ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΜΜΕ
(ΠΑΡΑΡΤΗΜΑ ΠΥΡΓΟΥ)

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

Καλόμοιρος Θέμελης

Κουφός Δημήτρης

ΕΙΣΗΓΗΤΗΣ: ΚΟΥΤΡΑΣ ΑΘΑΝΑΣΙΟΣ

2015

Πύργος, Μάρτιος 2015

Περιεχόμενα

Εικόνες

Πίνακες

Ευχαριστίες

Ευχαριστώ τη Ζωή.

Περίληψη

Αρχικά πραγματοποιείται σύγκριση του Android με άλλα λειτουργικά συστήματα για έξυπνα κινητά τηλέφωνα, όπως το iOs της Apple και το Windows Phone της Microsoft, σε διάφορους τομείς. Δεν θα μπορούσε να λείψει μία πλήρης αναφορά των εκδόσεων του Android που χρησιμοποιούνται σε συσκευές μέχρι και σήμερα. Στη συνέχεια παρουσιάζονται τα εργαλεία στα οποία ο developer θα στηριχθεί το «χτίσιμο» μίας εφαρμογής, οι τοποθεσίες στο διαδίκτυο για να πραγματοποιηθεί η λήψη αυτών καθώς και οι αναλυτικές οδηγίες για εγκατάσταση. Αργότερα περιγράφεται διεξοδικά η ανατομία μίας εφαρμογής του λειτουργικού συστήματος της Google όπως και η χρήση του τελευταίου API υπηρεσιών θέσης της εταιρείας. Αφού εγκαταληφθούν οι προηγούμενες περιγραφές, η εργασία προχωρά στην αναλυτική περιγραφή της τοπικής εφαρμογής KalymnosGuide, η οποία εύστοχα παρομοιάζεται με έναν τουριστικό οδηγό για χρήστες του Android. Από την σύλληψη της ιδέας, την προσπάθεια υλοποίησης αυτής, μέχρι και τη δημιουργία κλάσεων, δραστηριοτήτων κα. Η εργασία κλείνει με εκτενή ανάλυση του κώδικα των αρχείων που συγκροτούν την KalymnosGuide, όπως αρχεία δραστηριοτήτων, κλάσεων, πόρων κτλ, ενώ παράλληλα προβάλλονται τα τελικά συμπεράσματα των συντακτών.

Abstract

This thesis begins with a comparison of Android to other operating systems for smart phones, like Apple's iOs and Microsoft's Windows Phone, in various fields. It could not miss a complete reference about all the versions of Android that are being used in devices until today. The following topic talks about the tools that a developer should acquire in order to build an application, shows the sites for downloading as well as detailed instructions for installation. Later, the anatomy of an application of Google's operating system is being discussed in detail, and also the use of the latest of the company's API location services. Having abandoned the previous descriptions, work is proceeding to a detailed description of the application KalymnosGuide, which can be aptly likened to a tourist guide for the Android users. From conception, the implementation of this effort, up and creating classes, activities and more. The documend ends with an extensive analysis of the code from files that made up the KalymnosGuide application, such as activities, classes, resources, etc., while the final conclusions of the authors are being displayed.

1 Ανάπτυξη εφαρμογών για κινητά έξυπνα τηλέφωνα

1.1 Εισαγωγή στο Android (χρήση, σύγκριση με άλλα συστήματα κλπ)

Το έξυπνο τηλέφωνο (αγγλικά: smartphone) είναι ένα κινητό τηλέφωνο βασισμένο σε ένα λειτουργικό σύστημα κινητής τηλεφωνίας με περισσότερη προηγμένη υπολογιστική ικανότητα και συνδεσιμότητα σε σχέση με ένα απλό κινητό τηλέφωνο.

Τα πρώτα smartphones συνδύαζαν τις λειτουργίες ενός προσωπικού ψηφιακού βοηθού (PDA) και ενός κινητού τηλεφώνου. Σε μεταγενέστερα μοντέλα προστέθηκαν οι λειτουργίες των φορητών media players, low-end compact ψηφιακές φωτογραφικές μηχανές, βιντεοκάμερες τσέπης, καθώς και μονάδες πλοήγησης GPS, με αποτέλεσμα να διαμορφωθεί μια πολυχρηστική συσκευή. Πολλά σύγχρονα smartphones περιλαμβάνουν επίσης οθόνες αφής υψηλής ανάλυσης και web browsers που εμφανίζουν τυποποιημένες ιστοσελίδες, καθώς και βελτιστοποιημένες ιστοσελίδες για κινητά. Η πρόσβαση σε δεδομένα υψηλής ταχύτητας παρέχεται μέσω Wi-Fi και μέσω κινητών ευρυζωνικών υπηρεσιών. Τα τελευταία χρόνια, η ταχεία ανάπτυξη στην αγορά των εφαρμογών για κινητά και στο εμπόριο κινητών τηλεφώνων έχει γίνει οδηγός για την ευρεία υιοθέτηση των smartphones.

Τα λειτουργικά συστήματα (OS) των κινητών τηλεφώνων που χρησιμοποιούνται από τα σύγχρονα smartphones περιλαμβάνουν το Android της Google, το iOS της Apple, το Symbian της Nokia, το BlackBerry OS της RIM, το Bada της Samsung, τα Windows Phone της Microsoft, το webOS της Hewlett-Packard, καθώς και ενσωματωμένες διανομές Linux όπως το Maemo και το MeeGo. Τέτοιου είδους λειτουργικά συστήματα μπορούν να εγκατασταθούν σε πολλά διαφορετικά μοντέλα κινητών τηλεφώνων και συνήθως κάθε συσκευή μπορεί να λάβει πολλές ενημερωμένες εκδόσεις λογισμικού λειτουργικού συστήματος κατά τη διάρκεια ζωής της. Μερικά άλλα επερχόμενα λειτουργικά συστήματα είναι το Firefox OS της Mozilla, το Ubuntu Phone της Canonical Ltd's και το Tizen.

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

Τα τελευταία 5 χρόνια η πλατφόρμα Android έχει γίνει πολύ γνωστή στην αγορά των κινητών επικοινωνιών και κερδίζει όλο και περισσότερο κοινό. Έχει αρκετές βελτιώσεις σε ό,τι αφορά την λειτουργικότητα SDK, στην διαθεσιμότητα κινητών συσκευών και στο σύνολο χαρακτηριστικών. Στα χέρια των καταναλωτών βρίσκονται σήμερα πολλές τηλεφωνικές και άλλες συσκευές. Η πλατφόρμα Android χρησιμοποιείται σε ταμπλέτες, σε netbooks, σε συστήματα ανάγνωσης ηλεκτρονικών βιβλίων, στην Google TV, στις ψηφιακές κορνίζες και σε διάφορα άλλα ηλεκτρονικά προϊόντα. Οι εταιρίες και οι φορείς κινητών

επικοινωνιών αντιμετωπίζουν σοβαρά την πλατφόρμα και ξοδεύουν πολλά χρήματα σε διαφημιστικές εκστρατείες για τις συσκευές Android.

Τα τελευταία χρόνια, η πλατφόρμα Android έχει αλλάξει από μία πρώιμη πλατφόρμα σε μία πλατφόρμα, ισχυρά ανταγωνιστική των γνωστότερων πλατφόρμων. (Μιλάμε για πλατφόρμες όπως το iPhone και το BlackBerry). Το Android δεν είναι μόνο η υπ' αριθμόν ένα πλατφόρμα για έξυπνα τηλέφωνα, έχοντας υπερκεράσει μέχρι τα τέλη του 2013 το Symbian, αλλά έχει γίνει επίσης το πλέον επιθυμητό λειτουργικό σύστημα ανάμεσα σε καταναλωτές στις ΗΠΑ ένας ισχυρισμός που υποστηρίζεται από το 50% των νέων πωλήσεων smartphones.

	3 mo. Ending June 2012	3 mo. Ending June 2013
US Market	100,00%	100,00%
iOs	39,2	42,5
Android	52,6	51,5
BlackBerry	4	1,1
Windows	2,9	4
Other	1,3	0,9

Πίνακας 1-1: Android **και** iOs **κυριαρχούν στην αγορά των ΗΠΑ**

Θα μπορούσαμε να συγκρίνουμε την πλατφόρμα Android σε σχέση με τις άλλες; Δύσκολο γιατί κάθε άνθρωπος σκέφτεται και πράττει διαφορέτικα, σύμφωνα με τις ανάγκες του φυσικά. Οι άνθρωποι σε όλο τον κόσμο χρησιμοποιούν διαφορετικά τηλέφωνα, σε διαφορετικές θέσεις, για διαφερετικούς λόγους – λόγους όπως είναι η τιμή, η διαθεσιμότητα, η ποιότητα της κάλυψης, το σύνολο χαρακτηριστικών, η σχεδίαση, η εξοικείωση και η συμβατότητα. Δεν υπάρχει μία και μοναδική απάντηση σε αυτήν την συζήτηση.

Κάθε πλατφόρμα έχει σαφή πλεονεκτήματα επί των υπολοίπων και αυτά τα πλεονεκτήματα μπορούν να μεγιστοποιηθούν. Το κόλπο είναι να γνωρίζει κανείς ποια πλατφόρμα να χρησιμοποιήσει για ένα δεδομένο έργο. Μερικές φορές , η απάντηση είναι να χρησιμοποιηθούν όσο το δυνατό περισσότερες πλατφόρμες.Γιατί διαλέγουμε εμείς την πλατφόρμα android; Γιατί το Android είναι η πρώτη πλήρης, ανοικτή και δωρεάν διαθέσιμη πλατφόρμα κινητών επικοινωνιών. Οι προγραμματιστές έχουν ένα πλήρες κιτ ανάπτυξης λογισμικού, με άφθονα εργαλεία για ανάπτυξη ισχυρών, πλούσιων σε χαρακτηριστικά εφαρμογών. Η πλατφόρμα είναι ανοικτού πηγαίου κώδικα, και βασίζεται σε δοκιμασμένα και σωστά πρότυπα ανοικτού πηγαίου κώδικα, με τα οποία οι προγραμματιστές είναι εξοικειωμένοι. Το καλύτερο απ' όλα είναι ότι δεν υπάρχουν ακριβοί φραγμοί εισόδου για τους προγραμματιστές και δεν απαιτούνται χρεώσεις για άδειες χρήσης. Οι προγραμματιστές του Android έχουν πολλές επιλογές για διανομή και εμπορική εκμετάλλευση των εφαρμογών τους. Παρά ταύτα η πλατφόρμα Android δεν έχει φτάσει ακόμη στις πλήρεις δυνατότητες της. Αυτό σημαίνει συχνές ενημερώσεις του SDK, μία έκρηξη νέων συσκευών στην αγορά, και μία απαίτηση να παρακολουθεί κανείς συνέχεια για οτιδήποτε συμβαίνει στην αγορά του Android.

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. Ετσι διευκολύνονται στην ανάπτυξη πιο περίπλοκου και πιο πλούσιου λειτουργικού λογισμικού. Αυτό το λειτουργικό σύστημα κινητών τηλεφώνων (ή άλλων μικρών φορητών συσκεών που συνδέονται στο διαδικτυο) 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 ελεύθερων για χρήση προηγμένων εργαλείων λογισμικού.

Δεν υπάρχει διαφορά μεταξύ των λειτουργιών / εφαρμογών οι οποίες ειναι ενσωματωμένες στο τηλεφώνου απο τις εφαρμογές που δημιουργούνται και προστίθενται από τρίτους προγραμματιστές Android. Οι τελευταίες μπορούν και έχουν την ίδια πρόσβαση σε όλες τις κύριες λειτουργίες της συκευής 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 απο 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.

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.

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 απο έτοιμα εργαλεία αυξάνει σημαντικά την παραγωγικότητα των προγραμματιστών Android εφαρμογών και τους βοηθά να δημιουργήσουν εκπληκτικά πλούσιο λογισμικό γρηγορότερα και με λιγότερα λάθη.

Εκδόσεις Android
Android beta – 2007

Εκδόθηκε το Νοέμβριο του 2007 μαζί με το αντίστοιχο πακέτο ανάπτυξης εφαρμογών (SDK).

Android 1.0-2008

Εκδόθηκε το Σεπτέμβριο του 2008. Η επίσημη ονομασία του θα ήταν Astro αλλά λόγω πνευματικών δικαιωμάτων ονομάστηκε Android 1.0. Η πρώτη συσκευή με αυτό το λειτουργικό ήταν το HTC Dream και είχε τις εξής δυνατότητες:

· Δυνατότητα απόκτησης εφαρμογών και ενημερώσεων από το Android market.

· Διαδικτυακό πλοηγό με δυνατότητα zoom.

· Υποστήριξη κάμερας.

· Πλήρης πρόσβαση σε web mail server, POP3, SMTP και IMAp4

· Συγχρονισμό με τις υπηρεσίες της Google (Gmail, Contacts, Calendar, Maps, Sync, Search, Talk)

· Εφαρμογές instant messaging

· Φωνητικές κλήσεις

· Εφαρμογή παρακολούθησης βίντεο από το Youtube

· Bluetooth και WiFi

Android 1.1

Εκδόθηκε το Φεβρουάριο του 2009. Η επίσημη ονομασία του θα ήταν Bender αλλά λόγω πνευματικών δικαιωμάτων ονομάστηκε Android 1.1. Διόρθωνε κάποια μικροπροβλήματα και έδινε δυνατότητα αποθήκευσης των συνημμένων αρχείων των μηνυμάτων.

Cupcake 1.5

Εκδόθηκε τον Απρίλιο του 2009 και βασίστηκε στον πυρήνα 2.6.27 του Linux. Περιελάμβανε βελτιώσεις στο γραφικό περιβάλλον και νέες δυνατότητες. Κάποιες από αυτές ήταν:

· Υποστήριξη εικονικών πληκτρολογιών τρίτων κατασκευαστών με ενσωματωμένο λεξικό και προτεινόμενες λέξεις στο χρήστη.

· Υποστήριξη εργαλείων της επιφάνειας εργασίας που λαμβάνουν ενημερώσεις (Widgets)

Εγγραφή και αναπαραγωγή βίντεο σε μορφή Mp4 και 3GP

· Αυτόματη σύνδεση με συσκευές Bluetooth

· Δυνατότητα αντιγραφής και επικόλλησης στον διαδικτυακό πλοηγό

· Αυτόματη περιστροφή της εικόνας

· Δυνατότητα upload βίντε και εικόνων στο Youtube και το Picasa

Donut 1.6

Εκδόθηκε τον Σεπτέμβριο του 2009 και βασίστηκε στον πυρήνα 2.6.29 του Linux. Κάποιες από τις νέες δυνατότητες ήταν:

· Υποστήριξη οθονών μεγαλύτερων αναλύσεων

· Βελτιωμένη λειτουργία αναζήτησης που πλέον έγινε πιο γρήγορη και περιλαμβάνει αναζήτηση στο διαδίκτυο, στο ιστορικό του τηλεφώνου, στα μηνύματα, στο marketplace και σε άλλα

· Βελτιωμένη ταχύτητα στις εφαρμογές που χρησιμοποιούν την κάμερα

Eclair 2.0/2.1

Εκδόθηκε τον Οκτώβριο του 2009 και βασίστηκε στον πυρήνα 2.6.29 του Linux. Κάποιες από τις νέες δυνατότητες ήταν:

· Υποστήριξη πολλαπλών χρηστών για λογαριασμούς email

· Υποστήριξη Bluetooth 2.1

· Δυνατότητα επιλογής αποστολής μηνύματος, κλήσης, αποστολής email απλά με το άγγιγμα μιας επαφής

· Δυνατότητα αναζήτησης σε όλα τα αποθηκευμένα μηνύματα

· Πολλές νέες λειτουργίες γαι την κάμερα όπως υποστήριξη flash, ψηφιακό zoom, ιορροπία λευκού, εστίαση μάκρο και χρωματικά εφέ

· Μεγαλύτερη ταχύτητα πληκτρολόγησης στο εικονικό πληκτρολόγιο με πιο έξυπνο λεξικό και δυνατότητα εκμάθησης ανάλογα με τις λέξεις που πληκτρολογεί ο χρήστης

· Ανανεωμένο περιηγητή διαδικτύου με σελιδοδείκτες με εικόνες, δυνατότητα zoom το διπλό άγγιγμα και υποστήριξη HTML5

· Βελτιωμένη υποστήριξη και ταχύτητα υλικού σε όλους τους τομείς (οθόνες, επεξεργαστές, κάμερες)

· Βελτιωμένη εφαρμογή Google Maps

· Δυνατότητα διαχείρισης πολλαπλών σημείων αφής στην οθόνη

Froyo 2.2.x

Εκδόθηκε τον Μάιο του 2010 και βασίστηκε στον πυρήνα 2.6.32 του Linux. Η πρώτη συσκευή που το υιοθέτησε ήταν η Google Nexus One. Κάποιες από τις νέες δυνατότητες ήταν:

· Βελτιωμένη ταχύτητα και διαχείριση πόρων και κατά συνέπεια ταχύτερες εφαρμογές

· Υποστήριξη JavaScript

· Υποστήριξη οθονών υψηλών αναλύσεων (720p)

· Γρήγορη εναλλαγή μεταξύ γλωσσών και πληκτρολογιών

· Βελτιωμένη εφαρμογή Android Marke

· Υποστήριξη Bluetooth συσκευών για φωνητικές κλήσεις και σύνδεση με αυτοκίνητα

· Υποστήριξη εγκατάστασης εφαρμογών στην εξωτερική μνήμη

· Επιλογή απενεργοποίησης πρόσβασης σε δεδομένα μέσω δικτύων κινητής τηλεφωνίας

· Υποστήριξη Adobe Flash

Gingerbread 2.3.x

Εκδόθηκε τον Δεκέμβριο του 2010 και βασίστηκε στον πυρήνα 2.6.35 του Linux. Η πρώτη συσκευή που το υιοθέτησε ήταν η Google Nexus S. Κάποιες από τις αλλαγές και βελτιώσεις ήταν:

· Πιο απλό και πιο γρήγορο UI

· Εγγενής υποστήριξη VOIP

· Νέο διαχειριστή λήψεων

· Υποστήριξη πολλαπλών συσκευών κάμερας

· Καλύτερη διαχείριση ενέργειας

· Αλλαγή συστήματος αρχείων από YAFF σε ext4

· Σενεχές garbage collection που βοηθά πολύ τους προγραμματιστές

· Υποστήριξη φωνής και ήχου στο Google Talk

Honeycomb 3.x

Εκδόθηκε τον Φεβρουάριο του 2011 και βασίστηκε στον πυρήνα 2.6.36 του Linux. Το ιδιαίτερο χαρακτηριστικό του ήταν ότι απευθυνόταν αποκλειστικά σε Tablet PC. Η πρώτη συσκευή που το υιοθέτησε ήταν η Motorola Xoom. Επειδή απευθυνόταν αποκλειστικά σε Tablet PC δεν θα αναφέρουμε αλλαγές του λειτουργικού.

Ice Cream Sandwich 4.0.x

Εκδόθηκε τον Οκτώβριο του 2011 και βασίστηκε στον πυρήνα 3.0.1 του Linux. Θεωρητικά είναι συμβατό με οποιαδήποτε συσκευή είχε εγκατεστημένο το Android 2.3.x σαν εργοστασιακό λειτουργικό εκείνη τη στιγμή. Πολλές νέες δυνατότητες έγιναν διαθέσιμες με την έκδοση 4.0.x Μερικές από αυτές είναι:

· Βελτιωμένη μέθοδος διόρθωσης στο εικονικό πληκτρολόγιο

· Βελτιωμένη λειτουργία αντιγραφής και επικόλλησης

· Πολύ καλύτερη αναγνώριση φωνής και δυνατότητα μετατροπής της ομιλίας σε κείμενο σε πραγματικό χρόνο

· Δυνατότητα πρόσβασης στις εφαρμογές μέσω της λειτουργία κλειδώματος

· Δυνατότητα ξεκλειδώματος της συσκευής μέσω λογισμικού αναγνώρισης χαρακτηριστικών προσώπου

· Νέο περιηγητής διαδικτύου με δυνατότητα ταυτόχρονης χρήσης έως και 16 καρτελών

· Νέα γραμματοσειρά στο περιβάλλον, την Roboto

· Δυνατότητα τερματισμού εφαρμογών που καταναλώνουν πολλούς πόρους όταν δεν χρησιμοποιούνται

· Βελτιωμένη εφαρμογή χρήσης της κάμερας

· Ενσωματωμένη διαχείριση φωτογραφιών

· Wi-Fi Direct

· Android Beam

· Επιτάχυνση γραφικών του γραφικού περιβάλλοντος μέσω υλικού

· Εγγραφή βίντεο ποιότητας 1080p

Android 4.1 jelly bean

Η google ανακοίνωσε το android 4.1 στο συνέδριο στις 27 ιουνίου 2012, βασισμένο στον πυρήνα 3.0.31 του linux. Πρωταρχικός στόχος ήταν η βελτίωση της λειτουργικότητας και της απόδοσης της διεπαφής χρήστη. Η πρώτη συσκευή για να τρέξει το καινούργιο λειτουργικό κυκλοφόρησε στις 13 ιουλίου του 2012. Τα χαρακτηριστικά του καινούργιου λειτουργικού είναι:

• Ομαλότερη διεπαφή χρήστη:

• Vsync χρονική στιγμή σε όλα τα σχέδια και animation γίνεται από το πλαίσιο Android, συμπεριλαμβανομένων των rendering εφαρμογής, εκδηλώσεις αφής, τη σύνθεση και οθόνη ανανέωσης

•Triple buffering στον αγωγό γραφικών

• Βελτιωμένη προσβασιμότητα

• Διπλά κατευθυντήριο κείμενο και υποστήριξη διαφορετικών γλωσσών

• Δυνατότητα χειροκίνητης εγκατάστασης από το χρήστη χαρτών πληκτρολογίου

• Επεκτάσιμες ειδοποιήσεις

• Δυνατότητα απενεργοποίησης ειδοποιήσεων σε ειδική βάση εφαρμογής

• Συντομεύσεις και widgets μπορούν αυτόματα να αναδιορθωθούν ή να επαναπροσδιοριστούν για να επιτρέψουν σε νέα στοιχεία να ταιριάξουν στις οικιακές οθόνες

• Bluetooth μεταφορά δεδομένων για το Android Beam

• Ανενεργή υπαγόρευση φωνής

• Tablets με μικρότερες οθόνες χρησιμοποιούν τώρα μια διευρυμένη εκδοχή της διάταξης διασύνδεσης και στην αρχική οθόνη που χρησιμοποιείται από τηλέφωνα.

• Βελτιωμένη φωνητική αναζήτηση

• Βελτιωμένη εφαρμογή κάμερας

• Google Wallet (για το Nexus 7)

• Υψηλής ανάλυσης Google+ επαφή φωτογραφίες

• το Google now εφαρμογή αναζήτησης

• Πολυκάναλο ήχου

• USB audio (για εξωτερικός ήχος DACs)

• Αλυσιδωτό ήχο (επίσης γνωστή ως αναπαραγωγή χωρίς κενά)

• Διαθέσιμο πρόγραμμα Android browser αντικαθίσταται με την έκδοση Android mobile του Google Chrome σε συσκευές με προεγκατεστημένο το Android 4.1

• Δυνατότητα άλλων εκτοξευτών για να προσθέσετε widgets από το συρτάρι app χωρίς να απαιτείται root access

• Διορθώθηκε το bug για το Nexus 7 σχετικά με την αδυναμία να αλλάξετε τον προσανατολισμό της οθόνης σε οποιαδήποτε εφαρμογή

• Κλείδωμα / αρχικής οθόνης με υποστήριξη περιστροφής για το Nexus 7 [92]

• One-finger κίνηση για την επέκταση / σύμπτυξη κοινοποιήσεις [93]

• Νέο "Επιλογή όλων" ανάγλυφο κουμπί στο πληκτρολόγιο Swype

• Διορθώσεις σφαλμάτων και βελτιώσεις απόδοσης

4.2

• "Photo sphere" πανοραμικές φωτογραφίες [97]

• Πληκτρολόγιο με πληκτρολόγηση χειρονομία

• Βελτίωση στο κλείδωμα οθόνης, συμπεριλαμβανομένης της υποστήριξης widget και τη δυνατότητα να σύρετε απευθείας στη φωτογραφική μηχανή

• Ελέγχου ισχύος ειδοποίησης ("Quick Settings")

• "Daydream" screensavers , εμφανίζει πληροφορίες όταν είναι σε αδράνεια

• Πολλαπλοί λογαριασμοί χρηστών (tablets)

• Υποστήριξη για ασύρματη οθόνη (Miracast)

• Βελτιώσεις Προσβασιμότητα: triple-tap για να μεγεθύνετε ολόκληρη την οθόνη, pan και ζουμ με τα δύο δάχτυλα. Έξοδος ομιλίας και gesture mode πλοήγησης για τυφλούς χρήστες

• Νέο app ρολόι με ενσωματωμένο παγκόσμιο ρολόι και χρονόμετρο

• Όλες οι συσκευές χρησιμοποιούν πλέον την ίδια διάταξη interface, προηγουμένως προσαρμοστεί από τα τηλέφωνα για 4.1 για μικρότερες ταμπλέτες (με επίκεντρο τα κουμπιά του λογισμικού, το system bar στο πάνω μέρος της οθόνης, και μια αρχική οθόνη με μια αποβάθρα και επίκεντρο μενού της εφαρμογής), ανεξάρτητα από το μέγεθος της οθόνης

• Αύξηση του αριθμού των κοινοποιήσεων και actionable notificatios για περισσότερες εφαρμογές, επιτρέποντας στους χρήστες να ανταποκριθούν σίγουρα σε κοινοποιήσεις που βρίσκονται στη μπάρα κοινοποιήσεων και χωρίς την εκκίνηση του app απευθείας

• SELinux

• Always-on VPN

• Premium SMS επιβεβαίωσης

• Ομάδα Messaging

• Προστέθηκε Bluetooth χειριστήριο υποστηριζόμενο από HID

• Διόρθωση σφαλμάτων Bluetooth audio streaming σφάλματα ροής [103]

• USB λίστα επιτρεπόμενων debug

• Διορθώσεις σφαλμάτων και βελτιώσεις απόδοσης

Android Kit Kat 4.4

Η νέα έκδοση του λειτουργικού για smartphones της Google, το Android Kit Kat ή αλλιώς έκδοση Android 4.4 είναι εδώ και παρουσιάστηκε επίσημα απο την Google μαζί με το νέο smartphones της, το Google Nexus 5. Η νέα έκδοση είναι ελαφριά και φέρνει ένα νέο "ανάλαφρο" στυλ στο design του interface του αλλά και κάποια νέα χαρακτηριστικά που υπόσχονται να κάνουν την ζωή ενός χρήστη με Android κινητό ακόμα πιο όμορφη.

