι θα έπρεπε να πραγματοποιηθεί, ώστε να κατασκευάσουμε ένα καλύτερο κινητό τηλέφωνο.

Όσον αφορά τις εφαρμογές της Google, εκείνες αναπτύσσονται κατά κύριο λόγο σε γλώσσα προγραμματισμού JAVA χρησιμοποιώντας το λογισμικό Android Development Kit (ADT). Παρόλα αυτά υπάρχουν κι άλλα εργαλεία που είναι διαθέσιμα για τη δημιουργία και την ανάπτυξη των εφαρμογών όπως το ενσωματωμένο Software Development Kit (SDK) για εφαρμογές ή επεκτάσεις τους σε γλώσσα C ή C++. Τέλος το Google App Inventor, είναι ένα γραφικό περιβάλλον που παρέχει τη δυνατότητα δημιουργίας προγράμματος για το Android ακόμα και σε αρχάριους προγραμματιστές.

2.2 GlassFish

Εικόνα 2 Λογότυπο GlassFish

Ο GlassFish είναι μια ανοικτή πλατφόρμα server εφαρμογών όπου ξεκίνησε από τη Sun Microsystems για την Java EE πλατφόρμα και τώρα χρηματοδοτείται από την Oracle. Η υποστηριζόμενη έκδοση ονομάζεται Oracle GlassFish Server.

Ο GlassFish είναι η εφαρμογή αναφοράς της Java EE και ως εκ τούτου υποστηρίζει Enterprice JavaBeans, JavaServer Faces , JPA , JavaServer , JMS , RMI , JavaServer Pages , Servlets , κ.λπ. Αυτό επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές που είναι φορητές και επεκτάσιμες ώστε να ενσωματωθούν ως βασικοί πηλώνες της τεχνολογίας. Χτισμένο πάνω σε ένα modular πυρήνα τροφοδοτούμενο από OSGi , GlassFish οδηγεί κατευθείαν στην κορυφή του Apache Felix εφαρμογής. Τρέχει επίσης με Equinox OSGi ή Knopflerfish OSGi σε πραγματικό χρόνο. Το HK2 αφαιρεί το σύστημα OSGi για την παροχή εξαρτημάτων, το οποίο μπορεί επίσης να θεωρηθεί ως υπηρεσίες. Οι υπηρεσίες αυτές μπορούν να ανακαλυφθούν και να εγχέονται κατά το χρόνο εκτέλεσης.

Ο GlassFish βασίζεται σε πηγαίο κώδικα όπου κυκλοφόρησε από τη Sun και την Oracle Corporation 's TOPLINK. Χρησιμοποιεί ένα παράγωγο του Apache Tomcat ως servlet για την εξυπηρέτηση περιεχόμενου Web, με ένα πρόσθετο στοιχείο που ονομάζεται Grizzly, την οποία χρησιμοποιεί η Java New I / O (NIO) για επεκτασιμότητα και ταχύτητα.
Ο Sun Microsystems ξεκίνησε το έργο GlassFish στις 6 Ιουνίου 2005. Στις 4 Μαΐου 2006, το Project GlassFish κυκλοφόρησε την πρώτη έκδοση που υποστηρίζει την προδιαγραφή Java EE 5.

Στις 8 Μαΐου 2007 το project SailFin ανακοινώθηκε σε JavaOne ως υποέργο στο πλαίσιο του έργου GlassFish. Το έργο SailFin στοχεύει να προσθέσει Session Initiation Protocol (SIP) servlet στη λειτουργικότητα του GlassFish.

Στις 17 Σεπτεμβρίου 2007, η κοινότητα GlassFish κυκλοφόρησε την έκδοση 2 (aka Sun Java Application σύστημα Server 9.1) με δυνατότητες ομαδοποίησης ικανοτήτων, Microsoft-interoperable Web Services.

Στις 21 Ιανουαρίου 2009 η Sun Microsystems κυκλοφόρησε την έκδοση 2.1 του GlassFish (aka Sun GlassFish Enterprise Server 2.1), η οποία χρησιμεύει ως βάση για το έργο Sailfin SIP AppServer.

Στις 10 Δεκεμβρίου του 2009 το GlassFish v3 κυκλοφόρησε. Όντας η εφαρμογή αναφοράς Java EE, αυτή ήταν η πρώτη server εφαρμογή για την πλήρη εφαρμογή Java EE 6 JSR 316 . Η JSR 316, ωστόσο έχει εγκριθεί με επιφυλάξεις . Σε αυτή την έκδοση GlassFish προσθέτει νέα χαρακτηριστικά για να διευκολύνουν τη μετάβαση από Tomcat σε GlassFish. Τα άλλα κύρια νέα χαρακτηριστικά που είναι γύρω από την αυτονομία (GlassFish v3 έχουν

ήδη αποσταλεί με Apache Felix OSGi runtime), με χρόνο εκκίνησης (μερικά δευτερόλεπτα), ανάπτυξη για αλλαγή (παρέχεται από το NetBeans και Eclipse plugins).

Στις 25 Μαρτίου του 2010, αμέσως μετά την εξαγορά της Sun Microsystems, η Oracle εξέδωσε χάρτη πορείας για τις εκδόσεις 3.0.1, 3.1, 3.2 και 4.0 με θέματα που περιστρέφονται γύρω από την ομαδοποίηση, ψηφιοποίηση και ενοποίησης με συνοχή και άλλες τεχνολογίες της Oracle. Η κοινότητα του ανοικτού λογισμικού παραμένει ανεπηρέαστη.

Στις 28 Φεβρουαρίου 2011, η Oracle Corporation κυκλοφόρησε το GlassFish v3.1. Αυτή η έκδοση εισήγαγε και υποστηριζόταν για SSH που βασίζεται σε προβλέψεις, κεντρική διαχειριστή, clustering και load-balancing. Διατηρεί την υποστήριξή του τόσο για το Προφίλ Web και την πλήρη Java EE 6 Προδιαγραφών πλατφόρμας.

Στις 28 Ιούλη 2011, η Oracle Corporation κυκλοφόρησε το GlassFish 3.1.1. Αυτό είναι καινούργια έκδοση για την GlassFish v3.1 με πολλαπλές ενημερώσεις (Weld, Mojarra, Jersey, EclipseLink.κτλπ), το JDK 7 υποστηρίζει , AIX και περισσότερα.

Στις 29 Φεβρουαρίου του 2012, η Oracle Corporation κυκλοφόρησε το GlassFish v3.1.2. Αυτή η έκδοση περιλαμβάνει διορθώσεις σφαλμάτων και νέα χαρακτηριστικά, συμπεριλαμβανομένων βελτιώσεων κονσόλας διαχείρισης.

Στις 17 Ιουλίου 2012, η Oracle Corporation κυκλοφόρησε το GlassFish v3.1.2.2. Αυτή είναι μια πρώτη έκδοση για την αντιμετώπιση ορισμένων έκτακτων θεμάτων στο προϊόν.

Στις 12 Ιουνίου 2013, η Oracle Corporation κυκλοφόρησε το GlassFish 4.0. Η σημαντική αυτή έκδοση φέρνει την πλατφόρμα Java Platform, Enterprise Edition 7 support.

Στις 9 Σεπτεμβρίου του 2014, η Oracle Corporation κυκλοφόρησε το GlassFish 4.1. Αυτή η έκδοση περιλαμβάνει πολλές διορθώσεις σφαλμάτων (πάνω από χίλια) και τις τελευταίες κυκλοφορίες MR του CDI και WebSockets.

2.3 Google Play

Εικόνα 3 Λογότυπο Google Play

Το Google Play, όπου παλιότερα ονομαζόταν Android Market, είναι μία πλατφόρμα διαμοιρασμού ψηφιακού περιεχομένου διαχειριζόμενη από τη Google. Για τις συσκευές με λειτουργικό σύστημα Android, αυτή η εφαρμογή, αποτελεί το επίσημο κατάστημα λογισμικού και επιτρέπει στους χρήστες να αναζητήσουν και να κατεβάσουν στη συσκευή τους εφαρμογές οι οποίες έχουν αναπτυχθεί με το Android SDK και έχουν δημοσιευθεί και διανεμηθεί μέσω του Google.

Το Android Market ανακοινώθηκε από τη Google στις 28 Αυγούστου 2008 και έγινε διαθέσιμο για τους χρήστες στις 22 Οκτωβρίου. Η υποστήριξη για εφαρμογές επί πληρωμή ξεκίνησε στις 13 Φεβρουαρίου 2009 για προγραμματιστές των Ηνωμένων Πολιτειών και Ηνωμένου Βασιλείου, ενώ στις 30 Οκτωβρίου 2010 προστέθηκαν άλλες 29 χώρες. Το Δεκέμβριο του 2010 προστέθηκε ένα φίλτρο περιεχομένου στο Android Market, ενώ μειώθηκε δραματικά ο χρόνος που χρειαζόταν από τη στιγμή της πληρωμής του χρήστη για μία εφαρμογή έως τη διάθεσή της σε εκείνον, από 24-48 ώρες σε μόλις 15 λεπτά.

Μέσα στο 2011 έγιναν πολλές και σημαντικές αλλαγές για το τότε Android Market, όπως αυτή της δυνατότητας να κατεβάζει κάποιος χρήστης εφαρμογές μπαίνοντας μέσω του υπολογιστή του στο Web Site της πλατφόρμας λογισμικού και επιλέγοντας τη συσκευή Android που επιθυμεί να την εγκαταστήσει μόλις αυτή συνδεθεί στο Internet. Επίσης σημαντική είναι η πρόσθεση λιστών μέσα στο Android Market με τις κατηγορίες εφαρμογών "Κορυφαίες Πωλήσεις", "Κορυφαίοι Προγραμματιστές", "Τάσεις" και "Προτάσεις Συντακτών" έτσι ώστε μπαίνοντας στο Android Market να μπορεί ο χρήστης να βλέπει ποιες εφαρμογές και ποιοι προγραμματιστές είναι πιο ψηλά στις προτιμήσεις των υπόλοιπων χρηστών. Αλλά και των συντακτών ενώ παράλληλα προστέθηκαν περισσότερα φίλτρα αναζήτησης.

Τέλος στις 6 Μαρτίου 2012 έγινε η μετονομασία του Android Market σε Google Play Store ενώ άλλαξε και ο σχεδιασμός του για να γίνει σχεδόν όπως είναι σήμερα.

Τα εκτελέσιμα αρχεία των εφαρμογών στο Google Play Store έχουν κατάληξη .apk. Ενώ όμως στην αρχή περιορίζονταν το μέγεθος των αρχείων .apk που ανέβαζαν οι προγραμματιστές στο ηλεκτρονικό κατάστημα σε 50 Mbytes, από εκείνη τη χρονιά οι προγραμματιστές έχουν τη δυνατότητα να προσθέσουν δύο αρχεία των 100Mbytes σε ότι αφορά τα .apk ενώ μπορούν να προσθέσουν και άλλα 2 αρχεία των 2Gbytes το καθένα, άρα συνολικά 4,146Gbytes

Εκτός από το διαμοιρασμό εφαρμογών, το Google Play πλέον προτείνεται και ως ψηφιακό κατάστημα πολυμέσων προσφέροντας μουσική, περιοδικά, βιβλία, ταινίες και τηλεοπτικά προγράμματα. Οι χρήστες επίσης μπορούν να αγοράσουν και συσκευές μέσω αυτής της πλατφόρμας όπως Chromebooks, Google Nexus κινητές συσκευές, Chromecasts και αξεσουάρς.

Τα ονόματα των υπηρεσιών που λειτουργούν στη πλατφόρμα GooglePlay είναι GooglePlayMusic, GooglePlayBooks, GooglePlayNewsstand, GooglePlayMovies & TV και GooglePlayGames.

Οι εφαρμογές που διατίθενται από το Google Play χωρίζονται σε δύο βασικές κατηγορίες, αυτές που μπορεί να κατεβάσει ο χρήστης στη συσκευή του δωρεάν και αυτές που θα πρέπει να καταβάλει κάποιο ποσό. Για να κατεβάσει κάποιος τις εφαρμογές που τον ενδιαφέρουν μπορεί είτε να επισκεφτεί την εφαρμογή του Google Play για συσκευές Android ή για Google TV ή μπορεί να επισκεφτεί και να κατεβάσει τις εφαρμογές που τον ενδιαφέρουν από το διαδικτυακό τόπο του Google

Πολλές από τις εφαρμογές μπορεί να στοχεύουν σε συγκεκριμένους χρήστες με βάση τα ιδιαίτερα χαρακτηριστικά των συσκευών τους, για παράδειγμα μερικές εφαρμογές χρησιμοποιούν τον αισθητήρα κίνησης για να κινηθεί κάποιο αντικείμενο μέσα στην εφαρμογή ενώ άλλες εφαρμογές απαιτούν τη χρήση του δέκτη GPS της συσκευής όπως συμβαίνει στη περίπτωσή μας, για εντοπισμό θέσης, υπολογισμό ύψους και ταχύτητας κίνησης. Το κατάστημα Google Play έχει φτάσει τα 1,3 εκατομμύρια δημοσιευμένες εφαρμογές και 50 δισεκατομμύρια κατεβάσματα. Οι δωρεάν εφαρμογές είναι διαθέσιμες παγκοσμίως εκτός από χώρες που έχουν τεθεί σε εμπάργκο από τις Η.Π.Α., ενώ οι επί πληρωμή εφαρμογές είναι διαθέσιμες σε 135 χώρες. Σύμφωνα με την "AppBrain Stats" από τις 1,3 εκατομμύρια συνολικά διαθέσιμες εφαρμογές, το 1 εκατομμύριο διατίθενται δωρεάν ενώ μόνο 200 χιλιάδες είναι επί πληρωμή.

Σύμφωνα με μετρήσεις, προγραμματιστές από 45 χώρες έχουν τη δυνατότητα να διαμοιραστούν τις εφαρμογές τους στο Google Play επί πληρωμή. Για να μπορέσει όμως κάποιος προγραμματιστής να καταστήσει την εφαρμογή του διαθέσιμη στο ηλεκτρονικό κατάστημα της Google, θα πρέπει πρώτα να καταβάλει το ποσό των 25$. Στη περίπτωση όπου μία εφαρμογή διατίθεται επί πληρωμή, το 70% του χρηματικού ποσού πηγαίνει στο προγραμματιστή, ενώ το υπόλοιπο 30% πηγαίνει στα έξοδα διανομής και στα λειτουργικά τέλη. Το κέρδος που μπορεί κάποιος να έχει από μία εφαρμογή του στο Google Play τα λαμβάνει μέσω της υπηρεσίας Google Wallet ή μέσω του AdSense σε μερικές χώρες.

Το Google Play Store περιέχει φίλτρα που παρουσιάζουν μόνο τις εφαρμογές που είναι συμβατές με τη συσκευή του χρήστη έτσι ώστε να αποφευχθούν προβλήματα συμβατότητας. Επίσης οι προγραμματιστές των εφαρμογών έχουν τη δυνατότητα να περιορίζουν τις εφαρμογές που έχουν δημιουργήσει σε συγκεκριμένες χώρες. Σαν εφαρμογή όμως δεν είναι ανοιχτού κώδικα και μπορεί να εγκατασταθεί μόνο σε συσκευές όπου είναι συμβατές με τις απαιτήσεις της Google αφού αποδεχτεί τη δωρεάν χορήγηση άδειας εγκατάστασης και χρήσης από τη Google.

Ως εκ τούτου οι χρήστες κάποιων κινητών συσκευών τύπου tablet όπως για παράδειγμα το Kindle Fire της Amazon δε μπορούν να εγκαταστήσουν το Google Play Store αλλά χρησιμοποιούν το ήδη εγκατεστημένο λογισμικό ψηφιακού καταστήματος που τους χορηγεί η εταιρεία κατασκευής του tablet.

2.4 Google Developers

Εικόνα 4 Λογότυπο Google Developers

Πλέον στο Google Play είναι αρκετά εύκολο, οποιοσδήποτε προγραμματιστής Android εφαρμογών να ανεβάσει και να προωθήσει το application του σε αυτό. Η διαδικασία για να ανεβάσει ένας προγραμματιστής την εφαρμογή του είναι σχετικά εύκολη και χρειάζεται ένα ενδεικτικό ποσό καταβολής για μία και μόνο φορά. Το Google Play δίνει διάφορες δυνατότητες τόσο στους προγραμματιστές των εφαρμογών όσο και στους χρήστες παρέχοντας ένα σύστημα αξιολόγησης της εφαρμογής, εικόνες της εφαρμογής και άλλα στοιχεία όπως την έκδοση αυτής αλλά και το μέγεθός της. Επίσης, υπάρχουν πολλά στατιστικά όπως π.χ πόσοι χρήστες κατέβασαν την εφαρμογή σε συγκεκριμένο χρονικό διάστημα. Πιο συγκεκριμένα για να δημοσιεύσουμε την εφαρμογή μας στο Google Play θα πρέπει το application μας να τηρεί κάποιους βασικούς κανόνες που έχει θέσει η Google όπως για παράδειγμα.

- Η εφαρμογή μας δηλαδή το .apk αρχείο να μην υπερβαίνει τα 100 Mbytes.
- Πρέπει να διαγράψουμε τυχών Log κλήσεις από τον κώδικα μας και να μην υπάρχει το χαρακτηριστικό android: debuggable στο manifest αρχείο.
- Να κάνουμε έναν ενδελεχή έλεγχο της εφαρμογής σε τουλάχιστον μία κινητή συσκευή πριν την ανεβάσουμε στο Google Play.
- Να βεβαιωθούμε ότι δουλεύουν τα path μας και οι κλήσεις που κάνει η εφαρμογή σε κάποιον απομακρυσμένο server.
- Να παρέχουμε τιμές για τις μεταβλητές android :version Code και android :version Name στο manifest αρχείο μας.
- Να βεβαιωθούμε ότι χρησιμοποιούμε τα σωστά API keys στον κώδικα μας (κάθε API key πρέπει να αντιστοιχεί στο package name της εφαρμογής μας).

Εάν η εφαρμογή μας δεν τηρεί σωστά τον τελευταίο κανονισμό σχετικά με το API Key, τότε δεν έχουμε πρόσβαση στην υπηρεσία δικαιωμάτων της Google (Google Play Services) και επομένως δεν λειτουρεί.

2.5 Android

ToAndroid είναι μια ολοκληρωμένη, ανοιχτή και ελεύθερη πλατφόρμα για κινητά τηλέφωνα που περιλαμβάνει ένα λειτουργικό σύστημα (OS), το απαραίτητο ενδιάμεσο λογισμικό, βιβλιοθήκες και βασικές εφαρμογές. Το AndroidSystemDevelopmentKit παρέχει στους προγραμματιστές όλα τα εργαλεία και APIs για να αρχίσουν να αναπτύσσουν λογισμικό για την πλατφόρμα Android χρησιμοποιώντας τη γλώσσα προγραμματισμού Java.

Η ιστορία

Τα πράγματα ξεκίνησαν όταν ο ευφυής Andy Rubin θέλησε την Άνοιξη του 2005 να χρησιμοποιήσει την Google ως κατ' εξοχήν μηχανή αναζήτησης για το T-MobileSidekick, μια φερέλπιδα συσκευή κινητού, την οποία είχε αναπτύξει με την ομάδα συνεργατών του. Εν συνεχεία, ζήτησε να συναντηθεί με τον Larry Page, ο οποίος είναι ο ένας από τους δύο ιδρυτές της Google. Σε αυτήν τη συνάντηση ο Rubin παρουσίασε το Android ως ένα εν δυνάμει παγκόσμιο ανοικτό λειτουργικό σύστημα που θα άλλαζε για πάντα τον τρόπο που διαντιδρούνε οι χρήστες με το κινητό σας, τονίζοντας, ταυτόχρονα, τη σταθερή υπεροχή που παρατηρείται στις συνήθειες του αγοραστικού κοινού των κινητών τηλεφώνων, σε αντιδιαστολή με τις πωλήσεις ηλεκτρονικών υπολογιστών.

Την ίδια στιγμή, ο Page δεν ήθελε να γίνει απλώς ο υποστηρικτής του Android, ήθελε να γίνει ο ιδιοκτήτης του. Ο Andy βρήκε «την καλή», την ώρα που ένας ισχυρός παίκτης εμφανίστηκε στο προσκήνιο και έθεσε έτσι τους όρους του ανταγωνισμού σε άλλο επίπεδο. Ο καινούργιος «παίκτης» δεν είναι άλλος από εκείνον που τελικά λάνσαρε το καλοκαίρι του 2005, το iPhone της Apple.

Ο επιχειρηματικός-τεχνολογικός κόσμος περίμενε πως η Google θα απαντούσε με ένα gPhone, αλλά αυτό δεν έγινε, διότι έγινε κάτι άλλο, πολύ σημαντικότερο. Το Φθινόπωρο του 2005 ανακοινώνεται ότι 34 εταιρίες, όπως η TexasInstruments, η Intel, η T-Mobile και η SprintNextel, ενώνουν τις δυνάμεις τους με την Google για τη δημιουργία μιας πλατφόρμας ανοιχτού κώδικα που θα έχει ενσωματωμένο το λογισμικό Linux και θα εκπροσωπείται από μια νέα συστάδα εταιριών που θα καλείται OpenHandsetAlliance.

Στο χορό δεν άργησαν να μπουν και άλλες εταιρείες, όπως η HTC, η Motorola και η LG, ανακοινώνοντας την πρόθεσή τους να δώσουν προς πώληση στην αγορά smartphones με λειτουργικό σύστημα Android σε διάφορα σχήματα και μεγέθη, με τα οποία θα μπορεί να έχει ο χρήστης να ενσωματώνει στο κινητό του πλήθος εφαρμογών.

2.5.1 Εκδόσεις Android

Το Android που κυκλοφορεί διάφορες εκδόσεις με ονομασίες που σου ανοίγουν την όρεξη για νέα χαρακτηριστικά, όπως τα παλαιότερα CupCake (1.5), Donut (1.6), Éclair (2.0, 2.1), GingerBread (2.3) αλλά και FroYo (2.2) Honeycomb (3.0) που υλοποιείται σε ταμπλέτες ενώ υπάρχουν πλέον και οι εκδόσεις Honeycomb (3.1) και Honeycomb (3.2). Δεν πρέπει να ξεχάσουμε να αναφέρουμε και την επερχόμενη έκδοση IceCreamSandwich η οποία θα είναι διαθέσιμη από τον Οκτώβρη.

Από την «παρθενική» έκδοση Android 1.0, η οποία κυκλοφόρησε το Σεπτέμβριο του 2008, μέχρι την αμέσως επόμενη, 1.1 που παρουσιάστηκε το Φεβρουάριο του 2009, χρειάστηκε ένας χρόνος για να γίνει η έκρηξη των καινοτόμων εκδόσεων και των σημαντικών αλλαγών που επέφεραν για τον χρήστη.

Μερικές από τις χαρακτηριστικές αλλαγές των εκδόσεων που ακολούθησαν και οι εκπρόσωποί τους**:**

Android 1.0 και Android 1.1

Με την κυκλοφορία του πρώτου Androidsmartphone,του HTCDream έχουμε και την κυκλοφορία της έκδοσης 1.0 . Ενσωματωμένες εφαρμογές που υπήρχαν ήταν το ξυπνητήρι, ο περιηγητής, η αριθμομηχανή, η κάμερα, το email, οι χάρτες και η μουσική καθώς και κάποιες άλλες.

Android 1.5 Cupcake

Εικόνα 5- Android 1.5 Cupcake

Την άνοιξη του 2009 κυκλοφόρησε η έκδοση 1.5Cupcake και ουσιαστικά έχουμε την αρχή των ονομάτων με γλυκά. Η έκδοση αυτή ουσιαστικά σηματοδοτούσε την υποστήριξη των widgets και ως νέα χαρακτηριστικά του ήταν η εγγραφή video και playback σε μορφή MPEG-4 και 3GP και τα εφέ κίνησης κατά την περιήγηση στις διαφορετικές οθόνες. Επίσης υποστηρίζεται η δυνατότητα αυτόματης σύνδεσης ακουστικών headset σε συγκεκριμένη απόσταση, ανέβασμα εικόνων στο Picasa και βίντεο στο YouTube κατευθείαν από την κινητή συσκευή του χρήστη, ενώ παρέχει εικονικό πληκτρολόγιο με πρόβλεψη λέξεων και νέα widgets για την αρχική οθόνη. Διαθέτει κάμερα 5MP με αυτόματη εστίαση και κάρτα μνήμης microSD.Τέλος, το CupCake (1.5) «τρέχει» το HTCHero με οθόνη αφής TFT-LCD, 3,2”, ανάλυσης 320x480p (HVGA).

Android 1.6 Donut

Εικόνα 6 - Android 1.6 Donut

Το Σεπτέμβριο του 2009 κυκλοφόρησε η έκδοση 1.6 Donut συμπεριλαμβάνοντας βελτιώσεις όπως ευκολότερη αναζήτηση και δυνατότητα προεπισκόπησης εφαρμογών σε όσες συσκευές είχαν GooglePlay, δείκτες χρήσης της μπαταριάς και αυτόματη περιστροφή οθόνης. Με την έκδοσηDonut δίνεται έμφαση στην αναζήτηση από την αρχική οθόνη με bookmarks, ιστορικό, επαφές κ.ά. αλλά και στη φωνητική αναζήτηση, ενώ υποστηρίζονται και οθόνες αναλύσεων WVGA. Τηλέφωνα της έκδοσης αυτής είναι το LGGT 540, το οποίο υποστηρίζει οθόνη αναλύσεων WVGA και επιπλέον προσφέρει ευκολία στην εύρεση των επαφών, διαθέτει qwerty πληκτρολόγιο για γρήγορη αποστολή SMS και 3G και Wi-Fi για να είναι ο χρήστης συνδεδεμένος στο διαδίκτυο όπου και να βρίσκεται, και το SonyEricssonX10 με επεξεργαστή Snapdragon 1GHz της Qualcomm και οθόνη αφής 4”, 854x480pixels. Στην πίσω όψη βρίσκεται η κάμερα 8,1MP και το LEDFlash κάτω ακριβώς από τον φακό της.

Android 2.0 Éclair

Εικόνα7 - Android 2.0 Éclair

Τον Οκτώβριο του 2009 κυκλοφόρησε, μόλις ένα μήνα μετά το Donut. Συμπεριελάμβανε αρκετές μικρές βελτιώσεις, όπως Bluetooth 2.1, κινούμενο φόντο στην οθόνη του home, και πληκτρολόγιο με έξυπνο λεξιλόγιο που μαθαίνει ανάλογα με την χρήση των λέξεων. Με το όνομα Éclair έχουμε και επόμενες επίσης γλυκές εκδόσεις, τις Éclair 2.0 και Éclair 2.1 που προχωρούν ακόμη πιο πολύ, διαθέτοντας υποστήριξη HTML5,νέο browserUI, GoogleMaps 3.1.2, ψηφιακό ζουμ, ενσωματωμένη υποστήριξη για flash στην κάμερα, βελτιωμένο εικονικό πληκτρολόγιο, δυνατότητα αντίληψης multi-touch, livewallpapers, και Bluetooth 2.1. Το MotorolaDroid είναι ένας εκπρόσωπος της έκδοσης 2.0 (Éclair), με επεξεργαστή ArmCortexA8 550MHz. Ο σχεδιασμός του είναι slide με πλήρες qwerty πληκτρολόγιο και οθόνη αφής 3,7”, ανάλυσης 480×854p. Επίσης, έχουμε το SamsunggalaxyS, με οθόνη αφής 4.0″ SuperAMOLED που «τρέχει» Android 2.1 (Éclair) με επεξεργαστή 1GHz, εσωτερική μνήμη 8GB, Wi-Fi και Bluetooth 3.0, παρέχει ταχύτητα, χώρο και απαιτούμενες δυνατότητες σύνδεσης.

Android 2.2-2.2.3 FroYo

Εικόνα 8 - Android 2.2-2.2.3 FroYo

Το Μάιο του 2010 κυκλοφόρησε και το ακρώνυμο της έκδοσης αποτελεί συντόμευση της φράσης "FrozenYogurt" (παγωμένο γιαούρτι). Είναι η πρώτη έκδοση του Android που υποστήριζε AdobeFlash. Μερικές από τις βελτιώσεις ήταν σύνδεση μέσω USB και Wi-Fihotspot, η γρήγορη εναλλαγή γλώσσας κατά την πληκτρολόγηση και η δυνατότητα απενεργοποίησης της λειτουργίας δικτύου δεδομένων. Η έκδοση 2.2 (FroYo-FrozenYogurt) αναβάθμισε αισθητά την ταχύτητα του OS, αλλά και τη γενικότερη απόδοση, παρέχει υποστήριξη Adobeflash 10.1 και επιλογή εγκατάστασης εφαρμογών στην κάρτα μνήμης, διαθέτει Market με δυνατότητα αυτόματων updates, ενσωματώνει τον ChromeV8 JavaScript στα browsers applications. Στη FroYo βρίσκουμε και τη δυνατότητα χρήσης της συσκευής για διαμοιρασμό ίντερνετ μέσω Wi-Fi σε άλλες συσκευές (tethering). Εκπρόσωπος της έκδοσης αυτής είναι το LGOptimus 2x που διαθέτει οθόνη αφής 4″, λειτουργικό AndroidFroyo και διπύρηνο επεξεργαστή NvidiaTegra 2 με ισχυρή κάμερα 8MP, δυνατότητα εγγραφής και αναπαραγωγής βίντεο fullHD και συνδεσιμότητα HDMI και DLNA. Ένας άλλος εκπρόσωπος είναι το HTCDesireZ με AndroidFroyo και οθόνη αφής 3,7″, ανάλυσης WVGA (480x800p), συρόμενο Qwerty πληκτρολόγιο για γρήγορα e-mail, 5MP κάμερα και δέκτη GPS. Χάρη στον ενσωματωμένο AdobeFlashPlayer και προβάλλει κάθε ιστοσελίδα.

Android 2.3-2.3.7 Gingerbread

Εικόνα 9 - Android 2.3-2.3.7 Gingerbread

Τον Δεκέμβρη του 2010 Κυκλοφόρησε η πιο πολυπληθής έκδοση του Android.
Ήταν πολύ πιο γρήγορο και εύχρηστο από τις προηγούμενες εκδόσεις και έδινε στους δημιουργούς εφαρμογών μεγαλύτερες δυνατότητες. Οι βελτιώσεις περιλάμβαναν υποστήριξη πολλών καμερών στην συσκευή όπως και μεγαλύτερης ανάλυσης οθόνη. Η GingerBread (2.3), που κυκλοφόρησε το Δεκέμβριο του 2010, υποστηρίζει πλέον πολύ μεγάλα μεγέθη οθονών και αναλύσεων, διαθέτει επανασχεδιασμένο multi-touch πληκτρολόγιο, εγκατεστημένη υποστήριξη για τηλεφωνικές κλήσεις μέσω ίντερνετ (VoIP), downloadmanager για κατέβασμα μεγάλων αρχείων, λειτουργίες copy-paste σε όλο το λειτουργικό, καθώς και προεγκατεστημένη υποστήριξη για πολλαπλές κάμερες. Μεγάλος εκπρόσωπος της αποτελεί το SonyXperiaPlay, συσκευή προσανατολισμένη στο gaming, «τρέχει» GingerBread (2.3) με επεξεργαστή ScorpionARM7 και ταχύτητα στα 1GHz. Η οθόνη του είναι 4 ιντσών με ανάλυση 480x854p και η εσωτερική του μνήμη 400MB. Άλλοι εκπρόσωποι είναι τα HTCChaCha και HTCSalsa με GingerBread (2.3) που ενσωματώνουν ένα εξειδικευμένο Facebook πλήκτρο, για πρόσβαση με ένα άγγιγμα, στην υπηρεσία του Facebook μέσα από την εμπειρία HTCSense.

Android 3.0 – 3.1 Honeycomb

Τον Φεβρουάριο του 2011 κυκλοφόρησε και ήταν διαθέσιμη μόνο για tablets. Οι βελτιώσεις της περιλαμβάνουν γρήγορη πρόσβαση σε χαρακτηριστικά της κάμερας, καλύτερο πληκτρολόγιο κατάλληλο για μεγάλες οθόνες, εκτέλεση πολλαπλών λειτουργιών και εύκολη μετάβαση από την μια στην άλλη.
Η έκδοση Honeycomb (3.1) προσέθεσε την επιλογή να μεταφέρεται περιεχόμενο απευθείας από συσκευές USB, ενώ τέλος η έκδοση 3.2 προσέθεσε διάφορες δυνατότητες και ευκολίες για χρήστες και developers όπως τη μεταφορά αρχείων από κάρτες SD και δυνατότητα ZoomtoFill.

Εικόνα 10 - Android 3.0 – 3.1 Honeycomb

Android 4.0-4.0.2 IceCreamSandwich

Τον Οκτώβρη του 2011 κυκλοφόρησε η έκδοση 4.0 IceCreamSandwichκαι έφερε πληθώρα αλλαγών στο λειτουργικό σύστημα. Η δυνατότητα χρήσης "μαλακών" κουμπιών δηλαδή των κουμπιών πάνω στην οθόνη (πίσω, αρχική, κλπ) είναι πλέον πραγματικότητα καθώς μέχρι τότε όλα τα κινητά είχαν εξωτερικά κουμπιά για αυτές τις λειτουργίες. Μερικές άλλες δυνατότητες ήταν καλύτερη χρήση των φωνητικών εντολών, το FaceUnlock, βελτίωση της ταχύτητας απόκρισης και αναδιαμόρφωση του περιβάλλοντος χρήσης. Αυτή η έκδοση διαθέτει καλύτερο webbrowser με tabs, ανανεωμένο γραφικό περιβάλλον με αρκετά 3D στοιχεία και ανανεωμένο εικονικό πληκτρολόγιο. Δίνει ειδική έκδοση του Gmail για tablets, δυνατότητα βιντεοκλήσεων μέσω εφαρμογής GoogleTalk,
ανανεωμένη έκδοση GoogleMaps και βελτιστοποιημένη εφαρμογή για ανάγνωση Google-books. 3.4.9

Εικόνα 11 - Android 4.0-4.0.2 IceCreamSandwich

Android 4.1-4.3.1 JellyBean

Τον Ιούνιο του 2012 κυκλοφόρησε και αποτελεί την καλύτερη έκδοση του Android μέχρι σήμερα. Το περιβάλλον χρήσης και η απόκρισή του είναι πιο γρήγορη και καλοφτιαγμένη από ποτέ ενώ περιλαμβάνει πάρα πολλές μικρές βελτιώσεις σε όλο το σύστημα, όπως για παράδειγμα στην κάμερα και στην χρήση φωνής για υπαγόρευση κειμένου. Η έκδοση 4.1 JellyBean του OS σκαρφάλωσε στο 28,4% από 25% τον προηγούμενο μήνα και ξεπέρασε το ποσοστό του IceCreamSandwich, το οποίο ανέρχεται πλέον στο 27,5%. Παρόλα αυτά, η έκδοση 2.3 Gingerbread του λειτουργικού παραμένει στην πρώτη θέση και είναι εγκατεστημένη

Εικόνα 12 - Android 4.1-4.3.1 Jellybean

στο 38,5% των Android συσκευών. Το JellyBean εφαρμόστηκε για πρώτη φορά στο tabletGoogleNexus 7 ενώ η έκδοση Android 4.2 πρωτοεμφανίστηκε στα Nexus 4 και Nexus 10.

Android 4.4-4.4.4 KitKat

Εικόνα 13 - Android 4.4-4.4.4 KitKat

Τον Σεπτέμβρη του 2013 κυκλοφόρησε. Αν και αρχικά ήταν να ονομαστεί "KeyLimePie" ("KLP") κωδική ονομασία, το όνομα άλλαξε καθώς πολύ λίγοι άνθρωποι ξέρουν πραγματικά τη γεύση αυτής της πίτα. Το KitKat έκανε το ντεμπούτο του στο Nexus της Google 5 και έχει βελτιστοποιηθεί για να τρέχει σε ένα μεγαλύτερο εύρος συσκευών από τις προηγούμενες εκδόσεις του Android, αφού έχει ως συνιστώμενη ελάχιστη μνήμη RAM 512 .

Android 5.0 "Lollipop"

Εικόνα 14 - Android 5.0 "Lollipop"

Τον Ιούνιο του 2014 ανακοινώθηκε από την Google η νέα έκδοση του για επιλεγμένες συσκευές που τρέχουν Android, συμπεριλαμβανομένων τις συσκευές Nexus. Η Lollipop ως βασική αλλαγή που θα παρουσιάσει είναι ένα επανασχεδιασμένο περιβάλλον εργασίας χρήστη χτισμένο γύρω από μία διαδραστική σχεδιαστική γλώσσα που αποκαλείται ως «υλικό σχεδιασμού». Επιπλέον βελτιώσεις του συστήματος θα είναι οτι το notificationssystem θα επιτρέπει κοινοποιήσεις που θα μπορούν να προσπελαστούν από την lockscreen, και να εμφανίζονται μαζί με εφαρμογές ως banner πάνω την κορυφή της οθόνης. Εσωτερικές αλλαγές που θα γίνουν επίσης στην πλατφόρμα, πιο συγκεκριμένα, το AndroidRuntime (ART) θα αντικαταστήσει το Dalvik με μία πιο βελτιωμένη έκδοση την γνωστή ως ProjectVolta, η οποία θα παρέχει καλύτερη απόδοση των εφαρμογών πράγμα το οποίο σημαίνει ότι θα υπάρξουν και αλλαγές που αποσκοπούν στη βελτίωση και βελτιστοποίηση της χρήσης της μπαταρίας. Όπως παρατηρούμε και στις εικόνες κατα το πέρας των εκδόσεων , οι διαφορές του γραφικού περιβάλλοντος είναι αισθητές πόσο μάλιστα των δυνατοτήτων που μπορεί να παρέχει σε ένα χρήστη.

Το Android δεν είναι (μόνο) τηλέφωνο

Εικόνα 15 - Android συσκευές

Ο όρος Android δεν αναφέρεται σε smartphone, ωστόσο επικρατεί αυτή η εσφαλμένη αντίληψη από τον περισσότερο κόσμο. Εκτός από τα smartphone εξυπηρετεί και μία τεράστια γκάμα από tablet όπως το Nexus 7. Η ποικιλομορφία του όμως δεν σταματά εδώ, αφού μπορεί να τρέξει σ' όλων των ειδών τις συσκευές που υποστηρίζονται στο Linux από κάμερες, κονσόλες παιχνιδιών και τηλεοράσεις μέχρι συστήματααυτοκινήτων,ρολόγια και γυαλιά.

Επειδή λοιπόν είναι τόσο ευέλικτο κυκλοφορεί σε συσκευές για όλες τις ανάγκες και όλες τις τσέπες. Ανάλογα με το πορτοφόλι του καθενός άλλοι προτιμούν φθηνές συσκευές περιορισμένων δυνατοτήτων ενώ άλλοι δυσκολεύονται να αντισταθούν ακόμα και στα πιο ακριβά gadget όπως το GalaxyS5 ή το GalaxyNote 3 της Samsung.

Το Android όμως δεν είναι ίδιο σε όλες τις συσκευές. Ο μεγάλος ανταγωνισμός κάνει τις εταιρίες να το παραμετροποιούν με δικές τους ιδέες προσπαθώντας να προσελκύσουν περισσότερους καταναλωτές. Έτσι το "Android πρόσωπο" της Samsung διαφέρει από αυτό της LG, της Sony ακόμα και της Google! Οι πολλές μορφές του Android είναι ένας από τους λόγους που το κάνουν τόσο πετυχημένο καθώς ο κάθε χρήστης μπορεί να το διαμορφώσει όπως θέλει.

Ακόμα όμως και αν δε νιώθουμε άνετα με το γραφικό περιβάλλον της συσκευής μας, μπορούμε να αλλάξουμε την εμφάνισή του αντικαθιστώντας τον launcher η ακόμα και ολόκληρο το λειτουργικό σύστημα με μια customROM όπως η διάσημη CyanogenModROM.

2.5.2 Τι μπορούν να κάνουν οι συσκευές Android;

Τα τεχνικά χαρακτηριστικά διαφέρουν από συσκευή σε συσκευή αλλά όλα τρέχουν στο ίδιο λειτουργικό σύστημα: Android. Οι περισσότερες Android συσκευές παρέχουν υποστήριξη για κάμερα, GPS, Bluetooth, NFC, επιταχυνσιόμετρο, πυξίδα, γυροσκόπιο, και πολλά περισσότερα!

Σε επίπεδο λογισμικού, θα βρούμε και εκεί κάποια στάνταρ πραγματάκια όπως livewallpaper, widget, την εύχρηστη μπάρα ειδοποιήσεων το συρτάρι των εφαρμογών και άψογη ενσωμάτωση διαφόρων εφαρμογών που συνεισφέρουν στην ευκολία γρηγορότερου διαμοιρασμού πληροφοριών μέσω socialmedia, κλπ.

Στην πράξη οι Android συσκευές μπορούν να οργανώσουν τη ζωή μας, να μας φέρουν σε επικοινωνία με άλλους ανθρώπους με όλα τα μέσα (εκτός από σήματα καπνού), να μας βοηθήσουν να χαλαρώσουμε στον ελεύθερό μας χρόνο, να παίξουμε τα παιχνίδια μας, να οργανώσουμε και να δούμε τις ταινίες μας, να χειριστούμε άλλες συσκευές του σπιτιού και άλλα πολλά. Είναι μία πλήρης πλατφόρμα που χάρη στο χαρακτήρα του ανοιχτού κώδικα που διαθέτει, μας δίνει τη δυνατότητα να κάνουμε κυριολεκτικά τα πάντα!

Εικόνα 16 - Έξυπνα τηλέφωνα

AndroidDevelopment

Όλο και περισσότεροι κατασκευαστές υιοθετούν το Android ως βάση για τα mobile προϊόντα τους, με αποτέλεσμα το Android να αποτελεί το No1 λειτουργικό σύστημα σε φορητές συσκευές στον πλανήτη. Για τους προγραμματιστές αυτό δίνει μία τεράστια ευκαιρία να κερδίσουν χρήματα γράφοντας εφαρμογές που θα απευθύνονται σε εκατοντάδες εκατομμύρια χρήστες / πελάτες και προσθέτοντάς τις δωρεάν (μετά από ένα εφάπαξ ποσό των 25 δολαρίων) στο ράφι του καταστήματος που βρίσκεται μέσα σε όλες αυτές και ακούει στο όνομα GooglePlayStore.

Όπως κάθε αρχή και δύσκολη, έτσι και εδώ, το AndroidDevelopment θα μας φανεί λίγο δυσνόητο και προϋποθέτει ότι έχουμε κάποιες βασικές γνώσεις Java. Υπάρχουν όμως και άλλοι τρόποι που επιτρέπουν τη δημιουργία εφαρμογών ακόμα και χωρίς γνώση προγραμματισμού!

2.5.3 Γιατί Ανάπτυξη σε Android

2.5.3.1Λειτουργικότητα και ευελιξία.

Το Android είναι μια μοναδική πλατφόρμα που επιτρέπει την ανάπτυξη εφαρμογών λογισμικού το οποίο εκμεταλλεύεται πλήρως τις δυνατότητες μιας συμβατής συσκευής. Για παράδειγμα, οι προγραμματιστές εφαρμογών είναι ελεύθεροι να δημιουργήσουν εφαρμογές που χρησιμοποιούν οποιαδήποτε από τις βασικές λειτουργίες του τηλεφώνου όπως η αποστολή SMS, τηλεφωνικές κλήσεις, τη λήψη φωτογραφιών, το GPS κτλ. Έτσι διευκολύνονται στην ανάπτυξη πιο περίπλοκου και πιο πλούσια λειτουργικού λογισμικού. Αυτό το λειτουργικό σύστημα κινητών τηλεφώνων (ή άλλων μικρών φορητών συσκευών που συνδέονται στο διαδίκτυο) στηρίζεται στον ελεύθερο πυρήνα του Linux. Επιπλέον, η πλατφόρμα ανάπτυξης Android είναι μια πλατφόρμα multi-tasking, πράγμα που σημαίνει ότι κάθε εφαρμογή μπορεί να τρέξει στο τηλέφωνο ταυτόχρονα κάποια άλλη χωρίς να επηρεαστεί η απόδοσή τους, και αυτό είναι καλύτερο από το να περιορίζεται σε μία εφαρμογή κάθε φορά. Το Android είναι μια πλατφόρμα ανοικτού κώδικα, πράγμα που σημαίνει ότι μπορεί εύκολα να επεκταθεί και να τροποποιηθεί για να συμβαδίζει και να υιοθετεί τις τελευταίες τεχνολογίες και εξελίξεις. Το γεγονός ότι και η πηγή της πλατφόρμας είναι ανοικτή διασφαλίζει ότι η ανάπτυξη το Android θα έχει συνεχή πρόοδο και θα εξελίσσεται αφού ένας μεγάλος αριθμός ικανών android προγραμματιστών εργάζεται για τη δημιουργία ελευθέρων για χρήση προηγμένων εργαλείων λογισμικού.

2.5.3.2 Πλήρης παραμετροποίηση

Δεν υπάρχει διαφορά μεταξύ των λειτουργιών / εφαρμογών οι οποίες είναι ενσωματωμένες στο τηλεφώνου από τις εφαρμογές που δημιουργούνται και προστίθενται από τρίτους προγραμματιστές Android. Οι τελευταίες μπορούν και έχουν την ίδια πρόσβαση σε όλες τις κύριες λειτουργίες της συσκευής κάτι που επιτρέπει στους τελικούς χρήστες να απολαμβάνουν ένα ευρύ φάσμα εφαρμογών Android που μπορούν να χρησιμοποιηθούν για τη σχεδόν απεριόριστους σκοπούς. Με συσκευές χτισμένες στην πλατφόρμα Android, οι χρήστες έχουν τη δυνατότητα να προσαρμόσουν πλήρως τη συσκευή τους ανάλογα με τις ανάγκες και τις απαιτήσεις τους. Τυχόν εφαρμογές ακόμα και οι βασικές λειτουργίες μπορεί να τροποποιηθούν ή να αντικατασταθούν πλήρως από άλλες.
Για παράδειγμα, ο χρήστης μπορεί να χρησιμοποιήσει την επιθυμητή του εφαρμογή για να εμφανίσει τις φωτογραφίες που είναι αποθηκευμένες στο τηλέφωνό του, ή για να έχει πρόσβαση στην αλληλογραφία του.

2.5.3.3 Διαδραστικότητα

Οι προγραμματιστές Android μπορούν να δημιουργήσουν πολύπλοκες καινοτόμες εφαρμογές με σχεδόν απεριόριστη λειτουργικότητα. Για παράδειγμα, μια εφαρμογή μπορεί να μεταδώσει τα δεδομένα από το κινητό σας με το διαδίκτυο (κάτι που μπορεί να περιλαμβάνει το ημερολόγιο σας και τις προγραμματισμένες εκδηλώσεις, λίστα με τις

επαφές, τις φωτογραφίες σας και ακόμη και την τρέχουσα θέση σας, αλλά και παραγγελίες, τιμολόγια κτλ) και να λάβει όλα όσα μπορεί να χρειαστεί online και να εμφανίζονται στην οθόνη της συσκευής.

2.5.3.4 Απλούστερη Ανάπτυξη Εφαρμογών Κινητών.

Η πλατφόρμα παρέχει στο καθένα που ασχολείται με την ανάπτυξη εφαρμογών τη δυνατότητα χρησιμοποίησης μια μεγάλης ποικιλίας από βιβλιοθήκες και τα χρήσιμα εκείνα εργαλεία που μπορούν να χρησιμοποιηθούν για τη δημιουργία του πιο εξελιγμένου λογισμικού. Αυτή η ολοκληρωμένη δέσμη από έτοιμα εργαλεία αυξάνει σημαντικά την παραγωγικότητα των προγραμματιστών Android εφαρμογών και τους βοηθά να δημιουργήσουν εκπληκτικά πλούσιο λογισμικό γρηγορότερα και με λιγότερα λάθη.

2.5.4 Με ποιες γλώσσες μπορώ να γράψω Android εφαρμογές

Οι Android εφαρμογές είναι συνήθως γραμμένες σε Java καθώς είναι η πιο διάσημη γλώσσα προγραμματισμού στον κόσμο σήμερα και χαρακτηρίζεται ως αντικειμενοστραφής γλώσσα (object-oriented). Για όσους έχουν προηγούμενη εμπειρία με την φιλοσοφία του object-orientedprogramming σε άλλες γλώσσες, όπως C#, ObjectiveC ή Ruby τότε δεν θα αντιμετωπίσουν κάποιο ιδιαίτερο πρόβλημα στη μετάβαση σε Java.

Η Google παρέχει το **NativeDevelopmentKit (NDK)** το οποίο επιτρέπει στους developer να γράψουν σημαντικό μέρος των εφαρμογών τους σε native γλώσσες όπως είναι η C και η C++. Αυτό είναι ιδιαίτερα χρήσιμο καθώς μας επιτρέπει να επαναχρησιμοποιήσουμε τον υπάρχων κώδικα (δηλαδή κάποιες γνωστές βιβλιοθήκες) ή να βελτιώσουμε την κατανάλωση πόρων της εφαρμογής που φτιάχνουμε κάνοντας το μέγιστο δυνατό optimization (βελτιστοποίηση) στα σημεία που χρειάζεται.

Υπάρχουν κάποια εργαλεία τα οποία μας επιτρέπουν να γράψουμε εφαρμογές χρησιμοποιώντας και web γλώσσες όπως η HTML, η CSS και η Javascript. Με αυτά μπορούμε να δημιουργήσουμε εφαρμογές που θα τρέχουν μέσα σε ένα WebView, δηλαδή μία πλήρης webpage που τρέχει μέσα σε μία εφαρμογή αντί στον browser. Εργαλεία όπως τα **PhoneGap** και **AppceleratorTitanium** μας βοηθούν να γράψουμε κώδικα ο οποίος μεταγλωττίζεται στο παρασκήνιο σε nativecode. Υπάρχουν μειονεκτήματα και πλεονεκτήματα σ' αυτή την προσέγγιση και θα πρέπει να έχουμε πάντα υπόψη μας ότι χρησιμοποιώντας Java παίρνουμε τα καλύτερα αποτελέσματα, αλλά μπορούμε να φτιάξουμε εξίσου δυνατές εφαρμογές με όποιο εργαλείο μας βολεύει. Ένα ακόμη μειονέκτημα των άλλων γλωσσών προγραμματισμού είναι ότι τα 3rd party εργαλεία που χρησιμοποιούμε θα βρίσκονται πάντα ένα βήμα πίσω από τις τελευταίες εξελίξεις στον χώρο του Androiddevelopment.

Μερικά από αυτά είναι:

- το **Xamarin** για C#/.NET
- το **Ruboto** για Ruby
- το **Kivy** για Python

Είπαμε όμως ότι υπάρχουν και εργαλεία που δε χρειάζονται καθόλου γνώση προγραμματισμού. Με αυτά μπορούμε να φτιάξουμε απλές εφαρμογές αλλά και να μπούμε στη νοοτροπία του development χωρίς να ιδρώσουμε αλλά χρειάζονται επίσης μελέτη.