Με τον όρο "ελαφρύ" εννοούμε πως η νέα έκδοση του Android θα μπορεί να τρέξει και σε "υποδεέστερες" συσκευές που έχουν για παράδειγμα μόνο 512MB μνήμης. Αυτός είναι και

ο στόχος της Google η οποία θέλει να ανοιχθεί σε νέες αγορές και να απορροφήσει "άλλο ένα δισεκατομμύριο ανθρώπων" όπως λέει οι οποίοι έχουν αγοράσει μία πιο φθηνή συσκευή smartphone.

Τα νέα χαρακτηριστικά του Android Kit Kat 4.4 είναι τα εξής :

Εμφάνιση του cover ενός άλμπουμ που ακούει ο χρήστης ενώ έχει κλειδώσει την συσκευή του. Μπορεί κανείς πλέον να αλλάζει τραγούδια απο την οθόνη κλειδώματος. Διαφανής μπάρα εργαλείων στο κάτω μέρος, όπως και διαφανής Notification Area Σε Full Screen εφαρμογές πλέον εξαφανίζονται η Notification Bar αλλά και τα κουμπιά εργαλείων του launcher. Δυνατότητα εύρεσης τηλεφώνων εταιριών απο τον Dialer με βοήθεια απο το Google Maps. Πληκτρολογείτε το όνομα της εταιρίας και το Android ψάχνει να βρεί (μέσω Google Maps) το τηλέφωνο της εταιρίας. Το Hangouts ανανεώνεται, πλέον υποστηρίζει και SMS αλλά και η δυνατότητα Places είναι εκεί με το πάτημα ενός κουμπιού. Πλέον η κάμερα λειτουργεί σαν HDR+ . Εσείς τραβάτε μία φωτογραφία και η κάμερα επιλέγει τον καλύτερο συνδυασμό χρωμάτων και φτιάχνει το καλύτερο δυνατό αποτέλεσμα. Η δυνατότητα αυτή υπάρχει για την ώρα μόνο στο Google Nexus 5. Built In Google Cloud Print και υποστήριξη νέων εκτυπωτών της HP. Google Search σε κάθε Homescreen και φωνητικό Search με την φράση "Ok Google" όπως ακριβώς στα Google Glass. Ταχύτερο Multi-Tasking Emoji παντού Ανανεωμένο Google Now με swipe στην αριστερή πλευρά της οθόνης. Το Android 4.4 KitKat θα διατεθεί πολύ σύντομα για το Nexus 4, Nexus 7, Nexus 10 αλλά και για την Developer έκδοση του Galaxy S4 και HTC One.

1.2 Το περιβάλλον ανάπτυξης του Android

Τα τελευταία 10 χρόνια το Α και το Ω για τους java developers ήταν το eclipse, μόνο κ βασικό εργαλείο για την δημιουργία εφαρμογών Android. Λίγο πριν το τέλος του 2014, το Android Studio έκανε την εμφάνιση του, μετά από 2 χρόνια δουλείας απ' την Google και έβαλε σε 2η μοίρα το eclipse όπως ήταν και αναμενόμενο. Η Google για να μπορέσει να κόψει τους φανατικούς οπαδούς του eclipse ανακοίνωσε την διακοπή του plugin των Android development tools (ADT) στο eclipse, το οποίο ενσωματώνει μια σειρά από εργαλεία Android στο IDE. Καθώς το eclipse σβήνει, το android studio έρχεται να προσφέρει στους προγραμματιστές ένα Gradle build περιβάλλον και ένα βελτιωμένο σχεδιαστικό interface, καλύτερη παρακολούθηση της μνήμης και βελτίωση του editor για string translation.

Οπότε για να ξεκινήσουμε να δημιουργούμε εφαρμογές Android θα πρέπει να εγκαταστήσουμε το eclipse ή το android studio. Όποιο και να θελήσουμε να κατεβάσουμε πρώτο για να δουλέψουμε πρέπει απαραίτητα να έχουμε το java SE Development Kit 7u71 , οπότε το κατεβάζουμε για το ανάλογο σύστημα που διαθέτουμε και το κάνουμε εγκατάσταση από το www.oracle.com.

Java SE Development Kit 7u71

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

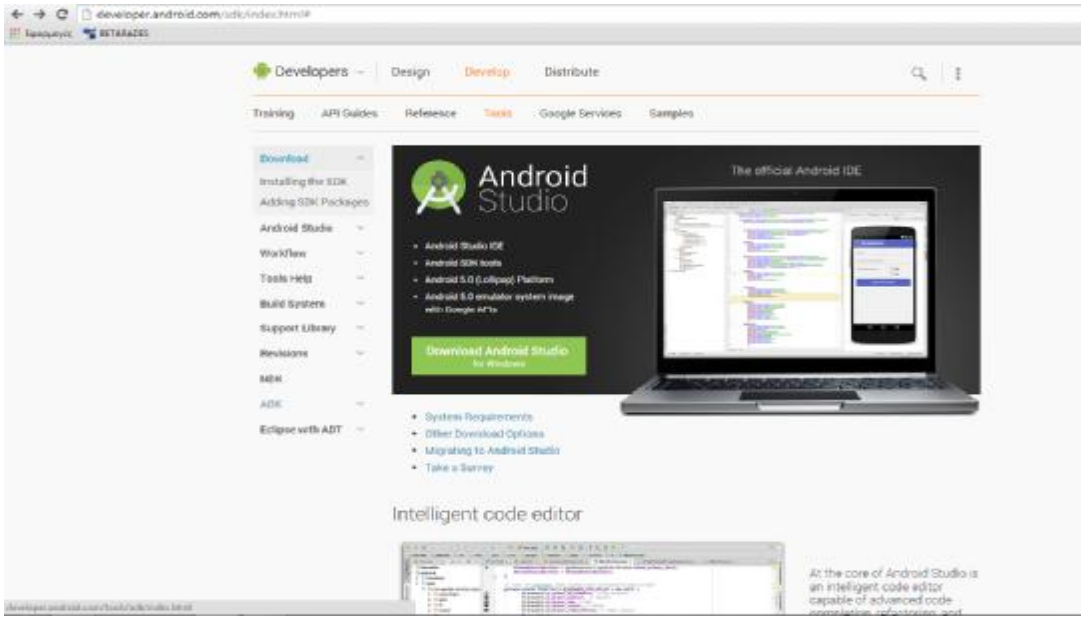
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	119.44 MB	jdk-7u71-linux-i586.rpm
Linux x86	136.76 MB	jdk-7u71-linux-i586.tar.gz
Linux x64	120.81 MB	jdk-7u71-linux-x64.rpm
Linux x64	135.63 MB	jdk-7u71-linux-x64.tar.gz
Mac OS X x64	185.84 MB	jdk-7u71-macosx-x64.dmg
Solaris x86 (SVR4 package)	139.36 MB	jdk-7u71-solaris-i586.tar.Z
Solaris x86	95.48 MB	jdk-7u71-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.68 MB	jdk-7u71-solaris-x64.tar.Z
Solaris x64	16.36 MB	jdk-7u71-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	138.74 MB	jdk-7u71-solaris-sparc.tar.Z
Solaris SPARC	98.62 MB	jdk-7u71-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	jdk-7u71-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.35 MB	jdk-7u71-solaris-sparcv9.tar.gz
Windows x86	127.78 MB	jdk-7u71-windows-i586.exe
Windows x64	129.52 MB	jdk-7u71-windows-x64.exe

Εικόνα 1-1: Java SE Development Kit 7u71

Κατέβασμα και εγκατάσταση Android Studio

Εισερχόμαστε στην τοποθεσία developer.android.com και κάνουμε download.



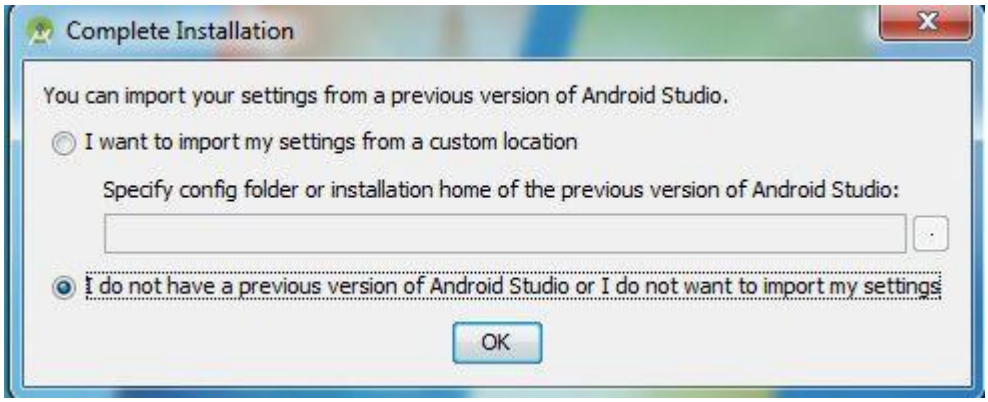
Εικόνα 1-2: developer.android.com/sdk/index.html#

Αφού κατέβει το αρχείο μας ξεκινάμε την εγκατάσταση.

Εικόνα 1-3: **Το παράθυρο διαλόγου του** Android Studio **που παραπέμπει στην εγκατάσταση του**

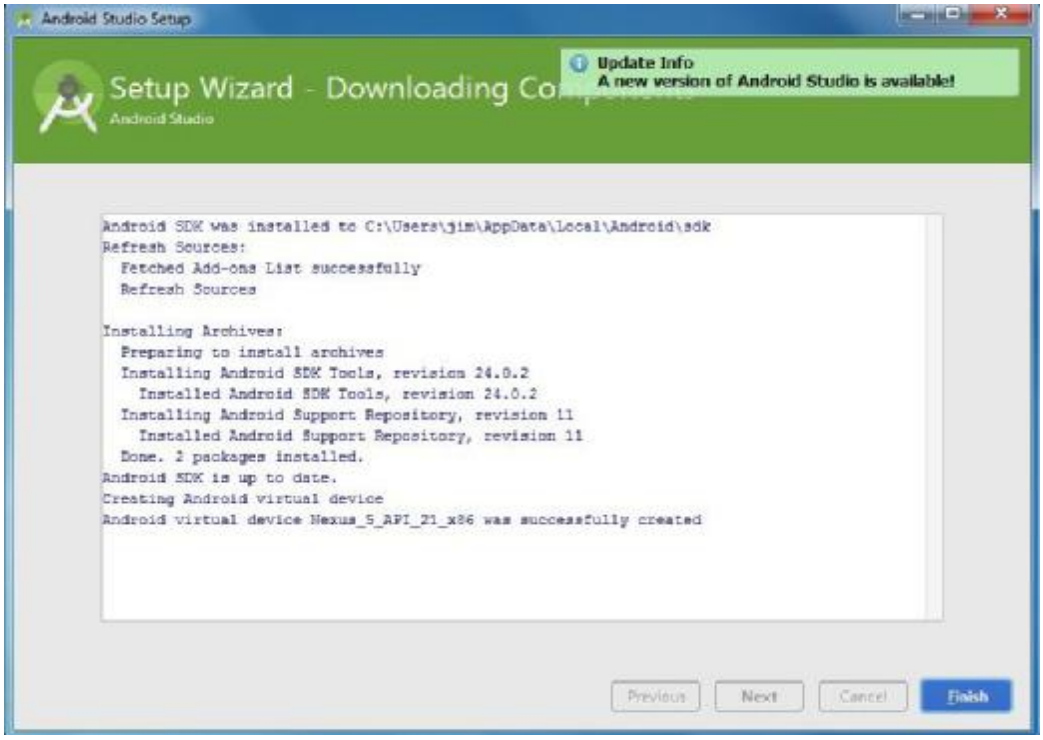
Μόλις τελειώσει η εγκατάσταση θα μας ζητηθεί να εισάγουμε δικές μας ρυθμίσεις.

Εικόνα 1-4



Εικόνα 1-5: **Εισαγωγή δικών μας ρυθμίσεων**

Εμείς δεν έχουμε οπότε πατάμε το δεύτερο και συνεχίζουμε. Κάνει install αυτόματα από το φάκελο του SDK τις λίστες, κατεβάζει τα SDK Tools και δημιουργεί το virtual device. Πατάμε finish και συνεχίζουμε.

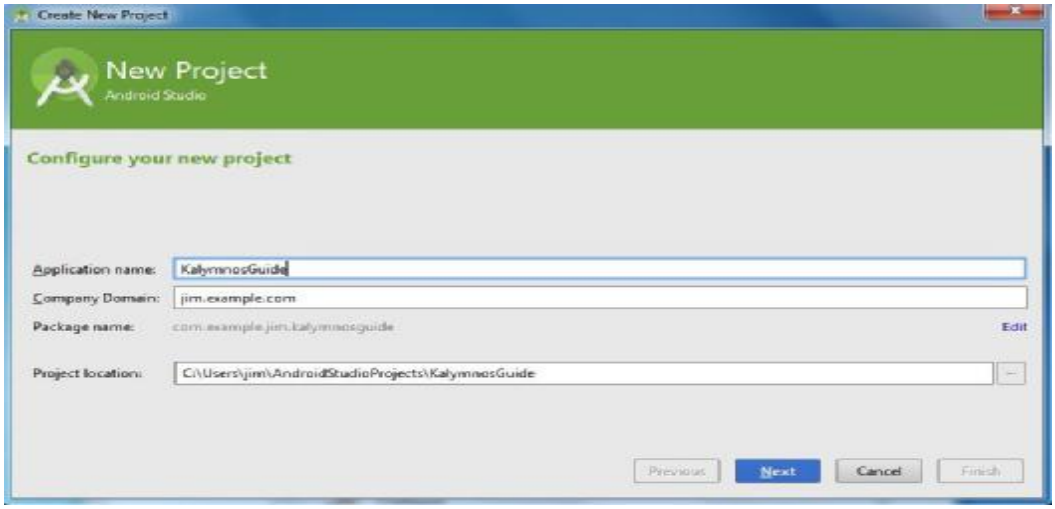


Εικόνα 1-6: **Αυτόματη εγκατάσταση σημαντικών στοιχείων όπως το** SDK

Δημιουργούμε ένα νέο project.

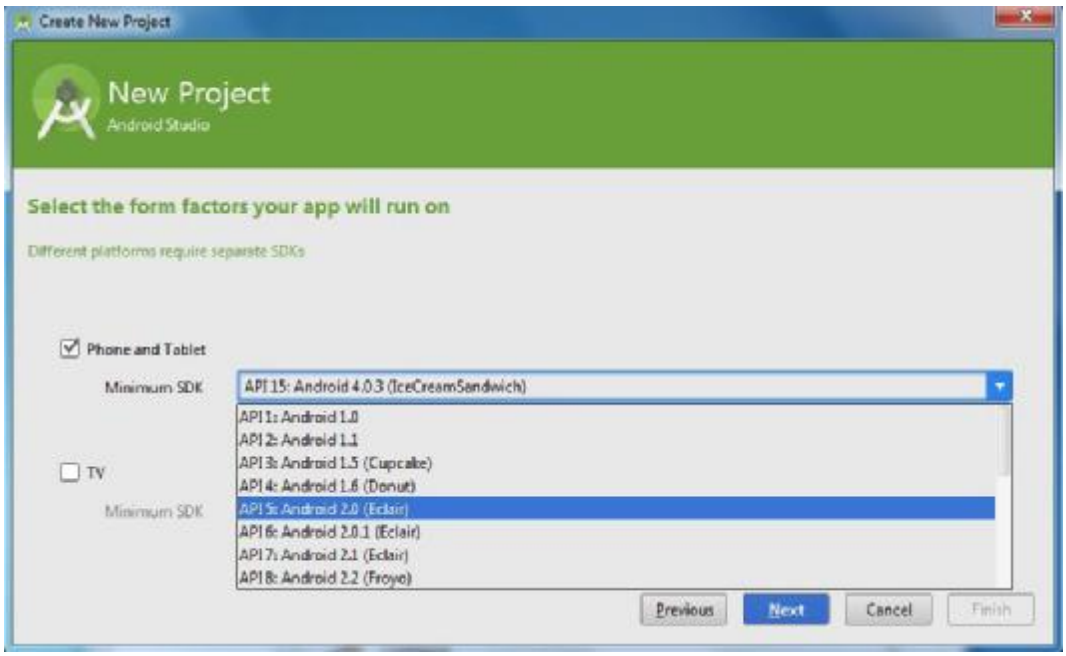
Εικόνα 1-7: **Δημιουργία νέου πρότζεκτ**

Στη συνέχεια δίνουμε το όνομα που θέλουμε να έχει το application πχ KalymnosGuide όπως έχουμε ονομάσει το δικό μας, το company domain και το project location.



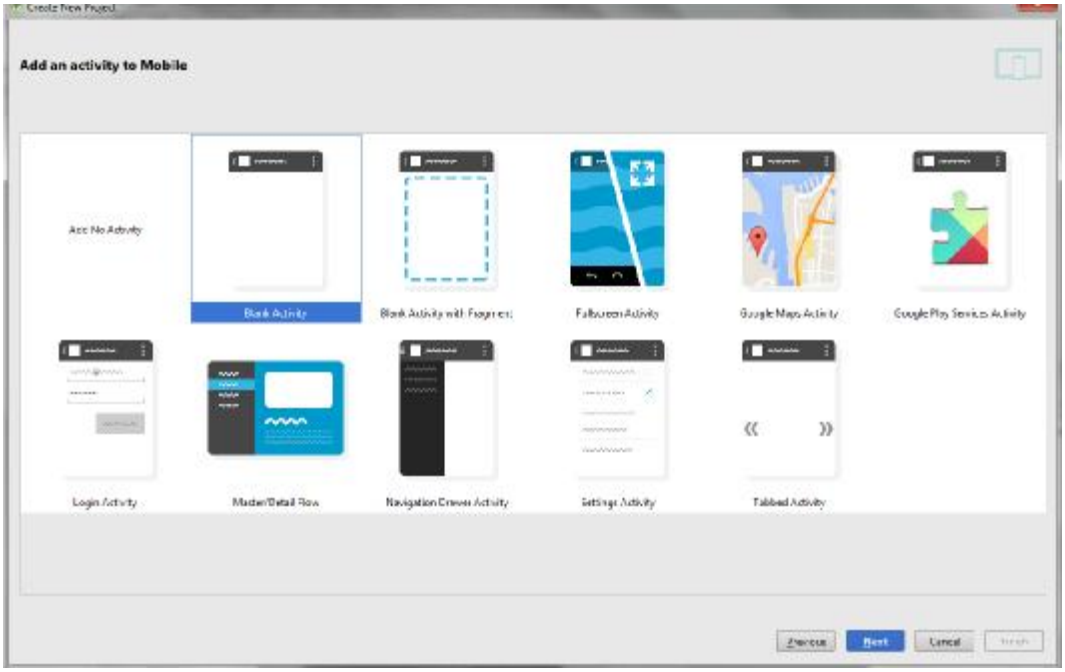
Εικόνα 1-8: **Ονομασία της εφαρμογής**

Μετέπειτα ρυθμίζουμε από ποια έκδοση και μετά θέλουμε να είναι διαθέσιμη η εφαρμογή μας. Ας βάλουμε απ' το API 5 και πάνω για να πιάσουμε όλα τα διαθέσιμα κινητά του αγοραστικού μας κοινού.



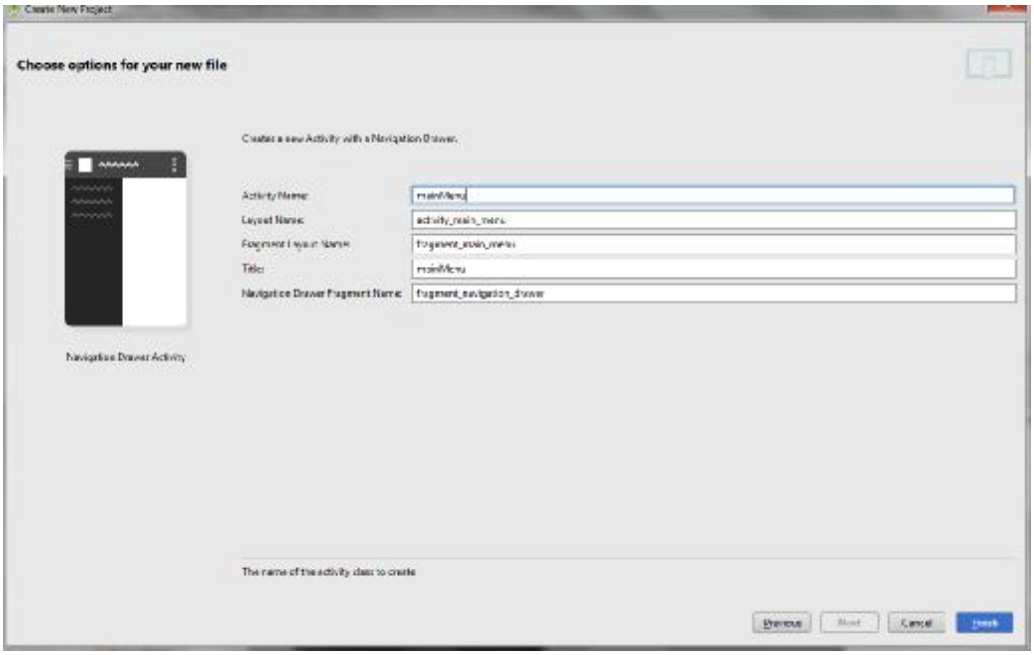
Εικόνα 1-9: **Ορισμός του ελάχιστου** Android SDK **που θα απευθύνεται η εφαρμογή**

Μας δίνει επιπλέον τη δυνατότητα να δημιουργήσουμε και την Activity που θέλουμε για να ξεκινήσουμε, ανάμεσα σε 10 πρότυπες μαζί με τον ανάλογο κώδικα. Ας επιλέξουμε την Navigation Drawer Activity και ξεκινάμε να δώσουμε το όνομα που θέλουμε εμείς.



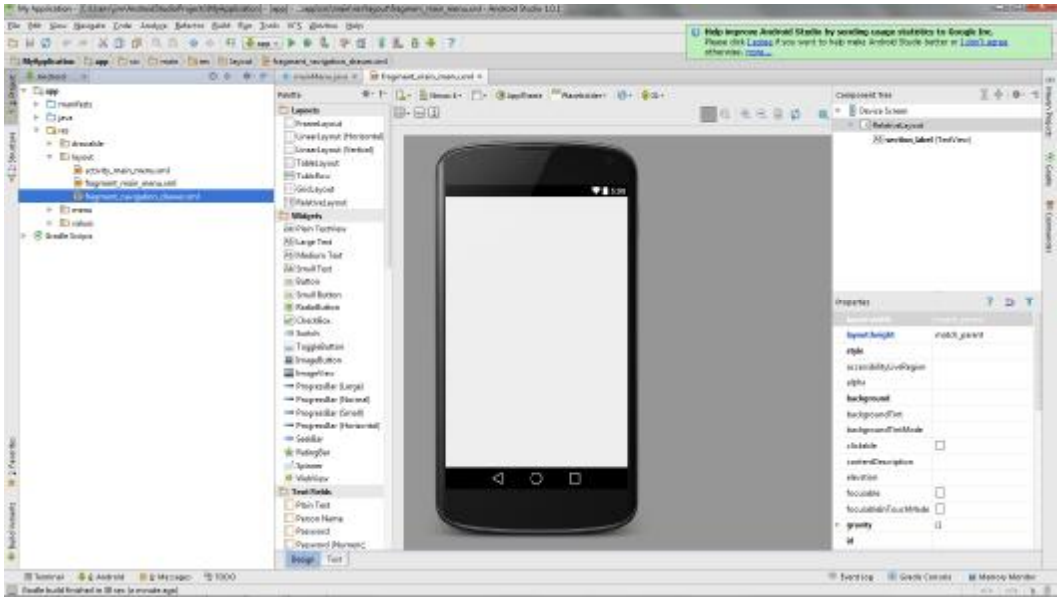
Εικόνα 1-10: **Προσθήκη δραστηριότητας**

Βάζουμε Activity Name mainMenu και πατάμε finish.



Εικόνα 1-11: **Λοιπές ρυθμίσεις της επιλεγμένης δραστηριότητας**

Μας ανοίγει το Android Studio έχοντας δημιουργήσει την Activity που του έχουμε ζητήσει, μαζί με τον κώδικα της. Τώρα μπορούμε να κάνουμε εμείς τις παρεμβάσεις που θέλουμε πάνω στην βάση που μας έχει δώσει.



Εικόνα 1-12: **Ο χώρος εργασίας του** Android Studio

Κατέβασμα και εγκατάσταση Eclipse IDE

Τώρα όσον αφορά το eclipse θα πρέπει να ξέρετε ότι όλα τα απαραίτητα εργαλεία που θα χρειαστούν είναι διαθέσιμα για κατέβασμα δωρεάν στο διαδίκτυο και γι' αυτό. Παρακάτω είναι όλα όσα θα χρειαστούν:

- Java JDK5 ή JDK6
- Android SDK
- Eclipse IDE for Java Developers
- Android Development Tools (ADT) Eclipse Plugin

Βήμα 1ο : **Εγκατάσταση** Java Development Kit (JDK)

Μπορείτε να κατεβάσετε τη τελευταία έκδοση της Java SDK από το website της Oracle στη καρτέλα "Java SE Downloads". Ακολουθήστε τις οδηγίες που θα βρείτε μαζί μόλις κατέβουν τα αρχεία και ολοκληρώστε την εγκατάσταση. Στο τέλος ορίστε τις μεταβλητές "PATH" και "JAVA_HOME" ώστε να αναφέρονται στη τοποθεσία που περιέχεται η Java και η Javac. Συνήθως είναι σε αυτή τη διεύθυνση: java_install_dir/bin και java_install_dir αντίστοιχα.