Αυτά είναι:

- το AppInventor για εφαρμογές και
- το Stencyl για παιχνίδια

2.5.5 Πώς θα ξεκινήσω να γράφω Android εφαρμογές

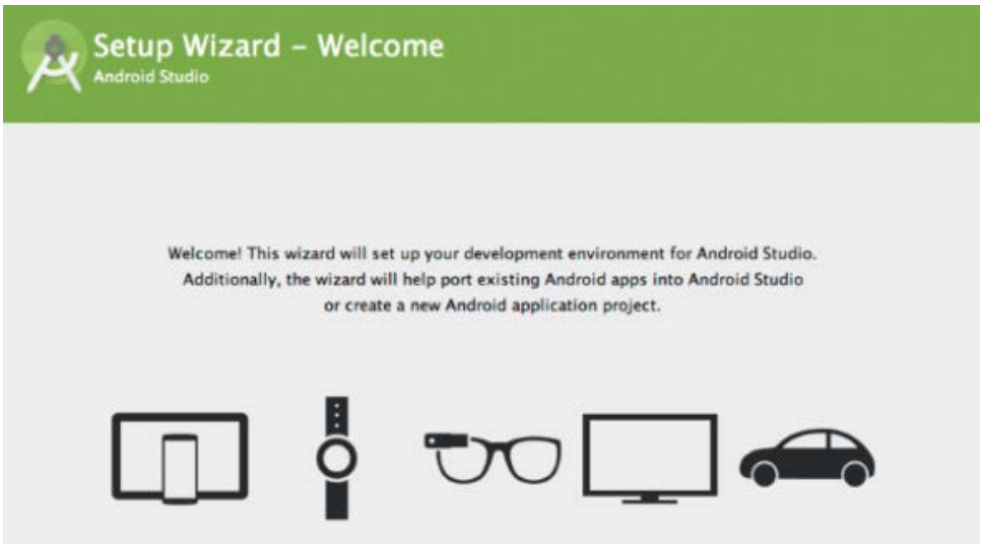
Στην επίσημη σελίδα των AndroidDeveloper θα βρούμε εκτενείς οδηγίες με θέματα όπως το πώς να ξεκινήσουμε, πώς να σχεδιάσουμε μια εφαρμογή και λεπτομέρειες σχετικά με όλα τα διαθέσιμα ApplicationProgrammingInterfaces (API) που μας δίνουν τη δυνατότητα να χρησιμοποιούμε υπηρεσίες άλλων εφαρμογών μέσα στη δική μας (π.χ. τους χάρτες του GoogleMaps). Τα εργαλεία για ανάπτυξη Android εφαρμογών είναι αυτό που λέμε cross-platform, δηλαδή διαθέσιμα για Windows, OSX και Linux, συνεπώς δεν απαιτείται η αγορά κάποιου επιπλέον hardware από εμάς.

Δεν υπάρχουν συγκεκριμένες οδηγίες στο πως θα στήσουμε το προγραμματιστικό περιβάλλον για Androiddevelopment, αλλά η διαδικασία είναι πάνω-κάτω η ίδια, αφού κάθε φορά πολύ μικρά πράγματα είναι αυτά που αλλάζουν. Η λογική σειρά των βημάτων που πρέπει να ακολουθήσουμε είναι η εξής:

1. Εγκαθιστούμε την Java στον υπολογιστή μας (μπορεί να είναι ήδη εγκατεστημένη).
2. Εγκαθιστούμε το Android SDK.
3. Εγκαθιστούμε το Eclipse, το οποίο είναι το Integrated Development Environment (IDE) που προτείνεται για Android development. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε και το Android Studio.
4. Εγκαθιστούμε το Android Development Tools (ADT) plugin.
5. Κατεβάζουμε και τα υπόλοιπα SDKcomponent του SDKManager.

2.6 AndroidStudio

Το AndroidStudio είναι η πρώτη επίσημη πλατφόρμα της εταιρείας που βασίζεται στο Intelli JIDEAJAVAIDE και οι developers θα έχουν την ευχέρεια επιλογής 4 διαφορετικών καναλιών (Stable, Beta, Dev, Canary) ανάλογα με το πόσο γρήγορα θέλουν να λαμβάνουν τις νέες λειτουργίες.



Εικόνα 17 - Android Studio

Το Android Studio φτιάχτηκε για να αντικαταστήσει το Eclipse με ADT, και βρίσκεται πλέον σε stable έκδοση. Μερικές από τις ευκολίες που προσφέρει το Android Studio είναι οι εξής:

Πρότυπα κώδικα για εύκολη δημιουργία κοινών λειτουργιών, πλούσιο layout editor με υποστήριξη για drag & drop theme editing, εργαλεία Lint για τη πρόληψη προβλημάτων (performance, usability, version compatibility), ProGuard και app-signing δυνατότητες, ενσωματωμένη υποστήριξη για την πλατφόρμα Google Cloud και άλλα πολλά.

Επιπλέον, η Google παρέχει οδηγίες μετάβασης από το Eclipse (περιβάλλον που χρησιμοποιούσαν αρκετοί Androiddevelopers μέχρι σήμερα) στο AndroidStudio, ενώ στη επίσημη ιστοσελίδα θα βρείτε όλες τις πληροφορίες για τις νέες λειτουργίες που είναι διαθέσιμες στην τελική έκδοση.

Μερικές από αυτές είναι οι:

- Intelligent code editor
- Code templates and GitHub integration
- Multi-screen app development
- Virtual devices of all shapes and sizes
- Flexible Gradle-based build system
- Expanded template support for Google Services and various device types

- Built-in support for Google Cloud Platform
- ProGuard and app-signing capabilities
- Support for creating multiple APKs with different features in the same project

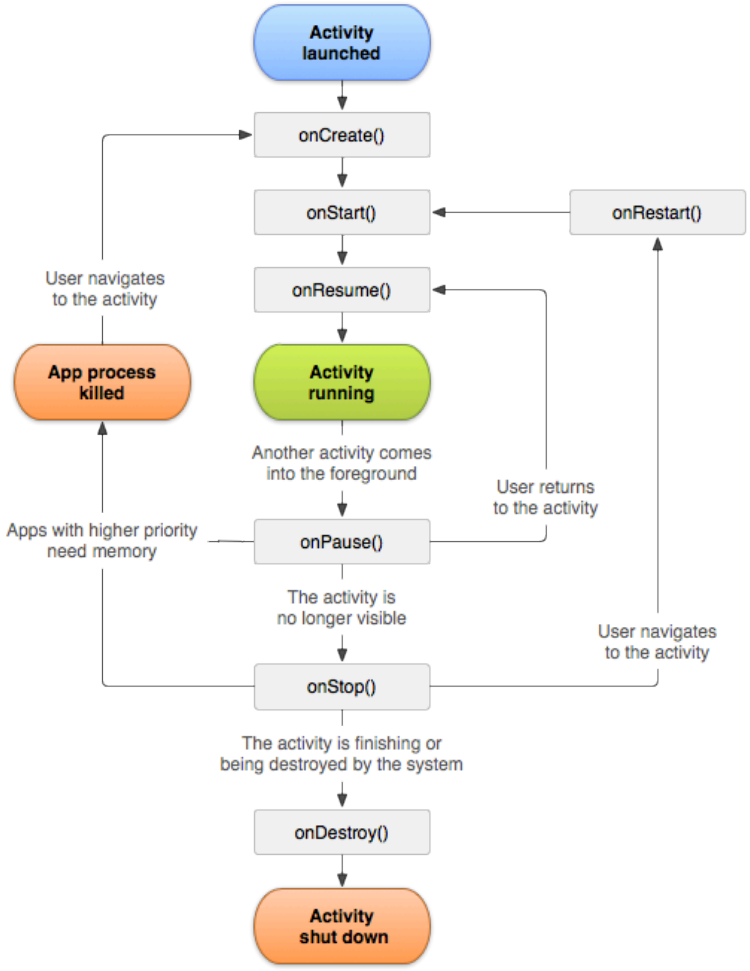
Κύκλος ζωής Activity

Τα activities στο σύστημα διαχειρίζονται ως μια στοίβα δραστηριοτήτων. Όταν μια νέα activity έχει ξεκινήσει, τοποθετείται στην κορυφή της στοίβας και γίνεται η activity που τρέχει εκείνη τη στιγμή - η προηγούμενη activityπαραμένει πάντα κάτω στη στοίβα, και δεν θα έρθει στο προσκήνιο και μέχρι τις νέα activity υπάρξει.

Μια activityέχει ουσιαστικά τέσσερις καταστάσεις:

- Αν μια activity του προσκηνίου (στο επάνω μέρος της στοίβας), είναι ενεργή ή τρέχει.
- Αν μια activityέχει χάσει την εστίαση, αλλά εξακολουθεί να είναι ορατή (δηλαδή, μια νέα μη-πλήρους μεγέθους ή διαφανές activityέχει επικεντρωθεί στην κορυφή της δραστηριότητάς σας), είναι σε παύση. Μια activity που είναι σε pause είναι ζωντανή (διατηρεί όλες τις πληροφοριών και θα παραμείνει προσκολλημένος στο διαχειριστή παραθύρων), αλλά μπορεί να σκοτωθεί από το σύστημα σε ακραίες καταστάσεις χαμηλής μνήμης.
- Αν μια activity είναι εντελώς επισκιασμένη από μια άλλη activity, έχει διακοπεί. Εξακολουθεί να διατηρεί όλες τις πληροφορίες, ωστόσο, δεν είναι πλέον ορατή στο χρήστη, είναι κρυφό παράθυρο και συχνά θα πρέπει να θανατωθούν από το σύστημα όταν η μνήμη είναι απαραίτητη αλλού.
- Αν μια activityέχει διακοπεί οριστικά ή προσωρινά, το σύστημα μπορεί να εγκαταλείψει τη activity από τη μνήμη είτε ζητώντας να τελειώσει, ή απλά τη θανάτωση του. Όταν εμφανίζεται και πάλι στο χρήστη, θα πρέπει να επανεκκινηθεί πλήρως και να αποκατασταθεί στην προηγούμενη κατάσταση.

Το παρακάτω διάγραμμα δείχνει τα σημαντικά μονοπάτια μιας activity. Τα τετράγωνα ορθογώνια αντιπροσωπεύουν τις μεθόδους επανάκτησης μπορείτε να εφαρμόσετε για να εκτελέσετε λειτουργίες, όταν η δραστηριότητα κινείται μεταξύ των κρατών. Τα χρωματιστά οβάλ είναι μεγάλα κράτη η δραστηριότητα μπορεί να είναι.



Εικόνα 18 Μονοπάτια ενός activity

AndroidManifest

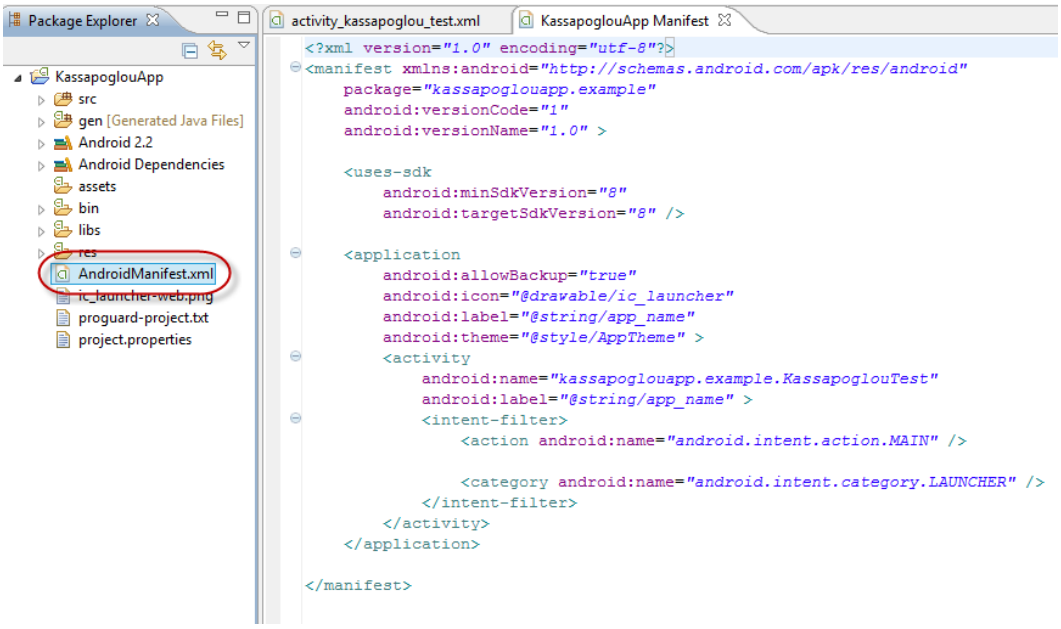
Το AndroidManifest αρχείο είναι το ένα και μοναδικό αρχείο που χρειάζεται κάθε Android εφαρμογή για να ξεκινήσει. Από την κατάληξη καταλαβαίνουμε ότι κάθε εντολή που περιέχει είναι γραμμένη σε XML μορφή. Το AndroidManifest.xml αρχείο είναι από εκεί που αρχικά διαβάζει το Android λειτουργικό για να ξέρει πώς να εκτελέσει την εφαρμογή μας με όλα τα στοιχεία που την αποτελούν.

Επίσης προσφέρει τις παρακάτω υπηρεσίες:

- Ονομάζει και αναγνωρίζει τα java πακέτα τις εφαρμογής μέσα στα οποία βρίσκονται οι java κλάσεις.
- Περιγράφει όλα τα στοιχεία από τα οποία αποτελείται μια εφαρμογή συμπεριλαμβανομένων και των ακόλουθων στοιχείων – activities, services, broadcastreceivers και contentproviders
- Προσδιορίζει τα processes που θα χρειαστούν να υποστηρίξουν την εφαρμογή.
- Προσδιορίζει τα δικαιώματα τα οποία πρέπει να έχει η τρέχον εφαρμογή για να μπορεί να αλληλοεπιδρά με άλλες εφαρμογές.

- Ορίζει τα δικαιώματα που άλλες εφαρμογές πρέπει να έχουν για να χρησιμοποιήσουν την εφαρμογή μας.
- Δηλώνει το ελάχιστο επίπεδο του AndroidAPI που απαιτεί η εφαρμογή.
- Παραθέτει τις βιβλιοθήκες που χρειάζεται η εφαρμογή για να τρέξει.
- Και τέλος καλεί και χρησιμοποιεί τις Instrumentation κλάσεις που παρέχουν πληροφορίες για την εκτέλεση και λειτουργία της εφαρμογής.

Κάνοντας διπλό κλικ επάνω στο αρχείο AndroidManifest.xml μπορούμε να δούμε το περιεχόμενο του.



Εικόνα 19 AndroidManifest

Από όλα αυτά τα elements δύο είναι αυτά που απαιτούνται – το *manifest* και το application. Αυτά τα δύο elements πρέπει πάντα να υπάρχουν και να εμφανίζονται μόνο μια φορά μέσα στο αρχείο. Οτιδήποτε άλλο element μπορεί να είναι γραμμένο περισσότερο από μια φορά ή και καθόλου.

Η γενική μορφή που μπορεί να έχει ένα element *manifest* είναι η εξής:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="string"
          android:sharedUserId="string"
          android:sharedUserLabel="string resource"
          android:versionCode="integer"
          android:versionName="string"
          android:installLocation=["auto" | "internalOnly" | "preferExternal"] >
    . . .
</manifest>
```

Εικόνα 20 element manifest γενικό παράδειγμα

Και στο δικό μας παράδειγμα έχει την ακόλουθη μορφή:

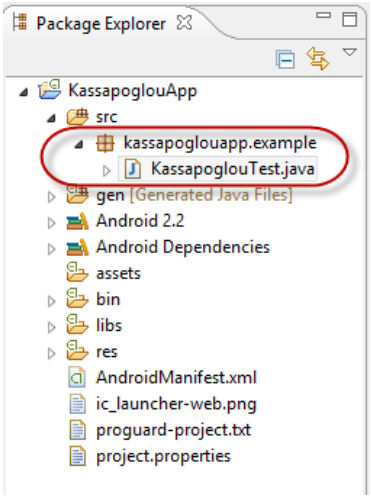
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kassapoglouapp.example"
    android:versionCode="1"
    android:versionName="1.0" >
```

Εικόνα 21 element manifest στον κώδικα μας

Τι είναι λοιπόν το *manifest*? Είναι το rootelement του αρχείου AndroidManifest.xml και πρέπει να περιέχει ένα *application* element. Ορίζει δύο attributes: το xmlns:android και το package.

Ορίζει το Androidnamespace και πρέπει πάντα να δείχνει στο
http://schemas.android.com/apk/res/android

Είναι το java πακέτο μέσα στο οποίο βρίσκεται η εκτελέσιμη java κλάση της εφαρμογής



Εικόνα 22 εκτελέσιμη java κλάση

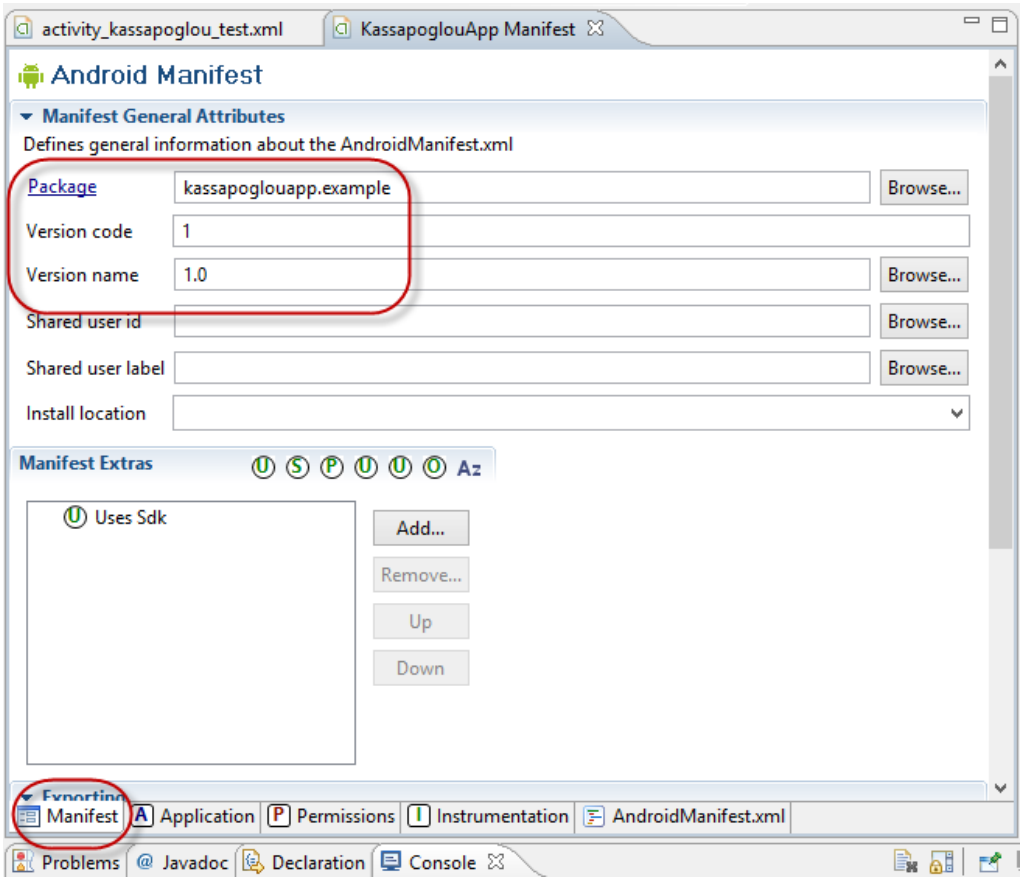
Android: versionCode

Αυτός είναι ένας εσωτερικός αριθμός που δείχνει την σε ποια έκδοση της εφαρμογής βρισκόμαστε. Το android:versionCode δεν εμφανίζεται στους χρήστες. Όσο μεγαλύτερος είναι ο αριθμός τόσο πιο πρόσφατη είναι η έκδοση της εφαρμογής. Ο αριθμός πρέπει να είναι ακέραιος, όπως π.χ. 4.

Android: versionName

Αυτός ο αριθμός εμφανίζεται στους χρήστες. Ο λόγος ύπαρξης αυτής της παραμέτρου δεν είναι κανένας άλλος από το να δείχνει στους χρήστες την έκδοση της εφαρμογής που τρέχουν οι χρήστες.

Για να αλλάξουμε την τιμή του android:versionCode όπως και εκείνη του android: versionName δεν έχετε παρά να αλλάξετε τον αριθμό απευθείας στο xml αρχείο ή πολύ πιο απλά πατήστε στο tab με το όνομα manifest και έχετε ένα παραθυρικό περιβάλλον διαχείρισης αυτών των μεταβλητών.



Εικόνα 23 περιβάλλον διαχείρισης των μεταβλητών

AndroidManifest - *uses-sdk*.

Ο ρόλος του *uses-sdk* είναι να τηρεί την συμβατότητα της εφαρμογής της οποίας αναπτύσσουμε με μία ή περισσότερες εκδόσεις της πλατφόρμας του Android μέσω του αριθμού API. Παρά το όνομα του, αυτό το στοιχείο χρησιμοποιείται για να καθορίσει το επίπεδο API και όχι τον αριθμό έκδοσης του SDK (software development kit). Στο παράδειγμα μας, ο αριθμός του API είναι 8 που σημαίνει ότι είμαστε συμβατοί τουλάχιστον με συσκευές Android με λειτουργικό έκδοσης 2.2 ή υψηλότερο.

Android: minSdkVersion

```
<uses-sdk android:minSdkVersion="integer"
          android:targetSdkVersion="integer"
          android:maxSdkVersion="integer" />
```

Εικόνα 24 Android Manifest

Είναι ο ακέραιος αριθμός που ορίζει το ελάχιστο επίπεδο API που απαιτείται για να τρέξει η εφαρμογή. Το λειτουργικό Android θα απαγορέψει το χρήστη να εγκαταστήσει οποιαδήποτε εφαρμογή αν το επίπεδο του API του συστήματος είναι χαμηλότερο από την τιμή που καθορίζεται στην android:minSdkVersion παράμετρο.

Android: targetSdkVersion

Ορίζει το επίπεδο API στο οποίο στοχεύει η εφαρμογή. Εάν δεν έχει οριστεί τότε η τιμή του είναι ίδια με εκείνη του minSdkVersion.

```
<application android:allowTaskReparenting=["true" | "false"]
             android:backupAgent="string"
             android:debuggable=["true" | "false"]
             android:description="string resource"
             android:enabled=["true" | "false"]
             android:hasCode=["true" | "false"]
             android:hardwareAccelerated=["true" | "false"]
             android:icon="drawable resource"
             android:killAfterRestore=["true" | "false"]
             android:largeHeap=["true" | "false"]
             android:label="string resource"
             android:logo="drawable resource"
             android:manageSpaceActivity="string"
             android:name="string"
             android:permission="string"
             android:persistent=["true" | "false"]
             android:process="string"
             android:restoreAnyVersion=["true" | "false"]
             android:supportsRtl=["true" | "false"]
             android:taskAffinity="string"
             android:theme="resource or theme"
             android:uiOptions=["none" | "splitActionBarWhenNarrow"] >
    . . .
</application>
```

Εικόνα 25 Android targetSdk Version

Αυτό το στοιχείο περιέχει σημαντικά στοιχεία που επηρεάζουν όλα τα κύρια χαρακτηριστικά της Android εφαρμογής όπως το εικονίδιο της εφαρμογής, την άδεια και ασφάλεια λειτουργίας της, και πολλές άλλες προκαθορισμένες τιμές. Πάμε όμως να δούμε πιο αναλυτικά τη δομή του στοιχείου. Στην πιο κάτω εικόνα βλέπουμε πως είναι οι default παράμετροι για την δική μας απλή εφαρμογή.

Android: allowBackup

Η τιμή "True" επιτρέπει στον χρήστη της εφαρμογής να κάνει backup την ίδια την εφαρμογή για να μπορέσει να την ανακτήσει ξανά.

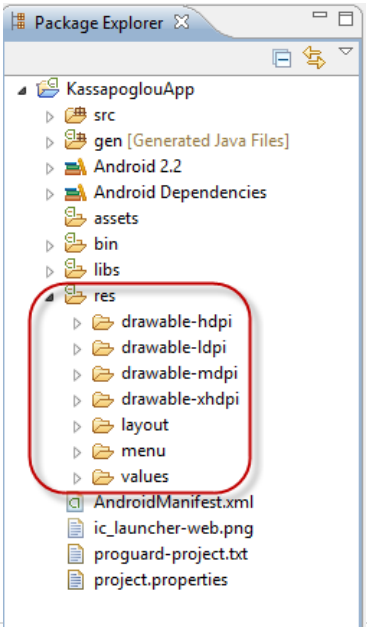
activity_kassapoglou_test.xml | KassapoglouApp Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kassapoglouapp.example"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="8" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="kassapoglouapp.example.KassapoglouTest"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Εικόνα 26 Android allowBackup

Android: icon

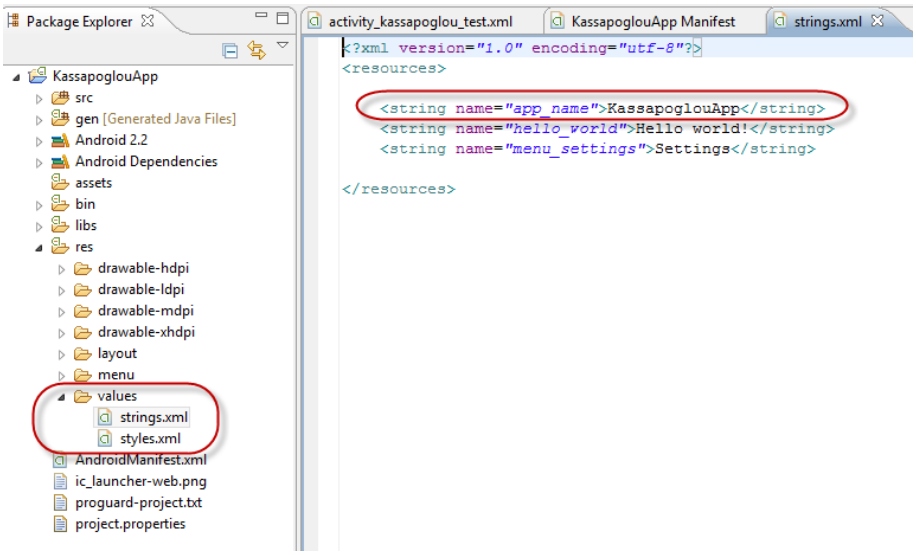
Εδώ ορίζεται το path του φάκελου από τον οποίο θα προέλθει το εικονίδιο της εφαρμογής. Εάν δεν ορίσουμε εμείς κάποιο εικονίδιο τότε η εφαρμογή θα βάλει το δικό της default που είναι το εικονίδιο του Android. Το εικονίδιο με το οποίο παρουσιάζεται η εφαρμογή όταν είναι εγκαταστημένη προέρχεται από ένα φάκελο το ο όνομα drawable. Για την εξυπηρέτηση όμως όλων των κινητών με διαφορετικά μεγέθη οθόνης, δημιουργούμε πολλαπλούς drawable φακέλους έτσι ώστε μέσα σε κάθε φάκελο έχουμε και το αντίστοιχο μέγεθος σε pixel του εικονιδίου. Οι φάκελοι drawable βρίσκονται μέσα στο φάκελο res.



Εικόνα 27 Android icon

Android: label

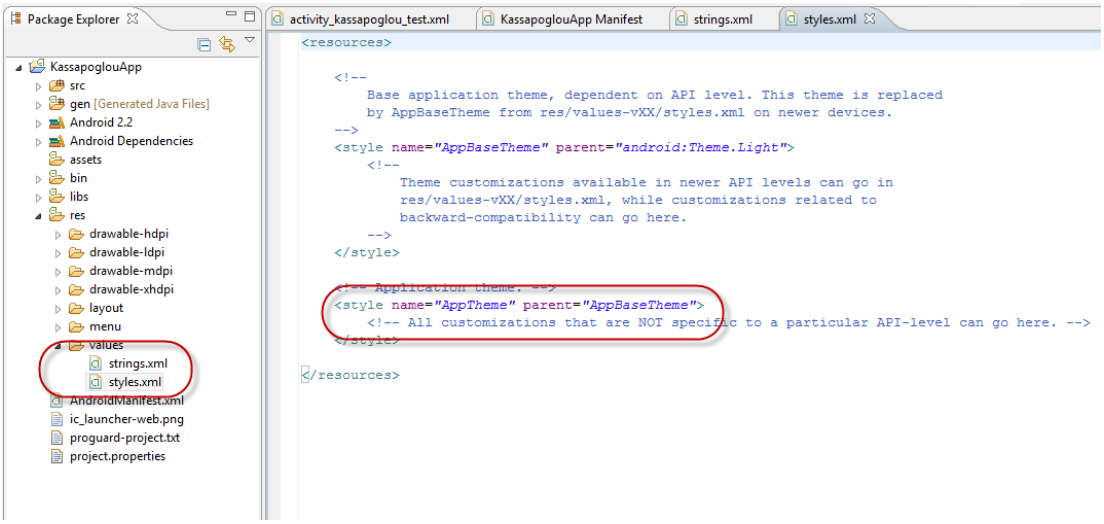
Εδώ ορίζεται το όνομα με το οποίο θα εμφανίζεται η εφαρμογή. Το όνομα προέρχεται από το strings.xml αρχείο που βρίσκεται μέσα στο φάκελο values.Στην πιο κάτω εικόνα βλέπουμε ότι το Androidmanifest αρχείο διαβάζει την μεταβλητή string με το όνομα app_name.



Εικόνα 28 Android label

Android: theme

Αναφέρεται στο defaulttheme της εφαρμογής το οποίο προέρχεται από το αρχείο styles.xml από τον φάκελο values όπως δείχνει και η πιο κάτω εικόνα.



Εικόνα 29 Android theme

Activity

Το *activity* ορίζει το που υπάρχει η javaactivity κλάση ή οποία είναι υπεύθυνη για την λειτουργία της εφαρμογής και για την δημιουργία του visualinterface που θα εμφανιστεί στην οθόνη της συσκευής. Με άλλα λόγια, το AndroidManifest είναι το αρχείο στο οποίο κοιτάει αρχικά η Android πλατφόρμα για να μπορέσει να βρει την activity κλάση που περιέχει το java πρόγραμμα εκκίνησης της εφαρμογής και να εμφανιστεί στην οθόνη μας. Οπότε τώρα έχει κάποια λογική το γεγονός ότι χρειάζονται οπωσδήποτε δύο παράμετροι να οριστούν μέσα στο *activity*– το android:name και το android:label.

Android: name

Αυτό είναι το όνομα της κλάσης μέσα στην οποία υλοποιείται το activity. Το όνομα πρέπει να είναι σε fullyqualifiedclass μορφή δίνοντας ακριβώς την τοποθεσία της κλάσης.

2.7 NetBeans

Εικόνα 30 –Λογότυπο Netbeans

Το NetBeans είναι ένα επιτυχημένο ερευνητικό έργο ανοιχτής πηγής (opensource) με μεγάλο αριθμό χρηστών, μια αναπτυσσόμενη κοινωνία, κοντά στους 100 συνεργάτες παγκοσμίως. Η SunMicrosystems ίδρυσε το ερευνητικό έργο ανοιχτής πηγής NetBeans τον Ιούνιο του 2000 και συνεχίζει να είναι ο κύριος ανάδοχος.

Το NetBeans είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης , IDE ή , για τη δημιουργία προγραμμάτων ηλεκτρονικών υπολογιστών σε μια σειρά από διαφορετικές γλώσσες. NetBeans ανάπτυξης αναφέρεται στη διαδικασία της χρησιμοποιώντας το NetBeans για να δημιουργήσουμε, να επεξεργαστούμε και να οργανώσουμε τον κωδικό σας, όπως μπορούμε να αναπτύξουμε ένα πρόγραμμα υπολογιστή. NetBeans ανάπτυξης μπορούν επίσης να αναφέρουν τη χρήση της πλατφόρμας NetBeans ως ένα πλαίσιο λογισμικού για τη δημιουργία νέων εφαρμογών.

Ανάπτυξη Γλώσσας

Το NetBeans υποστηρίζει την ανάπτυξη σε Java , PHP , HTML , JavaScript , CSS , Groovy και C + + . Στο πλαίσιο αυτό , η ανάπτυξη αναφέρεται στην υποστήριξη της κωδικοποίησης , τον εντοπισμό σφαλμάτων και την κατάρτιση κώδικα σε αυτές τις γλώσσες . Στη φάση της κωδικοποίησης , NetBeans επαληθεύει τον κωδικό σας σε πραγματικό χρόνο για να διασφαλιστεί η σωστή σύνταξη . Το NetBeans περιλαμβάνει μια σειρά από εργαλεία εντοπισμού σφαλμάτων για να μας βοηθάει να απομονώσουμε και να διορθώσουμε την σύνταξη που σχετίζονται με σφάλματα.

ΕικόνεςProjectManagement

Το NetBeans ομαδοποιεί όλα τα απαραίτητα αρχεία για την ανάπτυξη μιας ενιαίας αίτησης σε ένα αρχείο project. Τα αρχεία αυτά αποτελούνται από αρχικό κώδικα, μαζί με οποιοδήποτε εισαγόμενο κώδικα που μπορεί να εξαρτάται από αυτό. Η διαχείριση του έργου NetBeans καθιστά ευκολότερο να εργαστεί σε μεγάλα έργα με την άμεση δείχνοντάς σας πώς μια αλλαγή σε ένα μέρος του προγράμματος θα επηρεάσει το υπόλοιπο του προγράμματος .

Η Χαρακτηριστικά Κωδικοποίηση

Το NetBeans απλοποιεί τη διαδικασία κωδικοποίησης από το χρώμα κωδικοποίησης λέξεις-κλειδιά, αυτόματη μορφοποίηση των εισροών σας και τον έλεγχο του κώδικά σας για τη σωστή σύνταξη. Η αυτόματη διόρθωση τουNetBeansμπορεί να λύσει αυτόματα μια σειρά από κοινά προβλήματα κωδικοποίησης , συμπεριλαμβανομένης της αποτυχίας για την εισαγωγή σε μια τάξη , ανακριβείς δηλώσεις μεταβλητών και ελλειπουσών δηλώσεων επιστροφής . Η λειτουργία αυτόματης συμπλήρωσης θα εμφανίσει μια λίστα με όλες τις μεταβλητές και τις μεθόδους μέσα σε ένα αντικείμενο.

Εικόνων Πλατφόρμα

Με την πλατφόρμα NetBeans , μπορούμε να δημιουργήσουμε σύνθετες εφαρμογές Java μακριά από μια υπάρχουσα πλατφόρμα αντί να ξεκινάμε από το μηδέν . Η πλατφόρμα NetBeans χειρίζεται βασικές λειτουργίες εφαρμογής όπως η εξοικονόμηση την κατάσταση του προγράμματος , τη διαχείριση της γραφική διεπαφή χρήστη και την παροχή πρόσβασης στο σύστημα αρχείων . Η πλατφόρμα είναι ένα modular , επιτρέποντάς μας να χρησιμοποιήσουμε τις λειτουργίες που χρειαζόμαστε.

NetBeansIDE και NetBeansPlatform

Σήμερα δύο ερευνητικά έργα υπάρχουν: Το NetBeansIDE και το NetBeansPlatform.
Το NetBeansIDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE - ένα εργαλείο στους προγραμματιστές για να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java - αλλά μπορεί να υποστηρίξει όλες τις γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υποομάδων (modules) που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeansIDE. ToNetBeansIDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του.
Διαθέσιμο επίσης είναι το NetBeansPlatform; ένα εκτατό θεμέλιο αποτελούμενο από υπομονάδες (modular) που χρησιμοποιείται σαν βάση λογισμικού για τη δημιουργία μεγάλων επιτραπέζιων (desktop) εφαρμογών. Οι ISV συνεργάτες διαθέτουν προσθήκες, επιπρόσθετα

προγράμματα (plug-ins) που εύκολα συνενώνονται στο Platform και μπορούν επίσης να χρησιμοποιηθούν για την ανάπτυξη άλλων εργαλείων και λύσεων.
Και τα δύο τα προϊόντα είναι ανοιχτής πηγής (opensource) και ελεύθερα για εμπορική ή μη χρήση. Ο κώδικας πηγής (sourcecode) είναι διαθέσιμος για επαναχρησιμοποίηση κάτω από το Common Development and Distribution License (CDDL).

Netbeans.org

Το netbeans.org είναι το σπίτι της NetBeans κοινότητα ανοιχτής πηγής (opensourcecommunity) η οποία είναι αφοσιωμένη στο χτίσιμο ενός παγκοσμίας τάξεως IDE. Το netbeans.org επιτρέπει στους χρήστες περισσότερων από 160 χωρών παγκοσμίως να είναι σε επαφή με πηγές γνώσεων και άτομα που περιβάλλουν το NetBeans. Εδώ μπορείτε να κατεβάσετε τις τελευταίες εκδόσεις του NetBeans, να διαβάσετε online έγγραφα υποστήριξης, να καλλιεργήσετε τις προσωπικές σας γνώσεις πάνω στη Java, να ενημερώνεστε για τις τελευταίες εξελίξεις, να λάβετε μέρος σε mailinglist, να συνεισφέρετε με δικό σας κώδικα, να μάθετε για τα άτομα που συνδέονται με το ερευνητικό έργο, να κάνετε νέες επαφές, και πολλά άλλα.

2.8 Βάση Δεδομένων

Μια βάση δεδομένων (DataBase) αποτελεί μία οργανωμένη συλλογή ειδικά ταξινομημένων δεδομένων, σχεδιασμένη με τέτοιο τρόπο, ώστε να μπορεί να εξυπηρετήσει αποτελεσματικά πολλές εφαρμογές, μειώνοντας τις άσκοπες επαναλήψεις των δεδομένων. Αντί να δημιουργούνται ξεχωριστά αρχεία για κάθε εφαρμογή, όλα τα δεδομένα καταχωρούνται με τέτοια μέθοδο, ώστε να παρουσιάζονται στο χρήστη με έναν ενιαίο τρόπο. Κάθε εφαρμογή μπορεί να κάνει χρήση εκείνων των δεδομένων που είναι απαραίτητα για την επεξεργασία και την παραγωγή των αντίστοιχων πληροφοριών. Τα απλά συστήματα επεξεργασίας αρχείων ικανοποιούν πολλές απαιτήσεις σε επιχειρήσεις και οργανισμούς και κυρίως σε περιπτώσεις όπου υπάρχει μικρός αριθμός χρηστών και τα δεδομένα μπορούν να κατανεμηθούν εύκολα χωρίς να απαιτείται δημιουργία ολοκληρωμένων Πληροφοριακών Συστημάτων. Σε άλλες όμως περιπτώσεις, όπως σε ορισμένα συστήματα συναλλαγών, στα Πληροφοριακά Συστήματα Διοίκησης και στα Συστήματα Υποστήριξης Αποφάσεων, τα συστήματα επεξεργασίας αρχείων δεν μπορούν να ικανοποιήσουν τις απαιτήσεις καθώς όλα αυτά τα συστήματα περιλαμβάνουν μεγάλο αριθμό χρηστών, πολλά αρχεία με δεδομένα που σχετίζονται μεταξύ τους και διαφορετικές όψεις των ίδιων δεδομένων. Ακόμη περισσότερο, χρήστες αυτών των συστημάτων απαιτούν γρήγορες απαντήσεις σε μη προσχεδιασμένες ερωτήσεις μια λειτουργία που δεν υποστηρίζεται από τα απλά συστήματα επεξεργασίας.

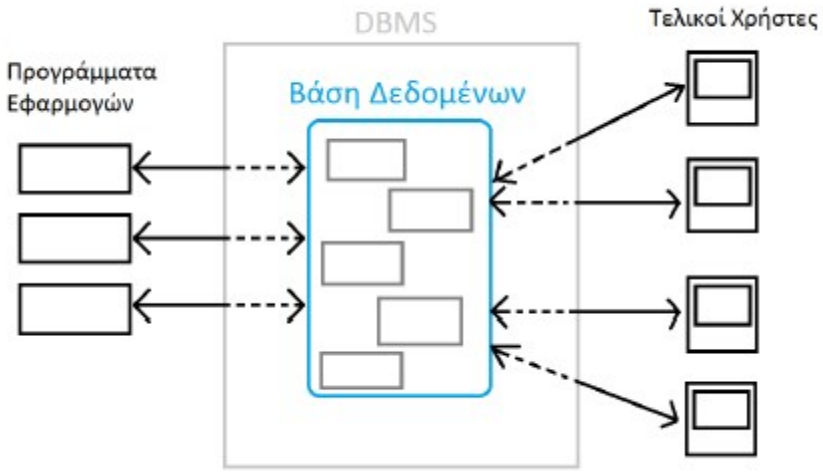
Τι σημαίνει βάση δεδομένων;

Με την απλούστερη έννοια, βάση δεδομένων (database) είναι μια συλλογή από εγγραφές και αρχεία, τα οποία είναι οργανωμένα έτσι ώστε να εξυπηρετούν ένα συγκεκριμένο σκοπό. Στον υπολογιστή σας μπορεί να κρατάτε τα ονόματα και τις διευθύνσεις όλων των φίλων ή πελατών σας. Ίσως να συλλέγετε και όλες τις επιστολές που γράφετε και να τις οργανώνετε με βάση τον παραλήπτη. Μπορεί να έχετε κι ένα άλλο σύνολο αρχείων στα οποία κρατάτε όλα σας τα οικονομικά στοιχεία-πελάτες ή προμηθευτές ή τις καταχωρήσεις των βιβλίων σας και τους ισολογισμού σας .Τα έγραφα του επεξεργαστή κειμένου που οργανώνετε με βάση το θέμα τους , είναι άλλο ένα είδος βάσης δεδομένων. Τα αρχεία του λογιστικού φύλλου που οργανώνετε σύμφωνα με τις χρήσεις τους, αποτελούν κι αυτά ένα είδος βάσης δεδομένων. Αν είστε πολύ οργανωμένοι, πιθανώς να μπορείτε να χειρίζεστε πολλές εκατοντάδες λογιστικών φύλλων, χρησιμοποιώντας φακέλους και υποφακέλους. Όταν το κάνετε αυτό, είστε εσείς οι διαχειριστές της βάσης δεδομένων. Ωστόσο, τι μπορείτε να κάνετε όταν τα προβλήματα που προσπαθείτε να λύσετε μεγαλώσουν πολύ; Πώς μπορείτε να συλλέγετε εύκολα πληροφορίες για όλους σας τους πελάτες και τις παραγγελίες τους όταν τα δεδομένα είναι αποθηκευμένα σε πολλά έγραφα και αρχεία λογιστικών φύλλων; Πώς μπορείτε να διατηρείτε συνδέσμους μεταξύ των αρχείων όταν εισάγετε νέες πληροφορίες; Πώς εξασφαλίζετε ότι τα δεδομένα εισάγονται σωστά; Τι γίνεται αν πρέπει να μοιραστείτε τις πληροφορίες σας με πολλά άτομα, αλλά δε θέλετε να μπορούν δυο άτομα να προσπαθήσουν να ενημερώσουν τα ίδια δεδομένα την ίδια στιγμή; Επειδή ακριβώς έχουμε να αντιμετωπίσουμε όλες αυτές τις προκλήσεις, χρειάζεστε ένα σύστημα διαχείρισης βάσεων δεδομένων.

Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)

Τα Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ),αγγλικός όρος DataBaseManagementSystems (DBMS), είναι προγράμματα τα οποία ακολουθούν μία πιο πολύπλοκη προσέγγιση στην αποθήκευση και ανάκτηση δεδομένων από αυτή που ακολουθούν τα απλά συστήματα επεξεργασίας αρχείων (FileManagementSystem-FMS). Στην πραγματικότητα, τα ΣΔΒΔ αρχίζουν εκεί που σταματούν τα απλά συστήματα επεξεργασίας αρχείων. Τα ΣΔΒΔ εκμεταλλευόμενα την ταχύτητα και την ευελιξία που προσφέρει η μέθοδος της άμεσης προσπέλασης, όσον αφορά την αποθήκευση και την ανάκτηση δεδομένων, οργανώνουν τα δεδομένα με πολύ διαφορετικό τρόπο από ότι τα απλά συστήματα. Ένα από τα μεγαλύτερα πλεονεκτήματα των ΣΔΒΔ είναι ότι το σύστημα βρίσκει και ανακτά τα δεδομένα χωρίς τη μεσολάβηση του χρήστη. Ακόμη και οι προγραμματιστές Η/Υ δεν απαιτείται να γνωρίζουν την τοποθεσία και την μορφή των δεδομένων καθώς μέσω των προγραμμάτων ζητούν να ανεβρεθούν τα δεδομένα και το σύστημα αναλαμβάνει να τα ανακτήσει για λογαριασμό τους. Τα συστήματα επεξεργασίας αρχείων όμως λειτουργούν διαφορετικά και με τρόπο που μοιάζει με την αναζήτηση ενός βιβλίου σε μια βιβλιοθήκη όπου βρίσκουμε κάποιο βιβλίο είτε ψάχνοντας τα ράφια (επεξεργασία μέσω σειριακής αναζήτησης), είτε ψάχνοντας στις καρτέλες της βιβλιοθήκης (επεξεργασία μέσω ευρετηρίου και άμεσης προσπέλασης). Όποιο τρόπο και αν επιλέξουμε από τους δύο θα πρέπει μόνοι μας να δούμε στα ράφια της βιβλιοθήκης για να πάρουμε το βιβλίο. Αντίστοιχα, με ένα ΣΔΒΔ είναι σαν να ζητούμε από τον υπάλληλο της βιβλιοθήκης να βρει το βιβλίο που επιθυμούμε και να μας το φέρει προσδιορίζοντας το τίτλο, το συγγραφέα και το έτος έκδοσης. Η διαδικασία που ακολουθεί ο υπάλληλος και ο τρόπος οργάνωσης των βιβλίων στα ράφια δεν

μας ενδιαφέρουν, δηλαδή δεν μας ενδιαφέρει αν ο υπάλληλος χρησιμοποίησε τις καρτέλες της βιβλιοθήκης για να βρει τη θέση του βιβλίου και αν τα βιβλία είναι οργανωμένα στα ράφια βάσει του τίτλου, του συγγραφέα, της ημερομηνίας έκδοσης ή του μεγέθους τους. Ακόμη περισσότερο μπορούμε με την ίδια ευκολία να ζητήσουμε από τον υπάλληλο να μας φέρει όλα τα βιβλία ενός συγκεκριμένου συγγραφέα ή τα άρθρα που δημοσιεύτηκαν πάνω σε ένα συγκεκριμένο θέμα την τελευταία πενταετία χωρίς να μας ενδιαφέρει πως θα γίνει αυτό. 10 Αν και τα ΣΔΒΔ παρέχουν ισχύ και ελαστικότητα, πέραν κάθε συμβατικού συστήματος, όλα τα συστήματα που υποστηρίζονται από Η/Υ δεν χρησιμοποιούν την τεχνολογία αυτή κυρίως εξαιτίας του κόστους που τα συνοδεύει, Ένας άλλος λόγος είναι ότι πολλά συστήματα αναπτύχθηκαν πριν την εμφάνιση της τεχνολογίας των Βάσεων Δεδομένων και οι επιχειρήσεις ή οργανισμοί δείχνουν απροθυμία να αλλάξουν τον τρόπο λειτουργίας τους. Ένας ακόμη λόγος είναι η πολυπλοκότητα ανάπτυξης και συντήρησης ενός τέτοιου συστήματος. Η απόφαση ανάπτυξης ενός ΣΔΒΔ πρέπει να είναι απόρροια μίας ανάλυσης κόστους /οφέλους καθώς τόσο τα ΣΔΒΔ όσο και τα απλά συστήματα επεξεργασίας αρχείων είναι κατάλληλα για διαφορετικές περιπτώσεις. Ο γενικός κανόνας που πρέπει να ακολουθείται στην επιλογή συστήματος είναι να ικανοποιούνται οι απαιτήσεις των χρηστών με το μικρότερο δυνατό κόστος Ένα ΣΔΒΔ είναι απλά το Λογισμικό το οποίο υποστηρίζει την συγκέντρωση των δεδομένων, την αποτελεσματική διαχείρισή τους και επιτρέπει τη πρόσβαση των εφαρμογών στις αποθηκευμένες αυτές πληροφορίες. Υπάρχουν τέσσερα συστατικά στον ορισμό της Βάσης Δεδομένων τα οποία πρέπει να τονιστούν ιδιαίτερα. Πρώτον, υπάρχει η φυσική υπόσταση των δεδομένων, τα οποία τηρούνται σε κάποιο μαγνητικό μέσο, συνήθως σε μαγνητικούς δίσκους. Στη Βάση Δεδομένων υπάρχουν τα δεδομένα που χρησιμοποιούνται από κάθε λειτουργική μονάδα του οργανισμού ή της επιχείρησης όπως π.χ. το Προσωπικό. Εν αντιθέσει με τα συμβατικά συστήματα, τα δεδομένα στα ΣΔΒΔ αποθηκεύονται σε μια τοποθεσία, προσδιορίζονται οριστικά και αμετάκλητα και χρησιμοποιούνται από όλες τις εφαρμογές που έχουν πρόσβαση στην συγκεκριμένη τοποθεσία. Το δεύτερο συστατικό ενός ΣΔΒΔ είναι το λογισμικό που χρησιμοποιείται διαμεσολαβητικά ανάμεσα στις εφαρμογές και στα φυσικά δεδομένα και τα αρχεία όπου αυτά είναι αποθηκευμένα.



Εικόνα 31 Σύστημα διαχείρισης Βάσης Δεδομένων

Όταν μία εφαρμογή ζητήσει κάποια πληροφορία, το ΣΔΒΔ τη βρίσκει και την μεταφέρει στην εφαρμογή χωρίς ο προγραμματιστής ή ο χρήστης να απαιτείται να προσδιορίσει την συγκεκριμένη πληροφορία ή να πει στο σύστημα που βρίσκεται. Όλες οι

εντολές αναζήτησης που χρησιμοποιούνται στις συμβατικές γλώσσες προγραμματισμού εδώ είναι περιττές.

Τα ΣΔΒΔ προσφέρουν στους προγραμματιστές μία γλώσσα προσδιορισμού δεδομένων η οποία μπορεί να χρησιμοποιηθεί για να παρέχει την σύνδεση μεταξύ εφαρμογών και αρχείων. Το τρίτο συστατικό είναι ένα σύνολο εφαρμογών όπου χρησιμοποιούνται για την ανεύρεση δεδομένων στη Βάση. Οι εφαρμογές αυτές είναι συνήθως γραμμένες σε κάποια συμβατική γλώσσα όπως η COBOL και τα δεδομένα, τα οποία απαιτούνται μέσω μίας εφαρμογής σε COBOL, ανευρίσκονται και μεταφέρονται στο ΣΔΒΔ χωρίς ο χρήστης να έχει προσδιορίσει ούτε πως, ούτε που μπορεί να βρίσκεται η πληροφορία αυτή. Ένα ΣΔΒΔ συνοδεύεται από ένα πακέτο λογισμικού το οποίο σε γενικές γραμμές περιλαμβάνει μία γλώσσα προσδιορισμού δεδομένων και μία γλώσσα διαχείρισης αυτών. Η γλώσσα προσδιορισμού των δεδομένων είναι η γλώσσα που χρησιμοποιείται από τους προγραμματιστές για την ανάπτυξη της Βάσης Δεδομένων.

2.9 Servlet

Εικόνα 32 Λοτόπυπο Servlet

Τα servlets πιο συχνά χρησιμοποιείται για: διαδικασία ή να αποθηκεύσει κλάση Java σε Java EE που συμμορφώνεται με το Java Servlet API, [2], ένα πρότυπο για την εφαρμογή Java κλάσεις που ανταποκρίνεται στα αιτήματα. Servlets θα μπορούσε κατ 'αρχήν να επικοινωνούν μέσω κάθε client-server πρωτόκολλο, αλλά χρησιμοποιούνται πιο συχνά με το πρωτόκολλο HTTP . Έτσι "servlet" χρησιμοποιείται συχνά για συντομία «HTTP servlet» Έτσι, ένας προγραμματιστής λογισμικού μπορεί να χρησιμοποιήσει ένα servlet για να προσθέσετε το δυναμικό περιεχόμενο σε ένα web server χρησιμοποιώντας την πλατφόρμα Java . Το περιεχόμενο που δημιουργείται είναι συνήθως HTML , αλλά μπορεί να είναι και άλλα στοιχεία, όπως η XML . Servlets μπορεί να διατηρήσει κατάσταση σε συνεδρία μεταβλητές σε πολλές συναλλαγές διακομιστή χρησιμοποιώντας τα HTTP cookies , ή URL ξαναγράφοντας .