Εάν είστε σε Windows και έχετε εγκαταστήσει το JDK στη διαδρομή C:\jdk1.6.0_15, θα πρέπει να ορίσετε τη παρακάτω εντολή στο αρχείο C:\autoexec.bat

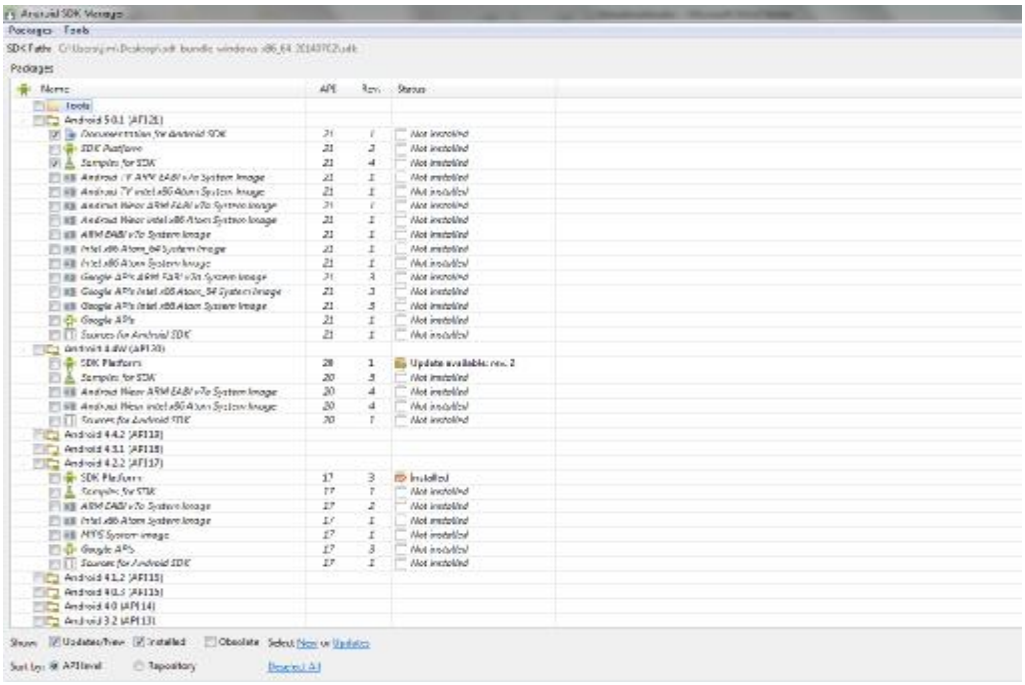
```
set PATH=C:\jdk1.6.0_15\bin;%PATH%

set JAVA_HOME=C:\jdk1.6.0_15
```

Εναλλακτικά, μπορείτε να κάνετε δεξί κλικ στον "Υπολογιστή μου", επιλέξτε "Ιδιότητες" - "Για προχωρημένους" - "Μεταβλητές Περιβάλλοντος". Εκεί βρείτε τη μεταβλητή "Path" και αντικαταστήστε τη διαδρομή στο πεδίο "Τιμή".

Βήμα 2ο : Εγκατάσταση Android SDK

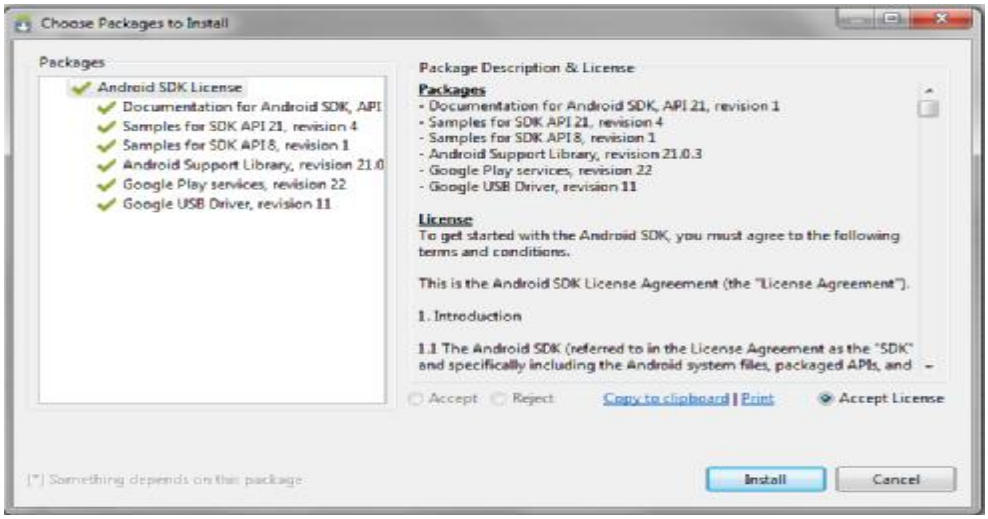
Μπορείτε να κατεβάσετε τη τελευταία έκδοση του Android SDK από το επίσημο site του Android εδώ: Android SDK Downloads. Εάν θέλετε να εγκαταστήσετε το SDK σε υπολογιστή με Windows, τότε θα πρέπει να βρείτε το αρχείο με όνομα installer_rXX-windows.exe και να το εγκαταστήσετε αφού το κατεβάσετε, και στη συνέχεια ακολουθήστε τις οδηγίες για να το εγκαταστήσετε επιτυχώς.

Ας τρέξουμε λοιπόν για πρώτη φορά το Android SDK Manager ανοίγοντας το από τη διαδρομή: Όλα τα προγράμματα > Android SDK Tools > SDK Manager. Θα εμφανιστεί το παρακάτω πλαίσιο:



Εικόνα 1-13: **Πρόσοψη του** Android SDK Manager

Αφού ανοίξαμε το SDK manager, είναι ώρα να εγκαταστήσουμε τα απαραίτητα πακέτα. Προεπιλεγμένα θα υπάρχουν 7 πακέτα προς εγκατάσταση, αλλά καλό είναι να από-επιλέξετε το Documentation for Android SDK και το Samples for SDK packages για να μειωθεί ο χρόνος εγκατάστασης. Έπειτα πατήστε το Install Packages button to proceed, και έπειτα θα εμφανιστεί το παρακάτω πλαίσιο:



Εικόνα 1-14: **Κλικ** install **για εγκατάσταση των επιλεγμένων πακέτων**

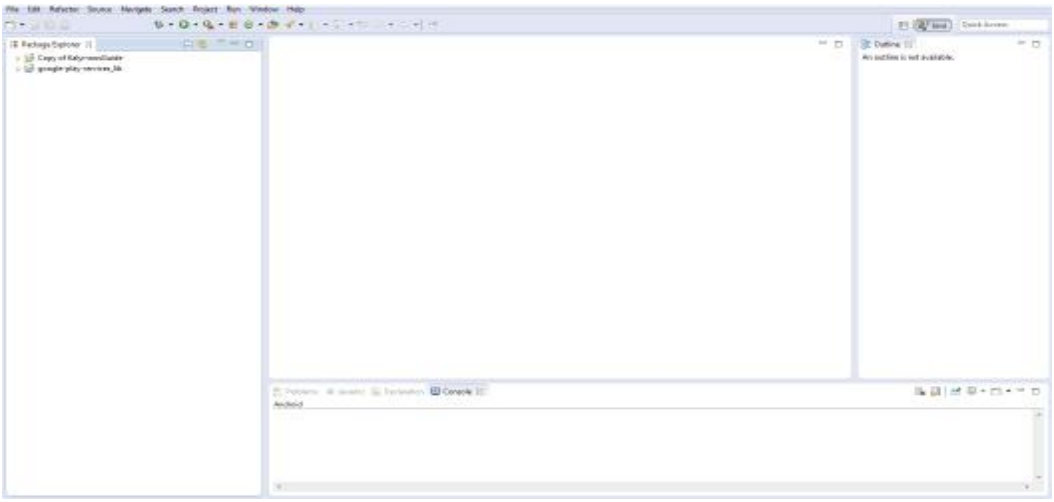
Στη συνέχεια επιλέξτε το Accept Licence και μετά το κουμπί Install για να ξεκινήσει η εγκατάσταση. Η ολοκλήρωση της εγκατάστασης θα πάρει αρκετή ώρα, οπότε περιμένετε μέχρι να δείτε ότι ολοκληρώθηκε. Μόλις γίνει αυτό, κλείστε το SDK manager πατώντας το "X" στο πάνω μέρος του παραθύρου.

Βήμα 3ο : **Εγκατάσταση** Eclipse IDE

Για να εγκαταστήσετε το Eclipse, κατεβάστε το από εδώ http://www.eclipse.org/downloads/. Όταν ολοκληρωθεί η εγκατάσταση, αποσυμπιέστε τα αρχεία σε μια τοποθεσία που σας βολεύει, για παράδειγμα: C:\eclipse στα windows, και τέλος ορίστε τη Path variable αντίστοιχα με τη διεύθυνση που αποσυμπιέσατε τα αρχεία.

Το Eclipse θα τρέξει σε Windows κάνοντας διπλό κλικ στο αρχείο eclipse.exe που βρίσκεται στα αρχεία που αποσυμπιέσατε.

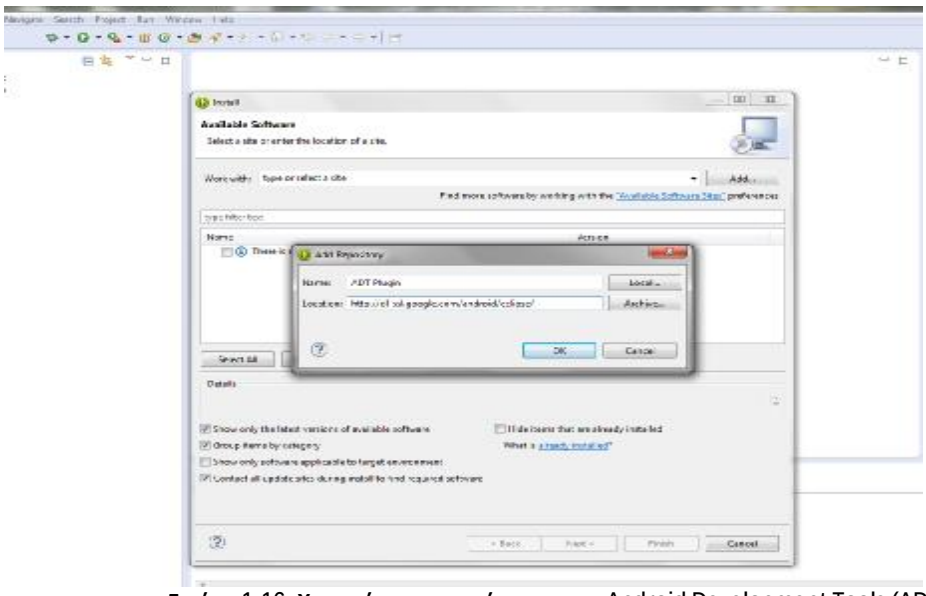
Στη συνέχεια, θα εμφανιστεί το παρακάτω πλαίσιο:



Εικόνα 1-15: **Ο χώρος εργασίας του** Eclipse IDE

Βήμα 4**ο** : **Εγκατάσταση** Android Development Tools (ADT)

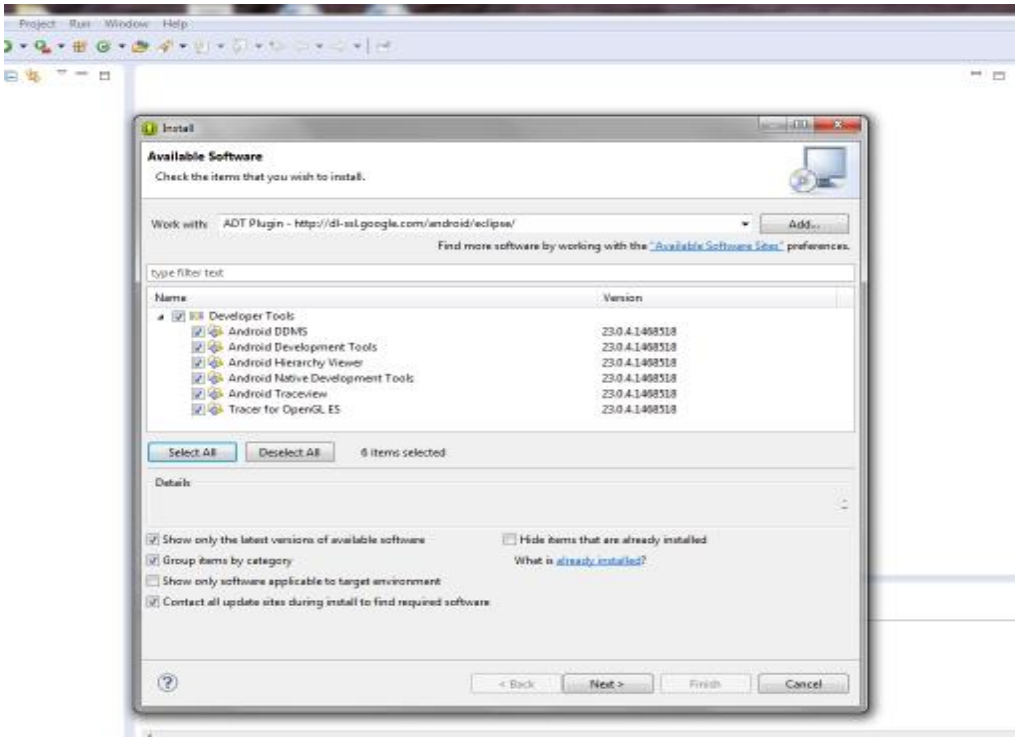
Σε αυτό το βήμα θα εισάγουμε το Android Development Tool plugin στο Eclipse. Ανοίγουμε λοιπόν το Eclipse επιλέγουμε: Help > Install New Software και θα εμφανιστεί το παρακάτω πλαίσιο:



Εικόνα 1-16: **Χειροκίνητη εγκατάσταση του** Android Development Tools (ADT)

Πιέστε το κουμπί Add και προσθέστε στο πεδίο Name: ADT Plugin και στο πεδίο Location:
https://dl-ssl.google.com/android/eclipse/

Πατήστε OK για να προστεθεί η τοποθεσία. Το Eclipse θα ψάξει για τα διαθέσιμα plugins και θα τα εμφανίσει στη λίστα που βλέπετε παρακάτω:

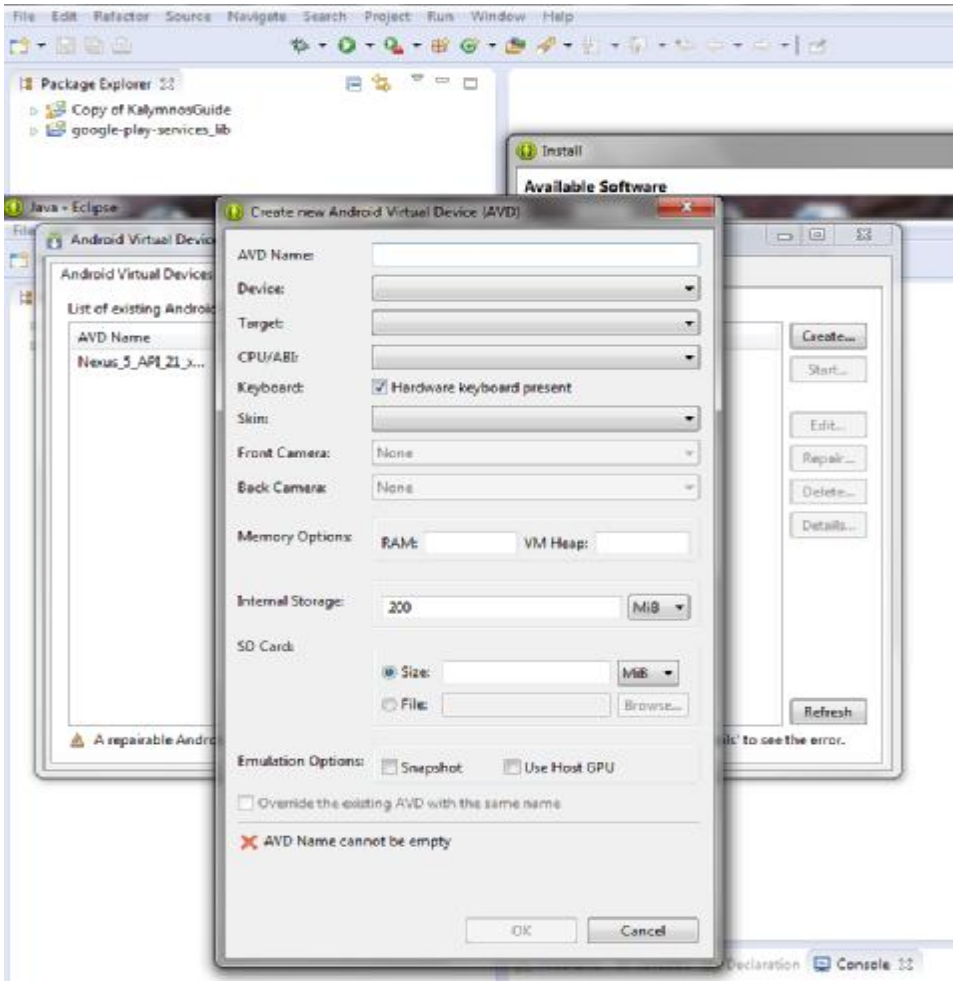


Εικόνα 1-17: **Εμφάνιση των διαθέσιμων** plugins

Επιλέξτε το κουμπί Select All για να επιλεγούν όλα και στη συνέχεια πατήστε το κουμπί Next το οποίο θα σας κατευθύνει για να ολοκληρωθεί η εγκατάσταση του Android Development Tools.

Βήμα 5ο : **Δημιουργία Εικονικής Συσκευής**

Για να ελέγξουμε την Android εφαρμογή που φτιάχνουμε στον υπολογιστή, χρειαζόμαστε μια εικονική συσκευή. Έτσι, πριν αρχίσουμε να γράφουμε κώδικα, ας δημιουργήσουμε μια συσκευή. Ανοίξτε το Android AVD Manager από το μενού του Eclipse: Window > AVD Manager. Πατήστε το κουμπί New για να δημιουργηθεί μια νέα εικονική συσκευή και εισάγετε τις παρακάτω τιμές προτού πατήσετε το κουμπί Create AVD.



Εικόνα 1-18: **Δημιουργία εικονικής συσκευής στο** Eclipse

Εάν η δημιουργία της εικονικής συσκευής δημιουργήθηκε με επιτυχία, τότε όλα είναι εντάξει και είστε έτοιμοι να δημιουργήσετε Android εφαρμογές. Καλό θα είναι εδώ να κάνετε μια επανεκκίνηση του υπολογιστή σας προτού αρχίσετε τη δημιουργία μιας εφαρμογής.

1.3 **Η ανατομία μίας εφαρμογής** Android

Δόμηση μίας εφαρμογής Android:
Η πλατφόρμα Android, όπως και κάθε άλλη πλατφόρμα, χρησιμοποιεί διαφορετική ορολογία για να περιγράψει τα συστατικά της εφαρμογής που παράγει. Όσον αφορά την δημοφιλή πλατφόρμα της Google, οι τρεις σηματικότερες κλάσεις είναι οι Context, Activity και Intent.

Ναι, σίγουρα, υπάρχουν και πολλά περισσότερα συστατικά, από τα οποία οι προγραμματιστές έχουν τη δυνατότητα να υλοποιήσουν πιο εξεζητημένες

λειτουργίες στις εφαρμογές τους, ωστόσο αυτές οι τρεις κλάσεις αποτελούν τους δομικούς λίθους για κάθε εφαρμογή στην πλατφόρμα του Android.

Context
Αυτή η κλάση αντιπροσωπεύει το περιβάλλον της εφαρμογής, όπου είναι η κεντρική θέση για όλη την λειτουργικότητα ανωτάτου επιπέδου της.

Το περιββάλλον της εφαρμογής μπορεί να χρησιμοποιηθεί για την προσπέλαση ρυθμίσεων και πόρων, στοιχεία που είναι κοινόχρηστα ανάμεσα σε στιγμιότυπα (αντικείμενα) πολαπλών δραστηριοτήτων.

Ακόμη μπορεί να ανακτηθεί με την τρέχουσα διεργασία, με τη βοήθεια της μεθόδου getApplicationContext( ), ως εξής: Context perivallon = getApplicationContext( );

Εφόσον ανακτήσουμε ένα έγκυρο περιβάλλον εφαρμογής τότε μπορούμε να το χρησιμοποιήσουμε για να προσπελάσουμε χαρακτηριστικά και υπηρεσίες επιπέδου εφαρμογής, όπως...

- Ανάκτηση πόρων Εφαρμογής (Context.getResources( ))
- Προσπέλαση Προτιμήσεων Εφαρμογής (Context.getSharedPreferences( ))
- Εκκίνηση στιγμιοτύπων (αντικειμένων) Activity
- Διαχείριση ιδιωτικών αρχείων, καταλόγων και βάσεων δεδομένων εφαρμογής
- Επιθεώρηση και επιβολλή αδειών εφαρμογής

1.3.1.1 Ανάκτηση πόρων εφαρμογής

Κάνοντας χρήση της μεθόδου getResources( ) του περιβάλλοντος εφαρμογής, μπορούμε να ανακτήσουμε πόρους της εφαρμογής.

Ένας από τους πιο εύκολους τρόπους για να ανακτήσουμε έναν πόρο είναι με τη χρήση του μοναδικού αναγνωριστικού πόρου, όπως ορίζεται στην αυτόματα παραγόμενη κλάση R.java. Ακολουθεί ένα παράδειγμα στο οποίο ανακτάται ένα String από τους πόρους της εφαρμογής, με βάση το αναγνωριστικό (ID) του πόρου.

Αφού έχουμε ορίσει ένα String <string name="app_name">Experience Kalymnos</string> σε ένα αρχείο xml του φάκελου /res της εφαρμογής μπορούμε να εργασθούμε ως εξής για να ανακτήσουμε τον πόρο, ας υποθέσουμε μέσα στο περιβάλλον μίας δραστηριότητας:

```
String appName = this.getResources( ).getString( R.string.app_name);
```

Για την ώρα θα μείνουμε ως εδώ για τους πόρους της εφαρμογής, μιας και θα μιλήσουμε για αυτούς εκτενέστερα παρακάτω.

*1.3.1.2 **Ανάκτηση προτιμήσεων εφαρμογής***

Οι προτιμήσεις κοινής χρήσης μίας εφαρμογής μπορούν να ανακτηθούν μέσω της μεθόδου getSharedPreferences( ) του περιβάλλοντος εφαρμογής.

Η συγκεκριμένη μέθοδος επιστρέφει ένα αντικείμενο της κλάσης SharedPreferences. Η κλάση SharedPreferences αποθηκεύει, με ένα σύστημα key-value, απλά δεδομένα εφαρμογής. Για την ακρίβεια ένα στιγμιότυπο της εν λόγω κλάσης αποθηκεύει μόνο τιμές int, boolean, long, float και String.

Θα δανειστούμε ένα παράδειγμα από τη δική μας εφαρμογή, στο οποίο αποθηκεύονται τιμές με χρήση των Προτιμήσεων της εφαρμογής. Στην Kalymnos Guide υπάρχει η δυνατότητα χαρακτηρισμού ενός αντικειμένου, όπως ένα εστιατόρειο, ως αγαπημένο. Έτσι για να κρατήσουμε τη συγκεκριμένη πληροφορία και να μπορούμε να την ανακτήσουμε από οπουδήποτε αλλού μέσα στην εφαρμογή θα εργαστούμε ως εξής:

Ο παρακάτω κώδικας δημιουργήσει ένα αρχείο προτιμήσεων κοινής χρήσης, με όνομα Favorites και θα αποθηκεύει την προτίμηση μας. Στην περίπτωση μας θα χρειαστούμε μόνο μία προτίμηση, αλλά εάν το επιθυμούμε μπορούμε να αποθηκεύσουμε περισσότερες.

*1.3.1.3 **Ανάκτηση προτιμήσεων εφαρμογής και δημιουργία νέου αρχείου χρησιμοποιώντας το περιβάλλον εφαρμογής***

```
SharedPreferences favorites = this.getSharedPreferences("Favorites", MODE_PRIVATE);
```

Η παράμετρος MODE_PRIVATE της μεθόδου getSharedPreferences( ) απαγορεύει την κοινή χρήση του αρχείου των προτιμήσεων εκτός του περιβάλλοντος της εφαρμογής. Θα μπορούσαμε για παράδειγμα να χρησιμοποιήσουμε την παράμετρο Context.MODE_WORLD_READABLE ή Context.MODE_WORLD_WRITABLE στην περίπτωση που θα θέλαμε και άλλες εφαρμογές να έχουν πρόσβαση στο αρχείο, ωστόσο μία τέτοια ενέργεια αποθαρρύνεται μιας και μπορεί να δημιουργήσει "τρύπες" στην εφαρμογή.