Servlet ιστορία API

Έκδοση Servlet API	Κυκλοφόρησε	Πλατφόρμα	Σημαντικές αλλαγές
Servlet 3.1	Μάιος 2013	JavaEE 7	Μη-blocking I / O, το πρωτόκολλο HTTP μηχανισμό αναβάθμισης (WebSocket) [5]
Servlet 3.0	Δεκέμβριος του 2009	JavaEE 6, JavaSE 6	Pluggability, Ευκολία ανάπτυξης, ασύγχρονη Servlet, Ασφάλεια, τη μεταφόρτωση του αρχείου
Servlet 2.5	Σεπτέμβριο του 2005	JavaEE 5, JavaSE 5	Απαιτεί JavaSE 5, υποστηρίζει σχολιασμό
Servlet 2.4	Νοέμβριος 2003	J2EE 1.4, J2SE 1.3	web.xml χρησιμοποιεί τη γλώσσα XML Schema
Servlet 2.3	Αύγουστος του 2001	J2EE 1.3, J2SE 1.2	Η προσθήκη του `φίλτρου`
Servlet 2.2	Αύγουστος 1999	J2EE 1.2, J2SE 1.2	Γίνεται μέρος της J2EE, εισήγαγε ανεξάρτητες εφαρμογές web σε .war αρχεία
Servlet 2.1	Νοέμβριος 1998	Απροσδιόριστος	Πρώτη επίσημη προδιαγραφή, πρόσθεσε `RequestDispatcher` , `ServletContext`
Servlet 2.0		JDK 1.1	Μέρος της Java Servlet Development Kit 2.0
Servlet 1.0	Ιουν 1997		

Εικόνα 33 Ιστορία του Servlet

Για να αναπτύξετε και να εκτελέσετε μια servlet, ένα δοχείο web πρέπει να χρησιμοποιείται. Ένα δοχείο ιστού (επίσης γνωστό ως ένα δοχείο servlet) είναι ουσιαστικά η συνιστώσα του web server που αλληλεπιδρά με τα servlets. Το δοχείο διαδίκτυο είναι υπεύθυνη για τη διαχείριση του κύκλου ζωής των Servlets, χαρτογραφώντας μια διεύθυνση URL σε ένα συγκεκριμένο Servlet και διασφαλίζοντας ότι ο αιτών διεύθυνση URL έχει τις σωστές δικαιώματα πρόσβασης.

Το Servlet API , που περιέχεται στο πακέτο Java ιεραρχία javax.servlet , ορίζει τα αναμενόμενα αλληλεπιδράσεις του δοχείου διαδίκτυο και ένα servlet.

Ένα Servlet είναι ένα αντικείμενο το οποίο υποβάλλεται η αίτηση και δημιουργεί μια απάντηση με βάση την εν λόγω αίτηση. Το βασικό πακέτο Servlet ορίζει αντικειμένων Java να εκπροσωπεί τα αιτήματα και τις απαντήσεις servlet, καθώς και αντικείμενα που να αντανακλούν το servlet είναι παράμετροι διαμόρφωσης και το περιβάλλον εκτέλεσης. Το πακέτο javax.servlet.http ορίζει HTTP -εξειδικευμένης υποκατηγορίες των γενικών στοιχείων servlet, συμπεριλαμβανομένων των αντικειμένων της διαχείρισης συνεδρία που παρακολουθούν πολλαπλά αιτήματα και απαντήσεις μεταξύ του web server και πελάτη. Servlets μπορούν να συσκευάζονται σε ένα αρχείο WAR ως web εφαρμογή .

Servlets μπορούν να παραχθούν αυτόματα από Java Server Pages (JSP) από τον compiler JavaServer Pages . Η διαφορά μεταξύ Servlets και JSP είναι ότι τα servlets συνήθως ενσωματώσετε HTML στο εσωτερικό κώδικα Java, ενώ JSPs ενσωματώσετε κώδικα Java σε HTML. Ενώ η άμεση χρήση Servlets να παράγουν HTML (όπως φαίνεται στο παρακάτω παράδειγμα) έχει γίνει σπάνιο, το υψηλότερο επίπεδο MVC πλαίσιο web σε Java EE (JSF) εξακολουθεί να χρησιμοποιεί ρητά την τεχνολογία servlet για την αίτηση χαμηλό επίπεδο / χειρισμού μέσω του FacesServlet απάντηση . Μια κάπως μεγαλύτερα χρήση είναι η χρήση servlets σε συνδυασμό με JSPs σε ένα μοτίβο που ονομάζεται " Model 2 », η οποία είναι μια γεύση του μοντέλου-view-ελεγκτή .

Η προδιαγραφή Servlet1 δημιουργήθηκε από τη Sun Microsystems , με την έκδοση 1.0 που ολοκληρώθηκε τον Ιούνιο του 1997. Από την έκδοση 2.3, η προδιαγραφή αναπτύχθηκε υπό την Java κοινοτική διαδικασία . JSR 53 ορίζεται τόσο το Servlet 2.3 και JavaServer Σελίδα 1.2 προδιαγραφές. JSR 154 καθορίζει τις Servlet 2.4 και 2.5 προδιαγραφές. Από τις 9 του Ιουνίου του 2015, η τρέχουσα έκδοση των προδιαγραφών Servlet είναι 3,1.

Τρεις μέθοδοι είναι κεντρικής σημασίας για τη διάρκεια του κύκλου ζωής ενός servlet. Αυτά init () , υπηρεσία () , και να καταστρέψουν () . Τίθενται σε εφαρμογή από κάθε servlet και γίνεται επίκληση σε συγκεκριμένες ώρες από το διακομιστή.

Κατά το πρώτο στάδιο προετοιμασίας του Servlet του κύκλου ζωής , το δοχείο web προετοιμάζει το παράδειγμα servlet, καλώντας την init () μέθοδο, περνώντας ένα αντικείμενο που υλοποιεί το javax.servlet.ServletConfig περιβάλλον. Αυτό το αντικείμενο διαμόρφωση επιτρέπει την servlet για να αποκτήσετε πρόσβαση ονόματος-τιμής παραμέτρους αρχικοποίησης από την εφαρμογή web.

Μετά την προετοιμασία, το παράδειγμα servlet μπορεί να εξυπηρετήσει τα αιτήματα πελατών. Κάθε αίτηση εξυπηρετείται στο δικό του ξεχωριστό νήμα της. Το δοχείο web καλεί την υπηρεσία () μέθοδο του servlet για κάθε αίτηση. Η υπηρεσία () μέθοδος αυτή προσδιορίζει το είδος του αιτήματος που γίνονται και τις αποστολές τους σε ένα κατάλληλο τρόπο για να χειριστεί την αίτηση. Ο δημιουργός του servlet πρέπει να παρέχουν μια εφαρμογή για τις μεθόδους αυτές.

Αν μια αίτηση γίνεται για μια μέθοδο που δεν υλοποιείται από το servlet, η μέθοδος της γονικής κλάσης ονομάζεται, συνήθως ως αποτέλεσμα ένα σφάλμα που επιστρέφεται στον αιτούντα.

Τέλος, το δοχείο web καλεί το καταστρέψουν () μέθοδος που λαμβάνει το servlet εκτός λειτουργίας. Η καταστροφή () μέθοδο, όπως η init () , καλείται μόνο μία φορά κατά τη διάρκεια του κύκλου ζωής ενός servlet.

Το ακόλουθο είναι ένα τυπικό σενάριο χρήσης των μεθόδων αυτών.
Ας υποθέσουμε ότι ένας χρήστης ζητά να επισκεφθείτε ένα URL .Το πρόγραμμα περιήγησης στη συνέχεια δημιουργεί μια αίτηση HTTP για αυτό το URL.Αυτή η αίτηση αποστέλλεται στη συνέχεια στον κατάλληλο διακομιστή. Το αίτημα HTTP λαμβάνεται από τον web server και προωθούνται στο δοχείο Servlet. Το δοχείο χαρτογραφεί το αίτημα αυτό σε μια συγκεκριμένη servlet. Το servlet δυναμικά ανακτώνται και τοποθετούνται στο χώρο διευθύνσεων του δοχείου. Το δοχείο επικαλείται την init () μέθοδο του servlet. Η μέθοδος αυτή εφαρμόζεται μόνον όταν το servlet πρώτα φορτωθεί στη μνήμη. Είναι δυνατόν να περάσετε ορίσματα αρχικοποίησης στον Servlet, έτσι ώστε να μπορεί να αυτορυθμιστεί. Το δοχείο επικαλείται την υπηρεσία () μέθοδο του servlet. Αυτή η μέθοδος καλείται να επεξεργαστεί την αίτηση HTTP. Το servlet μπορεί να διαβάσει τα δεδομένα που έχουν παρασχεθεί στην αίτηση HTTP. Το servlet μπορεί επίσης να διατυπώνει μια απόκριση HTTP για τον πελάτη. Το servlet παραμένει στο χώρο διευθύνσεων του περιέκτη και είναι διαθέσιμη για την επεξεργασία οποιωνδήποτε άλλων αιτήσεων HTTP που λαμβάνονται από τους πελάτες. Η υπηρεσία () μέθοδος καλείται για κάθε αίτηση HTTP. Το δοχείο μπορεί, κάποια στιγμή, να αποφασίσει να ξεφορτώσουν το servlet από τη μνήμη της.

Οι αλγόριθμοι με τους οποίους γίνεται η παρούσα απόφαση είναι ειδικές για κάθε δοχείο. Το δοχείο καλεί το servlet είναι να καταστρέψουν () μέθοδος για να αποποιηθούν οποιαδήποτε πόρους, όπως χειρισμού αρχείων που διατίθενται για το servlet, σημαντικά δεδομένα μπορούν να αποθηκευτούν σε μια επίμονη κατάστημα. Η μνήμη που διατίθεται για την Servlet και τα αντικείμενά της μπορεί στη συνέχεια να απορριμμάτων που συγκεντρώνονται.

2.10 SQL

Εικόνα 34 Λονότυπο SQL

Η SQL είναι μια γλώσσα υπολογιστή για εργασία με σύνολα δεδομένων και τις σχέσεις μεταξύ τους. Τα προγράμματα σχεσιακών βάσεων δεδομένων, όπως η MicrosoftOfficeAccess, χρησιμοποιούν την SQL για εργασία με δεδομένα. Σε αντίθεση με πολλές γλώσσες υπολογιστή, δεν είναι δύσκολο να διαβάσουμε και να κατανοήσουμε την SQL, ακόμα και αν είμαστε αρχάριοι. Όπως πολλές γλώσσες υπολογιστή, η SQL είναι ένα διεθνές πρότυπο που αναγνωρίζεται από φορείς προτύπων όπως ο ISO και ο ANSI.

Δεν είναι δύσκολο να διαβάσετε και να κατανοήσετε την SQL, ακόμα και εάν είστε αρχάριος. Μπορείτε να χρησιμοποιήσετε την SQL για να περιγράψετε σύνολα δεδομένων που μπορούν να σας βοηθήσουν να απαντήσετε σε ερωτήσεις. Όταν χρησιμοποιείτε την SQL, πρέπει να χρησιμοποιείτε τη σωστή σύνταξη. Η σύνταξη είναι το σύνολο των κανόνων με τους οποίους τα στοιχεία μιας γλώσσας συνδυάζονται σωστά. Η σύνταξη της SQL βασίζεται στη σύνταξη της Αγγλικής γλώσσας και χρησιμοποιεί πολλά ίδια στοιχεία με τη σύνταξη της VisualBasicforApplications (VBA).

Για παράδειγμα, μια απλή πρόταση SQL η οποία ανακτά μια λίστα επωνύμων για επαφές των οποίων το όνομα είναι Μαίρη μπορεί να μοιάζει με την εξής:

```
SELECT Last_Name
FROM Contacts
WHERE First_Name = 'Mary';
```

2.10.1 Προτάσεις SELECT

Για να περιγράψετε ένα σύνολο δεδομένων, χρησιμοποιώντας την SQL, μπορείτε να γράψετε μια πρόταση SELECT. Μια πρόταση SELECT περιέχει μια πλήρη περιγραφή ενός συνόλου δεδομένων που θέλετε να λάβετε από μια βάση δεδομένων. Περιλαμβάνει τα εξής:

- Ποιοι πίνακες περιέχουν τα δεδομένα.
- Με ποιον τρόπο σχετίζονται δεδομένα από διάφορες προελεύσεις.
- Ποια πεδία ή υπολογισμοί θα δημιουργήσουν τα δεδομένα.
- Κριτήρια με τα οποία πρέπει να ταιριάζουν τα δεδομένα που θα συμπεριληφθούν.
- Εάν και πώς θα γίνει ταξινόμηση των αποτελεσμάτων.

2.10.2 Όροι SQL

Όπως μια γραμματική πρόταση, έτσι και μια πρόταση SQL έχει όρους. Κάθε όρος εκτελεί μια λειτουργία για την πρόταση SQL. Ορισμένοι όροι είναι απαραίτητοι σε μια πρόταση SELECT. Ο παρακάτω πίνακας παραθέτει τους πιο συνηθισμένους όρους SQL.

SELECT	Παραθέτει τα πεδία που περιέχουν δεδομένα τα οποία σας ενδιαφέρουν.
FROM	Παραθέτει τους πίνακες που περιέχουν τα πεδία τα οποία παρατίθενται στον όρο SELECT.
WHERE	Καθορίζει το κριτήρια πεδίου που πρέπει να πληρούνται από κάθε εγγραφή για να συμπεριληφθεί στα αποτελέσματα.
ORDER BY	Καθορίζει τον τρόπο ταξινόμησης των αποτελεσμάτων.
GROUP BY	Σε μια πρόταση SQL που περιέχει συναρτήσεις συγκεντρωτικών αποτελεσμάτων, παραθέτει πεδία τα οποία δεν συνοψίζονται στον όρο SELECT.
HAVING	Σε μια πρόταση SQL που περιέχει συναρτήσεις συγκεντρωτικών αποτελεσμάτων, καθορίζει τις συνθήκες που ισχύουν για τα πεδία τα οποία συνοψίζονται στην πρόταση SELECT.

Select, From, Where

2.10.2.1 Ο όρος SELECT

SELECT [E-mail Address], Company

Πρόκειται για τον όρο SELECT. Αποτελείται από έναν τελεστή (SELECT) ακολουθούμενο από δύο αναγνωριστικά ([Διεύθυνση ηλεκτρονικού ταχυδρομείου] και Εταιρεία).Εάν ένα αναγνωριστικό περιέχει διαστήματα ή ειδικούς χαρακτήρες (όπως "Διεύθυνση ηλεκτρονικού ταχυδρομείου"), πρέπει να περικλείονται σε αγκύλες. Ένας όρος SELECT δεν χρειάζεται να αναφέρει τους πίνακες που περιέχουν τα πεδία και δεν μπορεί να καθορίζει τις συνθήκες που πρέπει να πληρούνται από τα δεδομένα που θα συμπεριληφθούν.

Ο όρος SELECT εμφανίζεται πάντα μπροστά από τον όρο FROM σε μια πρόταση SELECT.

2.10.2.2 Ο όρος FROM

FROM Contacts
Πρόκειται για τον όρο FROM. Αποτελείται από έναν τελεστή (FROM), ακολουθούμενο από ένα αναγνωριστικό (Επαφές).Ένας όρος FROM δεν παραθέτει τα πεδία που θα επιλεχθούν.

2.10.2.3 Ο όρος WHERE

WHERE City="Θεσσαλονίκη"

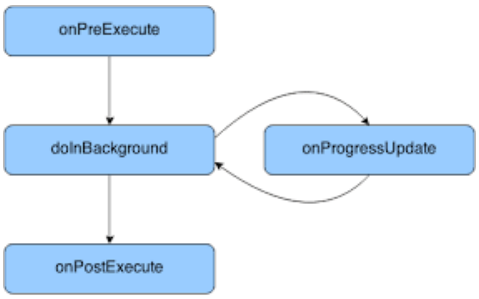
Πρόκειται για τον όρο WHERE. Αποτελείται από έναν τελεστή (WHERE) ακολουθούμενο από μια παράσταση (City = "Θεσσαλονίκη").

2.11 Json

Εικόνα 35 Λογότυπο Json

Το JSON Το JSON ή JavaScriptObjectNotation είναι ένα text-based ανοιχτό πρότυπο έτσι ώστε να είναι εύκολα αναγνώσιμο από τον χρήστη, με σκοπό την ανταλλαγή δεδομένων. Βασισμένο στη γλώσσα σεναρίων Javascript,το JSON είναι μια γλώσσα για να αναπαραστήσουμε απλές δομές δεδομένων και associativearrays,τα οποία ονομάζουμε αντικείμενα. Σε αντίθεση με τη JavaScript ,το JSON λειτουργεί ανεξαρτήτως γλώσσας και διαθέτει parsers για πάρα πολλές γλώσσες. [10] . Σε σχέση με το XML προτιμήθηκε γιατί λόγω του ότι απαιτεί πολύ λιγότερα σημεία σύνταξης(παρενθέσεις, άγκιστρα κτλ.) γλυτώνουμε έτσι την αποστολή περιττών δεδομένων. Επίσης η χρήση μιας βιβλιοθήκης gzip έτσι ώστε να μειώσουμε το μέγεθος και να το φτάσουμε στα επίπεδα του JSON δεν θα ήταν η καλύτερη επιλογή στην περίπτωση μας, καθώς θα αναγκάζαμε τη συσκευή μας σε περιττούς υπολογισμούς και σπατάλη της CPU άρα και της μπαταρίας χωρίς κάποιο σημαντικό λόγο και απλά για να πετύχουμε το ίδιο αποτέλεσμα.

2.12 AsyncTask



Εικόνα 36 AsyncTask διάγραμμα

Ορισμένες εφαρμογές απαιτούν μεγάλο χρόνο για να μπορέσουν να επεξεργαστούν τα δεδομένα ώστε να τα προβάλλουν στη διεπαφή χρήστη. Ένα τέτοιο παράδειγμα είναι η αίτηση δεδομένων από μία βάση. Το γεγονός αυτό σημαίνει ότι χρονοβόρες ενέργειες δεν πρέπει να επιβραδύνουν ή να μπλοκάρουν τη διεπαφή χρήστη (UI). Το Android παρέχει μία ειδική κλάση την AsyncTask η οποία επιτρέπει την επεξεργασία δεδομένων στο παρασκήνιο και προβολή του UI όταν η επεξεργασία τελειώσει. Μπορούμε να χρησιμοποιήσουμε την AsyncTask δημιουργώντας μία νέα εμφωλευμένη κλάση μέσα στην ήδη υπάρχουσα που κάνει «extends» την AsyncTask και υλοποιώντας τις μεθόδους onPreExecute(), doInBackground() και την onPostExecute(). Η μέθοδος onPreExecute() εκτελείται πριν ξεκινήσει η επεξεργασία δεδομένων στο παρασκήνιο. Η doInBackground() είναι υπεύθυνη για την επεξεργασία των δεδομένων στο παρασκήνιο και η μέθοδος onPostExecute() ενημερώνει τη διεπαφή χρήστη προσφέροντας τα δεδομένα που επεξεργάστηκαν. Τέλος για να χρησιμοποιήσουμε την κλάση που κάνει «extends» την AsyncTask πρέπει στην onCreate() της κύριας κλάσης να εκτελέσουμε την εντολή «new*Όνομα_κλάσης*.execute()».

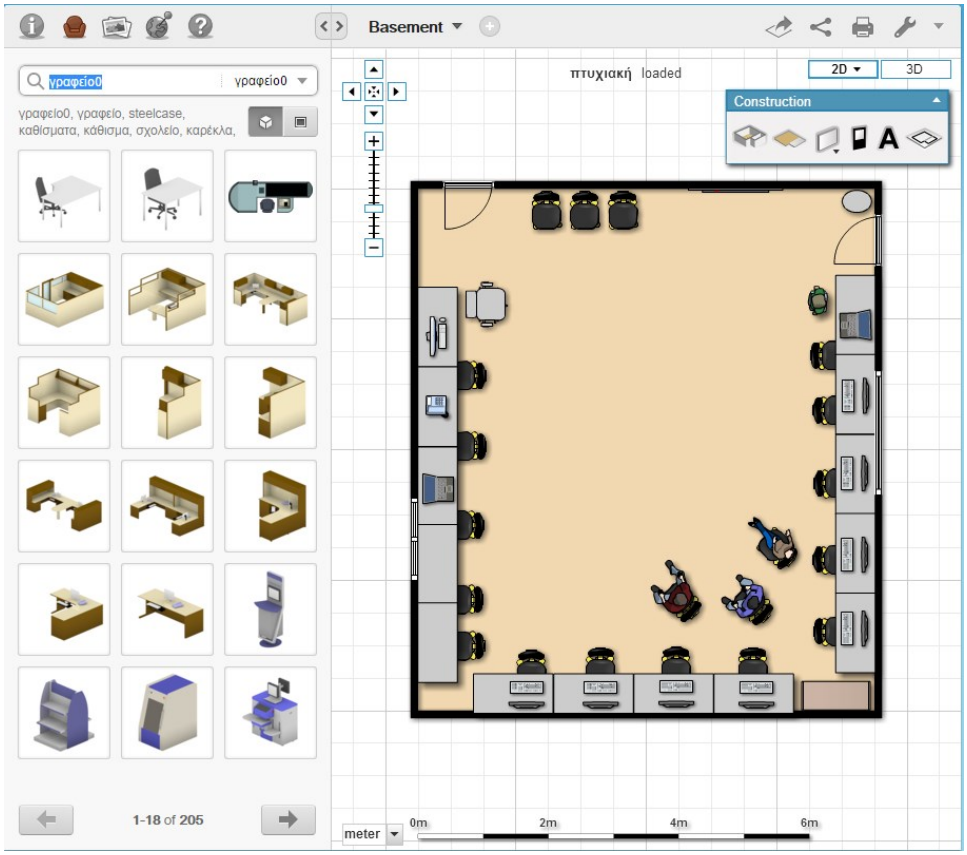
2.13 Floorplanner

Εικόνα 37 – Λογότυπο Floor Planer

Το Floorplanner είναι ο πιο εύκολος και όμορφος τρόπος για να δημιουργήσουμε και να μοιραστούμε διαδραστικές κατόψεις online. Η ανάγκη αυτή εμφανίστηκε όταν οι γονείς του ιδρυτή Jeroen έψαχναν για ένα νέο διαμέρισμα, δεν ήξεραν αν τα έπιπλά τους θα ταίριαζαν και κυρίως αν θα χωρούσαν στον χώρο. Ο μόνος τρόπος για να το καταλάβαιναν, ήταν να σχεδιάσουν μια κάτοψη με τα έπιπλα τους στον αντίστοιχο χώρο. Αυτό θα μπορούσε να υλοποιηθεί, είτε σε χαρτί - το οποίο παίρνει πολύ χρόνο - ή με CAD (βαρύ λογισμικό) - που διαρκεί ακόμη περισσότερο χρόνο. Έπρεπε να υπάρχει ένας καλύτερος τρόπος. Έτσι, ο Jeroen δημιούργησε το Floorplanner, ένα μέρος που μπορούμε να υλοποιήσουμε εύκολα μια κάτοψη ενός σπιτιού και να συμπληρώσουμε τα έπιπλα που επιθυμούμε. Το Floorplanner είναι τώρα η κύρια πλατφόρμα κάτοψης με πάνω από 10 εκατομμύρια εγγεγραμμένους χρήστες από όλο τον κόσμο και πελάτες όπως το ΙΚΕΑ, μεσίτες ακινήτων , υπηρεσίες σχεδίασης και τα λοιπά. Θεωρείται το πιο χρησιμοποιούμενο λογισμικό πρόγραμμα για σχεδίαση χώρου σε διαδικτυακή πλατφόρμα. Δημιουργήθηκε το 2007 και από τότε έχει βοηθήσει εκατομμύρια ανθρώπους και χιλιάδες εταιρείες στη δημιουργία λεπτομερών σχεδίων κάτοψης. Έχουν χρήστες σε 161 χώρες και παρέχουν τις υπηρεσίες του σε 14 γλώσσες.

Διαδικασία κάτοψης της αίθουσας

Βασισμένη στην πραγματική αίθουσα, γνωρίζοντας τις διαστάσεις αρχικά δημιουργήσαμε τους τοίχους, τις πόρτες και τα παράθυρα αντίστοιχα.ToFloorplanner όπως βλέπουμε και στην παρακάτω εικόνα μας παρέχει όλα τα έπιπλα που μπορούμε να χρειαστούμε κατά την σχεδίαση της αίθουσας ώστε να γίνει πιο ρεαλιστικό το αποτέλεσμα. Έπειτα έγινε η προσθήκη των τραπεζοκαθισμάτων περιφερειακά της αίθουσας καθώς και των ηλεκτρονικών υπολογιστών, αλλά και των αντικειμένων που ο χρήστης έχει τη δυνατότητα να σκανάρει και να περισυλλέξει.



Εικόνα 38 - Κάτοψη αίθουσας

2.14 Dijkstra

Ο αλγόριθμος του Ντάικστρα (Dijkstra) πήρε το όνομά του από τον Ολλανδό Έντσγκερ Ντάικστρα, ο οποίος τον επινόησε το 1956 και τον δημοσίευσε το 1959. Πρόκειται για έναν αλγόριθμο εύρεσης συντομότερων διαδρομών (single-source shortest path problem) από κοινή αφετηρία σε έναν (κατευθυνόμενο ή μη) γράφο με μη αρνητικά βάρη στις ακμές. Ο αλγόριθμος του Dijkstra είναι άπληστος. Δηλαδή, σε κάθε βήμα επιλέγει την τοπικά βέλτιστη λύση, ώσπου στο τελευταίο βήμα συνθέτει μια συνολικά βέλτιστη λύση. Αν ο γράφος περιέχει αρνητικά βάρη, ο αλγόριθμος του Ντάικστρα δεν δίνει σωστό αποτέλεσμα.

Ο αλγόριθμος του Ντάικστρα είναι πλέον ευρέως διαδεδομένος και χρησιμοποιείται σε πολλές εφαρμογές. Χρήση του αλγόριθμου αυτού κάνει το πρωτόκολλο OSPF, το οποίο είναι το εσωτερικό πρωτόκολλο πύλης δικτύου του Διαδικτύου.

Έχουμε έναν γράφημα G(V,E), όπου V το σύνολο των κόμβων του και E το σύνολο των ακμών του. Επίσης, έχουμε μια συνάρτηση βάρους $w : E \to \mathbb{R}^+ \cup \{0\}$ ορισμένη στις ακμές του γράφου. Αυτό σημαίνει ότι για να πάμε από έναν κόμβο του γράφου σε έναν άλλο, θα έχουμε κάποιο *κόστος*. Είναι σημαντικό τα βάρη να μην είναι αρνητικά, επειδή διαφορετικά ο αλγόριθμος δεν δίνει σωστό αποτέλεσμα. Ο αλγόριθμος του Ντάικστρα βρίσκει τα μονοπάτια που πρέπει να ακολουθήσουμε από έναν κόμβο-αφετηρία προς τους υπόλοιπους, ώστε να έχουμε το λιγότερο δυνατό κόστος.

Για τη λειτουργία του αλγόριθμου, σε ένα διάνυσμα d[] μεγέθους $|V| = n$ αποθηκεύουμε την *έως τώρα* υπολογισμένη απόσταση των κόμβων από την αφετηρία. Κατά την αρχικοποίηση, οι αποστάσεις σημειώνονται $d[s] = 0$ και $d[v] = +\infty$ για κάθε $v \neq s$, όπου s είναι ο κόμβος-αφετηρία. Επιπλέον ο αλγόριθμος διατηρεί μια ουρά προτεραιότητας, για την επεξεργασία των κόμβων του γραφήματος στη σωστή σειρά, και ένα σύνολο S, το σύνολο των κόμβων για τους οποίους ο αλγόριθμος έχει βρει την ελάχιστη διαδρομή. Στην Q εισάγονται όλοι οι κόμβοι του γραφήματος με κλειδί την τιμή d[*], ενώ το σύνολο S είναι αρχικά κενό. Τέλος, ο αλγόριθμος χρησιμοποιεί ένα ακόμη διάνυσμα, το prev[], μεγέθους n, στο οποίο για κάθε κόμβο u αποθηκεύεται ο αμέσως προηγούμενος κόμβος στο ελάχιστο μονοπάτι προς τον u. Για παράδειγμα, έστω ότι το ελάχιστο μονοπάτι από τον s στον b περνά πρώτα από τον a και αμέσως μετά φτάνει στον b. Τότε, prev[b]=a. Αρχικά, κάθε θέση του διανύσματος prev[] λαμβάνει την τιμή *null* (κενό).

Μετά την αρχικοποίηση των δομών που χρησιμοποιεί, ο αλγόριθμος εξάγει από την Q τον κόμβο x με το ελάχιστο d[*] και τον εισάγει στο σύνολο S. Στο πρώτο βήμα, για παράδειγμα, θα εξάγει τον κόμβο-αφετηρία s, αφού d[s]=0 ενώ όλοι οι υπόλοιποι κόμβοι έχουν άπειρο d[*]. Για κάθε γείτονα y (του x) που δεν ανήκει στο σύνολο S, αν $d[y] > d[x] + w(x,y)$ τότε ενημερώνει το d[y] καταχωρώντας του την τιμή d[x] + w(x,y) και θέτει prev[y]=x. Δηλαδή, αν ο αλγόριθμος υπολογίσει ένα ελαφρύτερο (από το ήδη υπολογισμένο) μονοπάτι για τον κόμβο y, τότε σημειώνει το κόστος του (d[y]) και τον αμέσως προηγούμενο κόμβο του νέου υπολογισμένου μονοπατιού (prev[y]). Με την αλλαγή του d[y] αλλάζει και η θέση του κόμβου y στην ουρά προτεραιότητας Q. Για την ακρίβεια, μεγαλώνει η προτεραιότητα του y, αφού κάθε νέα τιμή του d[y] είναι πάντα μικρότερη από την προηγούμενη. Αφού ο αλγόριθμος εξετάσει όλους τους γείτονες του x που δεν ανήκουν στο σύνολο S, εισάγει στο S τον κόμβο με το ελάχιστο d[*] από όλους όσους δεν ανήκουν στο S. Έπειτα, ο αλγόριθμος επιλέγει πάλι τον πρώτο σε προτεραιότητα κόμβο από την ουρά Q και επαναλαμβάνει τα βήματα αυτά μέχρι να αδειάσει η Q.

Όταν πλέον η Q θα έχει αδειάσει, ο αλγόριθμος θα έχει βρει τα ελάχιστα μονοπάτια από τον κόμβο s προς τους όλους τους υπόλοιπους και τα κόστη τους.

Ο αλγόριθμος είναι άπληστος (greedy): σε κάθε βήμα, εξετάζει μόνο τους γειτονικούς ενός κόμβου (τοπικότητα). Βρίσκει τον κόμβο για τον οποίο έχει υπολογίσει την ελάχιστη διαδρομή και τον εισάγει στο σύνολο S, χρησιμοποιώντας πληροφορίες από προηγούμενα βήματα (σύνθεση). Στο τέλος δίνει ένα αποτέλεσμα για όλους τους κόμβους.

2.14.1 Σε βήματα

Μια περισσότερο τυποποιημένη περιγραφή του αλγόριθμου είναι η παρακάτω, η οποία δείχνει τη λειτουργία του αλγόριθμου σε βήματα.

1. Σημείωσε σε κάθε κόμβο μια *ετικέτα απόστασης* (d[*]) με τιμή 0 στον αρχικό κόμβο και τιμή *άπειρο* σε όλους τους υπόλοιπους. Επίσης, σημείωσε μια *ετικέτα προηγούμενου κόμβου* (prev[*]) και βάλε της την κενή τιμή για όλους τους κόμβους. Η ετικέτα αυτή χρειάζεται για τον υπολογισμό της ζητούμενης διαδρομής στο τέλος.
2. Σημείωσε όλους τους κόμβους μη-επεξεργασμένους ($S = \emptyset$). Ο τρέχων κόμβος είναι ο αρχικός.
3. Για τον τρέχων κόμβο, εξέτασε όλους τους μη-επεξεργασμένους γείτονές του και υπολόγισε το συνολικό άθροισμα απόστασής τους από τον αρχικό κόμβο. Για παράδειγμα, αν ο τρέχων κόμβος έχει απόσταση 6 από τον αρχικό και ο γείτονας του τρέχοντος κόμβου, που εξετάζει αυτή τη στιγμή ο αλγόριθμος, έχει απόσταση 2 από τον τρέχων, το συνολικό άθροισμα απόστασης του γείτονα από τον αρχικό κόμβο είναι 6+2=8. Αν αυτή η απόσταση είναι μικρότερη από την ετικέτα απόστασης που είχε σημειωθεί, αντικατάστησέ τη με τη νέα υπολογισμένη τιμή και σημείωσε τον τρέχων κόμβο στην *ετικέτα προηγούμενου κόμβου*.
4. Όταν τελειώσεις με την εξέταση όλων των γειτόνων του τρέχοντος κόμβου, σημείωσέ τον ως επεξεργασμένο. Ένας επεξεργασμένος κόμβος δεν εξετάζεται ποτέ ξανά από τον αλγόριθμο. Η ετικέτα απόστασής της είναι η ελάχιστη και θα παραμείνει σταθερή.
5. Ο επόμενος τρέχων κόμβος θα είναι ο μη-επεξεργασμένος κόμβος με τη μικρότερη ετικέτα απόστασης.
6. Αν όλοι οι κόμβοι έχουν σημειωθεί ως επεξεργασμένοι, προχώρα στο επόμενο βήμα. Διαφορετικά, συνέχισε από το βήμα 3.
7. Ξεκινώντας από τον κόμβο-προορισμό (ο οποίος είναι ο τελευταίος τρέχων κόμβος) εκτύπωσε τον κόμβο που αναγράφεται στην *ετικέτα προηγούμενου κόμβου*. Επανέλαβε μέχρι η *ετικέτα προηγούμενο κόμβου* που θα συναντήσεις να είναι άδεια.

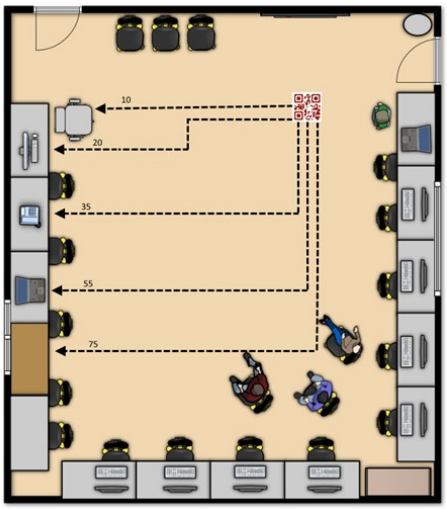
Ακολουθεί περιγραφή του αλγορίθμου που χρησιμοποιήθηκε, ο οποίος υπολογίζει τις αποστάσεις μεταξύ του σημείου αφετηρίας και των γειτονικών του κόμβων. Οι αποστάσεις από κάθε κόμβο προς κάθε κόσμο, έχουν εξ αρχής οριστεί σε δισδιάστατο πίνακα. Συνολικά έχουν οριστεί τέσσερις πίνακες για τα τέσσερα διαφορετικά σημεία αφετηρίας. Οι αποστάσεις αποθηκεύονται σε ένα ArrayList όπου στη συνέχεια ταξινομούνται και κατ αυτό τον τρόπο προχωράμε στην διαδικασία του Mapping.

```
public static void dijkstra_algorithm(int adjacency_matrix[][], int source)
        {
            int evaluationNode;
            for (int i = 1; i < length +1 ; i++)
               for (int j = 1; j < length +1 ; j++)
               adjacencyMatrix[i][j] = adjacency_matrix[i][j];

            for (int i = 1; i <= length; i++)
            {
                        distances_original[i] = Integer.MAX_VALUE;
            }
            unsettled.add(source);
            distances_original[source] = 0;
            while (!unsettled.isEmpty())
            {
               evaluationNode = getNodeWithMinimumDistanceFromUnsettled();
                        unsettled.remove(evaluationNode);
                        settled.add(evaluationNode);
                        evaluateNeighbours(evaluationNode);
            }
            for(int i=1;i<distances_original.length;i++)
            {
               distances_tmp[i] = distances_original[i];
            }
            for(int k=1;k<distances_pos.length;k++)
            {
               distances_pos[k]=0;
                        my_array[k]=0;
        }
             Arrays.sort(distances_tmp);
          }
```

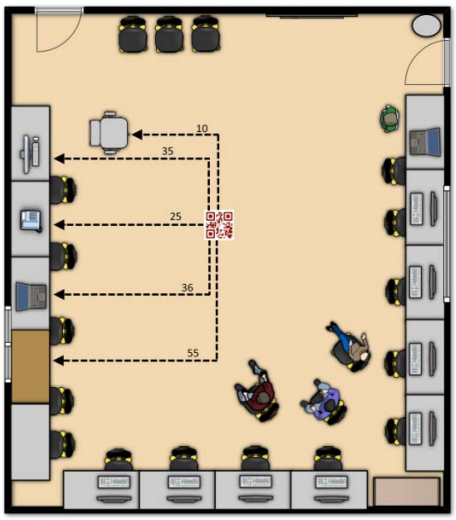
2.14.2 Εικόνες κάτοψης αίθουσας με τις αφετηρίες

Στην περίπτωση μας έχουμε 4 αφετηρίες, οι οποίες προσδιορίζονται βάση των τεσσάρων διαφορετικών QR_Codes που βρίσκονται σε σταθερές θέσεις μέσα στην αίθουσα. Κάθε ένα από αυτά μπορεί να οριστεί ως αφετηρία (αρκεί να το σαρώσει ο χρήστης)στον γράφο με κόμβους τα αντικείμενα τα οποία έχει ήδη σαρώσει ο χρήστης. Για κάθε ένα από τα διαφορετικά σημεία αφετηρίας έχουν δηλωθεί εκ των προτέρων οι αποστάσεις μεταξύ των σημείων, προκειμένου να γίνει η κατανομή των κοντινότερων σημείων ως προς το σημείο αφετηρίας.

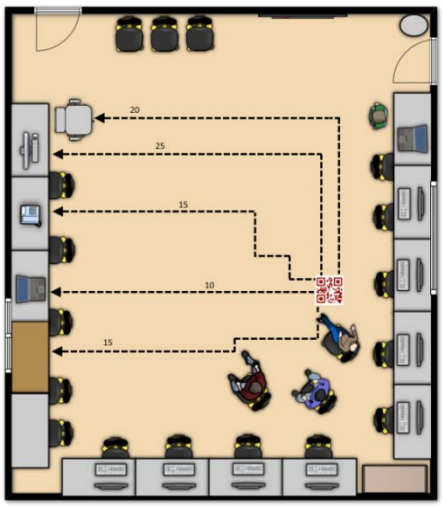
Παρακάτω ακολουθούν εικόνες με την κάτοψη της αίθουσας, οι οποίες περιλαμβάνουν τις τέσσερις διαφορετικές θέσεις των QRCodes στο χώρο, καθώς και τις αποστάσεις μεταξύ των σημείων.



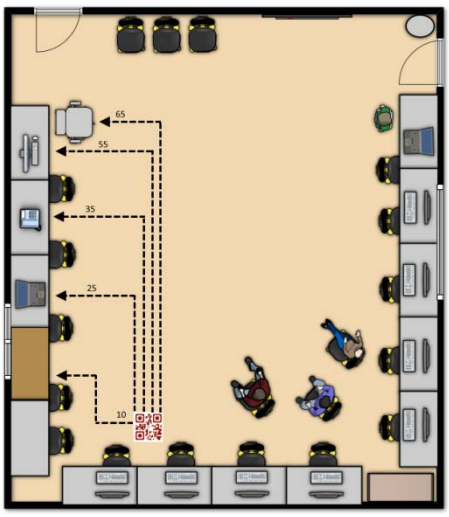
Εικόνα 41 Κάτοψη αίθουσας απο θέση 1



Εικόνα 40 Κάτοψη αίθουσας απο θέση 2



Εικόνα 42 Κάτοψη αίθουσας απο τη θέση 3



Εικόνα 39 Κάτοψη αίθουσας απο τη θέση 4

2.15 QR_Codes

Εικόνα 43 Λογότυπο QRCode

Τα QR Codes δημιουργήθηκαν το 1994 από Denso Wave , μια ιαπωνική θυγατρική του Ομίλου Toyota. Η χρήση αυτής της τεχνολογίας είναι τώρα ελεύθερη . Το QR Code δεν είναι το μόνο δισδιάστατο barcode στην αγορά , ένα άλλο παράδειγμα είναι ο κωδικός Data Matrix.

Το QR Code είναι η πιο διάσημη barcode 2D στον κόσμο . Έχει κερδίσει την επιτυχία της στην Ιαπωνία από τη δεκαετία του 2000 όπου τώρα είναι ένα πρότυπο . Το 2011 , κατά μέσο όρο 5 QR Codes σαρώθηκαν καθημερινά από κάθε ιάπωνα - περισσότερο από το μέσο αριθμό των SMS που στέλνονται ! Το 2010 τα QR Codes άρχισαν να επεκτείνεται στις ΗΠΑ , και στη συνέχεια, στην Ευρώπη.

Σήμερα τα QR Codes μπορούμε να τα δούμε σε φυλλάδια , αφίσες , περιοδικά και ούτω καθεξής. Τα QR Codes μας επιτρέπουν να αλληλεπιδράσουμε με τον κόσμο χρησιμοποιώντας το Smartphones. Συγκεκριμένα, ένα QR Code επεκτείνει τα στοιχεία που έχει στη διάθεση σε οποιοδήποτε φυσικό αντικείμενο και να δημιουργήσει ένα ψηφιακό βαθμό στις επιχειρήσεις εμπορίας. Αυτή η τεχνολογία επιτρέπει και επιταχύνει τη χρήση των υπηρεσιών κινητής τηλεφωνίας στο διαδίκτυο.

Όταν σαρώνουμε ένα QR Code χρησιμοποιώντας το έξυπνο τηλέφωνο, μπορούμε να πάρουμε άμεση πρόσβαση στο περιεχόμενό του. Διαβάζοντας το QR Code μπορούμε στη συνέχεια να μεταφέρουμε μια ενέργεια, όπως το άνοιγμα browser σε μια συγκεκριμένη διεύθυνση URL. Άλλες ενέργειες που μπορούμε να πραγματοποιήσουμε είναι η αποθήκευση μιας επαγγελματικής κάρτας στη λίστα επαφών του Smartphone μας , ή σύνδεση σε ένα ασύρματο δίκτυο.

2.15.1 Δημιουργία QR_Codes – Unitag

Εικόνα 44 Λογότυπο Unitag

Στην εφαρμογή αυτή επέλεξα να χρησιμοποιήσω το πρόγραμμα unitag για τη δημιουργία των QR_Codes μου. Αρχικά επιλέγουμε από τα tags το QR_Codes και δημιουργούμε ένα καινούργιο. Στη συνέχεια από τις ρυθμίσεις επιλέγουμε να είναι static η πληροφορία και όχι Dynamic, καθώς το δυναμικό ναι μεν μπορεί να επεξεργαστεί σε σχέση με το στατικό qr_code αλλά κατά την σάρωση δεν εμφανίζει το περιεχόμενο απευθείας, αλλά το εμφανίζει ανοίγοντας ένα url, πράγμα που εμείς δεν επιθυμούμε. Στη καρτέλα επιλέγουμε τι μορφής θέλουμε να είναι, παραδείγματος χάρη Text ,Geolocation, Calendar κτλπ. Επιλέγοντας την κατηγορία (στην περίπτωση μας Text) πληκτρολογούμε ότι θέλουμε να κατανοεί το σύστημα κατά την σάρωση του. Έπειτα από το μενού Προσαρμογή έχουμε τη δυνατότητα να χρησιμοποιήσουμε όποιο χρώμα επιθυμούμε για την εμφάνιση του qr_code, εμείς έχουμε επιλέξει ένα σκούρο κόκκινο. Μπορούμε επίσης να τοποθετήσουμε εικόνα στο κέντρο του καθώς και να αλλάξουμε το μοντέλο του από τετράγωνο σε κυκλικό και άλλα.

3 Σχεδιασμός

3.1 Περιγραφή

Η Android εφαρμογή "Tema" είναι προσβάλαμε από έξυπνα τηλέφωνα που περιλαμβάνουν λογισμικό Android και χαρακτηρίζεται από την AndroidΕκδοση 5.0 Lollipop. Η εφαρμογή «Tema» έχει ως βασικό στόχο, την διευκόλυνση και την εξυπηρέτηση των χρηστών της, οι οποίοι ενδιαφέρονται για την συλλογή προϊόντων από χώρο στον οποίο δεν έχουν άμεσα πρόσβαση (αποθήκη).Αυτό επιτυγχάνεται με την σάρωση του ειδικού Barcode οποίο χαρακτηρίζει κάθε αντικείμενο και το οποίο βρίσκεται σε διαφορετικό χώρο (έκθεση).

Ο χρήστης έχοντας εγκαταστήσει την εφαρμογή στο κινητό του (το οποίο διαθέτει Camera) και έχοντας πρόσβαση στο διαδίκτυο μπορεί να περιηγηθεί στον χώρο (έκθεση) και να σαρώσει το QRCode του αντικειμένου που τον ενδιαφέρει. Αυτή η διαδικασία μπορεί να επαναληφθεί για όσες φορές επιθυμεί ο χρήστης. Κάθε QRCode που προκύπτει ως αποτέλεσμα της σάρωσης του χρήστη καταχωρείται στη βάση. Αξίζει να σημειωθεί πως στην περίπτωση που ο χρήστης σαρώσει το ίδιο αντικείμενο πάνω από μια φορές δεν καταχωρείται στη βάση, αλλά ενημερώνεται από την εφαρμογή πως ήδη έχει προβεί στην ενέργεια αυτή.

Στην συνέχεια τα σαρωμένα αντικείμενα προβάλλονται ως λίστα στον χρήστη. Ο χρήστης έχει την δυνατότητα να δει περισσότερες πληροφορίες για κάθε αντικείμενο ξεχωριστά, καθώς και να επεξεργαστεί την λίστα του διαγράφοντας κάποιο αντικείμενο. Επιπλέον δυνατότητα που παρέχει η εφαρμογή στον χρήστη είναι η προβολή του ιστορικού χρήστης.

Τέλος όταν ο χρήστης έχει ολοκληρώσει την διαδικασία σάρωσης των προϊόντων μπορεί να προχωρήσει στο επόμενο βήμα, το οποίο είναι η συλλογή των αντικειμένων (αποθήκη). Κατά την διαδικασία αυτή εμφανίζεται η κάτοψη του χώρου (αποθήκη) από τον οποίο καλείται ο χρήστης να συλλέξει τα αντικείμενα που σάρωσε στο προηγούμενο βήμα. Έπειτα από την συλλογή όλων των προϊόντων από τον χώρο, τα αντικείμενα ατά χαρακτηρίζονται ως ήδη συλλεγμένα και η διαδικασία ολοκληρώνεται. Στο σημείο αυτό η εφαρμογή είναι έτοιμη να ξεκινήσει εκ νέου την διαδικασία αυτή από την αρχή.

3.2 Προδιαγραφές - Περιγραφή

3.2.1 Σάρωση

Προκειμένου να επιτευχθεί η διαδικασία της σάρωσης έγινε χρήση της Data που διαθέτει το κινητό, συνεπώς απαραίτητη προϋπόθεση για την χρήση της εφαρμογής είναι η συσκευή να περιλαμβάνει Camera. Κατά την διαδικασία αυτή το QRCode ανιχνεύεται και αναγνωρίζεται από το σύστημα.

3.2.2 Καταχώρηση/Ανάκτηση στη-από βάση

Τα QRCodes που προκύπτουν ως αποτέλεσμα, καταχωρούνται στη βάση μέσω HttpClient που δημιουργείται από την συσκευή Android, προς το μηχάνημα το οποίο περιμένει για requests και το οποίο τρέχει τον serverApache. Όλα τα απαραίτητα request περιλαμβάνουν sqlqueries τα οποία εκτελούνται από την μεριά του server. Η ικανότητα αυτή επιτυγχάνεται μόνο με τη χρήση Internet από την μεριά του client και του server. Μια διευκρίνηση επί τούτου είναι πως η android εφαρμογή «Tema» έχει υλοποιηθεί προς το παρόν να λειτουργεί μόνο όταν client και server βρίσκονται στο ίδιο δίκτυο. Αυτό συμβαίνει διότι ο server τρέχει σε τοπικό μηχάνημα και δεν είναι προσβάσιμο από συσκευές πέρα της εμβέλειας που αναγνωρίζει.

3.2.3 Συλλογή αντικειμένων

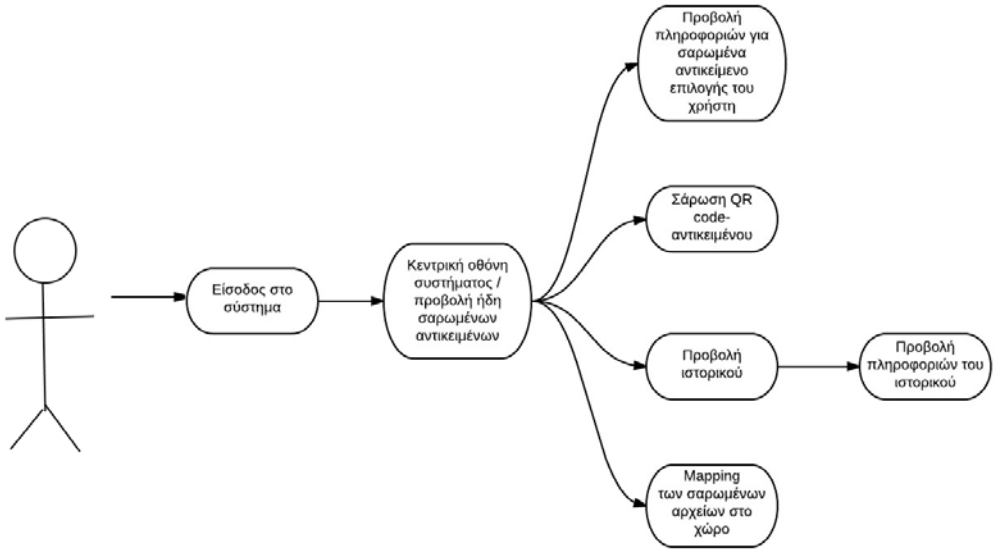
Όσο αφορά την διαδικασία της συλλογής των αντικειμένων από τον χώρο (αποθήκη), έχει χρησιμοποιηθεί ο αλγόριθμος Dijkstra. Ο Dijkstra είναι υπεύθυνος για την εύρεση του κοντινότερου μονοπατιού των σαρωμένων αντικειμένων, έχοντας ως firstpoint την θέση του χρήστη στο χώρο. Αυτό επιτυγχάνεται ελέγχοντας τους γειτονικούς κόμβους των σαρωμένων αντικειμένων, και δεδομένου την σταθερά απόσταση η οποία χαρακτηρίζει τα αντικείμενα.

3.2.4 Προβολή αντικειμένων στο χώρο

Για την προβολή των αντικειμένων στο χώρο έγινε χρήση του Canvas που παρέχεται από την Android εφαρμογή. Συγκεκριμένα κάθε αντικείμενο χαρακτηρίζεται από συγκεκριμένες συντεταγμένες στον χώρο. Οι συντεταγμένες αυτές χρησιμοποιούνται εν συνεχεία στην απεικόνιση του κοντινότερου μονοπατιού από την θέση του χρήστη. Κάθε φορά που ο χρήστης συλλέγει κάποιο αντικείμενο το Canvas ανανεώνεται.