Στη συνέχεια "ανοίγουμε" τον Editor (interface της κλάσης SharedPreferences, SharedPreferences.Editor) φορτώνουμε τις τιμές που θέλουμε και επιβεβαιώνουμε την αποθήκευση χρησιμοποιώντας τη μέθοδο commit( ) του Editor.

```
SharedPreferences.Editor editor = favorites.edit( );

editor.putBoolean("Key", true);

editor.commit( );
```

Σε αυτή τη φάση έχουμε φορτώσει στο αρχείο προτιμήσεων “Favorites” την τιμή true με κλειδί αναγνώρισης το String “Key”. Τώρα έχουμε τη δυνατότητα να ανακτήσουμε την συγκεκριμένη πληροφορία οπουδήποτε στην εφαρμογή μας, χρησιμοποιώντας το περιβάλλον εφαρμογής όπως παρακάτω:

```
SharedPreferences favorites = getSharedPreferences(“Favorites, Context.MODE_PRIVATE”);

boolean favoriteResult = favorites.getBoolean(“Key”, true);
```

Activity

Μία τυπική εφαρμογή Android δεν είναι τίποτα άλλο από μία συλλογή εργασιών, οι οποίες καλούνται δραστηριότητες (activities). Η κλάση Activity είναι κεντρική για κάθε εφαρμογή της συγκεκριμένης πλατφόρμας.

Συνηθέστερα, θα ορίζουμε και θα υλοποιούμε μία δραστηριότητα για κάθε οθόνη συστήματος στην εφαρμογή μας. Κάθε μία από αυτές έχει ένα μοναδικό σκοπό και μία μοναδική διεπαφή χρήστη μέσα σε μία εφαρμογή. Δηλαδή μία activity πρέπει να έχει το δικό της αντίστοιχο αρχείο διάταξης (.xml), το οποίο είναι αποθηκευμένο μέσα στους πόρους της εφαρμογής.

Τις περισσότερες φορές χρησιμοποιούμε ένα αρχείο διάταξης για κάθε μία δραστηριότητα, ωστόσο κανείς δεν μας εμποδίζει να δημιουργήσουμε μία διάταξη για δύο ή περισσότερες δραστηριότητες. Πάντως, σε αυτήν την περίπτωση, οι πόροι του αρχείου διάταξης, παραδείγματος χάρη εικόνες ή κείμενο, της εκάστοτε δραστηριότητας θα τίθενται κατά το χρόνο εκτέλεσης.

*1.3.1.4 **Εκκίνηση δραστηριότητας***

Στο Android ο προγραμματιστής μπορεί να εκκινήσει με αρκετούς τρόπους μία δραστηριότητα, συμπεριλαμβανομένων των παρακάτω:

- Καθορισμός μίας δραστηριότητας εκκίνησης μέσα στο manifest.xml
- Εκκίνηση μίας activity μέσω του περιβάλλοντος εφαρμογής
- Εκκίνηση θυγατρικής δραστηριότητας από μία πατρική της με σκοπό τη λήψη ενός αποτελέσματος

1.3.1.4.1 Εκκίνηση δραστηριότητας μέσα στο αρχείο manifest

Μία δραστηριότητα μπορεί να ξεκινήσει μέσω του αρχείου manifest του Android. Στην πραγματικότητα είναι υποχρεωτικό για κάθε εφαρμογή να έχει καθορίσει μία προεπιλεγμένη δραστηριότητα εκκίνησης μέσα στο manifest.

Ρίχνοντας μία ματιά στο manifest της δικής μας εφαρμογής θα διαπιστώσουμε πως η δραστηριότητα MainMenu έχει καθοριστεί ως η προεπιλεγμένη δραστηριότητα.

Το παραπάνω μεταφράζεται ως εξής: Όταν ο χρήστης τρέξει την εφαρμογή η πρώτη δραστηριότητα που θα εμφανιστεί στο προσκήνιο θα είναι η MainMenu. Το γεγονός αυτό το πετύχαμε χρησιμοποιώντας ένα φίλτρο Intent (<intent-filter>) στο manifest, καθορίζοντας με αυτόν τον τρόπο τη δραστηριότητα MainMenu ως το κύριο σημείο εισόδου της εφαρμογής.

Κάποιος που γνωρίζει από Java θα παρατηρήσει ομοιότητες με την μέθοδο εκκίνησης main (public static void main(String args[])).

Θα πρέπει να σημειωθεί πως και άλλες κλάσεις Activity μπορούν να καθοριστούν στο manifest, ώστε να εκκινούν υπό συγκεκριμένες συνθήκες. Ωστόσο πρόκειται για δευτερεύοντα σημεία εισόδου και δεν θα μας απασχολήσουν σε αυτή την εργασία.

1.3.1.4.2 Εκκίνηση δραστηριότητας μέσω του περιβάλοντος εφαρμογής

Τις περισσότερες φορές εκκινούμε μία δραστηριότητα χρησιμοποιώντας την μέθοδο startActivity( ) του περιβάλλοντος εφαρμογής. Η μέθοδος αυτή παίρνει μία παράμετρο Intent. Θα αναφερθούμε αναλυτικά στις προθέσεις (Intents) παρακάτω, μιας και αποτελούν το τρίτο μέρος των δομικών λίθων μίας εφαρμογής Android. Για την ώρα όμως θα εξετάσουμε μία κλήση της μεθόδου startActivity( ).

Ας υποθέσουμε ότι βρισκόμαστε “μέσα” στην δραστηριότητα MainMenu της εφαρμογής μας, δηλαδή επεξεργαζόμαστε το αρχείο MainMenu.java. Εφόσον ζητάμε να “προχωρήσουμε” σε μία άλλη δραστηριότητα, ας πούμε την SubMainMenu, η πρώτη μας ενέργεια θα πρέπει να είναι η δημιουργία μίας πρόθεσης η οποία θα καταδεικνύει ως στόχο τη δραστηριότητα που θέλουμε να ξεκινήσουμε. Πιο απλά, αν βρισκόμαστε στη δραστηριότητα MainMenu και θέλουμε να “φθάσουμε” στην SubMainMenu, εργαζόμαστε ως εξής:

1.3.1.4.2.1 Δημιουργία της Πρόθεσης

```
Intent goToSubMainMenu = new Intent(getApplicationContext(), SubMainMenu.class);
```

Στον παραπάνω κώδικα δημιουργήσαμε την πρόθεση goToSubMainMenu η οποία παίρνει ως πρώτη παράμετρο το περιβάλλον της εφαρμογής και ως δεύτερη ένα αντικείμενο Class.

Το αντικείμενο Class αντιπροσωπεύει την δραστηριότητα στην οποία θέλουμε να φθάσουμε, στην περίπτωση μας τη SubMainMenu.

Εδώ θα πρέπει να σημειώσουμε πως η κλάση SubMainMenu ορίζεται κάπου στο πακέτο της εφαρμογής και καταγράφεται σαν δραστηριότητα μέσα στο αρχείο manifest του Android. Σε διαφορετική περίπτωση το λειτουργικό δεν θα την αναγνωρίζει ως δραστηριότητα κατά το χρόνο εκτέλεσης. Μιας λοιπόν και μας δόθηκε η ευκαιρία στο σημείο αυτό, αμέσως μετά τη δημιουργία μίας

δραστηριότητας θα πρέπει αυτή να δηλώνεται στο manifest ως δραστηριότητα της εφαρμογής.

1.3.1.4.2.2 Εκκίνηση της Δραστηριότητας

Τώρα που το Android γνωρίζει την πρόθεση μας, η οποία είναι η εκκίνηση της SubMainMenu, μένει μόνο να του δώσουμε την εντολή για να πράξει αναλόγως.

```
startActivity(goToSubMainMenu);
```

1.3.1.4.2.3 Εκκίνηση Δραστηριότητας για Λήψη Αποτελέσματος

Θα υπάρξουν φορές όπου θα πρέπει να εκκινήσουμε μία εφαρμογή, να την αναγκάσουμε να καθορίσει κάτι, και στη συνέχεια να επιστρέψει χρήσιμες πληροφορίες στην καλούσα δραστηριότητα. Στην KalymnosGuide δεν συναντάμε μία τέτοια περίπτωση, ωστόσο καλό θα ήταν να αναφερθεί ένας ακόμη τρόπος εκκίνησης δραστηριότητας.

Στην περίπτωση λοιπόν που μία δραστηριότητα χρειάζεται ένα αποτέλεσμα από μία άλλη δραστηριότητα, η πρώτη θα πρέπει να κάνει χρήση της μεθόδου της κλάσης Activity startActivityForResult(Intent intent, int requestCode).

Το αποτέλεσμα που ζητάται θα επιστραφεί στην παράμετρο Intent της μεθόδου onActivityResult(int requestCode, int resultCode, Intent data).

1.3.1.5 Ο Κύκλος Ζωής μίας Δραστηριότητας

Κατά τη διάρκεια πλοήγησης του χρήστη σε διάφορες εφαρμογές της πλατφόρμας του Android, τα στιγμιότυπα των δραστηριοτήτων παιρνούν από διάφορα μεταβατικά στάδια του κύκλου της ζωής τους.

Παραδείγματος χάρη υποθέτουμε ότι ένας χρήστης πλοηγείται μέσα στην δική μας εφαρμογή, την KalymnosGuide. Η δραστηριότητα που εμφανίζεται πρώτη, από τη στιγμή που θα “κλικάρει” το εικονίδιο της εφαρμογής ο χρήστης, είναι η MainMenu.

Ωστόσο η πλοήγηση του χρήστη μπορεί να διακοπεί από μία εισερχόμενη κλήση, οπότε η εφαρμογή προς το παρόν διακόπεται και μεταφέρεται στο παρασκήνιο.

Οι εφαρμογές διακόπτονται όταν αυτών προηγούνται συμβάντα υψηλότερης προτεραιότητας, όπως προηγουμένως αυτό της τηλεφωνικής κλήσης. Μόνο μία ενεργή εφαρμογή μπορεί να υπάρχει ανά πάσα στιγμή, και για να ακριβολογήσουμε, μόνο μία δραστηριότητα εφαρμογής μπορεί να βρίσκεται στο προσκήνιο ανά πάσα στιγμή.

Η διαχείριση της κατάστασης, της μνήμης, των πόρων και των δεδομένων μίας εφαρμογής είναι ευθύνη της ίδιας. Αν, ας πούμε, σε κάποια φάση η μνήμη της συσκευής κυμανθεί σε χαμηλά επίπεδα, το Android έχει τη δυνατότητα να τερματίσει ανά πάσα στιγμή μία δραστηριότητα (να καταστρέψει το αντικείμενο

της) που έχει διακοπεί προσωρινά, έχει τερματιστεί ή έχει καταστραφεί. Έτσι οι προγραμματιστές θα πρέπει να παρακολουθούν την κατάσταση μίας δραστηριότητας.

Αυτό θα το πετύχουν χρησιμοποιώντας τις λεγόμενες "lifecycle callback methods". Ο προγραμματιστής έχει τη δυνατότητα να διαχειριστεί τον κύκλο ζωής μιας δραστηριότητας, προς δικό του όφελος. Θα αναφέρουμε ένα απλό παράδειγμα, παρμένο από τα tutorials της Google, που λέει το εξής:

Στην περίπτωση που θέλουμε να κατασκευάσουμε μία εφαρμογή που θα προβάλει streaming video, δηλαδή έναν streaming video player, έχουμε τη δυνατότητα να παύσουμε το βίντεο (pause) και να τερματίσουμε την σύνδεση στο ίντερνετ την ώρα που ο χρήστης μεταφέρεται σε άλλη εφαρμογή.

Σε αυτό το σημείο βρισκόμαστε σε μία δεδομένη χρονική στιγμή όπου η εφαρμογή μας βρίσκεται στο παρασκήνιο και ο χρήστης αλληλεπιδρά με μία άλλη.

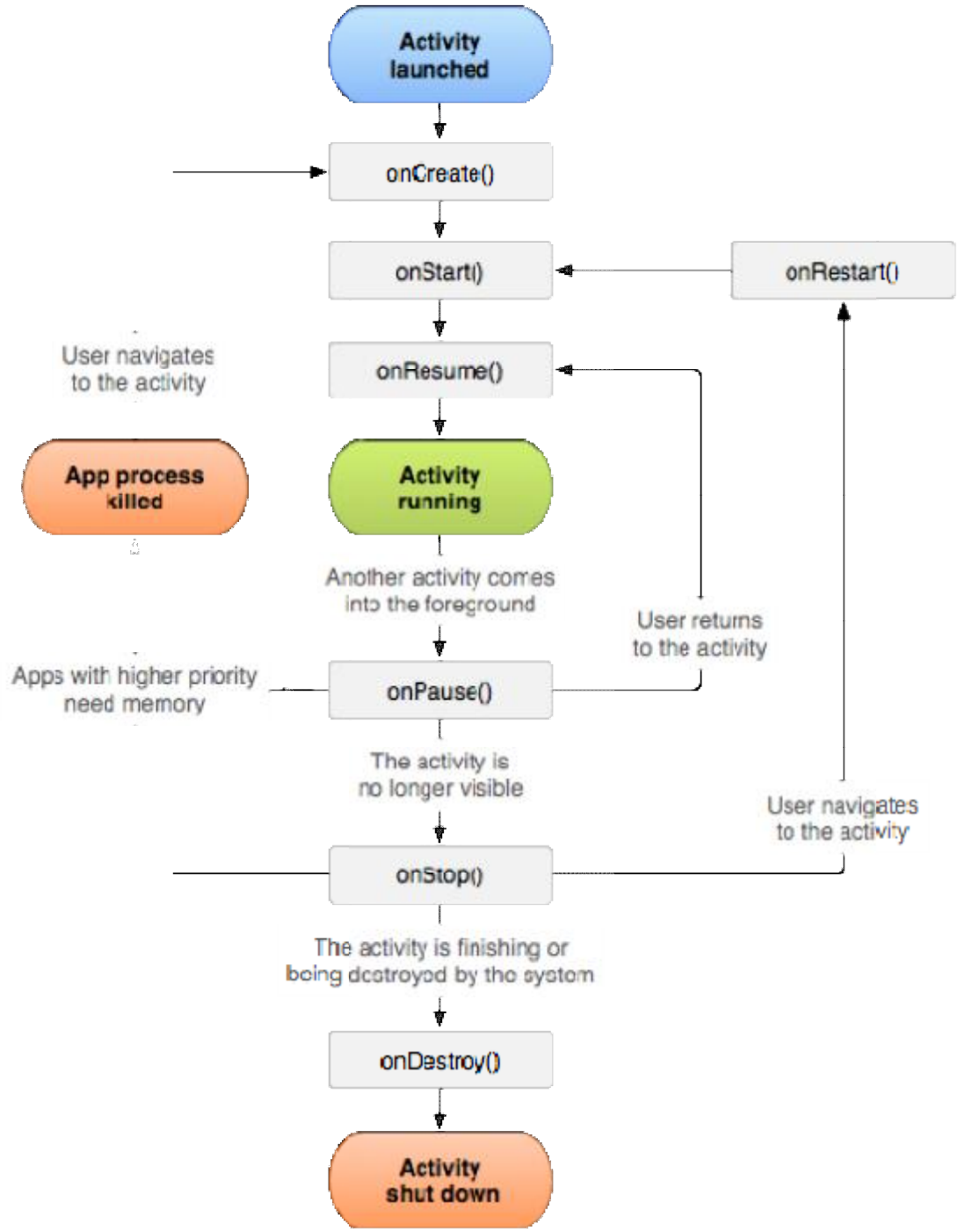
Πως θα μας οφελούσε η χρήση των μεθόδων του κύκλου ζωής της δραστηριότητας; Εφόσον έχουμε ήδη σταματήσει το βίντεο και την σύνδεση με το ίντερνετ, για την περίπτωση που ο χρήστης έχει "μετακομίσει" σε άλλη εφαρμογή, απενεργοποιούμε και απελευθερώνουμε σημαντικούς πόρους. Γιατί το γεγονός πως ένα στιγμιότυπο μίας δραστηριότητας βρίσκεται στο παρασκήνιο δεν σημαίνει ότι έχει τερματιστεί (δηλαδή το αντικείμενο της Activity δεν καταστρέφεται), εκτός αν εμείς επιθυμήσουμε τον τερματισμό του.

Τώρα, από την άλλη πλευρά, χρησιμοποιώντας τις μεθόδους του κύκλου ζωής μίας δραστηριότητας, μπορούμε να πετύχουμε την συνέχεια της αναπαραγωγής του βίντεο και της επανασύνδεσης στο ίντερνετ, αφού ο χρήστης επιστρέψει στην εφαρμογή μας. Κάτι που σημαίνει πως η δραστηριότητα που ήταν στο παρασκήνιο θα μεταφερθεί στο προσκήνιο και ο χρήστης θα συνεχίσει την παρακολούθηση του βίντεο από το σημείο που το είχε αφήσει.

Αν οι παραπάνω ενέργειες στις κατάλληλες μεθόδους του lifecycle μίας activity δεν είχαν συμβεί, τότε την στιγμή που ο χρήστης θα πλοηγούνταν στην ξένη εφαρμογή, η δική μας θα συνέχιζε να αναπαράγει το βίντεο και να είναι συνδεδεμένη στο διαδίκτυο, καταναλώνοντας σημαντικούς πόρους, άσκοπα. Ως αποτέλεσμα μία συσκευή που δεν έχει αρκετά γρήγορο επεξεργαστή ή αρκετή μνήμη για να τρέξει και τις δύο διεργασίες θα παρουσίαζε πρόβλημα.

1.3.1.5.1 Ανακοινώσεις κύκλου ζωής δραστηριότητας

Η κλάση Activity μας παρέχει αρκετές μεθόδους οι οποίες ανταποκρίνονται σε διάφορα συμβάντα που μπορούν να προκύψουν ανά πάσα στιγμή. Οι σημαντικότερες εξ αυτών είναι οι onCreate( ), η onResume( ), η onPause( ) και η onDestroy( ).



Εικόνα 1-19: **Ο κύκλος ζωής μίας δραστηριότητας**

Η παραπάνω εικόνα παρουσιάζει την ροή ενός κύκλου ζωής μίας δραστηριότητας. Το κύριο νήμα συχνά καλείται νήμα UI, επειδή εκεί γίνεται εσωτερικά η επεξεργασία για σχεδίαση του UI.

Αρκετά σημαντική πρέπει να γίνει η κατανόηση του γεγονότος ότι μία δραστηριότητα θα πρέπει να εκτελέσει οποιαδήποτε διεργασία εκτελείται μέσα σε ορισμένες μεθόδους του κύκλου ζωής της το ταχύτερο δυνατόν. Σκοπός είναι να μην μπλοκαριστεί το κύριο νήμα UI για αρκετό χρόνο, γιατί σε αυτήν την περίπτωση

το λειτουργικό σύστημα Android ίσως προβεί στον τερματισμό της εν λόγω δραστηριότητας εξαιτίας της έλλειψης αποκρίσεως

Αναλυτικά οι σημαντικότερες μέθοδοι ειδοποίησης του κύκλου ζωής μίας δραστηριότητας:

onCreate( )

Ο κύκλος ζωής μίας δραστηριότητας αρχίζει με την μέθοδο onCreate( ). Στη συγκεκριμένη μέθοδο ο προγραμματιστής συνήθως προβαίνει στις παρακάτω ενέργειες. Αρχικοποιεί τα στατικά δεδομένα της εφαρμογής, δεσμεύει τα δεδομένα και τους πόρους που απαιτούνται ενώ θέτει τη διάταξη της δραστηριότητας (layout) κάνοντας χρήση της μεθόδου Activity.setContentView( ).

Στην ουσία η onCreate( ) καλείται όταν μία δραστηριότητα εκκινεί ή επανεκκινεί όπως θα δούμε παρακάτω.

onStart( )

Εν συνεχεία περνάμε στην onStart( ) η οποία καλείται μετά την onCreate( ) ή την onRestart( ), όταν δηλαδή η δραστηριότητα είχε πάψει να λειτουργεί, αλλά τώρα είναι έτοιμη να εμφανιστεί και πάλι στο προσκήνιο και να αλληλεπιδράσει με τον χρήστη. Θα την ακολουθήσει η onResume( ).

onResume()

Στην πρώτη ροή του κύκλου ζωής μίας δραστηριότητας η onResume( ) καλείται μετά την onStart( ), αλλά σε διαφορετική περίπτωση μπορεί να καλεσθεί και μετά την onPause( ) και την onRestoreInstanceState( ).

Σε αυτή τη φάση η δραστηριότητα γίνεται δραστηριότητα παρασκηνίου και αλληλεπιδρά με τον χρήστη. Η onResume( ) είναι ένα καλό μέρος για να λαμβάνει αποκλειστικούς πόρους, να ξεκινάει animations, ήχο, βίντεο ή να ανοίγει “exclusive access devices” όπως η κάμερα κ.α.

Μία μικρή επισήμανση εδώ για την onResume( ). Χρησιμοποιώντας την συγκεκριμένη μέθοδο δεν λαμβάνουμε μία στέρεα ένδειξη ότι η δραστηριότητα μας είναι ορατή στον χρήστη. Θα μπορούσε, για παράδειγμα, ένα παράθυρο του συστήματος να είναι μπροστά, όπως ένα keyguard. Αν θέλουμε βεβαιωθούμε 100% ότι η δραστηριότητα μας αλληλεπιδρά με τον χρήστη θα πρέπει να χρησιμοποιήσουμε την onWindowFocusChanged(boolean).

onPause( )

Κατά τη διάρκεια της εκτέλεσης της εφαρμογής υπάρχει η πιθανότητα η δραστηριότητα που βρίσκεται στο προσκήνιο να “ενοχληθεί” από διάφορα άλλα

συμβάντα, όπως στο προαναφερθέν παράδειγμα της τηλεφωνικής κλήσης. Έτσι μετά την onResume( ) εκτελείται η onPause( ), η οποία μεταφέρει, προς το παρόν, τη δραστηριότητα μας στο παρασκήνιο.

Σε αυτό το σημείο μερικές από τις εργασίες που θα πρέπει να πραγματώσει ο προγραμματιστής περιλαμβάνουν την αποθήκευση των δεδομένων που δεν έχουν υποβληθεί, την απενεργοποίηση ή απελευθέρωση σημαντικών πόρων και τον τερματισμό τυχόν ήχων, animation ή βίντεο.

onStop( )

Σε αντίθεση με την κατάσταση παύσης, η οποία προσδιορίζει μία μερική απόφραξη του κυρίου νήματος UI, η κατάσταση τερματισμού (onStop( )) μας εγγυάται πως ο χρήστης βρίσκεται σε μία ξεχωριστή δραστηριότητα ή σε μία εντελώς διαφορετική εφαρμογή.

Στην onStop( ) ο προγραμματιστής καθαρίζει τυχόν στατικά δεδομένα της εφαρμογής και απελευθερώνει, όπως και στην onPause( ), τυχόν ληφθέντες πόρους.

Εφόσον η δραστηριότητά μας έχει σταματήσει, το σύστημα θα μπορούσε να καταστρέψει το στιγμιότυπο της δραστηριότητας, σε περίπτωση που χρειάζεται να ανακτήσει μνήμη.

Μέχρι εδώ που φθάσαμε, στην onStop( ) δηλαδή, υπάρχουν δύο ενδεχόμενα. Το πρώτο είναι αυτό της επανεκκίνησης της δραστηριότητας, μέσω της onRestart( ), ενώ το δεύτερο είναι η καταστροφή του αντικειμένου της δραστηριότητας, μέσω της onDestroy( ).

onRestart()

Όταν μία δραστηριότητα επανέρχεται στο προσκήνιο από την κατάσταση τερματισμού, τότε καλείται η μέθοδος onRestart( ). Στη συνέχεια καλείται και πάλι η onStart( ) και μετά η onResume( ).

Είναι σπάνιες οι περιπτώσεις που θα χρεαστεί να καλέσουμε την onResume( ), μιας και στη μέθοδο πραγματοποιούμε εργασίες αποκατάστασης οι οποίες είναι απαραίτητες μόνο όταν η εφαρμογή βρισκόταν σε κατάσταση τερματισμού. Μετά την onStop( ) δηλαδή. Εξάλλου στις απλές εφαρμογές ακόμα και η χρήση της onStop( ) σπανίζει, μιας και η onPause( ) “βγάζει τη δουλειά”.

onDestroy( )

Καλείται όταν μία δραστηριότητα τερματίζεται, καταστρέφοντας το αντικείμενο της. Ακόμη καθαρίζει τυχόν στατικά δεδομένα της εφαρμογής και απελευθερώνει πόρους. Εκτελεί με λίγα λόγια μία τελική εκκαθάριση πριν την καταστροφή του στιγμιοτύπου της δραστηριότητας.

Το συγκεκριμένο σενάριο μπορεί να συμβεί επειδή η activity τελειώνει (κάποιος έκανε κλήση της μεθόδου finish( ) ή το σύστημα προσωρινά καταστρέφει το αντικείμενο της δραστηριότητας για να ελευθερώσει χώρο, τα δύο σενάρια μπορούν να διακριθούν με τη μέθοδο isFinishing( )).

Intent

Για να κατανοήσουμε καλύτερα τις προθέσεις (intents) στο Android θα εξηγήσουμε τι αντιπροσωπεύει ένα αντικείμενο Intent. Ένα αντικείμενο της κλάσης Intent αντιπροσωπεύει μία αίτηση εργασίας, μία πρόθεση όπως αναφέρει και το όνομα της, που χρησιμοποιείται από το λειτουργικό σύστημα Android.

Επί παραδείγματι: Όταν καλούμε την μέθοδο Activity.startActivity(Intent intent), η παράμετρος intent υποδηλώνει την πρόθεση τη δική μας να ξεκινήσουμε μία νέα δραστηριότητα. Ο ορισμός της δραστηριότητας στόχου θα δηλωθεί κατά την δημιουργία του αντικειμένου Intent (new Intent(Context packageContext, Class<?> cls)). Έτσι το σύστημα Android ταιριάζει την ενέργεια Intent με την κατάλληλη δραστηριότητα. Αυτή η δραστηριότητα κατόπιν εκκινεί.

Όπως αναφέρεται και στην τεκμηρίωση του Android SDK, μία πρόθεση δεν είναι απαραίτητα κάτι συγκεκριμένο, μία συγκεκριμένη ενέργεια όπως δείξαμε στο παραπάνω παράδειγμα. Είναι δυνατό μία Intent να δηλώνει μία ασαφή πρόθεση μας, ζητώντας την εκκίνηση οποιασδήποτε δραστηριότητας η οποία συμφωνεί με ορισμένα κριτήρια.