3.2.5 User Case Diagram

Στο User Case διάγραμμα απεικονίζεται η αλληλουχία και συνοχή των λειτουργειών με τις οποίες έρχεται να αλληλεπίδραση ο χρήστης κατά την είσοδο του στο σύστημα «Tema».



4 Κύριο Μέρος εφαρμογής

4.1 Οθόνες εφαρμογής

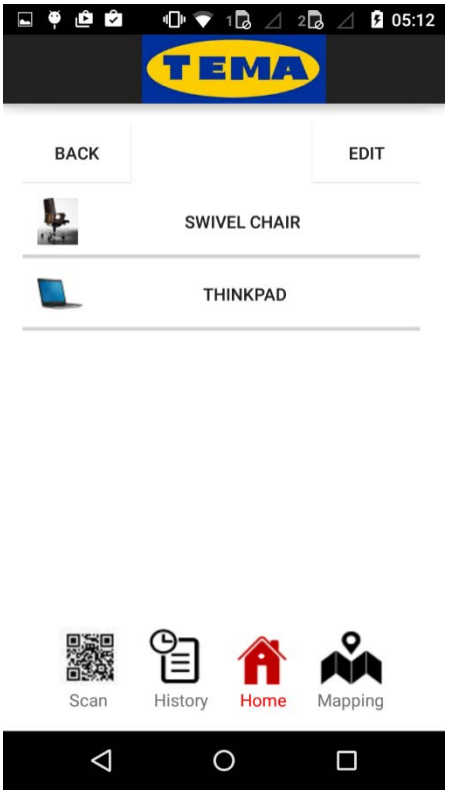
4.1.1 1η οθόνη (Είσοδος στο σύστημα)

Κατά την είσοδο στην εφαρμογή, ο χρήστης μπορεί να ενημερωθεί για τους όρους χρήσης αυτής, καθώς και του δίνεται η δυνατότητα εισαγωγής στην κεντρική οθόνη της εφαρμογής.

Εικόνα 45 1η οθόνη εφαμρογής

4.1.2 2η οθόνη (Κεντρική οθόνη εφαρμογής)

Στην οθόνη αυτή προβάλλεται η λίστα από τα αντικείμενα που έχει σκανάρει ο χρήστης κατά την πλοήγηση του στο χώρο. Η λίστα αυτή ενδέχεται να είναι κενή αν ο χρήστης εισέρχεται για πρώτη φορά στο σύστημα, ειδάλλως εμφανίζονται τα αντικείμενα στα οποία έχει ανιχνεύσει το QRcode τους κατά την πλοήγηση του στο χώρο, αλλά δεν έχει προβεί ακόμη στην συλλογή τους. Εάν ο χρήστης χρησιμοποιεί την εφαρμογή για πρώτη φορά, τότε δημιουργείται αυτόματα ένα μοναδικόuser_id, εάν όχι, τότε το σύστημα μας βασιζόμενο σε αυτό το μοναδικό id εντοπίζει από τη βάση τα σκαναρισμένα αντικείμενα που αντιστοιχούν σε αυτόν τον χρήστη και του προβάλλει τη λίστα αυτών. Επιπλέον ο χρήστης έχει τη δυνατότητα να διαγράψει κάποιο αντικείμενο από τη λίστα του που δεν το επιθυμεί πλέον, καθώς και να ενημερωθεί για περαιτέρω πληροφορίες για το αντικείμενο που έχει σαρώσει(θα τα δούμε αναλυτικά παρακάτω. Σε αυτήν την οθόνη δίνεται η δυνατότητα στον χρήστη να προβεί στις λειτουργίες σάρωσης αντικειμένων, προβολή ιστορικού καθώς και περιήγησης στο χώρο (θα τα δούμε αναλυτικά παρακάτω)



Εικόνα 46 2η οθόνη εφαρμογής

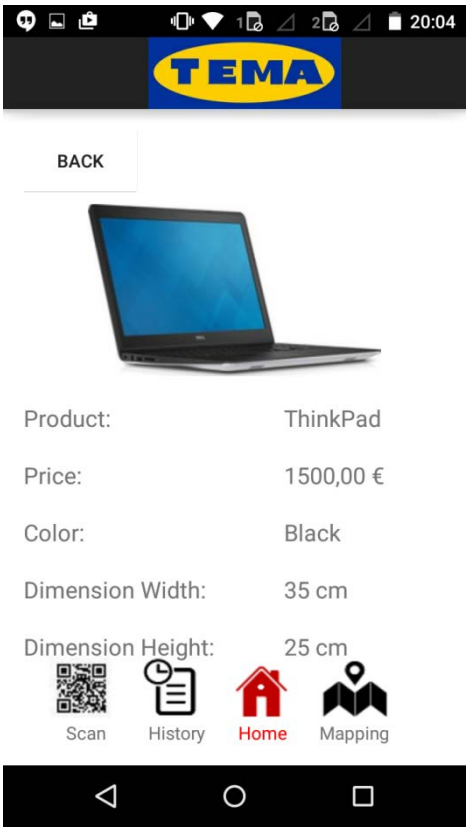
4.1.3 3η οθόνη(BarcodeReader)

Κατά τη διαδικασία της σάρωσης ενεργοποιείται η κάμερα της συσκευής, προκειμένου να γίνει η σάρωση του QRCode που αντιστοιχεί στο προϊόν το οποίο ο χρήστης θέλει μετέπειτα να συλλέξει από τον χώρο της αποθήκης.

Εικόνα 47 3η οθόνη εφαμρογής

4.1.4 4η οθόνη (Πληροφορίες των σαρωμένων αντικειμένων)

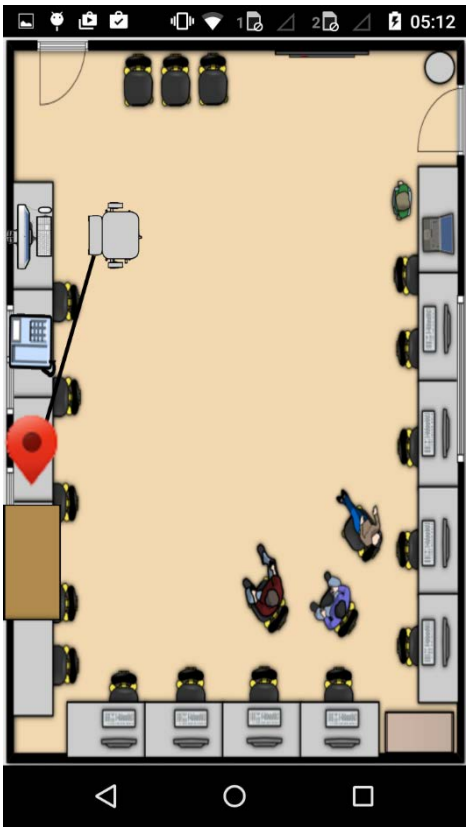
Σε αυτήν την οθόνη ο χρήστης ενημερώνεται με περεταίρω πληροφορίες για το αντικείμενο που έχει επιλέξει (το όνομα του προϊόν, την τιμή, το χρώμα, τις διαστάσεις μήκους/πλάτους καθώς και μία φωτογραφία του). Οι πληροφορίες προσκομίζονται από την αντίστοιχη βάση δεδομένων η οποία περιλαμβάνει όλα τα αντικείμενα που έχει τη δυνατότητα ο χρήστης να σαρώσει.



Εικόνα 48 4η οθόνη εφαρμογής

4.1.5 5η οθόνη (Mapping)

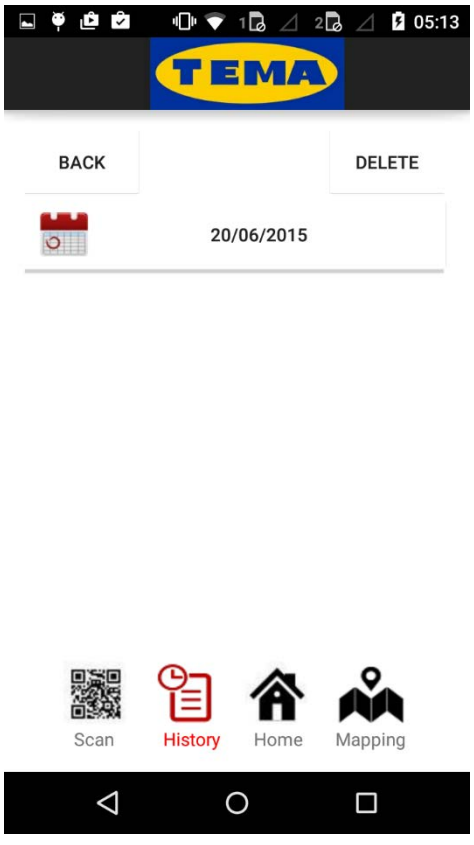
Στην οθόνη αυτή εμφανίζεται η αντιστοίχηση των αντικειμένων στο χώρο, βάση της κοντινότερης απόστασης τους από τον αυτόν ώστε να ακολουθήσει η συλλογή τους. Δηλαδή μόλις ο χρήστης ολοκληρώσει τη σάρωση των αντικειμένων, σκανάρει κάποιο qr_code το οποίο θα προσδιορίσει την ακριβή θέση του μέσα στον χώρο. Στη συνέχεια εμφανίζεται ο χάρτης με τις διαδοχικές διαδρομές προς το πλησιέστερο αντικείμενο τη φορά.



Εικόνα 49 5η οθόνη εφαρμογής

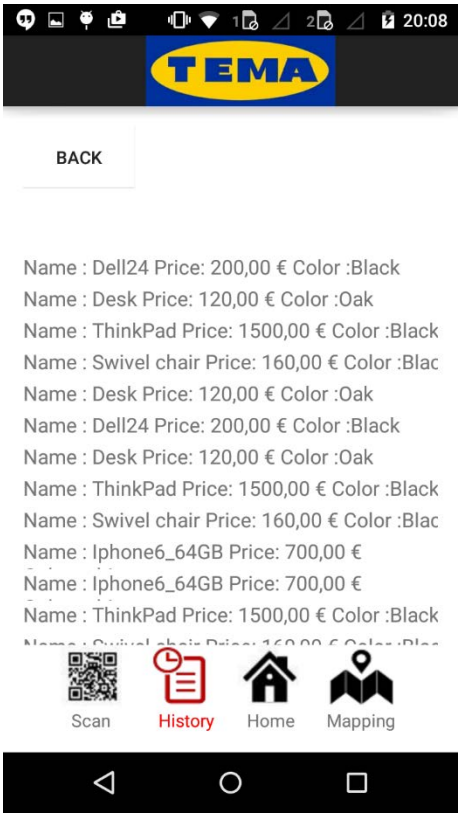
4.1.6 6η οθόνη (Ιστορικό)

Στην οθόνη αυτή προβάλλεται η λίστα από τα αντικείμενα που έχει σαρώσει ο χρήστης σε παλαιότερη επίσκεψη στο χώρο και τα έχει συλλέξει από τον χώρο αποθήκης. Η λίστα αυτή ενδέχεται να είναι κενή αν ο χρήστης εισέρχεται για πρώτη φορά στο σύστημα ή αν δεν έχει ολοκληρώσει την πλοήγηση ως προς τη συλλογή των αντικειμένων από την αποθήκη στο χώρο παλαιότερα. Αυτόματα όταν ο χρήστης μαζέψει το τελευταίο αντικείμενο από τα επιθυμητά που έχει επιλέξει και ενημερώσει το σύστημα, η λίστα του ιστορικού ανανεώνεται προβάλει την ημερομηνία που διεξάχθηκε η πράξη αυτή.



Εικόνα 50 6η οθόνη εφφαμρογής

4.1.7 7η οθόνη (Πληροφορίες Ιστορικού)



Εδώ εμφανίζονται αναλυτικά όλα τα προϊόντα που ο χρήστης έχει συλλέξει σε μια συγκεκριμένη ημερομηνία στο παρελθόν. Αναγράφεται το όνομα του προϊόντος η τιμή καθώς και το χρώμα του.

4.2 Ανάλυση των πινάκων της βάσης δεδομένων

4.2.1 Table USERS

όνομα πεδίου	**τύπος**	**μέγεθος**	**περιγραφή**
User_id	Varchar	50	Υποδεικνύει τον μοναδικό αναγνωριστικό του χρήστη που δημιουργείτε κατά την είσοδό του για πρώτη φορά στο σύστημα. Αποτελείται από την ημέρα, τον μήνα, τον χρόνο, την ώρα, τα λεπτά και τα δευτερόλεπτα που έγινε η είσοδος.
Product_code	Varchar	50	Το κάθε προϊόν έχει και αυτό αντίστοιχα το δικό του μοναδικό κωδικό όπου το έχουμε ορίσει εμείς.
Date_scanning_product	Varchar	50	Αναγράφετε η ημερομηνία που έχει σαρώσει ο χρήστης το αντίστοιχο αντικείμενο.
Second_done_is_selected	Varchar	50	Το πεδίο αυτό αρχικά κατά την σάρωση αντικειμένου αντιστοιχεί με -. Μόλις ολοκληρωθεί η πλοήγηση και ο χρήστης έχει μαζέψει όλα τα αντικείμενα του τότε αλλάζει σε 1.
Date_of_second_done	Varchar	50	Αντίστοιχα και το πεδίο αυτό αρχικά είναι άδειο και κατά την ολοκλήρωση της πλοήγησης αναγράφει την εκάστοτε ημερομηνία.

4.2.2 Table PRODUCTS

όνομα πεδίου	**Τύπος**	**μέγεθος**	**περιγραφή**
Product_id	Varchar	50	Αναγράφει το όνομα του κάθε προϊόντος.
Product_code	Varchar	50	Το qr_code αντιστοιχεί σε ένα μοναδικό product_code, όπου στη συνέχεια με τη σάρωση του qr_code γνωρίζουμε αντίστοιχα σε ποιο productcode αναφέρεται.
Product_price	Varchar	50	Περιέχει την τιμή του κάθε προϊόν μας.
Product_color	Varchar	50	Περιγράφει το χρώμα του προϊόν.
Product_dim_width	Varchar	50	Αναλύει τις διαστάσεις του προϊόν ως προς το μήκος.
Product_dim_depth	Varchar	50	Περιγράφει τις διαστάσεις του προϊόν ως προς το βάθος.
Product_dim_height	Varchar	50	Αναγράφει τις διαστάσεις του προϊόν ως προς το ύψος.
Packing_dim_length	Varchar	50	Περιέχει τις διαστάσεις τις συσκευασίας που περιέχει το προϊόν, ως προς το μήκος.
Packing_dim_width	Varchar	50	Περιγράφει τις διαστάσεις τις συσκευασίας που περιέχει το προϊόν, ως προς το πλάτος.
Packing_dim_height	Varchar	50	Αναγράφει τις διαστάσεις τις συσκευασίας που περιέχει το προϊόν, ως προς το ύψος.
Packing_dim_weight	Varchar	50	Αναλύει τις διαστάσεις τις συσκευασίας που περιέχει το προϊόν, ως προς το βάρος.
Product_num	Varchar	50	Περιέχει τους αριθμούς με τη σειρά που έχουμε ορίσει για το κάθε αντικείμενο ξεχωριστά.
Position	Varchar	50	Αναγράφει τις συντεταγμένες των αντικειμένων (χ,y) που χρησιμεύει κατά την πλοήγηση ώστε να γνωρίζουμε το κάθε προϊόν σε ποια θέση στον χάρτη βρίσκεται για να μπορέσουμε να τραβήξουμε σωστά τις γραμμές από τον χρήστη στο αντικείμενο αλλά και στα αντικείμενα μεταξύ τους.

4.3 Απεικόνιση βάσης

Ακολουθούν φωτογραφίες με τις απεικονίσεις των βάσεων δεδομένων

4.3.1 Users

Πίνακας **USERS**. Κάθε γραμμή αντιστοιχεί σε μια καταχώρηση σάρωσης αντικειμένου από τον χρήστη.

select * from EVE.USERS ×

Page Size: 50 Total Rows: 385 Page: 1 of 8 Matching Rows:

#	USER_ID	PRODUCT_CODE	DATE_SCANNING_PRODUCT	SECOND_DONE_IS_SELECTED	DATE_OF_SECOND_DONE
1	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
2	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
3	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
4	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
5	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
6	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
7	16/06/2015_22:30:44	002.148.75	17/06/2015	1	17/06/2015
8	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
9	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
10	16/06/2015_22:30:44	002.148.75	17/06/2015	1	17/06/2015
11	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
12	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
13	16/06/2015_22:30:44	002.148.75	17/06/2015	1	17/06/2015
14	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
15	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
16	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
17	16/06/2015_22:30:44	002.148.75	17/06/2015	1	17/06/2015
18	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
19	16/06/2015_22:30:44	001.105.69	17/06/2015	1	17/06/2015
20	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
21	16/06/2015_22:30:44	200.925.05	17/06/2015	1	17/06/2015
22	16/06/2015_22:30:44	002.148.75	17/06/2015	1	17/06/2015

Εικόνα 51 Πίνακας Users

4.3.2 Products

Πίνακας **PRODUCTS**. Κάθε γραμμή αντιστοιχεί σε μια καταχώρηση αντικειμένου με όλες τις απαραίτητες πληροφορίες.

select * from EVE.PRODUCT... ×

Page Size: 50 Total Rows: 5 Page: 1 of 1 Matching Rows:

#	PRODUCT_ID	PRODUCT_CODE	PRODUCT_PRICE	PRODUCT_COLOR	PRODUCT_DIM_WIDTH	PRODUCT_DIM_DEPTH
1	Desk	002.139.87	120,00	Oak	150	60
2	Iphone6_64GB	200.925.05	700,00	white	7,78	0,71
3	ThinkPad	001.135.70	1500,00	Black	35	25
4	Swivel chair	002.148.75	160,00	Black	60	65
5	Dell24	001.105.69	200,00	Black	56	5

Εικόνα 52 Πίνακας Products

4.4 Queries

```
String q = "select * from USERS where USER_ID='"+user_id+"' and
SECOND_DONE_IS_SELECTED = '0'";
        ResultSet rs = stm.executeQuery(q);

        while(rs.next())
        {
           String q1 = "select * from PRODUCTS where
PRODUCT_CODE='"+rs.getString(2)+"' ";
           ResultSet rs1 = stm1.executeQuery(q1);
           if(rs1.next())
           {}
        }
```

Στο πρώτο selectεπιλέγουμε από τον πίνακα USERS όπου το USER_IDνα είναι ίδιο με το user_id που έχει ο συνδεδεμένος χρήστης και το SECOND_DONE_IS_SELECTED να είναι μηδέν. Εάν ισχύει αυτό επιλέγουμε από τον πίνακα PRODUCTS τα προϊόντα που έχει σαρώσει ο χρήστης με το παραπάνω user_id με βάση του PRODUCT_CODE τους και παίρνουμε πληροφορίες ώστε να τα εμφανίσουμε στον χρήστη με μορφή λίστας.

```
String q = "delete from USERS where USER_ID = ? ";
```

Διαγράφουμε από τον πίνακα USERSτα πεδία όπου το USER_ID ισούται με το ερωτηματικό του συνδεδεμένου χρήστη (?).

```
String query = "delete from USERS where PRODUCT_CODE = ? and USER_ID = ? and
SECOND_DONE_IS_SELECTED = '0'";
```

Διαγράφει από τον πίνακα USERS τα πεδία όπου το PRODUCT_CODEισούται με το ερωτηματικό (?), το USER_IDεπίσης είναι ίσο με το ερωτηματικό (?) και το SECOND_DONE_IS_SELECTED ισούται με μηδέν (0).

```
String q = "select * from USERS where USER_ID='"+user_id+"' and
SECOND_DONE_IS_SELECTED = '1'";
```

Επιλέγει από τον πίνακα USERSόλες τις καταχωρήσεις όπου το USER_ID ισούται με το αναγνωριστικό του χρήστη που είναι συνδεδεμένος στο σύστημα εκείνη τη στιγμή και SECOND_DONE_IS_SELECTED ισούται με 1, δηλαδή τα συγκεκριμένα αντικείμενα έχουν ήδη συλλεχθεί από τον χρήστη.

```
String q1 = "select * from PRODUCTS where PRODUCT_CODE="'+code+'" ";
ResultSet rs = stm.executeQuery(q);
          while(rs.next())
         {
            String q1 = "select * from PRODUCTS where
PRODUCT_CODE="'+rs.getString(2)+'" ";
            ResultSet rs1 = stm1.executeQuery(q1);
            if(rs1.next())
            {}
         }
```

Επιστρέφονται από τον πίνακα PRODUCTSόλες τις καταχωρήσεις όπου το PRODUCT_CODE ισούται με το αναγνωριστικό του αντικειμένου

```
String q = "select * from USERS where USER_ID="'+user_id+'"
```

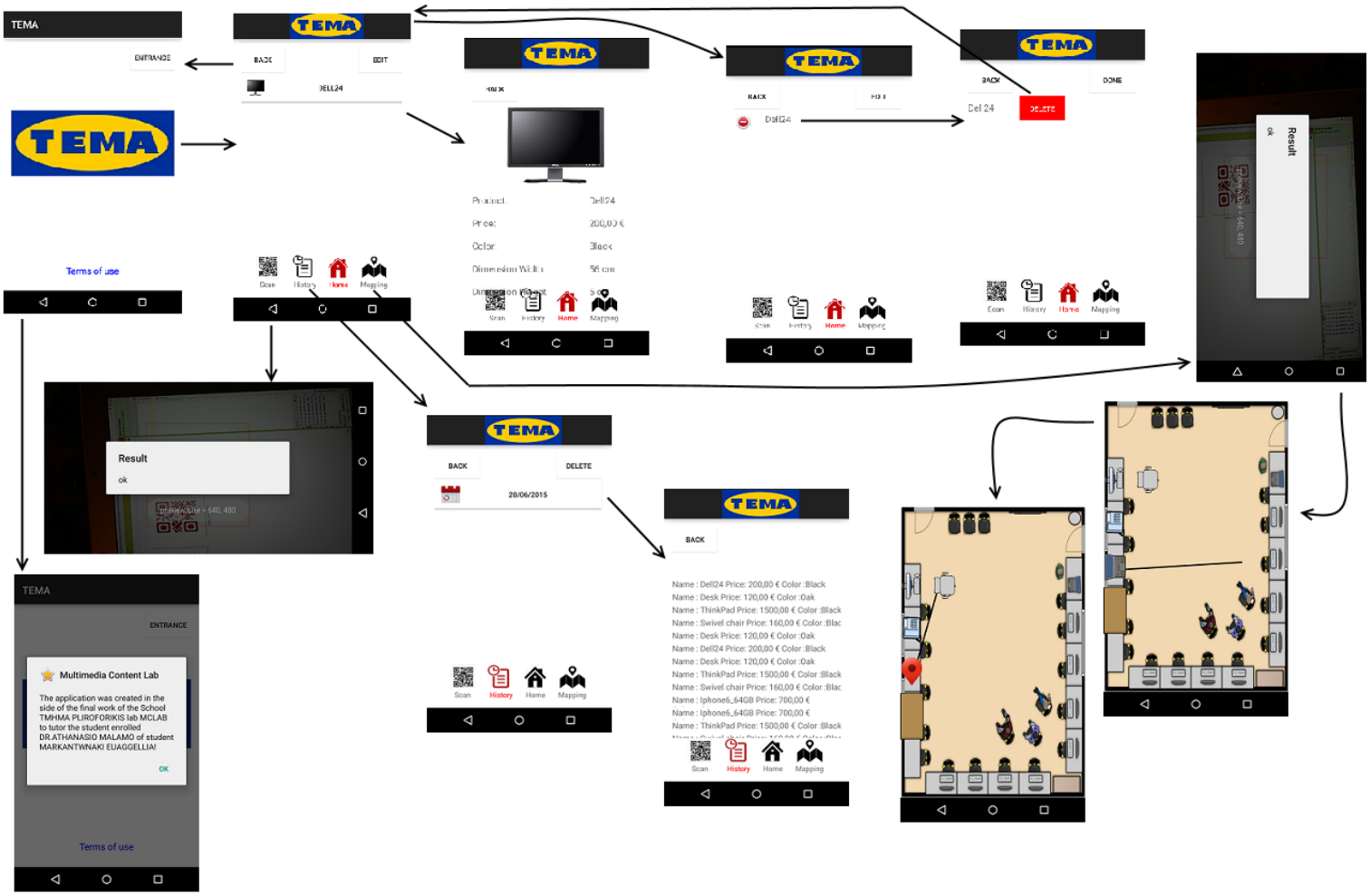
Επιστρέφονται από τον πίνακα USERSόλες τις καταχωρήσεις όπου το USER_ID ισούται με το αναγνωριστικό του χρήστη.

```
String q = "select * from USERS where USER_ID="'+user_id+'"
          ResultSet rs = stm.executeQuery(q);
          ResultSet rs1;
          while(rs.next())
          {
               if(rs.getString("SECOND_DONE_IS_SELECTED").equals("0"))
                 {
                    String pc = rs.getString("PRODUCT_CODE");

                    String q1 = "select * from PRODUCTS where PRODUCT_CODE="'+pc+'" ";
                    rs1 = pst1.executeQuery(q1);
                    while(rs1.next())
                    {
                       data.add(rs1.getString(12)+"_"+rs1.getString(13));
                    }

                    pst.executeUpdate("UPDATE USERS SET DATE_OF_SECOND_DONE =
'"+done_date+'" where SECOND_DONE_IS_SELECTED = '0'");
                    pst.executeUpdate("UPDATE USERS SET
SECOND_DONE_IS_SELECTED = '1' where SECOND_DONE_IS_SELECTED = '0' ");
}

          }
```

Πραγματοποιείται UPDATE στον πίνακα PRODUCTS όπου DATE_OF_SECOND_DONE ισούται με την ημερομηνία όπου ο χρήστης επέλεξε να κάνει Mapping στο σύστημα προκειμένου να προχωρήσει στη συλλογή τους και where SECOND_DONE_IS_SELECTED = 0 δηλαδή το συγκεκριμένο αντικείμενο δεν έχει συλλεχθεί ακόμη.