Στο Android θα συναντάμε δύο τύπους μίας πρόθεσης και τρεις θεμελιώδης τρόπους χρήσης της. Όσον αφορά τους τύπους έχουμε implicit και explicit Intents και όσον αφορά τους τρόπους χρήσης έχουμε την εκκίνηση μίας activity, την εκκίνηση μίας service και την παράδοση ενός broadcast.

1.3.1.6 Τύποι μίας Πρόθεσης

Explicit Intents

Οι explicit intents καθορίζουν το όνομα του στοιχείου (fully-qualified class name) που θα εκκινήσει, είτε αυτό είναι μία δραστηριότητα ή μία υπηρεσία.

Implicit Intents

Στην περίπτωση των implicit intents δεν χρειάζεται να ονομάσουμε ένα συγκεκριμένο στοιχείο, απλώς να δηλώσουμε μία γενική ενέργεια η οποία θα πρέπει να πραγματοποιηθεί από μία διαφορετική εφαρμογή από τη δική μας.

Για παράδειγμα έστω ότι θέλουμε να δείξουμε στον χρήστη μία τοποθεσία σε έναν χάρτη. Μπορούμε να χρησιμοποιήσουμε μία implicit intent η οποία θα ζητήσει από μία άλλη εφαρμογή, ικανή να εμφανίσει ένα τέτοιο περιεχόμενο, να το κάνει. Η Google Maps θα μπορύσε να είναι μία από αυτές τις εφαρμογές ή οποιαδήποτε

άλλη εφαρμογή η οποία είναι ικανή να εμφανίσει μία τοποθεσία στον χάρτη. Το ίδιο θα μπορούσε να συμβεί και με την αναπαραγωγή ενός αρχείου μουσικής ή ενός βίντεο.

*1.3.1.7 **Τρόποι χρήσης μίας** Intent:*

Start an activity:

Μία δραστηριότητα εκπροσωπεί μία οθόνη στην εφαρμογή. Μπορούμε να εκκινήσουμε ένα νέο στιγμιότυπο μίας Activity περνώντας μία παράμετρο Intent στην μέθοδο startActivity, όπως αναφέραμε σε παράδειγμα προηγουμένως. Η πρόθεση περιγράφει την δραστηριότητα που θα εκκινήσει και αν είναι απαραίτητο μεταφέρει και πληροφορία.

Στην περίπτωση που θέλουμε να πάρουμε κάποιο αποτέλεσμα από μία δραστηριότητα όταν εκείνη τερματιστεί, αντί της startActivity καλούμε την startActivityForResult( ). Η δραστηριότητα μας θα λάβει το αποτέλεσμα ως ένα ξεχωριστό Intent αντικείμενο στην μέθοδο onActivityResult( ).

Start a service:

Στο Android μία υπηρεσία εκτελεί εργασίες στο παρασκήνιο, χωρίς την ύπαρξη ενός user interface. Σε αυτές τις εργασίες ο χρήστης δεν χρειάζεται να αλληλεπιδράσει με την εφαρμογή, όπως ένα κατέβασμα ενός αρχείο ή η αναπαραγωγή ενός ήχου ως background.

Μπορούμε να εκκινήσουμε μία service η οποία θα πραγματοποιήσει μία one-time operation (όπως το κατέβασμα ενός αρχείου), περνώντας μία Intent στην μέθοδο startService( ) της κλάσης Activity.

Η πρόθεση περιγράφει την υπηρεσία που θα ξεκινήσει και φέρει οποιαδήποτε σημαντική πληροφορία (data).

Deliver a broadcast:

Ένα broadcast είναι ένα μήνυμα όπου μπορεί να ληφθεί από κάθε εφαρμογή. Το Android στέλνει διάφορα τέτοια μηνύματα για συμβάντα συστήματος, όπως για παράδειγμα η φόρτιση της μπαταρίας.

Μπορούμε να στείλουμε broadcasts σε άλλες εφαρμογές περνώντας ένα αντικείμενο Intent στις μεθόδους sendBroadcast( ), sendOrderedBroadcast( ), ή sendStickyBroadcast( ).

1.3.1.8 ***Μεταβίβαση Πληροφοριών μέσω*** *Intent*

Εκτός των κύριων λειτουργιών μία πρόθεση Intent χρησιμοποιείται για τη μεταβίβαση δεδομένων ανάμεσα σε δραστηριότητες.

Τα επιπρόσθετα δεδομένα σε μία πρόθεση καλούνται extras και μπορούμε να τα συμπεριλάβουμε σε μία πρόθεση κάνοντας χρήση της μεθόδου putExtra( ) της κλάσης Intent, πάντα με τον κατάλληλο τύπο δεδομένων που θέλουμε να περιλάβουμε.

Τα extras των προθέσεων παίζουν βασικό ρόλο κατά την πλοήγηση του χρήστη στην εφαρμογή KalymnosGuide. Σχεδόν πάντα, πριν εκκινήσει η δραστηριότητα στόχου, η δραστηριότητα παρασκηνίου αποθηκεύει σημαντικές πληροφορίες στην κατάλληλη Intent. Αυτές οι πληροφορίες παίζουν καταλυτικό ρόλο στην φύση της δραστηριότητας στόχου.

putExtra()

Δανειζόμαστε ένα παράδειγμα από την εφαρμογή μας:

```
Intent intent = new Intent(MainMenu.this, SubMainMenu.class);

intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.eatanddrink);

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

MainMenu.this.startActivity(intent);
```

Αρχικά δημιουργούμε το αντικείμενο της πρόθεσης και του καθορίζουμε τη δραστηριότητα που θέλουμε να καταλλήξουμε, οπότε πρόκειται για explicit intent.

Στη συνέχεια περνάμε την extra πληροφορία μέσω της putExtra( ) (εδώ θα περάσουμε έναν integer), χρησιμοποιώντας ένα ζευγάρι ονόματος-τιμής, όπως ακριβώς και με την αποθήκευση δεδομένων στις κοινές προτιμήσεις SharedPreferences.

Τέλος ξεκινάμε την δραστηριότητα στόχου χρησιμοποιώντας την startActivity( ).

get...Extra()

Συνεχίζουμε στο ίδιο παράδειγμα σύμφωνα με το οποίο έχει εκκινηθεί η δραστηριότητα SubMainMenu. Πως μπορεί όμως να πάρει την χρήσιμη πληροφορία που της έστειλε η MainMenu μέσω της πρόθεσης intent;

Η μέθοδος getIntent( ) της κλάσης Activity επιστρέφει το αντικείμενο της Intent το οποίο ξεκίνησε την παρούσα δραστηριότητα. Οπότε μέσα στην τεκμηρίωση της SubMainMenu μπορούμε να λάβουμε την καλούσα πρόθεση ως εξής:

```
Intent callingIntent = this.getIntent( );
```

Και πως μπορούμε να πάρουμε την πληροφορία που χρειαζόμαστε; Μέσω μίας εκ των μεθόδων get...Extra( ) της κλάσης Intent, αναλόγως τη φύση της πληροφορίας που αποθηκεύσαμε ως extra στην πρόθεση. Στο δικό μας παράδειγμα είχαμε προσθέσει έναν integer, οπότε θα εξάγουμε έναν integer:

```
int pliroforia = callingIntent.getIntExtra("IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU", 0);
```

Όπως παρατηρούμε θα πρέπει να δώσουμε την ίδια τιμή String που χρησιμοποιήσαμε ως κλειδί για να πάρουμε την πληροφορία που θέλουμε.

Άλλες μεθόδοι που εξάγουμε πληρορία από μία πρόθεση είναι οι getDoubleExtra( ), getFloatExtra( ), getStringExtra( ) κτλ.

1.3.1.9 Χρήση Πρόθεσης για Εκκίνηση άλλων Εφαρμογών

Μέχρι στιγμής είδαμε πως μία εφαρμογή μπορεί να εκκινήσει δραστηριότητες που ορίζονται στο δικό της πακέτο, χρησιμοποιώντας προθέσεις. Ωστόσο, οι εφαρμογές μπορούν να εκκινήσουν και εξωτερικές δραστηριότητες οι οποίες βρίσκονται μέσα σε άλλες εφαρμογές, με τις κατάλληλες άδειες φυσικά.

Στην τεκμιρίωση του Android θα βρούμε πολλές καθορισμένες ενέργειες προθέσεων, για πολλές συνηθισμένες εργασίες χρηστών. Για παράδειγμα:

- Εκκίνηση ενσωματομένου προγράμματος περιήγησης στο Web και παροχή μίας διεύθυνσης URL
- Εκκίνηση προγράμματος περιήγησης στο Web και παροχή μίας συμβολοσειράς αναζήτησης
- Εκκίνηση ενσωματωμένης εφαρμογής Dialer και παροχή ενός αριθμού τηλεφώνου
- Εκκίνηση ενσωματωμένης εφαρμογής Maps και παροχή μιας θέσης
- Εκκίνηση ενσωματωμένης εφαρμογής Camera σε τρόπο λειτουργίας φωτογραφίας ή βίντεο
- Εκκίνηση ενός επιλογέα ήχου κλήσης
- Εγγραφή ενός ήχου

Παρακάτω θα δώσουμε ένα παράδειγμα για το πως μπορούμε να εκκινήσουμε ένα πρόγραμμα περιήγησης στο Web με ένα καθορισμένο URL.

Ξεκινάμε με την δημιουργία ενός αντικειμένου URL:

```
Uri address = Uri.parse("http://developer.android.com/training/index.html");
```

Στη συνέχεια δημιουργούμε μία πρόθεση με μία προκαθορισμένη ενέργεια (ACTION_VIEW). Στην τεκμιρίωση της κλάσης Intent μπορούμε να βρούμε όλες τις προκαθορισμένες ενέργειες.

```
Intent intent = new Intent(Intent.ACTION_VIEW, address);
```

Τέλος, ξεκινάμε την δραστηριότητα.

```
this.startActivity(intent);
```

Κατά τον χρόνο εκτέλεσης, σε αυτό το σημείο, εκκινεί η δραστηριότητα του προγράμματος περιήγησης και εισέρχεται στο προσκήνιο, στέλνοντας την αρχική καλούσα δραστηριότητα προς προσωρινή παύση στο παρασκήνιο. Ο χρήστης τελειώνοντας την εργασία του με το πρόγραμμα περιήγησης, αφού κάνει Back, επανεκκινεί η αρχική δραστηριότητα.

Σε αυτό το σημείο θα πρέπει να σημειώσουμε πως μία εφαρμογή έχει τη δυνατότητα να δημιουργήσει δικούς της τύπους προθέσεων και να επιτρέψει σε άλλες εφαρμογές να την καλούν, κάτι που καθιστά δυνατή την ανάπτυξη ισχυρά ενοποιημένων πακέτων εφαρμογών.

Παράθυρα Διαλόγου

Το λειτουργικό σύστημα Android το "τρέχουν" εκατοντάδες διαφορετικές συσκευές, smartphones, tablets και πολλές άλλες, με αποτέλεσμα οι οθόνες των συσκευών αυτών να ποικίλουν σε μέγεθος.

Παλαιότερα που το λειτουργικό το έτρεχαν ως επί το πλείστον έξυπνα κινητά τηλέφωνα οι οθόνες, οι οθόνες των συσκευών αυτών ήταν μικρές και ο χώρος της διεπαφής χρήστη ήταν και είναι πολύτιμος.

Θα υπάρξουν, λοιπόν, περιπτώσεις όπου θα χρειαστεί να χειριστούμε μία μικρή ποσότητα αλληλεπίδρασης, αλλά η δημιουργία μίας ξεχωριστής δραστηριότητας θα ήταν κάτι υπερβολικό για μία τόσο μικρή χρήση. Κάπου εδώ έρχονται τα παράθυρα διαλόγου να μας λύσουν τα χέρια.

Στην τεκμιρίωση του Android ένα παράθυρο διαλόγου ορίζεται ως ένα μικρό παράθυρο το οποίο προτρέπει τον χρήστη να πάρει μία απόφαση (για παράδειγμα "ναι" ή "όχι") ή να εισάγει επιπλέον πληροφορία. Ένα παράθυρο διαλόγου δεν γεμίζει την οθόνη και συνήθως χρησιμοποιείται σε γεγονότα που απαιτείται από τους χρήστες να αναλάβουν μία δράση για να μπορέσουν να προχωρήσουν. Επίσης το παράθυρο διαλόγου ανήκει στην δραστηριότητα και εκπροσωπείται από την κλάση Dialog.

Μία δραστηριότητα μπορεί να περιλαμβάνει ένα ή περισσότερα παράθυρα διαλόγου και κάθε παράθυρο διαλόγου μπορεί να δημιουργηθεί και μετά να χρησιμοποιηθεί πολλές φορές. Κάτι τέτοιο μπορεί να υλοποιηθεί με τη χρήση των μεθόδων της κλάσης Activity που αφορούν τα παράθυρα διαλόγου, ωστόσο, σήμερα είναι deprecated (use the new DialogFragment class with FragmentManager instead). Αυτό βέβαια δεν σημαίνει πως μας στερείται η δυνατότητα χρήσης, οπότε αξίζει τουλάχιστον να τις αναφέρουμε.

1.3.1.10 ***Μέθοδοι Παραθύρων Διαλόγου της κλάσης*** *Activity*

- showDialog() Εμφανίζει ένα παράθυρο διαλόγου, αν χρειάζεται το δημιουργεί

- onCreateDialog() Είναι μια ειδοποίηση, όταν ένα παράθυρο διαλόγου δημιουργείται για πρώτη φορά και προστίθεται στην δεξαμενή παραθύρων διαλόγου της δραστηριότητας

- onPrepareDialog() Είναι μία ειδοποίηση για την ενημέρωση ενός παραθύρου διαλόγου στα γρήγορα. Ένα παράθυρο διαλόγου δημιουργείται μία φορά και μπορεί να χρησιμοποιηθεί πολές φορές από μία δραστηριότητα. Η onPrepareDialog( ) επιτρέπει στο παράθυρο διαλόγου να ενημερώνεται, ακριβώς πριν να εμφανιστεί για κάθε κλήση της showDialog( )

- dismissDialog() Κλείνει ένα παράθυρο διαλόγου και επιστρέφει στην δραστηριότητα. Το παράθυρο διαλόγου συνεχίζει να είναι διαθέσιμο, για να ξαναχρησιμοποιηθεί και πάλι με κλήση της showDialog( )

- removeDialog() Αφαιρεί το παράθυρο διαλόγου πλήρως από την δεξαμενή παραθύρων διαλόγου. Δηλαδή καταστρέφει το αντικείμενο της Dialog class

Τα παράθυρα διαλόγου είναι αρκετά χρήσιμα μιας και είναι ικανά να δημιουργήσουν πολύ απλές διεπαφές χρήστη, που δεν απαιτούν να λειτουργήσει μία τελείως νέα οθόνη ή δραστηριότητα. Αντ' αυτού, η καλούσα δραστηριότητα αποστέλλει ένα παράθυρο διαλόγου, το οποίο μπορεί να έχει την δική του διάταξη και διεπαφή χρήστη, με κουμπιά και μηχανισμούς ελέγχου εισόδου.

Η Dialog class είναι η βασική κλάση για τα παράθυρα διαλόγων, αλλά η τεκμιρίωση του Android μας προτρέπει να αποφεύγουμε την απευθείας αρχικοποίηση της. Μας δίνει όμως κάποιες χρήσιμες subclasses της Dialog όπως η AlertDialog (παράθυρο διαλόγου που έχει έναν τίτλο, μέχρι τρία κουμπιά και μία λίστα από selectable items), η DatePickerDialog ή TimePickerDialog (παράθυρο διαλόγου με προσχεδιασμένο UI για την επιλογή ώρας-ημερομηνίας), ProgressDialog, CharacterPickerDialog κτλ.

Τέλος, μπορούμε ακόμη να δημιουργήσουμε ένα παράθυρο διαλόγου προσαρμοσμένο στις δικές μας ανάγκες, σχεδιάζοντας ένα αρχείο διάταξης XML και κάνοντας χρήση της μεθόδου Dialog.setContentView( ). Για να ανακτήσουμε μηχανισμούς ελέγχου από την διάταξη του παραθύρου διαλόγου, απλώς χρησιμοποιούμε την Dialog.findViewById( ).

Fragments

Η έννοια των fragments εμφανίζεται αργότερα στο Android και παρουσιάστηκε στην 3.0 έκδοση του λειτουργικού (API lvl 11), κυρίως για την υποστήριξη πιο δυναμικών και ευέλικτων σχέδιων UI σε μεγάλες οθόνες, όπως αυτές των tablet.

Μπορούμε να ορίσουμε πολύ απλά ένα fragment ως μπλοκ UI, με τον δικό του κύκλο ζωής, το οποίο μπορεί να επαναχρησιμοποιηθεί μέσα σε διαφορετικές δραστηριότητες. Πολλά fragments μπορούν να συνδιαστούν σε μία μόνο δραστηριότητα χτίζοντας ένα UI πολλαπλών παραθύρων.

Τα fragments είναι αρθρωτά κομμάτια μίας δραστηριότητας, που, όπως προαναφέραμε, έχουν τον δικό τους κύκλο ζωής, δέχονται τα δικά τους input events και επίσης μπορούμε να τα προσθέσουμε ή να τα αφαιρέσουμε κατά το χρόνο εκτέλεσης της δραστηριότητας.

Είναι άρρηκτα συνδεδεμένα με την δραστηριότητα που “υπηρετούν”, μιας και πάντα υλοποιούνται μέσα στην εν λόγω activity. Ακόμη και ο κύκλος ζωής τους επηρεάζεται από τον κύκλο ζωής της δραστηριότητας που τα φιλοξενεί. Για παράδειγμα, όταν μία δραστηριότητα βρίσκεται σε κατάσταση paused, τότε και όλα τα fragments που περιέχονται σε εκείνη θα είναι paused, όπως επίσης όταν η δραστηριότητα καταστραφεί, έτσι και τα fragments. Ωστόσο, θα πρέπει να σημειωθεί πως όταν η δραστηριότητα τρέχει (δηλαδή βρίσκεται στη resume φάση του lifecycle της), εμείς μπορούμε να διαχειριστούμε κάθε fragment ξεχωριστά, όπως προσθέτοντας τα ή αφαιρόντας τα.

Ένα συνηθισμένο παράδειγμα χρήσης fragment είναι αυτό της εφαρμογής ενός ειδησεογραφικού πρακτορείου, όπου διαφαίνεται η αλλαγή ροής UI ανάμεσα σε κατακόρυφο και οριζόντιο προσανατολισμό. Αν μία διεπαφή έχει μία προβολή λίστας στοιχείων και μία προβολή λεπτομερειών, η λίστα και οι λεπτομέρειες μπορούν να είναι και οι δύο, fragments. Το πρώτο fragment θα παρουσιάζει τους τίτλους των ειδήσεων ενώ το δεύτερο την είδηση. Κλικάροντας σε έναν τίτλο του πρώτου fragment θα εμφανίζεται η σχετική είδηση στο δεύτερο fragment και, το σημαντικό, όλα αυτά μέσα στην ίδια δραστηριότητα!

*1.3.1.11 **Δημιουργία ενός** Fragment*

Για να δημιουργήσουμε ένα fragment θα πρέπει να δημιουργήσουμε μία νέα κλάση η οποία θα κληρονομεί την κλάση Fragment.

Ένα αντικείμενο της Fragment class έχει τον δικό του κύκλο ζωής, ο οποίος είναι "δεμένος" με αυτόν της δραστηριότητας. Έτσι πολλές από τις μεθόδους του κύκλου ζωής μίας δραστηριότητας είναι παρόμοιες με αυτές της κλάσης Fragment, όπως για παράδειγμα η onCreate( ), η onStart( ), onPause( ), onStop( ) κτλ.

Συνήθως πρέπει να υλοποιήσουμε τουλάχιστον τις παρακάτω βασικές ειδοποιήσεις του κύκλου ζωής ενός τεμαχίου:

- onCreate( ) Καλείται κατά τη δημιουργία του fragment
- onCreateView( ) Καλείται όταν το fragment πρέπει να δημιουργήσει την δική του προβολή
- onStart( ) Καλείται όταν το fragment γίνεται ορατό στον χρήστη
- onPause( ) Παρόμοιοα με την Activity.onPause( )
- onStop( ) Καλείται όταν το fragment δεν είναι ορατό στον χρήστη
- onDestroy( ) Διαγραφή fragment

Η αρθρωτή φύση των fragments τα κάνει ένα πολύ ισχυρό δομικό στοιχείο των σύγχρονων εφαρμογών. Αξίζει να σημειωθεί πως η API Fragment διατίθεται και ως μία στατική βιβλιοθήκη συμβατότητας, που μπορεί να χρησιμοποιηθεί με παλαιότερες εκδόσεις του Android, μέχρι και την 1.6, οπότε τα χαρακτηριστικά της κλάσης Fragment μπορούν να χρησιμοποιηθούν από τις περισσότερες εφαρμογές του Android.

Πόροι - Resources

Μία εφαρμογή Android εμπεριέχει γραφικά, συμβολοσειρές και άλλους διάφορους τύπους πόρων οι οποίοι είναι απαραίτητοι για να χτιστούν στιβαρές διεπαφές χρήστη.

*1.3.1.12 **Πόροι Εφαρμογής και Πόροι Συστήματος***

Μελετώντας τους πόρους της πλατφόρμας του Android από μία αφαιρετική οπτική θα διαπιστώσουμε ότι χωρίζονται σε δύο μεγάλες κατηγορίες. Οι πόροι της εφαρμογής και οι πόροι του συστήματος.

Ο προγραμματιστής ορίζει τους πόρους εφαρμογής μέσα στα αρχεία του έργου του Android και είναι ειδικοί για το application.

Στην αντίπερα όχθη, οι πόροι του συστήματος είναι κοινοί πόροι, ορίζονται από την πλατφόρμα και είναι προσπελάσιμοι από όλες τις εφαρμογές.

Και οι δύο τύποι πόρων μπορούν να προσπελαστούν κατά τον χρόνο εκτέλεσης, φορτώνοντας τους μέσα στον κώδικα Java, τις περισσότερες φορές μέσα σε μία activity.

1.3.1.13 ***Διαχείριση Πόρων Εφαρμογής***

Οι πόροι της εφαρμογής αποθηκεύονται μέσα στα αρχεία του έργου του Android και πιο συγκεκριμένα στον φάκελο /res (resources).

Οργανώνονται, ορίζονται και μεταγλωττίζονται μαζί με το πακέτο της εφαρμογής, χρησιμοποιώντας μία σωστά καθορισμένη και ευέλικτη δομή καταλόγου.

Επίσης, οι πόροι της εφαρμογής δεν μπορούν να χρησιμοποιηθούν από το υπόλοιπο σύστημα Android, παρά μόνο από την εφαρμογή.

```
MyProject/
  src/
    MyActivity.java
  res/
    drawable/
      icon.png

       layout/
      main.xml
      info.xml
    values/
      strings.xml
```

Όπως παρατηρούμε στο παραπάνω παράδειγμα ο φάκελος /res περιέχει όλους τους πόρους σε υποφακέλους. Ένα image resource, δύο layout resource και ένα string resource.

Τα ονόματα των υποφακέλων είναι σημαντικά και ανάλογα με αυτά καθορίζεται ο τύπος του πόρου που θα αποθηκευτεί στο ανάλογο subdirectory. Σύμφωνα με την προηγούμενη πρόταση, στον υποφάκελο με όνομα /drawable θα αποθηκεύονται μόνο αρχεία εικόνων, στον υποφάκελο /layout μόνο αρχεία διάταξης xml κτλ.

1.3.1.14 ***Όλοι οι υποφακέλοι του*** */res directory:*

- /animator Αποθηκεύει αρχεία XML τα οποία ορίζουν ιδιότητες αρχείων σχεδιάσιμων γραφικών

- /anim Αποθηκεύει αρχεία XML τα οποία ορίζουν αρχεία σχεδιάσιμων γραφικών

- /color Αποθηκεύει αρχεία XML τα οποία ορίζουν χρώματα

- /drawable Αποθηκεύει αρχεία (.png, .9.png, .jpg, .gif) ή αρχεία XML τα οποία μεταγλωττίζονται ως τα παρακάτω drawable resource subtypes:

Nine-Patches (re-sizable bitmaps)

State lists

Shapes

Animation drawables

Other drawables

- /layout Αποθηκεύει αρχεία XML τα οποία ορίζουν ένα user interface layout

- /menu Αποθηκεύει αρχεία XML τα οποία ορίζουν ένα application menu

- /raw Αποθηκεύει ακατέργαστα αρχεία όλων των τύπων

- /values Αποθηκεύει XML αρχεία τα οποία περιέχουν απλές τιμές όπως strings, integers, και colors.

- /xml Αποθηκεύει ακατέργαστα αρχεία XML τα οποία αναγιγνώσκονται κατά τον χρόνο εκτέλεσης

Μέχρι την ώρα που γράφεται αυτή η εργασία, θα πρέπει να σημειωθεί πως, στο Eclipse IDE με το οποίο δημιουργήσαμε την δική μας εφαρμογή, ορισμένοι υποκατάλογοι του /res, όπως οι /drawable, /layout και /values, δημιουργούνται προεπιλεγμένα κατά τη δημιουργία νέου έργου Android. Ωστόσο δεν συμβαίνει το ίδιο και με άλλους υποκαταλόγους, οι οποίοι θα πρέπει να προστεθούν από τον προγραμματιστή, όταν χρειάζεται, όπως ο /raw.

Τώρα όσον αφορά τους τύπους των πόρων, η πλατφόρμα του Android υποστηρίζει διάφορους από αυτούς, οι οποίοι μπορούν να συνδυαστούν και να δημιουργήσουν στιβαρές εφαρμογές.

Μία εφαρμογή μπορεί να περιλαμβάνει πολλά και διαφορετικά είδη πόρων όπως συμβολοσειρές, χρώματα, διαστάσεις, αρχεία σχεδιάσιων γραφικών-animations, αρχεία διάταξης-layout και ακατέργαστα αρχεία παντός τύπου.

1.3.1.15 ***Τύποι Πόρων Εφαρμογής***

- Animation Resources

Οι tween animations αποθηκεύονται στον res/anim/ και μπορούμε να αποκτήσουμε πρόσβαση μέσω της R.anim class.

Οι frame animations αποθηκεύονται στον res/drawable/ και είναι προσβάσιμοι μέσω της R.drawable class.

- Color State List Resource

Οι color resources αποθηκεύονται στον res/color/ και είναι προσβάσιμοι από την R.color class.

- Drawable Resources

Αποθηκεύονται στον res/drawable/ και πρόσβαση από την R.drawable class.

- Layout Resource

Ορισμός ενός layout για το UI του application.

Αποθήκευση στον res/layout/ και πρόσβαση από την R.layout class.

- Menu Resource

Ορισμός περιεχομένου μενού εφαρμογής.

Αποθήκευση στον res/menu/ πρόσβαση από την R.menu class.

- String Resources

Ορισμός strings και string arrays.

Αποθήκευση στον res/values/ και πρόσβαση από την R.string, R.array.

- Style Resource

Ορισμός του look και του format των στοιχείων του UI.

Αποθήκευση στον res/values/ και πρόσβαση από την the R.style class.

- More Resource Types

Ορισμός απλών τιμών όπως booleans, integers, dimensions, colors, και άλλων array.

Αποθήκευση στον res/values/ αλλά πρόσβαση από την μοναδική R sub-class (όπως R.bool, R.integer, R.dimen, κτλ.).

1.3.1.16 Αναφορά στους Πόρους της Εφαρμογής

Οι πόροι της εφαρμογής μπορούν να χρησιμοποιηθούν προγραμματιστικά και επίσης μπορούν να αναφερθούν μέσα σε άλλους πόρους εφαρμογών.

Στην αναφορά των διαφόρων τύπων πόρων της εφαρμογής που κάναμε παραπάνω, μιλήσαμε για τον υποφάκελο που θα αποθηκεύεται ο εκάστοτε πόρος αλλά και την κλάση η οποία παρέχει την πρόσβαση στον πόρο. Υπεύθυνη για την πρόσβαση των πόρων drawable είναι η κλάση R.drawable, για τους πόρους menu η R.menu, για τα strings η R.string κτλ. Συμπαιρένουμε λοιπόν πως για πρόσβαση σε κάποιον πόρο θα αναφερθούμε σε μία υποκλάση της κλάσης R. Ποια είναι όμως αυτή η κλάση R;

1.3.1.17 Η κλάση R

"Με έναν σύντομο, δικό μας, ορισμό, η κλάση R (το "R" για Resources) του android είναι μία αυτόματα παραγόμενη κλάση η οποία περιέχει όλους τους ορισμούς των πόρων για ένα συγκεκριμένο πακέτο εφαρμογής.

Το δικό μας πακέτο εφαρμογής είναι το com.Skemelio.kalymnosguide, οπότε και παράγεται αυτόματα η κλάση com.Skemelio.kalymnosguide.R."

Οι πόροι της εφαρμογής μπορούν να προσπελαστούν προγραμματιστικά χρησιμοποιώντας το αυτόματα παραγόμενο αρχείο κλάσης R.java. Έτσι για να αναφερθούμε σε έναν πόρο μέσα σε μία δραστηριότητα θα πρέπει να ανακτήσουμε το αντικείμενο της κλάσης Resources, κάνοντας χρήση της μεθόδου getResources( ) και μετά θα προχωρήσουμε στην κατάλληλη κλήση της μεθόδου, με βάση τον τύπου του πόρου, που θέλουμε να ανακτήσουμε.

Παράδειγμα: Έστω ότι θέλουμε να ανακτήσουμε έναν πόρο string της KalymnosGuide, ο οποίος ορίζεται μέσα στο αρχείο /res/values/strings.xml.

```
String appName = this.getResources( ).getString(R.string.app_name);
```

Με τον παραπάνω τρόπο αναφερθήκαμε στον πόρο μέσα στον κώδικα Java. Στην περίπτωση που θέλουμε να αναφερθούμε στον ίδιο πόρο από ένα άλλο μεταγλωττισμένο πόρο, π.χ. ένα αρχείο διάταξης τότε θα εργαστούμε ως εξής:

@[τύπος πόρου]/[όνομα πόρου], δηλαδή στην δική μας περίπτωση

@string/app_name.

1.3.1.18 Διαχείριση Πόρων Συστήματος

Εκτός από τους πόρους εφαρμογής τους οποίους εμείς επιλέγουμε και εργαζόμαστε με αυτούς σε ένα έργο Android, υπάρχουν και οι πόροι του συστήματος της πλατφόρμας.

Μία εφαρμογή έχει τη δυνατότητα να προσπελάσει τους πόρους του συστήματος Android πέρα από τους ιδιωτικούς πόρους. Οι πόροι του συστήματος είναι

κοινόχρηστοι σε όλες τις εφαρμογές, παρέχοντας στον χρήστη κοινά στυλ και άλλα χαρακτηριστικά, όπως συνήθης χρησιμοποιούμενες συμβολοσειρές και χρώματα.

Αντιστοίχως με τους πόρους εφαρμογής, οι οποίοι αποθηκεύονται στο πακέτο "όνομα_πακέτου_εφαρμογής.R", οι πόροι του συστήματος αποθηκεύονται στο πακέτο android.R. Έτσι, όπως είχαμε το παράδειγμα με την στατική κλάση R.string για τις συμβολοσειρές των πόρων εφαρμογής, εδώ θα έχουμε την αντίστοιχη κλάση android.R.string.

Η διαφορές είναι μικρές και όσον αφορά την ανάκτηση ενός πόρου συστήματος, ας πούμε μέσα από μία δραστηριότητα. Έστω ότι θέλουμε να ανακτήσουμε το String "Cancel" από τους πόρους του συστήματος. Θα εργασθούμε ως εξής:

```
String cancel = this.getResources().getSystem().getString(android.R.string.cancel);
```

Όπως παρατηρούμε την διαφορά με τους πόρους εφαρμογής κάνει η μέθοδος getSystem( ) πριν από την κλήση της getString( ), η οποία ανακτά το καθολικό αντικείμενο συστήματος της κλάσης Resource και επίσης στην παράμετρο της μεθόδου getString( ) αντί για R.string έχουμε android.R.string.

Παρόμοια και με την αναφορά σε έναν πόρο συστήματος για την XML.

@android:[τύπος πόρου]/[όνομα πόρου], έτσι για το παράδειγμα του cancel έχουμε:

```
@android:string/cancel
```

*1.3.1.19 **Τιμές Απλών Πόρων***

Πρόκειται για τους πόρους εφαρμογής που αποθηκεύονται στον υποφάκελο /values του /res, συμβολοσειρές, χρώματα και διαστάσεις.

Οι τιμές των απλών πόρων ορίζονται μέσα σε αρχεία XML του /res/values. Αυτά τα αρχεία πόρων χρησιμοποιούν ειδικές σημάνσεις XML, που παριστούν ζεύγη ονόματος/τιμής. Οι συγκεκριμένοι πόροι μεταλωττίζονται μέσα στο κατάλληλο πακέτο, κατά τον χρόνο δόμησης.

Τέλος μπορούμε να τους διαχειριστούμε με την χρήση κάποιου IDE, όπως του Eclipse ή του Android Studio, ή απλώς να επεξεργαστούμε απευθείας τα αρχεία XML.

*1.3.1.20 **Συμβολοσειρές***

Οπουδήποτε χρειαστεί η εφαρμογή μας κείμενο θα χρησιμοποιήσουμε πόρους συμβολοσειράς.

Αυτά τα strings θα τα ορίσουμε με την σήμανση <string>, θα τους αναγνωρίσουμε με την ιδιότητα name και θα τα αποθηκεύσουμε στο αρχείο /res/values/strings.xml.

Θα πάρουμε ένα δείγμα από την δική μας εφαρμογή. Αν στην KalymnosGuide ανοίξουμε το αρχείο /res/values/strings.xml θα εμφανιστούν πολλοί ορισμοί συμβολοσειρών, συμπεριλαμβανομένων των παρακάτω:

```
<!-- Ονόματα Επιχειρήσεων -->

<string name="bar_domus">Domus</string>

<string name="bar_cantina">Cantina</string>

<string name="bar_scorpion">Scorpion</string>

<string name="tavern_pantelis">O Pantelis</string>

<string name="tavern_aigeopelagitiko">Aigeopelagitiko</string>

<string name="ouzo_manias">O Manias</string>

<string name="ouzo_mamouzelos">Mamouzelos</string>
```

Οι παραπάνω συμβολοσειρές αντιπροσωπεύουν ονόματα επιχειρήσεων τα οποία θα χρησιμοποιήσει η εφαρμογή KalymnosGuide. Ορίζουμε τον πόρο συμβολοσειράς χρησιμοποιώντας τη σήμανση <string>, μέσα στην σήμανση η ιδιότητα name η οποία θα αναγνωρίζει τον πόρο και ανάμεσα στην σήμανση <string name=”paradeigma”>value</string> η τιμή της συμβολοσειράς.

Επομένως, αν εργαζόμαστε σε μία activity και χρειαστούμε τον πρώτο πόρο του άνω παραδείγματος δεν έχουμε παρά να τον αποθηκεύσουμε σε ένα String, όπως παρακάτω:

```
String barDomus = this.getResources( ).getString( R.string.bar_domus);
```

Η τιμή του άνω String ισοδυναμεί με “Domus”.

1.3.1.21 ***Χρώματα***

Οι πόροι χρώματος αποθηκεύονται στο αρχείο /res/values/color.xml, ορίζονται με την σήμανση <color> και αναγνωρίζονται με την ιδιότητα name, παρόμοια με τις συμβολοσειρές.

Το αρχείο /res/values/color.xml ανήκει στα αρχεία τα οποία δεν δημιουργούνται προεπιλεγμένα, όπως προαναφέραμε παραπάνω, οπότε πρέπει να τα δημιουργήσουμε εμείς.

Το Android υποστηρίζει χρώματα 12-bit και 24-bit σε μορφοποίηση RGB. Η τιμή ενός χρώματος ξεκινάει πάντα με μία δίαιση (#) και στη συνέχεια ακολουθείτε με την Alpha-Red-Green-Blue (το alpha αναφέρεται στην διαφάνεια) πληροφορία στα παρακάτω formats, σύμφωνα με την τεκμηρίωση το Android:

- #RGB ϝ�� 12-bit
- #ARGB ϝ�� 12-bit alpha
- #RRGGBB ϝ�� 24-bit
- #AARRGGBB ϝ�� 24-bit alpha

Το παρακάτω δείγμα είναι παρμένο από την KalymnosGuide. Παρατηρούμε ότι οι τιμές των χρωμάτων είναι δεκαεξαδικές και είναι λογικό κάποιος να μην είναι οικείος με το δεκαεξαδικό σύστημα. Λύσεις υπάρχουν στο web, όπου μπορεί να αναζητήσει την δεκαεξαδική τιμή του χρώματος που θέλει (http://html-color-codes.info/) ή ακόμα μπορεί να εκμεταλλευτεί την παλέττα του Photoshop ή κάποιου αντίστοιχου λογισμικού επεξεργασίας εικόνων.

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

<resources>

<color name="white_transparent_60">#99FFFFFF</color>

<color name="eatanddrink_transparent40">#66ddf100</color>

<color name="placestostay_transparent40">#66f19f00</color>

<color name="history_transparent40">#66f14f00</color>

<color name="howtoreach_transparent40">#660077f1</color>

<color name="islandbeauties_transparent40">#66000bf1</color>

<color name="whattodo_transparent40">#661cf100</color>

<color name="culture_transparent40">#66c6f100</color>

<color name="usefullinfo_transparent40">#66ddf100</color>

</resources>
```

Στον παρακάτω κώδικα θα ανακτήσουμε έναν από τους παραπάνω πόρους χρωμάτων μέσα σε μία κλάση Activity, χρησιμοποιώντας την μέθοδο getColor( ).

```
int color = this.getResources.getColor(R.color.culture_transparent40);
```

Μπορούμε να πάρουμε και έναν έτοιμο πόρο χρώματος από το Android:

```
int color = MainMenu.this.getResources().getSystem().getColor(android.R.color.black);
```

1.3.1.22 ***Διαστάσεις***

Οι πόροι διαστάσεων είναι πολύ σημαντικοί στο Android και τους χρειαζόμαστε για να καθορίσουμε διάφορα μεγέθη όπως αυτό μίας συμβολοσειράς, το μέγεθος ενός μηχανισμού ελέγχου διεπαφής χρήστη, όπως μία ImageView, ένα Button ή μία TextView κτλ.

Οι πόροι διαστάσεων ορίζονται με την σήμανση <dimen>, αναγνωρίζονται με την ιδιότητα name -όπως όλοι οι άλλοι πόροι- και αποθηκεύονται μέσα στο αρχείο πόρου /res/values/dimens.xml. Δυστυχώς και αυτό το αρχείο θα πρέπει να δημιουργηθεί από εμάς.

```
<resources>

<!-- Default screen margins, per the Android Design guidelines. -->

<dimen name="activity_horizontal_margin">16dp</dimen>

<dimen name="activity_vertical_margin">16dp</dimen>

</resources>
```

Το παραπάνω δείγμα δείχνει τις δύο καταχωρημένες διαστάσεις του αρχείου /res/values/dimens.xml της KalymnosGuide. Αυτό που παρατηρούμε στις διαστάσεις είναι πως η τιμή κάθε πόρου διάστασης πρέπει να τελειώνει με μία μονάδα μέτρησης. Ακολουθούν οι μονάδες μέτρησης διαστάσεων που υποστηρίζει το Android καθώς και η περιγραφή αυτών:

Τύπος	**Περιγραφή**	**Συμβολοσειρά**
Pixels	Pixels της οθόνης	px
Inches	Φυσική μέτρηση	in
Millimeters	Φυσική μέτρηση	mm
Points	Κοινή μέτρηση γραμματοσειράς	pt
Pixels ανεξάρτητα πυκνότητας	Pixels σχετικά προς τα 160 dpi	dp
Pixels ανεξάρτητα κλίμακας	Προτεινόμενη για εμφάνιση κλιμακούμενων γραμματοσειρών	sp

Πίνακας 1-2: **Μονάδες μέτρησης διαστάσεων που υποστηρίζει το** Android

Έφθασε η ώρα για την ανάκτηση ενός πόρου διάστασης κάνοντας χρήση της κλάσης getDimension( ):

```
float dimen = this.getResources( ).getDimension(R.dimen.activity_vertical_margin);
```

1.3.1.23 ***Σχεδιάσιμοι Πόροι - Εικόνες***

Ο φάκελος /res/drawable είναι υπεύθυνος για την αποθήκευση σχεδιάσιμων πόρων, όπως είναι οι εικόνες.

Στην πραγματικότητα το Android παρέχει πολλαπλές εκδόσεις των ίδιων γραφικών για οθόνες διαφορετικών συσκευών, όπου η πυκνότητα τους σε pixels ποικίλει. Για παράδειγμα, ένα προεπιλεγμένο έργο Android περιέχει διάφορους καταλόγους /drawable. Τον /drawable-ldpi για χαμηλή πυκνότητα σε pixels, /drawable-mdpi για μέση πυκνότητα, /drawable-hdpi για υψηλή πυκνότητα.

Το λειτουργικό υποστηρίζει, σε ένα γενικευμένο πλαίσιο, τέσσερις διαφορετικά μεγέθη οθονών και έξι διαφορετικές πυκνότητες οθόνης. Όσον αφορά το μέγεθος έχουμε small, normal, large, και xlarge. Όσον αφορά την πυκνότητα έχουμε:

ldpi (low) ~120dpi

mdpi (medium) ~160dpi

hdpi (high) ~240dpi

xhdpi (extra-high) ~320dpi

xxhdpi (extra-extra-high) ~480dpi

xxxhdpi (extra-extra-extra-high) ~640dpi

Το Android είναι εκείνο που θα επιλέξει την σωστή έκδοση του πόρου, με βάση την συσκευή στην οποία θα εκτελεστεί η εφαρμογή. Εδώ θα πρέπει να σημειωθεί πως όλες οι εκδόσεις του ίδιου πόρου θα πρέπει να φέρουν το ίδιο όνομα σε κάθε κατάλογο drawable.

Τώρα, όσον αφορά τους τύπους που σχεδιάσιμων πόρων που χρησιμοποιούμε στο Android, συνήθως είναι αρχεία εικόνων τύποω bitmap, όπως PNG και JPG. Στον παρακάτω πίνακα εμφανίζονται αναλυτικά οι μορφοποιήσεις εικόνων που υποστηρίζονται στο Android.

Μορφοποίηση Εικόνας	Περιγραφή	Επέκταση
Portable Network Graphics	Προτιμώμενη μορφοποίηση, χωρίς απώλειες	.png (PNG)
Nine-Patch Stretchable Images	Προτιμώμενη μορφοποίηση, χωρίς απώλειες	.9.png (PNG)
Joint Photographic Experts Group	Αποδεκτή μορφοποίηση, με απώλειες	.jpg (JPEG/JPG)
Graphics Interchange Format	Μη συνιστώμενη, αλλά υποστηριζόμενη, χωρίς	.gif(GIF)

	απώλειες	

Πίνακας 1-3: **Τύποι σχεδιάσιμων πόρων του** Android

1.3.1.24 ***Αναφορά Πόρων Εικόνων***

Προγραμματιστικά μπορούμε να ενθυλακώσουμε έναν πόρο εικόνας μέσα σε ένα αντικείμενο της κλάσης BitmapDrawable, κλάση η οποία κληρονομεί την android.graphics.drawable.Drawable.

Έτσι, σε μία δραστηριότητα μπορούμε να χρησιμοποιήσουμε την μέθοδο getDrawable( ) ως εξής:

```
BitmapDrawable eikona = (BitmapDrawable) this.getResources().getDrawable(R.drawable.airport);
```

Το παράδειγμα είναι παρμένο από την εφαρμογή μας και όπως παρατηρούμε, πρέπει γίνει "cast" σε BitmapDrawable, μιας και η μέθοδος getDrawable της κλάσης Resources επιστρέφει αντικείμενο Drawable.

1.3.1.25 ***Άλλοι Τύποι Σχεδιάσιμων Πόρων***

Τέλος, αξίζει να αναφέρουμε πως στο Android εκτός των αρχείων γραφικών, μπορούμε να δημιουργήσουμε ειδικά μορφοποιημένα αρχεία XML, για να οποία περιγράψουμε άλλες υποκλάσεις της Drawable, όπως η ShapeDrawable.

Περισσότερες πληροφορίες στην τεκμηρίωση του Android για το πακέτο android.graphics.drawable (http://developer.android.com/reference/android/graphics/drawable/package-summary.html).

1.3.1.26 ***Διατάξεις*** *– Layouts*

Οι διατάξεις είναι ειδικά μορφοποιημένα αρχεία XML τα οποία ορίζουν το πως θα εμφανιστεί ένα τμήμα της οθόνης διεπαφής του χρήστη ή ολόκληρη η οθόνη, μέσα σε μία εφαρμογή Android.

Τις περισσότερες φορές συμπληρώνουμε τους πόρους διατάξεων με διάφορους τύπους μηχανισμών ελέγχου προβολής, όπως TextViews, ImageViews, Buttons κτλ, οι οποίοι με τη σειρά τους αναφέρονται σε άλλους πόρους, π.χ συμβολοσειρές, χρώματα, διαστάσεις κα.

Ένα αρχείο πόρου διάταξης αποθηκεύεται στον φάκελο /res/layout και, όπως συμβαίνει και με τα drawables, υπάρχουν διάφοροι σχετικοί φάκελοι οι οποίοι αποθηκεύουν πόρους διάταξης. Ωστόσο, στην περίπτωση των πόρων διατάξεων οι φάκελοι χωρίζονται, όχι ανάλογα με την πυκνότητα, αλλά με την ανάλυση μίας οθόνης. Έτσι έχουμε:

xlarge οθόνες οι οποίες είναι τουλάχιστον 960dp x 720dp

large οθόνες οι οποίες είναι τουλάχιστον 640dp x 480dp

normal οθόνες οι οποίες είναι τουλάχιστον 470dp x 320dp

small οθόνες οι οποίες είναι τουλάχιστον 426dp x 320dp

Οπότε στην περίπτωση της xlarge οθόνης δημιουργούμε τον σχετικό πόρο διάταξης και τον αποθηκεύσουμε στον φάκελο /res/layout-large. Ο πόρος θα φέρει, βεβαίως, το ίδιο όνομα με αυτόν που είναι αποθηκευμένος στον /res/layout. Στη συνέχεια για large οθόνες έχουμε τον υποφάκελο του /res, /layout-large και τέλος για small οθόνες έχουμε τον υποφάκελο /layout-small.

Σκοπός είναι να πετύχουμε "density independence" στην εφαρμογή μας, ώστε το application να εμφανίζεται περίπου το ίδιο σε οθόνες με διαφορετική πυκνότητα και ανάλυση. Ως αποτέλεσμα αυτού ο χρήστης έχει την ψευδαίσθηση ότι η εφαρμογή δημιουργήθηκε με βάση τα χαρακτηριστικά της δικής του δικής του συσκευής.

Εφόσον η εφαρμογή μας δεν πετύχει το “density independece” οι απώλειες μπορεί να μην είναι σημαντικές, αλλά μπορεί και να αποβούν μοιραίες. Για παράδειγμα θα μπορούσε ένας μηχανισμός ελέγχου, όπως ένα κουμπί, να φαίνεται πολύ μικρό σε μία μεγάλη οθόνη με μεγάλη αριθμό πυκνότητας ή αντιστρόφως να υπερκαλύπτει μία μικρή οθόνη με μικρό αριθμό πυκνότητας.

Οι εικόνες που ακολουθούν βοηθούν στην ευκολότερη κατανόηση του παραπάνω αξιώματος. Στην πρώτη εικόνα φαίνεται μία οθόνη εφαρμογής η οποία δεν υποστηρίζει διαφορετικές πυκνότητες, ενώ στην δεύτερη μία που υποστηρίζει.

*1.3.1.27 **Εργασία με Πόρους Διατάξεων***

Στην πράξη, οι διατάξεις μεταγλωττίζονται μέσα στο πακέτο της εφαρμογής ως πόροι XML ή δημιουργούνται κατά τον χρόνο εκτέλεσης, μέσα στην Activity, μέσω των κατάλληλων κλάσεων διάταξης που παρέχει το Android SDK. Βέβαια στις περισσότερες των περιπτώσεων χρησιμοποιούνται τα αρχεία πόρων διάταξης XML, τα οποία βελτιώνουν κατά πολύ τη σαφήνεια, το ευανάγνωστο και τη δυνατότητα επαναχρησιμοποίησης κώδικα.

Ας δώσουμε ένα παράδειγμα τμήματος ενός αρχείου πόρου διάταξης, παρμένο από την KalymnosGuide:

```
<LinearLayout

android:id="@+id/linearlayout1_imageviews_mainmenu"

android:layout_width="match_parent"

android:layout_height="wrap_content"
```

```
android:layout_alignParentTop="true"
android:layout_marginTop="45dp"
android:orientation="horizontal"
android:weightSum="2" >
<ImageView
android:id="@+id/imageview1_mainmenu"
android:layout_width="150dp"
android:layout_height="150dp"
android:layout_weight="1"
android:src="@drawable/eatanddrink" />
<ImageView
android:id="@+id/imageview2_mainmenu"
android:layout_width="150dp"
android:layout_height="150dp"
android:layout_weight="1"
android:src="@drawable/placestostay" />
</LinearLayout>
```

Το τμήμα του άνω πόρου διάταξης ανήκει στο αρχείο mainmenu.xml, το οποίο βρίσκεται στον /res/layout και είναι υπεύθυνο για την περιγραφή της διάταξη της δραστηριότητας MainMenu.java.

Ο μηχανισμός ελέγχου LinearLayout χρησιμοποιείται ως υποδοχέας για όλους τους εσωτερικούς μηχανισμούς ελέγχου. Εδώ περιέχει δύο ImageView, ωστόσο θα μπορούσε να περιέχει περισσότερους και διαφορετικούς μηχανισμούς ελέγχου, όπως ένας TextView ή ακόμα και άλλον έναν LinearLayout.

Ακόμη βλέπουμε πως εσωτερικά του αρχείου διάταξης mainmenu.xml γίνονται αναφορές σε άλλους πόρους, όπως η αναφορά στο αρχείο εικόνας eatanddrink (@drawable/eatanddrink).

1.3.1.28 Layout Resource Editor

Χρησιμοποιώντας τον επεξεργαστή πόρων διάταξης (layout resource editor) του Eclipse IDE έχουμε τη δυνατότητα να σχεδιάσουμε μεταγλωττισμένους πόρους διάταξης καθώς και να κάνουμε προεπισκόπιση.

Ο Layout Resource Editor του Eclipse έχει δύο καρτέλες:

Η πρώτη είναι η Graphical Layout, η οποία παρέχει την δυνατότητα οπτικής σχεδίασης με τη γνωστή μέθοδο μεταφοράς και απόθεσης (drag n drop), ενώ η δεύτερη είναι ο επεξεργαστής του αρχείου .xml. Δηλαδή αν το αρχείο ονομάζεται mainmenu.xml τότε η δεύτερη καρτέλα φέρει το ίδιο όνομα και επιτρέπει την απευθείας επεξεργασία του ΧΜΛ της διάταξης.

Σε αυτό το σημείο κάποιος που δεν γνωρίζει πως να επεξεργαστεί απευθείας το αρχείο xml δεν θα πρέπει να ανακουφιστεί λόγω του Layout Resource Editor καθώς, βάση και της δικής μας εμπειρίας, τις περισσότερες φορές θα πρέπει να εναλλασσόμαστε μεταξύ του graphical layout και του επεξεργαστή του xml αρχείου.

Στην πραγματικότητα η επεξεργασία της καρτέλας graphical layout κάνει ακριβώς την ίδια δουλειά με τον επεξεργαστή xml, καθώς όποια ενέργεια εκτελέσουμε στην λειτουργία των γραφικών οι ανάλογες αλλαγές θα υπάρχουν και στον κώδικα του αρχείου xml. Ωστόσο, όσον αφορά την προεπισκόπιση στην καρτέλα graphical layout θα πρέπει να σημειώσουμε πως δεν είναι ποτέ ακριβής. Θα πρέπει πάντα να γίνονται δοκιμές σε μία πραγματική συσκευή, ή , έστω σε μία εικονική συσκευή (virtual device) που παρέχει το Eclipse.

Η λογική είναι η ίδια και για το Android Studio με πολύ μικρές διαφορές. Για παράδειγμα στο νέο IDE έχουμε “Design” αντί του Graphical Layout του Eclipse και την καρτέλα “Text” αντί της “.xml”.

*1.3.1.29 **Εργασία με Αρχεία***

Ένα έργο του Android μπορεί να περιέχει και αρχεία ως πόρους, εκτός τους πόρους συμβολοσειράς, γραφικών, διάταξης και τα λοιπά. Αυτά τα αρχεία μπορούν να έχουν οποιαδήποτε μορφοποίηση.

1.3.1.29.1 Αρχεία XML

Αυθαίρετα αρχεία XML μπορούν να περιλαμβάνονται ως πόροι στο Android. Αυτοί οι πόροι αποθηκεύονται στον κατάλογο /res/xml και είναι η προτιμώμενη μορφοποίηση για ποιαδήποτε δομημένα δεδομένα απαιτεί η εφαρμογή.

Για να προσπελάσουμε προφραμματιστικά μέσα σε μία δραστηριότητα ένα αρχείο πόρου xml, έστω το /res/xml/archive.xml, θα εργαστούμε ως εξής:

```
XmlResourceParser antikeimeno = this.getResources( ).getXml(R.xml.archive);
```

Για όσους γνωρίζουν xml, αφού προσπελαστεί παραπάνω αντικείμενο της XmlResourceParser, μπορεί να αναλυθεί η XML και να εξάγουμε τα κατάλληλα στοιχεία δεδομένων.

1.3.1.29.2 Ακατέργαστα αρχεία

Ως πόροι μπορούν να συμπεριληφθούν και ακατέργαστα αρχεία σε μία εφαρμογή. Αρχεία ήχου, βίντεο και κάθε άλλη μορφοποίηση που μπορεί να φανεί χρήσιμη για την εφαρμογή.

Τα ακατέργαστα αρχεία αποθηκεύονται στον κατάλογο πόρων /res/raw, ωστόσο αν πρόκειται να περιλάβομε πόρους αρχείων μέσων θα πρέπει πρώτα να συμβουλευθούμε την τεκμιρίωση του Android προκειμένου να καθοριστούν οι μορφοποιήσεις μέσων και οι κωδικοποιήσεις που υποστηρίζονται από στις συσκευές στόχου της εφαρμογής. Η τοποθεσία http://developer.android.com/guide/appendix/media-formats.html δίνει μία γενική λίστα των υποστηριζόμενων μορφοποιήσεων για συσκευές Android.

Όσον αφορά την προσπέλαση ενός πόρου ακατέργαστου αρχείου προγραμματιστικά, πάντα μέσα από μία δραστηριότητα, αυτή είναι επιτυχής κάνοντας χρήση της μεθόδου openRawResource( ) της κλάσης Resources. Έστω ότι έχουμε το αρχείο /res/raw/mousiki.mp3.