4.5 Φωτογραφία συστήματος

Ακολουθεί φωτογραφία με την συνοχή των οθονών του συστήματος.



Εικόνα 53 Φωτογραφία συστήματος

5 Αποτελέσματα

5.1 Περιθώρια βελτίωσης

Το σύστημα ΤΕΜΑ θα μπορούσε να βελτιωθεί σε μεγάλο βαθμό όσο αφορά την επικοινωνία μεταξύ της εφαρμογής και του server. Προς στιγμήν λειτούργει μόνο σε τοπικό επίπεδο και μόνο εάν η συσκευή και ο server βρίσκονται στο ίδιο δίκτυο.

Ακόμη θα μπορούσε να βελτιωθεί και σε επίπεδο χρηστικότητας με περισσότερα αντικείμενα προς σάρωση. Αυτό θα ήταν μια παράμετρος που θα επέφερε αύξηση της πολυπλοκότητας του αλγόριθμου Dijkstra.

Βιβλιογραφία

http://el.wikipedia.org/wiki/Android

http://www.doctorandroid.gr/p/iphone.html

http://www.adds.gr/company/technology/what-is-android/

http://www.allaboutandroid.gr/?p=6362

http://www.mydroid.gr/2013/02/leitourgiko-android-ekdoseis/

http://nefeli.lib.teicrete.gr/browse/stef/epp/2014/ArvanitisKonstantinos,KalykakisEleftherios/attached-document-1418805373-237616-9302/ArvanitisKonstantinos_KalykakisEleftherios2014.pdf

http://osarena.net/android/aguides/mathenontas-to-android-androidmanifest-xml-is-all-about-mathimata-4-5-6.html

http://www.techgear.gr/android-studio-now-available-94869/

http://apothetirio.teiep.gr/xmlui/bitstream/handle/123456789/998/fin_20040053.pdf?sequence=1

http://el.wingwit.com/Programming/java-programming/89253.html

https://support.office.com/el-gr/article/%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE-%CF%83%CF%84%CE%B7%CE%BD-SQL-%CF%84%CE%B7%CF%82-Access-d5f21d10-cd73-4507-925e-bb26e377fe7e?ui=el-GR&rs=el-GR&ad=GR#bm1

http://kddlab.di.uoa.gr/mmd/sites/default/files/PtyxiakiMaroulis.pdf

https://en.wikipedia.org/wiki/GlassFish

https://en.wikipedia.org/wiki/Java_servlet

6 Παράρτημα

6.1 Υπογραφές των συναρτήσεων από κάθε οθόνη (Clientside)

6.1.1 1η οθόνη - Είσοδος στο σύστημα

protected void onCreate(Bundle savedInstanceState) {}

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχηςactivity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

public void Go_To_Second_Screen(View view) {}

Σε αυτήν την συνάρτηση συνδέουμε την πρώτη οθόνη (MainActivity) με τη δεύτερη (MainActivity2), όταν πατηθεί το αντίστοιχο button από τον χρήστη.

Private void initiatePopupWindow() {}

Η συνάρτηση αυτή καλείται όταν ο χρήστης πατήσει το button “Terms of Use”, και εμφανίζεται ένα pop-up window με πληροφορείς της εφαρμογής.

6.1.2 2η οθόνη – Κεντρική οθόνη συστήματος

protected void onCreate(Bundle savedInstanceState) {}

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

public void Creation_Of_User() {}

Δημιουργεί έναν μοναδικό αναγνωριστικό για τον χρήστη με βάση την ημέρα τον μήνα τον χρόνο την ώρα και τα δευτερόλεπτα.

public void Get_Users_Scanned_Products(String user_id){}

Παίρνει ως όρισμα το αναγνωριστικό του χρήστη και καλεί την συνάρτηση όπου δημιουργεί ένα HTTPRequest προς στην βάση δεδομένων, όπου έχουν αποθηκευτεί τα σαρωμένα αντικείμενα του κάθε χρήστη και τα επιστρέφει.

protected void Progress_Bar_Visible() {}

Εμφανίζεται μια μπάρα φόρτωσης όση ώρα απαιτείται ώστε να φορτωθούν τα σαρωμένα αντικείμενα.

protected void Progress_Bar_Invisible() {}

Μόλις τα αντικείμενα φορτώσουν τότε κάνουμε την μπάρα που αναφέραμε παραπάνω μη εμφανή.

public class Get_Event_Products_from_Database extends AsyncTask<String, Void, Void> {}

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}
Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

private void Get_Products(final String output) {}
Η μέθοδος αυτή παίρνει ως όρισμα την επιστρεφόμενη τιμή από την βάση δεδομένων με την μορφή JSONstring. Στη συνέχεια γίνεται επεξεργασία αυτού του JSONstring το οποίο εντέλει δημιουργεί ένα ArrayList ώστε στη συνέχεια να γίνει η προβολή των αντικειμένων στην οθόνη του χρήστη βάση αυτού του ArrayList.

public void Add_Product(String product_name,String code) {}

Η συνάρτηση αυτή είναι υπεύθυνη για την προβολή των σαρωμένων αντικειμένων. Συγκεκριμένα δημιουργείται ένα κουμπί με συγκεκριμένες διαστάσεις που αντιστοιχεί σε κάθε ένα σαρωμένο αντικείμενο.

public void Product_More(String code) {}

Η συνάρτηση αυτή παίρνει ως όρισμα το αναγνωριστικό του σαρωμένου αντικειμένου και βάση αυτού επικαλείται κλίση στη βάση προκειμένου να επιστραφούν πληροφορίες για το συγκεκριμένο αντικείμενο.

public void Go_To_First_Screen(View view) {}

Σε αυτήν την συνάρτηση συνδέουμε την δεύτερη οθόνη με τη πρώτη , όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_Scan_Screen(View view) {}

Σε αυτήν την συνάρτηση συνδέουμε την δεύτερη οθόνη με την οθόνη σάρωσης όπου ανοίγει η camera, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_History_Screen(View v) {}

Σε αυτήν την συνάρτηση συνδέουμε την δεύτερη οθόνη με την οθόνη ιστορικού του χρήστη, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_Mapping_Screen(View view) {}

Σε αυτήν την συνάρτηση συνδέουμε την δεύτερη οθόνη με την οθόνη προβολής του mapping, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Edit_XML_First_Time(View v) {}

Μόλις πατηθεί το κουμπί **Edit** δημιουργούμε καινούργια στοίχιση στην οθόνη μας αφαιρούμε την προηγούμενη εμφάνιση καθώς και αριστερά από κάθε κουμπί εμφανίζεται μια εικόνα_κουμπί που υποδεικνύει τη διαγραφή .

public void Edit_XML_Second_Time(int id) {}

Μόλις πατηθεί η εικόνα_κουμπί που αναφέραμε παραπάνω, τότε αντίστοιχα αφαιρούμε την προηγούμενη εμφάνιση και δεξιά εμφανίζεται ένα κουμπί με κείμενο "**Delete**" (Διαγραφή)

Class DeleteProductFromDataBase extends AsyncTask<String, Void, Void> {}

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

Protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα γίνεται κλίση στην βάση δεδομένων, ελέγχει το product_code του προϊόν καθώς και το user_id του χρήστη και διαγράφει την αντίστοιχη στήλη από τη βάση.

protected void onPostExecute(Void feed) {}

Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.
Καλεί την **UpdateScreen** (δες παρακάτω)

Public voidUpdateScreen()

Κάνει επαναφόρτιση την οθόνη μας.

6.1.3 3η οθόνη - BarcodeReader

protected void onCreate(Bundle savedInstanceState) {}

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

Protected void onPause() { }

Η onPause επιτρέπει να σταματήσει τις συνεχιζόμενες ενέργειες που δεν θα πρέπει να συνεχιστούν, όπως βίντεο, κάθε πληροφορία που θα πρέπει να αποθηκεύονται μόνιμα σε περίπτωση που ο χρήστης συνεχίζει να αφήνει την εφαρμογή. Εάν ο χρήστης επιστρέφει στη δραστηριότητά από την κατάσταση παύσης, το σύστημα επανέρχεται και καλεί τη μέθοδο onResume ().

Protected void onResume() { }

Η συνάρτηση αυτή επιτρέπει την συνέχιση του activity.

6.1.4 4η οθόνη – Πληροφορίες σαρωμένων αντικειμένων

Protected void onCreate(BundlesavedInstanceState) {}

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

Public void Go_To_Second_Screen(View view) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την κεντρική, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_Scan_Screen(View view) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την οθόνη σάρωσης, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_History_Screen(View v) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την οθόνη ιστορικού, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_Mapping_Screen(View view) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την οθόνη mapping, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Get_More_Info () { }

Καλεί την Get_More_Info_About_Products (βλέπε παρακάτω)

public class Get_More_Info_About_Products extends AsyncTask<String, Void, Void> {

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}

Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

private void Get_Products(final String output) { }

Η μέθοδος αυτή παίρνει ως όρισμα την επιστρεφόμενη τιμή από την βάση δεδομένων με την μορφή JSONstring. Στη συνέχεια γίνεται επεξεργασία αυτού του JSONstring το οποίο εντέλει δημιουργεί ένα ArrayList ώστε στη συνέχεια να γίνει η προβολή των αντικειμένων στην οθόνη του χρήστη βάση αυτού του ArrayList.

private void GetInfo(ArrayList<String> list) { }

Η συνάρτηση αυτή επεξεργάζεται κατάλληλα το ArrayList και επιλέγει ανάλογα την πληροφορία που επρόκειτο να εμφανιστεί.

public void Add_Product(String name, String price, String color, String width ,String height) { }

Εμφανίζουμε με βάση τον κωδικό του προϊόντος την εικόνα , το όνομα, την τιμή , το χρώμα και τις διαστάσεις του.

6.1.5 5η οθόνη - Mapping

protected void onCreate(Bundle savedInstanceState) { }

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

class Set_Product_List_As_Done extends AsyncTask<String, Void, Void>{}

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}

Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

public void SendInfoToDatabase(View v) { }

Σε αυτήν την συνάρτηση ανανεώνουμε στην βάση δεδομένων την πληροφορία που χαρακτηρίζει τα αντικείμενους collapsed.

public void Go_To_MainActivity2() { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την κεντρική οθόνη, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

6.1.5.1 Προετοιμασία Canvas

public PrepareCanvasView( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για την αρχικοποίηση του Canvas, προκειμένου έχοντας γνώση για την κατάταξη των αντικειμένων, (μέσω του αλγορίθμου Dijkstra)προς την απόσταση στο χώρος στη συνέχει να εμφανιστεί ένα Path του μονοπατιού μεταξύ του χρήστη και των αντικειμένων.

protectedvoidonDraw( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για την απεικόνιση του μονοπατιού στον Canva

public void Draw_In_Canvas( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για την ανανέωση του μονοπατιού.

public void Find_Points_For_Canvas( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για τον προσδιορισμό των σημείων που πρόκειται να απεικονισθούν στο μονοπάτι.

publicvoidclearCanvas( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για τον καθαρισμό του Canvas ώστε να ανανεωθεί το μονοπάτι , όταν πλέον ο χρήστης έχει επιλέξει το συγκεκριμένο αντικείμενο και θέλει να προχωρήσει στο επόμενο.

public boolean Is_Button_Clicked( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για τον έλεγχο, εάν ο χρήστης έχει συλλέξει το αντικείμενο από το μονοπάτι ώστε να προχωρήσει στην ένδειξη του επόμενου συντομότερου μονοπατιού.

privatevoidinitiatePopupWindow( ) { }

Η συνάρτηση αυτή είναι υπεύθυνη για την ενημέρωση του χρήστη μέσω της εμφάνισης ενός pop-upwindow στην περίπτωση που ο χρήστης έχει επιλέξει όλα τα αντικείμενα του μονοπατιού .

public void Go_To_Main_Screen( )

Η συνάρτηση αυτή είναι υπεύθυνη για την μετάβαση του χρήστη στην κεντρική οθόνη.

6.1.5.2 MainActivity_Mapping_Continue

protected void onCreate( ) { }

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

class Set_Product_List_As_Done extends AsyncTask<String, Void, Void> {}

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}

Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

private void Get_Scannes_Products( ) { }

Η συνάρτηση είναι υπεύθυνη για την ανάκτηση των σαρωμένων αντικειμένων από τη βάση δεδομένων ώστε να προωθηθούν στον αλγόριθμο Dijkstra ώστε να υπολογιστεί το συντομότερο μονοπάτι.

public void Preparation_For_Dijkstra( ) { }

Η συνάρτηση είναι υπεύθυνη για την ανάκτηση των συντεταγμένων των σαρωμένων αντικειμένων από τη βάση δεδομένων ώστε να προωθηθούν στον αλγόριθμο Dijkstra ώστε να υπολογιστεί το συντομότερο μονοπάτι.

publicvoidGoToCanvasView( ) { }

Η συνάρτηση είναι υπεύθυνη για την αρχικοποίηση του Canvas ώστε στη συνέχεια να απεικονιστεί διαδοχικά το συντομότερο μονοπάτι μεταξύ των αντικειμένων που ο χρήστης σάρωσε.

publicvoidSendInfoToDatabase( ) { }

Η συνάρτηση είναι υπεύθυνη για την ενημέρωση της βάσης δεδομένων στην περίπτωση που ο χρήστης σύλλεξε το αντικείμενο ώστε να προχωρήσει η διαδικασία της εύρεσης του συντομότερου μονοπατιού.

6.1.5.3 Αλγόριθμος Dijkstra

public static boolean CheckIfArrayContainsNum( ) { }

Η συνάρτηση είναι υπεύθυνη για την εύρεση των αντικείμενων που είναι σαρωμένα

public static void dijkstra_algorithm( ) { }

Η συνάρτηση είναι υπεύθυνη για την εύρεση του συντομότερου μονοπατιού μέσα από τις κλίσης των ακόλουθων συναρτήσεων.

privatestaticintgetNodeWithMinimumDistanceFromUnsettled( ) { }

Η συνάρτηση είναι υπεύθυνη για την εύρεση του γειτονικού κόμβου με την μικρότερη απόσταση. Αυτό το επιτυγχάνει καλώντας την συνάρτηση evaluateNeighbours

privatestaticvoidevaluateNeighbours( ) { }

Η συνάρτηση είναι υπεύθυνη για την επιλογή του γειτονικού κόμβου με την μικρότερη απόσταση

public static void Initialize_adjacency_matrix( ) { }

Η συνάρτηση είναι υπεύθυνη για την αρχικοποίηση των αποστάσεων μεταξύ των κόμβων του γράφου.

public static void The_Final_Distances_1( ) { }

Η συνάρτηση είναι υπεύθυνη για την αρχικοποίηση του πίνακα με τις αποστάσεις στις οποίες θα βασιστεί ο αλγόριθμος Dijkstraώστε να υπολογίσει το μονοπάτι. Η αρχικοποίηση προκύπτει μέσω του πίνακα που περιλαμβάνει όλες τις αποστάσεις μεταξύ των κόμβων του γράφου.

6.1.5.4 CanvasView

protectedvoidonCreate( ) { }

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

public boolean Check_If_Button_Is_Scanned( ) { }

Η συνάρτηση είναι υπεύθυνη για τον έλεγχο εάν το αντικείμενο που αντιστοιχεί σε κόμβο του γράφου έχει σαρωθεί από τον χρήστη ώστε να απεικονιστεί στο σχήμα.

6.1.6 6η οθόνη – Ιστορικό

protected void onCreate(Bundle savedInstanceState) { }

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

publicvoidGetUserHistory() { }

Καλεί την συνάρτηση **GetHistoryFromDatabase**(βλέπε παρακάτω)

protected void Progress_Bar_Visible() {}

Εμφανίζεται μια μπάρα φόρτωσης όση ώρα απαιτείται ώστε να φορτωθούν τα σαρωμένα αντικείμενα.

protected void Progress_Bar_Invisible() {}

Μόλις τα αντικείμενα φορτώσουν τότε κάνουμε την μπάρα που αναφέραμε παραπάνω μη εμφανή.

class GetHistoryFromDatabase extends AsyncTask<String, Void, Void> { }

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}
Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

protected void Progress_Bar_Visible() {}

Αυτή η συνάρτηση καλείται προκειμένου να εμφανίσει το progressbar καθόλη την διάρκεια αναμονής του χρήστη μέχρι να φορτώσουν τα αποτελέσματα που αναμένει.

protected void Progress_Bar_Invisible()

Αυτή η συνάρτηση καλείται προκειμένου να εξαφανίσει το progressbar εφόσον τα αποτελέσματα εμφανίστηκαν στην οθόνη του.

private void Get_History(final String output) { }

Η μέθοδος αυτή παίρνει ως όρισμα την επιστρεφόμενη τιμή από την βάση δεδομένων με την μορφή JSONstring. Στη συνέχεια γίνεται επεξεργασία αυτού του JSONstring το οποίο εντέλει δημιουργεί ένα ArrayList ώστε στη συνέχεια να γίνει η προβολή των αντικειμένων στην οθόνη του χρήστη βάση αυτού του ArrayList.

private void Get_History (ArrayList<String> list) { }

Η συνάρτηση αυτή επεξεργάζεται κατάλληλα το ArrayList και επιλέγει ανάλογα την πληροφορία που επρόκειτο να εμφανιστεί.

public void Go_To_First_Screen (View v) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την κεντρική οθόνη, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Delete_History(View v) { }

Καλεί την συνάρτηση **Delete_History**όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_Scan_Screen(View v) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την οθόνη σάρωσης , όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_HomePage(View v) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την κεντρική οθόνη, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Go_To_Mapping_Screen(View v) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την οθόνη mapping, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

Public void Delete_History() { }

Καλεί την συνάρτηση **DeleteHistoryFromDatabase**(βλέπε παρακάτω)

class DeleteHistoryFromDatabase extends AsyncTask<String, Void, Void> {}

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}

Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

publicvoidUpdateScreen() { }

Κάνει ανανέωση της οθόνης**MainActivity2**

public void Add_History_To_XML(final String product_name) { }

Η συνάρτηση Add_History_To_XML είναι υπεύθυνη για την δημιουργία και προβολή του ιστορικού στην οθόνη του χρήστη.

PublicvoidHistory_More(Stringdate) { }

Η συνάρτηση αυτή είναι υπεύθυνη ώστε να τοποθετεί στο κουμπί κείμενο, την ημερομηνία που έγινε το δεύτερο done στην οθόνη πλοήγησης.

6.1.7 7η οθόνη Πληροφορίες Ιστορικού

protected void onCreate(Bundle savedInstanceState) { }

Καλείται όταν το activity δημιουργείται για πρώτη φορά. Η συνάρτηση αυτή μας ορίζει το View της αντίστοιχης activity και επιτρέπει την δήλωση μεταβλητών. Η μέθοδος αυτή παρέχει επίσης μια δέσμη που περιέχει προηγούμενο στιγμιότυπο της activity αυτής.

public void Go_To_History_Screen(View view) { }

Σε αυτήν την συνάρτηση συνδέουμε την οθόνη που βρίσκεται ο χρήστης με την οθόνη ιστορικού, όταν πατηθεί το αντίστοιχο button από τον χρήστη.

public void Get_More_InfoHistory() { }

Η συνάρτηση αυτή είναι υπεύθυνη για την κλίση της κλάσης για να ανακτηθεί το ιστορικό του χρήστη από τη βάση

class Get_More_Info_History_from_Database extends AsyncTask<String, Void, Void> {}

Η συνάρτηση αυτή επικαλείται την κλίση προς τη βάση δεδομένων και περιλαμβάνει τις εξής συναρτήσεις:

protected Void doInBackground(String... urls) {}

Όταν απαιτείται η εκτέλεση λειτουργίας για πολύ μεγάλο χρονικό διάστημα, όπως ένα HTTPRequest επιβάλλεται η κλίση της μεθόδου αυτής. Συγκεκριμένα **κ**άνει κλίση στην βάση δεδομένων μας και με βάση τον μοναδικό κωδικό του χρήστη (user_id).

protected void onPostExecute(Void feed) {}

Αυτή η συνάρτηση καλείται μετά τη μέθοδο doInBackground και ολοκληρώνει την επεξεργασία. Τα αποτελέσματα από την doInBackground περνά σε αυτή τη μέθοδο.

private void Get_History_Info(final String output) { }

Η μέθοδος αυτή παίρνει ως όρισμα την επιστρεφόμενη τιμή από την βάση δεδομένων με την μορφή JSONstring. Στη συνέχεια γίνεται επεξεργασία αυτού του JSONstring το οποίο εντέλει δημιουργεί ένα ArrayList ώστε στη συνέχεια να γίνει η προβολή των αντικειμένων στην οθόνη του χρήστη βάση αυτού του ArrayList.

private void Get_Products_Of_History_Info_from_DB (ArrayList<String> list) { }

Η συνάρτηση αυτή επεξεργάζεται κατάλληλα το ArrayList και επιλέγει ανάλογα την πληροφορία που επρόκειτο να εμφανιστεί

public void Add_Product(String tmp) { }

Η συνάρτηση αυτή είναι υπεύθυνη για την προβολή των αντικειμένων στην οθόνη, ως όρισμα έχει το όνομα του αντικειμένου

6.2 Server side

Collect_All_Products_Of_User_Id

public class Collect_All_Products_Of_User_Id extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την συλλογή των σαρωμένων αντικειμένων του χρήστη και να τα επιστρέψει στη μεριά του χρήστη, ώστε να προβληθούν.

DeleteHistoryFromDatabase

public class DeleteHistoryFromDatabase extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την διαγραφή του ιστορικού του χρήστη από τη βάση δεδομένων.

Delete_User_Scanned_Product

public class Delete_User_Scanned_Product extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την διαγραφή ενός συγκεκριμένου ήδη σαρωμένου αντικειμένου από τη βάση δεδομένων.

GetHistoryFromDatabase

public class GetHistoryFromDatabase extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την ανάκτηση του ιστορικού του χρήστη από τη βάση δεδομένων.

Get_More_Info_About_Products

public class Get_More_Info_About_Products extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την ανάκτηση πληροφοριών για αντικείμενο που έχει επιλέξει ο χρήστης από τη βάση δεδομένων.

More_History_Info

public class More_History_Info extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την ανάκτηση των πληροφοριών του ιστορικού για μια συγκεκριμένη ημερομηνία όπου ο χρήστης σύλλεξε τα αντικείμενα του

SetProductListAsDone

public class SetProductListAsDone extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την ενημέρωση της βάσης δεδομένων ώστε να χαρακτηρίσει τα σαρωμένα αντικείμενα του χρήστη ως ήδη συλλεγμένα

UsersScannedObjects

public class UsersScannedObjects extends HttpServlet {}

Η συνάρτηση αυτή είναι υπεύθυνη για την εισαγωγή στη βάση δεδομένων του αντικειμένου που σαρώνει ο χρήστης.

6.3 Κώδικας

6.3.1 μονοπατιού στον Canvas

Ο ακόλουθος κώδικας είναι υπεύθυνος για την δημιουργία του μονοπατιού στον Canvas και αυτό επιτυγχάνεται ορίζοντας και προστίθοντας σε έναν πίνακα από Points το σημείο αφετηρίας και το σημείο προορισμού.

```
public void Draw_In_Canvas()
{
      Point[] points = null;
      if(from_point_x == to_point_x)
      {
          if(from_point_y<to_point_y)
          {
                    points = new Point[]{new Point(from_point_x, from_point_y), new
                    Point(from_point_x + 100, from_point_y),
                   new Point(to_point_x + 100, to_point_y ), new
                   Point(to_point_x,to_point_y)};
          }
          else if(from_point_y > to_point_y)
          {
                    points = new Point[]{new Point(from_point_x, from_point_y), new
                    Point(from_point_x + 100, from_point_y),
                   new Point(from_point_x + 100, from_point_y - 180), new Point(from_point_x
                   - 100, from_point_y - 180)};
          }
      }
else
 {
                    points = new Point[]{new Point(from_point_x, from_point_y), new
                    Point(to_point_x, to_point_y) };
      }
      boolean first = true;
      for(Point point : points)
{
         if(first)
{
          first = false;
                    mPath.moveTo(point.x, point.y);
          }
else
{
                    mPath.lineTo(point.x, point.y);
          }
      }
      mCanvas.drawPath(mPath, mPaint);
   }
```

6.3.2 δημιουργία του HTTPRequest

Ο ακόλουθος κώδικας είναι υπεύθυνος για την δημιουργία του HTTPRequest. Αυτό επιτυγχάνεται αρχικοποιώντας έναν HTTPClientκαι ορίζοντας την σύνδεση με τον server. Στη συνέχεια μέσω του inputStreamReader λαμβάνουμε το responseαπό τη μεριά του server.

```
public static String executeHttpGet(String url) throws Exception
{
      BufferedReader in = null;
      try {
         HttpClient client = getHttpClient();
         HttpGet request = new HttpGet();
         request.setURI(new URI(url));
         HttpResponse response = client.execute(request);

         in = new BufferedReader(new
          InputStreamReader(response.getEntity().getContent()));

         StringBuffer sb = new StringBuffer("");
         String line = "";
         String NL = System.getProperty("line.separator");
         while ((line = in.readLine()) != null)
 {
            sb.append(line + NL);
         }
         in.close();

         String result = sb.toString();
         return result;
      } finally {
         if (in != null) {
            try {
               in.close();
            } catch (IOException e) {
e.printStackTrace();
            }
         }
      }
   }
```

Οκώδικας που ακολουθεί είναι υπεύθυνος για την μετατροπή του JsonString σε JsonArray και μετέπειτα η μετατροπή του JsonArray σε ArrayList

```
private void Get_Products(final String output)
    {
      MainActivity2.this.runOnUiThread(new Runnable()
      {
        @Override
        public void run()
        {
          if(output!=null && output.length()>0 )
          {
            String jsonString = output;
JSONArray jsonArray;

            try {
              jsonArray = new JSONArray(jsonString);
              list = new ArrayList<String>();
              for (int i = 0; i < jsonArray.length(); i++)
              {
                list.add(jsonArray.getString(i));
              }
            } catch (JSONException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
            }
            Get_Products_from_DB(list);
}
        }
      });
    }
```