```
InputStream file = this.getResources( ).openRawResource(R.raw.mousiki);
```

*1.3.1.30 **Άλλοι Τύποι Πόρων***

Εκτός των παραπάνω αναφερθέντων τύπων πόρων υπάρχουν και άλλοι διαθέσιμοι τύποι οι οποίοι χρησιμοποιούνται λιγότερο. Μερικοί από αυτούς είναι οι παρακάτω:

- Γενικοί(τιμές boolean και ακέραιοι)
- Πίνακες(πίνακες συμβολοσειρών, πίνακες ακεραίων κτλ)
- Μενού
- Ακολουθίες κίνησης
- Σχεδιάσιμα σχήματα
- Στυλ και θέματα
- Μηχανισμοί ελέγχου προσαρμοσμένων διατάξεων

Αναλυτικά για όλους τους τύπους πόρων στην ιστοθέση http://developer.android.com/guide/topics/resources/available-resources.html.

Το **Αρχείο M**anifest

Κάθε εφαρμογή Android περιλαμβάνει ένα ειδικό αρχείο που ονομάζεται manifest. Το λειγουργικό χρησιμοποιεί το AndroidManifest.xml, το οποίο βρίσκεται μέσα στο πακέτο της εφαρμογής, για να καθορίσει διάφορες ρυθμίσεις διαμόρφωσης της εφαμοργής, οι οποίες περιλαμβάνουν την ταυτότητας της και τις άδειες (permissions) που απαιτεί η εφαρμογή για να εκτελεστεί.

1.3.1.31 AndroidManifest.xml

Όπως διαπιστώνουμε και από την σχετική κατάλληξη του αρχείου, το AndroidManifest.xml είναι ένα αρχείο XML το οποίο:

1. εγκαθιστά και ενημερώνει το πακέτο της εφαρμογής
2. εμφανίζει λεπτομέρειες της εφαρμογής στους χρήστες
3. εκκινεί τις δραστηριότητες της εφαρμογής
4. διαχειρίζεται τις άδειες της εφαρμογής
5. χειρίζεται αρκετές άλλες προχωρημένες διαμορφώσεις της εφαρμογής

Επεγεργαστής Πόρου Manifest του Eclipse

Το αρχείο manifest μπορούμε να το επεξεργαστούμε απευθείας από την XML ή χρησιμοποιώντας τον ειδικό επεξεργαστή πόρου του αρχείου manifest που παρέχει το Eclipse, το IDE που χρησιμοποιήθηκε για την κατασκευή της KalymnosGuide. Στο Android Studio διαφέρουν τα πράγματα.

Όσον αφορά τον επεξεργαστή πόρου του AndroidManifest.xml στο Eclipse, αυτός χωρίζεται σε πέντε καρτέλες: Την καρτέλα Manifest, την Application, την Permissions, την καρτέλα Instrumentation και τέλος την AndroidManifest.xml.

*1.3.1.32 **Καρτέλα** Manifest*

Η καρτέλα Manifest παρέχει τις ρυθμίσεις επιπέδου πακέτου της εφαρμογής, όπως το όνομα του πακέτου, τις πληροφορίες έκδοσης κτλ.

*1.3.1.33 **Καρτέλα** Application*

Η καρτέλα Application παρέχει τις ρυθμίσεις επιπέδου εφαρμογής, όπως την ετικέτα καιτο εικονίδιο της εφαρμογής, καθώς και πληροφορίες για τα συστατικά του application, όπως οι activities, τα intent-filters και διάφορες άλλες λειτουργικότητες τις εφαρμογής.

*1.3.1.34 **Καρτέλα** Permissions*

Η καρτέλα Permissions περιέχει τους τυχόν κανόνες αδειών που απαιτούνται για την εγκατάσταση της εφαρμογής από τον χρήστη. Ακόμη η συγκεκριμένη καρτέλα μπορεί να επιβάλει προσαρμοσμένες άδειες, που δημιουργούνται ειδικά για την εφαρμογή.

Σε αυτό το σημείο θα πρέπει να διαφοροποιηθεί η καρτέλα Permissions του manifest με την αναδυόμενη λίστα Permissions της καρτέλας Application του αρχείου, μιας και υπάρχει κίνδυνος σύγχησης. Η καρτέλα χρησιμοποιείται για να προσπελάσει η εφαρμογή τους πόρους ή τα API που χρειάζεται, ενώ η αναδυόμενη λίστα χρησιμοποιείται για να ορίσει

άδειες που απαιτούντανι από άλλες εφαρμογές, προκειμένου να προσπελάσουν εκτιθέμενους πόρους και API της εφαρμογής μας.

*1.3.1.35 **Καρτέλα** Instumentation*

Η καρτέλα Instrumentation χρησιμοποιείται για την δήλωση τυχόν κλάσεων οργάνων μέτρησης και για παρακολούθηση της εφαρμογής

Καρτέλα AndroidManifest.xml

Εδώ είναι η περιοχή όπου μπορούμε να επεξεργαστούμε απευθείας το αρχείο manifest, μέσα από τον επεξεργαστή πόρων του Eclipse. Ουσιαστικά οι παραπάνω καρτέλες αποτελούν έναν πιο ευέλικτο τρόπο για να πραγματοποιηθούν οι αλλαγές εδώ μέσα, στην καρτέλα AndroidManifest.xml.

*1.3.1.36 **Λειτουργίες του αρχείου** AndroidManifest.xml*

Ήρθε η ώρα να αναλύσουμε το περιβόητο αρχείο manifest της εφαρμογής, δανειζόμενοι κώδικα από το AndroidManifest.xml της εφαρμογής KalymnosGuide.

Σε ότι αφορά τον ορισμό της xml, αυτός αρχίζει πάντα με μία επικεφαλίδα όπως παρακάτω:

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

Στη συνέχεια ακολουθεί η σήμανση η οποία <manifest> η οποία εμπεριέχει την εξίσου σημαντική σήμανση <application>. Πολλές από τις σημαντικές ρυθμίσεις που απαιτεί η εφαρμογή μας τίθενται χρησιμοποιώντας χαρακτηριστικά και θυγατρικές σημάνσεις των δύο παραπάνω μπλοκ, όπως:

Ονομασία Πακέτων Android

Η ονομασία του πακέτου Android αναφέρεται ως χαρακτηριστικό της σήμανσης <manifest>, μέσα στο αρχείο AndroidManifest.xml.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.Skemelio.kalymnosguide"

android:versionCode="1"

android:versionName="1.0" >
```

*1.3.1.37 **Εκδόσεις Εφαρμογής***

Στο Android χρησιμοποιούμε τις εκδόσεις μιας εφαρμογής για οργάνωση και παρακολούθηση των χαρακτηριστικών της, καθώς επίσης και για τη διαχείριση των αναβαθμίσεων της εφαρμογής. Αυτός είναι και ο λόγος που η σήμανση <manifest> περιέχει δύο ξεχωριστά χαρακτηριστικά έκδοσης: Έναν κωδικό έκδοσης και ένα όνομα έκδοσης.

1.3.1.37.1 Καθορισμός ονόματος έκδοσης

Πρόκειται για το χαρακτηριστικό android:versionName="1.0" της <manifest>. Πρόκειται για τις παραδοσιακές πληροφορίες έκδοσης, που χρησιμοποιούνται για την παρακολούθηση των δομήσεων της εφαρμογής.

Για να έχει ουσία η έκδοση του ονόματος θα πρέπει να υποστηρίζουμε και αναβαθμίζουμε την εφαρμογή μας, μιας και αυτό το το πεδίο είναι ορατό στον χρήστη.

1.3.1.37.2 Καθορισμός κωδικού έκδοσης

Το χαρακτηριστικό android:versionCode="1" της σήμανσης <manifest> χρησιμοποιείται από το Android, αλλά και από τις θέσεις αγορών του Android, προκειμένου να διαχειριστούν αναβαθμίσεις και υποβαθμίσεις εφαρμογών.

Η τιμή του ξεκινά πάντα από το 1 και αυξάνεται κατά ένα με κάθε νέα έκδοση της εφαρμογής, χωρίς ωστόσο να εμφανίζεται στους χρήστες.

1.3.1.37.3 Minimum SDK version και Target SDK version

Χρησιμοποιούμε την σήμανση <uses-sdk>, η οποία βρίσκεται μέσα στην <manifest>, προκειμένου να καθορίσουμε το ελάχιστο SDK που απαιτείται να έχει μία συσκευή για "τρέξει" σωστά την εφαρμογή που κατασκευάσαμε.

```
<uses-sdk

android:minSdkVersion="14"

android:targetSdkVersion="18" />
```

Το χαρακτηριστικό minSdkVersion καθορίζει τα προαναφερθέντα, ενώ το targetSdkVersion την έκδοση του Android SDK της οποία έχουμε θέσει ως στόχο.

1.3.1.37.4 Ονομασία Εφαρμογής

Η ονομασία μίας εφαρμογής Android καθορίζεται μέσα στο χαρακτηριστικό android:label="@string/app_name" της σήμανσης <application>.

Όπως παρατηρούμε, το όνομα ορίζεται ως μία συμβολοσειρά, στην προκειμένη περίπτωση ως ενός πόρου συμβολοσειράς.

1.3.1.37.5 Εικονίδιο Εφαρμογής

Το εικονίδιο της εφαρμογής καθορίζεται από το χαρακτηριστικό android:icon="@drawable/kalymnos_logo" της σήμανσης <application> του αρχείου manifest της εφαρμογής.

1.3.1.37.6 Περιγραφή Εφαρμογής

Η συνοπτική περιγραφή μιας εφαρμογής Android παριστάται από μία συμβολοσειρά η οποία θα χρησιμοποιηθεί από το χαρακτηριστικό android:label="@string/description" της σήμανσης <application>.

Η KalymnosGuide δεν διαθέτει κάποια περιγραφή.

1.3.1.38 Δήλωση Δραστηριοτήτων στο Manifest

Το λειτουργικό σύστημα Android αναγνωρίζει μία κλάση ως δραστηριότητα μόνον στην περίπτωση που αυτή έχει καταχωρηθεί μέσα στο αρχείο manifest με το όνομα της κλάσης της, πριν μπορέσει να εκτελεστεί στην συσκευή. Οπότε, κάθε φορά που προστίθεται μία δραστηριότητα θα πρέπει να δηλώνεται αμέσως στο αρχείο manifest.

Τώρα όσον αφορά τον τρόπο καταχώρησης μίας δραστηριότητας στο αρχείο manifest, μπορεί να επιτευχθεί με δύο τρόπους:

Πρώτον, επιλέγοντας την κλάση μέσα από την καρτέλα Application του manifest ή δηλώνοντας την δραστηριότητα απευθείας από την XML, χρησιμοποιώντας την σήμανση <activity>.

Για παράδειγμα η δραστηριότητα MainMenu δηλώθηκε στο manifest της KalymnosGuide ως εξής:

```
<activity android:name="com.Skemelio.kalymnosguide.MainMenu"
android:label="@string/app_name" android:screenOrientation="portrait" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

Η δήλωση της δραστηριότητας MainMenu εμπεριέχει και ένα φίλτρο Intent το οποίο καθορίζει την MainMenu ως σημείο εκκίνησης της εφαρμογής KalymnosGuide.

Το φίλτρο Intent για την εκκίνηση μιας δραστηριότητας προεπιλεγμένα πρέπει να διαμορφωθεί χρησιμοποιώντας μία σήμανση <intent-filter>, με μία <action> MAIN και με <category> LAUNCHER.

1.3.1.39 Άδειες Εφαρμογής

Το Android έχει δομηθεί σε έναν πυρήνα Linux, οπότε χρησιμοποιεί το ενσωματομένο σύστημα ασφάλειας ως τμήμα του μοντέλου ασφάλειας του.

Κάθε μία από τις εφαρμογές του Android υπάρχει μέσα στην δική της εικονική μηχανή και λειτουργεί μέσα στον δικό της λογαριασμό χρήστη Linux. Έτσι αν μία εφαρμογή θέλει να έχει πρόσβαση σε κοινόχρηστους και προνομιούχους πόρους της εκάστοτε συσκευής θα πρέπει να έχει δηλώσει τις συγκεκριμένες άδειες μέσα στο manifest.

Είναι προφανές πως μία εφαρμογή δεν χρειάζεται καμία άδεια για να προσπελάσει τα δικά της ιδιωτικά αρχεία, ωστόσο χρειάζεται άδειες για να να προσπελάσει κοινόχρηστους ή ειδικούς πόρους, χρησιμοποιώντας την σήμανση <uses-permission> στο manifest. Ακριβώς η ίδια εργασία μπορεί να γίνει ευκολότερα στην καρτέλα Permissions του επεγεργαστή

πόρου του αρχείου manifest, όπως είχαμε αναφέρει στην περιγραφή της σχετικής καρτέλας.

Η παρακάτω άδεια επιτρέπει την προσπέλαση υλικού της συσκευής από την εφαρμογή, στην συγκεκριμένη περίπτωση την κάμερα:

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

1.4 Χρήση API υπηρεσιών θέσης (google maps – geolocation)

Μία από τις πιο ενδιαφέρουσες δυνατότητες του Android είναι η χρήση των υπηρεσιών θέσης, γεγονός που εύκολα γίνεται αντηλιπτό ρίχνοντας μία ματιά στις δημοφιλέστερες εφαρμογές της πλατφόρμας, διαπιστώνοντας την ευρεία χρήση τους.

Ως χρήστες λοιπόν, λίγο-πολύ, όλοι γνωρίζουμε την δημοφιλέστερη εφαρμογή χαρτών, τους Google Maps, όπως επίσης όλοι έχουμε παρατηρήσει εφαρμογές, όπως για παράδειγμα εφαρμογές κοινωνικής δικτύωσης, οι οποίες παρακολουθούν τη θέση του χρήστη. Στη συγκεκριμένη υποενότητα αφήνουμε τη σκοπιά του χρήστη και κοιτάμε με τα μάτια του developer.

Σαν προγραμματιστές λοιπόν, θα πρέπει, καταρχήν, να αναλογιστούμε τη σημαντικότητα της γνώσης της θέσης του χρήστη. Γνωρίζοντας την είμαστε ικανοί να δημιουργήσουμε εξυπνότερες εφαρμογές οι οποίες θα προσφέρουν ποιοτικότερες υπηρεσίες και θα προσφέρουν καλύτερες και περισσότερες πληροφορίες. Υλοποιώντας τις λεγόμενες Location Services σε συνδυασμό με την κορυφαία εφαρμογή Google Maps μπορούμε να πετύχουμε εκπληκτικά πράγματα.

Η χρήση των παραπάνω δυνατοτήτων μέσα σε μία εφαρμογή επιτυγχάνεται χρησιμοποιώντας τις κλάσεις του πακέτου android.location (μεταγενέστερα το Google Play services location APIs) και φυσικά το Google Maps Android API.

Location Services

Οι Location Services επιτρέπουν στις εφαρμογές να λαμβάνουν περιοδικές ενημερώσεις της γεωγραφικής θέσης της συσκευής, ή να στείλουν μία πρόθεση Intent όταν η συσκευή εισέρχεται στην εγγύτητα μιας δεδομένης γεωγραφικής θέσης.

Όπως αναφέραμε προηγουμένως οι εφαρμογές Android έχουν πρόσβαση στις υπηρεσίες θέσης μέσα από τις κλάσεις του πακέτου android.location.

Ξεκινάμε την περιήγηση μας στο πακέτο με την κεντρική κλάση LocationManager η οποία είναι αυτή που δίνει πρόσβαση στις λεγόμενες “system location services”. Η κλάση LocationManager δεν αρχικοποιείται με την παραδοσιακή μέθοδο, καλώντας έναν δημιουργό (constructor). Το σύστημα θα δημιουργήσει για εμάς ένα νέο στιγμιότυπο της κλάσης μέσω της μεθόδου getSystemService(Context.LOCATION_SERVICE). Αυτή η σύνταξη της μεθόδου getSystemService επιστρέφει ένα νέο στιγμιότυπο του LocationManager. Υποθέτουμε ότι η εργασία γίνεται μέσα σε μία δραστηριότητα:

```
LocationManager locMgr = this.getSystemService(Context.LOCATION_SERVICE);
```

Από τη στιγμή που η εφαρμογή μας διαθέτει έναν LocationManager μπορεί να κάνει τα εξής τρία:

1. Να ζητήσει μία λίστα με όλους τους διαθέσιμους LocationProviders με σκοπό τη γνώση της τελευταίας γνωστής θέσης του χρήστη.

2. Να ειδοποιείται για περιοδικές ενημερώσεις της τρέχουσας θέσης του χρήστη από έναν πάροχο

3. Να ειδοποιείται με μία πρόθεση Intent όταν η συσκευή εισέλθει στην εγγύτητα μιας δεδομένης γεωγραφικής θέσης

Πέραν της LocationManager το πακέτο android.Location διαθέττει δύο ακόμα πολύ σημαντικές κλάσεις, την LocationListener και την Location.

Το interface LocationListener χρησιμοποιείται με σκοπό την παροχή ενημερώσεων αλλαγής θέσης μέσω του LocationManager.

Η κλάση Location απλώς αναπαριστά μία γεωγραφική τοποθεσία, η οποία εμπεριέχει πληροφορίες όπως το γεωγραφικό της ύψος, γεωγραφικό πλάτος, μήκος και άλλα.

Ο παρακάτω κώδικας μας φανερώνει τον τρόπο που ορίζουμε έναν LocationListener και πως ζητάμε location updates:

```
// Ξεκινάμε δημιουργώντας ένα στιγμιότυπο του LocationManager

LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);


// Ορίζουμε έναν listener ο οποίος θα ακροάζεται τα location updates

LocationListener locationListener = new LocationListener() {

  public void onLocationChanged(Location location) {

    // Καλείται όταν βρεθεί μία νέα location από τον network location provider

    makeUseOfNewLocation(location);

  }


  public void onStatusChanged(String provider, int status, Bundle extras) {}


  public void onProviderEnabled(String provider) {}
```

```
  public void onProviderDisabled(String provider) {}
};

// Register τον listener στον Location Manager για να ζητήσει location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
```

Last Known Location

Τις περισσότερες φορές η τρέχουσα θέση του χρήστη είναι η ίδια με την τελευταία γνωστή θέση της συσκευής. Επομένος το μόνο που θα χρειαστεί από εμάς είναι η λήψη της λεγόμενης last known location (τελευταίας γνωστής τοποθεσίας)!

Μία εφαρμογή Android μπορεί να ζητήσει την τελευταία γνωστή θέση της συσκευής του χρήστη χρησιμοποιώντας τα Google Play services location APIs. Πιο συγκεκριμένα θα χρησιμοποιήσουμε το "fused location provider" για να λάβουμε την τελευταία γνωστή τοποθεσία.

Ο fused location provider είναι ένα από τα location APIs του Google Play Services. Διαχειρίζεται την υποκείμενη τεχνολογία εντοπισμού θέσης και παρέχει ένα απλό API έτσι ώστε να μπορούμε να καθορίσουμε τις απαιτήσεις μας σε υψηλό επίπεδο. Παράδειγμα των απαιτήσεων αυτών είναι η υψηλή ακρίβεια της θέσης ή κατανάλωση χαμηλής ισχύος της μπαταρίας κτλ.

Θα πρέπει να γίνουν τα εξής τέσσερα βήματα για να λάβουμε την τελευταία γνωστή τοποθεσία μέσω του fused location provider.

1. Εγκατάσταση του Google Play Services
2. Ορισμός Αδειών Εφαρμογής
3. Σύνδεση στο Google Play Services
4. Λήψη της τελευταίας γνωστής τοποθεσίας

*1.4.1.1 **Εγκατάσταση του** Google Play Services*

Για να έχουμε πρόσβαση στον fused location provider το project της εφαρμογής μας θα πρέπει να περιλαμβάνει τις Google Play Services. Τις "κατεβάζουμε" και τις εγκαθιστούμε μέσω του SDK Manager και συνδέουμε τη σχετική βιβλιοθήκη με το έργο μας.

Όσον αφορά το Eclipse, το περιβάλλον ανάπτυξης της δικής μας εφαρμογής, η εγκατάσταση των Google Play Services περιλαμβάνει τα παρακάτω βήματα:

1. Άνοιγμα του SDK manager
2. Κατέβασμα των Google Play Services μέσω του SDK manager

3. Import του πρότζεκτ της βιβλιοθήκης ...\adt-bundle-windows-x86_64\sdk\extras\google\google_play_services\libproject

4. Σύνδεση του πρότζεκτ της εφαρμογής με τη βιβλιοθήκη

5. Μετάβαση στο αρχείο manifest της εφαρμογής και επικόληση του

"<meta-data android:name="com.google.android.gms.version"

android:value="@integer/google_play_services_version" />"

μέσα στο <application>

*1.4.1.2 **Ορισμός Αδειών Εφαρμογής***

Οι εφαρμογές που χρησιμοποιούν τις υπηρεσίες τοποθεσίας θα πρέπει να φέρουν και τις συγκεκριμένες άδειες.

Η πλατφόρμα του Android προσφέρει δύο από αυτές για την συγκεκριμένη περίπτωση, την ACCESS_COURSE_LOCATION και την ACCESS_FINE_LOCATION. Η πρώτη άδεια παρέχει τη δυνατότητα σε μία εφαρμογή να λάβει την τοποθεσία μέσω πύργων κινητής τηλεφωνίας και Wi-Fi, ενώ η δεύτερη περιλαμβάνει και GPS.

Σημείωση: *Κατά τη διάρκεια της ανάπτυξης μίας "location-aware" εφαρμογής στο Android, έχουμε τη δυνατότητα να χρησιμοποιήσουμε είτε το GPS, εφόσον διαθέτει η συσκευή, είτε το Android Network Location Provider για να λάβουμε την τοποθεσία του χρήστη. Ή και τα δύο!*

Οι διαφορές ανάμεσα στο GPS και το το Android Network Location Provider είναι σημαντικές. Το GPS παρέχει ακριβέστερα αποτελέσματα, όμως λειτουργεί μόνο σε εξωτερικούς χώρους και "καταπίνει" την μπαταρία της συσκευής. Ένα ακόμη μειονέκτημα του GPS είναι πως δεν επιστρέφει την τοποθεσά τόσο γρήγορα όσο θα ήθελε ο χρήστης.

Από την άλλη πλευρά, το Android Network Location Provider καθορίζει τη θέση του χρήστη χρησιμοποιώντας πύργους κινητής τηλεφωνίας και σήματα Wi-Fi, παρέχοντας έτσι πληροφορίες για τη θέση του χρήστη σε εσωτερικούς και εξωτερικούς χώρους, ανταποκρινόμενο ταχύτερα και χρησιμοποιώντας λιγότερη ενέργεια μπαταρίας. Επομένως το εργαλείο που θα είναι υπεύθυνο για τη λήψη της τοποθεσίας του χρήστη θα επιλεχθεί από τον ίδιο τον προγραμματιστή και σύμφωνα με τα δικά του κριτήρια.

Η αίτηση για τις παραπάνω άδειες γίνεται χρησιμοποιώντας το στοιχείο uses-permission στο manifest της εφαρμογής, όπως παρακάτω:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.google.android.gms.location.sample.basiclocationsample" >
                .....
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
                .....
```

```
</manifest>
```

*1.4.1.3 **Σύνδεση στο** Google Play Services*

Προκειμένου να συνδεθούμε στο Google Play Services API θα χρειαστούμε ένα στιγμιότυπο του Google Play Services API Client. Αυτή η εργασία θα πρέπει να λάβει χώρα μέσα στην μέθοδο onCreate( ) της κλάσης Activity.

Το στιγμιότυπο που θα δημιουργηθεί θα είναι αντικείμενο της κλάσης (interface για την ακρίβεια) GoogleApiClient και για την αρχικοποίηση του θα πρέπει να κάνουμε χρήση της κλάσης GoogleApiClient.Builder.

Πριν περάσουμε όμως στην υλοποίηση της παραπάνω εργασίας θα πρέπει να σημειωθεί πως η κλάση της δραστηριότητας θα υλοποιεί δύο απαραίτητα interface. Το GoogleApiClient.ConnectionCallBacks και το GoogleApiClient.onConnectionFailedListener.

```
public class MainActivity extends FragmentActivity implements GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener
```

Τα πρώτο interface ακροάζεται τη σύνδεση ή την αποσύνδεση του client από την υπηρεσία, ενώ το δεύτερο ακροάζεται το σενάριο αποτυχίας σύνδεσης στην υπηρεσία.

1.4.1.3.1 Αρχικοποίηση του αντικειμένου GoogleApiClient

Τώρα περνάμε στον κώδικα που θα μας δώσει την αρχικοποίηση του αντικειμένου GoogleApiClient, το οποίο είναι απαραίτητο για να συνδεθούμε στο Google Play Services API:

```
protected synchronized void buildGoogleApiClient() {

        mGoogleApiClient = new GoogleApiClient.Builder(this)

          .addConnectionCallbacks(this)

          .addOnConnectionFailedListener(this)

          .addApi(LocationServices.API)

          .build();

        Toast.makeText(this, "Connected to Google Play Services", Toast.LENGTH_SHORT).show();

    }
```

Ξεκινώντας την μελέτη του παραπάνω τμήματος κώδικα θα πρέπει να σημειωθεί πως η αναφορά του αντικειμένου mGoogleApiClient έχει οριστεί μέσα στην κλάση Activity, μαζί με τα άλλα fields της. Ακόμη το μήνυμα Toast χρησιμοποιήθηκε για λόγους debbuging.

Για να αρχικοποιήσουμε λοιπόν το νέο αντικείμενο GoogleApiClient θα πρέπει να κάνουμε χρήση της κλάσης GoogleApiClient.Builder. Η τελευταία θα επιστρέψει ένα αντικείμενο GoogleApiClient στην περίπτωση που καλέσει την μέθοδο build( ).

Η addConnectionCallbacks( ) προσθέτει έναν listener ο οποίος θα ακροάζεται την σύνδεση και την αποσύνδεση του client από την υπηρεσία και στην προκειμένη περίπτωση μπαίνει η παράμετρος this, καταδεικνύοντας το GoogleApiClient.ConnectionCallBacks ως το υπεύθυνο interface για το σκοπό αυτό και φυσικά επειδή υλοποιείται (implements) από την κλάση δραστηριότητας. Παρόμοια και με την addOnConnectionFailedListener( ).

Στο τέλος καλούμε την build( ) και έτοιμη η αρχικοποίηση του mGooglePlayClient.

1.4.1.4 Σύνδεση στην υπηρεσία

Εφόσον έχουμε ένα αντικείμενο GoogleApiClient μπορούμε να συνδεθούμε στο Google Play Services, μέσω της μεθόδου connect( ) του αντικειμένου:

```
//Δημιουργία του αντικειμένου GoogleApiClient

this.buildGoogleApiClient();

//Σύνδεση στην υπηρεσία

this.mGoogleApiClient.connect();
```

1.4.1.5 Λήψη της τελευταίας γνωστής θέσης

Έχουμε συνδεθεί στο Google Play Services και ο τελικός σκοπός μας είναι η λήψη της τελευταίς γνωστής τοποθεσίας του χρήστη, η οποία όπως προαναφέραμε στις περισσότερες των περιπτώσεων ισοδυναμεί με την πραγματική θέση του χρήστη στον χάρτη.

Η λήψη της τελευταίας γνωστή θέσης θα γίνει με τη χρήση της μεθόδου getLastKnowLocation( ) του fused location provider (interface FusedLocationProviderApi).

Η εργασία αυτή θα λάβει χώρα στην μέθοδο onConnected( ), η οποία αποτελεί μέρος της κλάσης δραστηριότητας χάρη στο GoogleApiClient.ConnectionCallBacks interface.

Αναλυτικά:

```
@Override

        public void onConnected(Bundle connectionHint) {

                Location                mLastLocation                =
LocationServices.FusedLocationApi.getLastLocation(this.mGoogleApiClient);

        }
```

Σε αυτό το σημείο η τελευταία γνωστή τοποθεσία είναι το αντικείμενο mLastLocation. Εμείς μπορούμε να αξιοποιήσουμε τις ιδιότητες και τις μεθόδους της κλάσης Location προς δικό μας όφελος. Για παράδειγμα, έστω ότι η εφαρμογή μας υλοποιούσε έναν Google Map. Θα μπορούσαμε να μετακινήσουμε τον χάρτη στο σημείο της τελευταίας γνωστής τοποθεσίας,

επειδή είμαστε σε θέση να λάβουμε το γεωγραφικό πλάτος, το γεωγραφικό μήκος και άλλες πληροφορίες ως εξής:

```
double latitude = mLastLocation.getLatitude();

double longitude = mLastLocation.getLongitude();
```

Ο τρόπος που θα γινόταν αυτή η εργασία αφορά το επόμενο ζήτημα, τους Google Maps.

Google Maps Android API

Οι χάρτες της Google αποτελούν αναμφισβήτητα μία από τις πιο εντυπωσιακές και δημοφιλής εφαρμογές της εταιρείας. Ωστόσο εδώ δεν θα εστιάσουμε στην εφαρμογή Google Maps, αλλά στο Google Maps Android API και συγκεκριμένα στη νεότερη έκδοση του, το Google Maps Android API v2. Η εφαρμογή KalymnosGuide κάνει χρήση του Google Maps Android API V2, εκμεταλλευόμενη μερικά από τα χαρακτηριστικά του.

Με το Google Maps Android API έχουμε τη δυνατότητα να προσθέσουμε χάρτες σε μία εφαρμογή οι οποίοι, όπως είναι λογικό, αντλούν δεδομένα από τους Google Maps. Το API διαχειρίζεται αυτόματα την πρόσβαση στους servers των χαρτών της Google, το κατέβασμα των δεδομένων (data downloading), την απεικόνιση του χάρτη, και τα συμβάντα αγγίγματος σε αυτόν.

Ακόμη, χρησιμοποιώντας το εν λόγω API μπορούμε να προσθέσουμε markers στο χάρτη, να σχεδιάσουμε γραμμές, πολύγωνα, να αλλάξουμε τον τύπο του χάρτη και άλλα.

Η κλάση κλειδί για τα Google Maps APIs είναι η MapView, ωστόσο εμείς στην εφαρμογή KalymnosGuide χρησιμοποιήσαμε fragment και συγκεκριμένα την κλάση MapFragment ή την SupportMapFragment, οι οποίες κληρονομούν την Fragment.

Τέλος, η δραστηριότητα η οποία θα υλοποιεί έναν χάρτη θα πρέπει να κληρονομεί την κλάση FragmentActivity, η οποία με τη σειρά της κληρονομεί την Activity.

Πριν προχωρήσουμε στην περεταίρω ανάλυση θα πρέπει να γίνει κατανοητό πως τα Google Maps Android APIs δεν περιλαμβάνονται στην πλατφόρμα του Android. Μπορούμε να τα βρούμε σε κάθε συσκευή η οποία διαθέτει Google Play Store και τρέχει Android 2.2 ή μεγαλύτερο, μέσω των Google Play Services.

Για να ενσωματώσουμε τους Google Maps μέσα στην εφαρμογή μας θα πρέπει να εγκαταστήσουμε την βιβλιοθήκη των Google Play services.

1.4.1.6 Όροι Χρήσης των Google Maps APIs

Υπάρχουν κάποιοι πολύ σημαντικοί κανόνες που πρέπει να γνωρίζουμε, εφόσον θα χρησιμοποιήσουμε το συγκεκριμένο API.

Το API και οι υπηρεσίες που περιγράψαμε και θα περιγράψουμε είναι δωρεάν, ωστόσο υπάρχουν κάποιοι πολύ σημαντικοί όροι χρήσης που πρέπει να γνωρίζουμε.

Οι όροι χρήσης βρίσκονται συγκεντρωμένοι στην εξής ηλεκτρονική τοποθεσία: https://developers.google.com/maps/terms, ωστόσο θα αναφέρουμε σύντομα μερικά πράγματα που θα πρέπει να γνωρίζει ένας developer.

- Θα πρέπει να διαθέτει λογαριασμό google
- Θα πρέπει να εμφανίσει στην εφαρμογή του μία υποσημείωση σχετικά με τους όρους χρήσης του API (παρέχεται από μία utility class)
- Θα πρέπει να γνωρίζει ότι οι υπηρεσίες είναι δωρεάν μέχρι ενός σημείου, για παράδειγμα μία εφαρμογή μπορεί να ζητήσει 2500 αιτήσεις geocoding την ημέρα
- Θα πρέπει να μην κάνει απλώς ένα re-implement των Google Maps ή του Google Earth
- Θα πρέπει να μην κρύβει διαφημίσεις ή copyright κτλ

*1.4.1.7 **Υλοποίηση του** Google Maps Android API v2*

Απαιτούνται διάφορα βήματα προτού μία εφαρμογή Android χρησιμοποιήσει το Google Maps Android API v2.

1. Κατέβασμα και εγκατάσταση του Google Play services SDK, το οποίο περιλαμβάνει και το Google Maps Android API
2. Απόκτηση ενός API key. Για να αποκτήσουμε ένα κλειδί θα πρέπει να καταγράψουμε το πρότζεκτ της εφαρμογής στην Google APIs Console
3. Προσθήκη ενός χάρτη στην εφαρμογή

1.4.1.7.1 Εγκατάσταση και ρύθμιση του Google Play services SDK

Ακριβώς η ίδια διαδικασία που περιγράφεται παραπάνω στην Last Known Location.

1.4.1.7.2 Απόκτηση ενός API-key

Για να αποκτήσουμε πρόσβαση στους διακομιστές των Google Maps μέσω του Maps API, θα πρέπει να προσθέσουμε ένα API key στην εφαρμογή.

Το κλειδί είναι δωρεάν και μπορούμε να το χρησιμοποιήσουμε σε οποιαδήποτε εφαρμογή η οποία θα χρησιμοποιήσει το Maps API. Επιπροσθέτως το κλειδί υποστηρίζει απεριόριστο αριθμό χρηστών.

Προμηθευόμαστε το Maps API key απο την Google APIs Console δίνοντας την υπογραφή πιστοποιητικού της εφαρμογής (signing certificate) και το όνομα του πακέτου της. Στη συνέχεια θα προσθέσουμε το κλειδί στην εφαρμογή μας τοποθετώντας το στο αρχείο manifest.

1.4.1.7.3 Ακολουθούν τα βήματα για τη λήψη ενός Map API key:

1. Ανάκτηση του πιστοποιητικού της εφαρμογής (signing certificate)
2. Εγγραφή του πρότζεκτ της εφαρμογής στην Google API Console
3. Αίτηση ενός ή περισσοτέρων κλειδιών

4. Προσθήκη του κλειδιού στην εφαρμογή

5. Λοιπές ρυθμίσεις στο AndroidManifest.xml

1.4.1.7.4 Ανάκτηση του πιστοποιητικούτης εφαρμογής

Η κάθε εφαρμογή φέρει μία "ψηφιακή υπογραφή", γνωστή ως "SHA-1 fingerprint". Αυτό το ψηφιακό "δακτυλικό αποτύπωμα" δεν είναι τίποτα άλλο από ένα String το οποίο δημιουργείται από τον αλγόριθμο SHA-1. Κάθε ένα από αυτά τα αποτυπώματα είναι μοναδικό για κάθε εφαρμογή και γι αυτό το λόγο οι Google Maps το χρησιμοποιούν για να την εντοπίσουν.

Για να εμφανίσουμε όμως το σωστό API key θα πρέπει πρώτα να σιγουρευτούμε ότι χρησιμοποιούμε το κατάλληλο πιστοποιητικό. Υπάρχουν δύο ειδών πιστοποιητικά, το debug certificate και το release certificate.

Το debug certificate θα χρησιμοποιηθεί για την ανάκτηση ενός API key το οποίο θα λειτουργεί μόνο για debuging, ενώ όταν θα έρθει η ώρα για τη δημοσίευση της εφαρμογής στο Google Play Store, τότε θα πρέπει να χρησιμοποιήσουμε το release certificate.

1.4.1.7.5 Ανάκτηση του debug certificate

Στα Windows η τεκμηρίωση του Android δίνει τον εξής τρόπο ανάκτησης του debug certificate.

Αρχικά εντοπίζουμε το αρχείο debug.keystore το οποίο βρίσκεται στη θέση C:\Users\your_user_name\.android\. Στη συνέχεια ανοίγουμε το debug.keystore το με το πρόγραμμα εντολών των Windows (C:\Windows\System32\cmd.exe) και έπειτα πληκτρολογούμε το "keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android". Στο πέρας αυτής της διαδικασίας θα πρέπει να εμφανιστούν τα εξής:

Alias name: androiddebugkey

Creation date: Jan 01, 2013

Entry type: PrivateKeyEntry

Certificate chain length: 1

Certificate[1]:

Owner: CN=Android Debug, O=Android, C=US

Issuer: CN=Android Debug, O=Android, C=US

Serial number: 4aa9b300

Valid from: Mon Jan 01 08:04:04 UTC 2013 until: Mon Jan 01 18:04:04 PST 2033

Certificate fingerprints:

MD5: AE:9F:95:D0:A6:86:89:BC:A8:70:BA:34:FF:6A:AC:F9

SHA1: BB:0D:AC:74:D3:21:E1:43:07:71:9B:62:90:AF:A1:66:6E:44:5D:75

Signature algorithm name: SHA1withRSA

Version: 3

Στο σημείο αυτό ανακτούμε το αποτύπωμα SHA-1 (αυτό που έχουμε υπογραμμισμένο).

Σε αυτή τη φάση η χρήση του Eclipse έχει ένα πλεονέκτημα, μιας και η διαδικασία ανάκτησης του SHA-1 fingerprint μπορεί να γίνει πολύ ευκολότερη. Αρκεί να επιλέξουν από το μενού Windows > Prefs > Android > Build και το αποτύπωμα SHA-1 εμφανίζεται μπροστά τους. Υπενθυμίζουμε πως θα λειτουργήσει για την ανάκτηση ενός API key το οποίο θα λειτουργεί για debug mode.

1.4.1.7.6 Ανάκτηση του release certificate

Όσον αφορά την ανάκτηση του release certificate τα πράγματα είναι διαφορετικά, μιας και δεν υπάρχει προκαθορισμένο αρχείο .keystore. Θα πρέπει να το δημιουργήσουμε εμείς.

1. Από το μενού επιλέγουμε File > Export > Android Application και στη συνέχεια επιλέγουμε το πρότζεκτ που επιθυμούμε να κάνουμε export και κλικ στο next

2. Σε αυτό το σημείο θα μας ζητηθεί από το παράθυρο διαλόγου η δημιουργία ενός αρχείου keystore, εκτός και αν έχουμε ήδη ένα διαθέσιμο, οπότε και το επιλέγουμε.

3. Στο τελευταίο παράθυρο διαλόγου, ακριβώς ένα βήμα πριν από το export, φαίνονται οι υπογραφές της εφαρμογής

Επομένως, όταν έρθει η ώρα της δημοσίευσης της εφαρμογής μας στο Google Play Store, θα πρέπει να κάνουμε export την εφαρμογή μας. Από αυτή τη διαδικασία θα "αλλιέυσουμε" και το realease certificate.

1.4.1.7.7 Εγγραφή του πρότζεκτ της εφαρμογής στην Google API Console

Ήρθε η ώρα για την εγγραφή της εφαρμογής στο Google API Console. Ακολουθούν τα σχετικά βήματα τα οποία αφορούν τη νέα Google API Console. Η διαδικασία ελάχιστα διαφέρει από την παλαιά κονσόλα:

1. Σε έναν browser ανοίγουμε την Google API Console (https://code.google.com/apis/console). Στην περίπτωση που χρησιμοποιούμε την κονσόλα για πρώτη φορά, θα χρειαστεί να δημιουργήσουμε ένα νέο πρότζεκτ για να παρακολουθούμε την χρήση του.

2. Από τη στιγμή που έχει δημιουργηθεί ένα project η οθόνη της κονσόλας θα μεταβεί αυτόματα στην προεπισκόπιση του. Από τις επιλογές αριστερά εντοπίζουμε την κατηγορία APIs & auth και την επιλέγουμε. Στη συνέχεια θα εμφανιστεί ένα αναδυόμενο μενού με επιλογές. Επιλέγουμε την APIs

3. Σε αυτό το σημείο στην οθόνη μας εμφανίζονται τα διαθέσιμα APIs που παρέχονται από την Google Console στο project μας. Μας ενδιαφέρει το Google Maps Android API v2, το εντοπίζουμε και θέτουμε την κατάσταση του σε "on"!

1.4.1.7.7.1 Αίτηση ενός ή περισσοτέρων κλειδιών

Από τη στιγμή που έχουμε ενεργοποιήσει το Google Maps Android API v2 στο πρότζεκτ της εφαρμογής μας στην κονσόλα, έχουμε την δυνατότητα αιτήσεων ενός ή και περισσοτέρων κλειδιών.

Στην οθόνη προεπισκόπισης του πρότζεκτ επιλέγουμε και πάλι από τις κατηγορίες αριστερά την APIs. Αυτή την φορά όμως από το αναδυόμενο μενού θα επιλέξουμε την κατηγορία "Credentials".Επιλέγουμε το μπλε button "Create New Key" και στη συνέχεια κάνουμε κλικ στην επιλογή "Android Key".

Από εκεί και πέρα θα εμφανιστεί ένα νέο παράθυρο διαλόγου το οποίο θα ζητήσει το αποτύπωμα SHA-1 της εφαρμογής (ο τρόπος απόκτησης του αναφέρθηκε παραπάνω). Θα το κάνουμε επικόληση και στη συνέχεια θα πληκτρολογήσουμε ένα ελληνικό ερωτηματικό το οποίο θα ακολουθήται (χωρίς κενά) από το όνομα πακέτου της εφαρμογής, όπως φαίνεται και στο παράδειγμα παρακάτω:

45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example

Αφού εκτελεστεί και αυτή η εργασία επιλέγουμε "Create" και ανακτούμε το κλειδί μας.

1.4.1.7.7.2 Προσθήκη του κλειδιού στην εφαρμογή

Μετά την απόκτηση ενός API key έρχεται η τοποθέτηση του στην εφαρμογή, η οποία θα γίνει στο αρχείο manifest.

Επομένως μεταβαίνουμε στο manifest και θα επικολήσουμε το εξής στοιχείο meta-data μέσα στο tag <application>.

```
<meta-data

android:name="com.google.android.maps.v2.API_KEY"

android:value="AIzaSyD_6G9_OjnNydabEk768dtjwIFDlQR" />
```

Όπως παρατηρούμε, το API key έχει επικοληθεί μέσα στην ιδιότητα value του στοιχείου meta-data.

1.4.1.7.8 Λοιπές ρυθμίσεις στο AndroidManifest.xml

Θα πρέπει να συνεχίσουεμ την προσθήκη ρυθμίσεων στο AndroidManifest.xml της εφαρμογής.

Πρωτίστως, εάν δεν έχουμε τις κατάλληλες άδειες, θα πρέπει να της δηλώσουμε μέσω του uses-permission. Αυτή την ενέργεια θα μπορούσαμε να την είχαμε κάνει και νωρίτερα.

```
<permission

 android:name="com.Skemelio.kalymnosguide.permission.MAPS_RECEIVE"
```

```
android:protectionLevel="signature" />


<uses-permission android:name="com.Skemelio.kalymnosguide.permission.MAPS_RECEIVE"
/>

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

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

<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

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

Στη συνέχεια, δεν είναι απαραίτητο, αλλά θα πρέπει να προσθέσουμε την απαίτηση του OpenGL ES version 2. Το OpenGL ES version 2 είναι αυτό που θα «ζωγραφίσει» τους χάρτες στην εφαρμογή μας, οπότε θα πρέπει να απαιτείται από την συσκευή η εγκατάσταση του, πριν την εγκατάσταση της εφαρμογής KalymnosGuide.

Η απαίτηση θα δηλωθεί μέσω του στοιχείου uses-feature το οποίο θα βρίσκεται στο tag <manifest>:

```
<uses-feature

    android:glEsVersion="0x00020000"

    android:required="true" />
```

Τέλος, όπως έχουμε αναφέρει παραπάνω, απαιτείται το εξής στοιχείο,

```
<meta-data

      android:name="com.google.android.gms.version"

      android:value="@integer/google_play_services_version" />
```

το οποίο προσθέτει το Google Play services version στην εφαρμογή.

1.4.1.7.9 Προσθήκη ενός χάρτη στην εφαρμογή

Έφθασε η στιγμή της προσθήκης του χάρτη, προγραμματιστικά, στην εφαρμογή!

Ο παρακάτω σχετικός κώδικας θα εμφανιστεί στην απλούστερη μορφή του για να γίνει κατανοητή η προσθήκη του χάρτη.

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

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.google.android.gms.maps.MapFragment"/>
```

Οι παραπάνω γραμμές κώδικα αφορούν το νέο αρχείο activity_main.xml που δημιουργήσαμε στον φάκελο /res/layout και ουσιαστικά θα αποτελέσει το layout της δραστηριότητας, η οποία υλοποιεί έναν χάρτη τύπου GoogleMap(κλάση com.google.android.gms.maps.GoogleMap).

Στην ιδιότητα name του fragment γράφεται η κλάση η οποία θα το υλοποιήσει. Στη συγκεκριμένη περίπτωση είναι η MapFragment.

Στο επόμενο βήμα θα γράψουμε τον παρακάτω κώδικα στην δραστηριότητα που θα υλοποιήσει τον χάρτη.

```
package com.example.mapdemo;
    import android.app.Activity;
    import android.os.Bundle;

    public class MainActivity extends Activity {

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

Τρέχοντας την εφαρμογή θα πρέπει να εμφανίζεται ένας χάρτης Google Map, σε διαφορετική περίπτωση θα έχουμε κάνει κάποιο λάθος με το API-key ή με όλα τα παραπάνω που αναφέρθηκαν.

Γεοκωδικοποίηση - The Google Maps Geolocation API

Με τον όρο γεοκωδικοποίηση (geocoding) ορίζουμε τη διαδικασία αντιστοίχισης μιας διεύθυνσης στο ζεύγος γεωγραφικών συντεταγμένων (γεωγραφικό μήκος-πλάτος) όπου υπάγεται.

Θα συναντήσουμε και τον όρο αντίστροφη γεοκωδικοποίηση (reverse geocoding), όπου μέσω των γεωγραφικών συντεταγμένων εντοπίζουμε την διεύθυνση. Η αντίστροφη διαδικασία δηλαδή.

Την συγκεκριμένη υπηρεσία δεν θα τη βρούμε στην κονσόλα της Google, μιας και διατίθεται αυτόματα και δωρεάν, όμως με μερικούς περιορισμούς. Για παράδειγμα τα request που μπορεί να ζητήσει μία εφαρμογή την ημέρα είναι περιορισμένα, εκτός και αν κάποιος αγοράσει το «Geolocation API for Business license», όπου θα του επιτρέπονται περισσότερα κτλ.

Όσον αφορά την υπηρεσία για την πλατφόρμα του Android, θα χρησιμοποιήσουμε την βασική κλάση Geocoder (android.location.Geocoder). Αφού την αρχικοποιήσουμε θα χρησιμοποιήσουμε μία εκ των μεθόδων getFromLocation(double latitude, double longitude, int maxResults) ή getFromLocationName(String locationName, int maxResults).

Η getFromLocation( ) θα επιστρέψει μία λίστα με διευθύνσεις (List<Address>) αφού έχουμε δώσει ως παράμετρο τις γεωγραφικές συντεταγμένες, ενώ η getFromLocationName( ) αφού έχουμε δώσει ως παράμετρο μία String διεύθυνση. Ο integer maxResults υποδηλώνει τον μέγιστο αριθμό των Address αντικειμένων που θα περιέχει η λίστα List<Address>.

Ακολουθεί ο κώδικας για γεοκωδικοποίηση μέσω της μεθόδου geolocate( ) και για την αντίστροφη γεοκωδικοποίηση η μέθοδος reverseGeolocate( ):

```
public void geolocate() throws IOException{

        Geocoder gc = new Geocoder(this);

        String name = "kalymnos";

        List<Address> lista = gc.getFromLocationName(name, 1);

        Address address = lista.get(0);

        String onomasia_perioxis = address.getLocality();

        String taxidromikos_kwdikas = address.getPostalCode();

        String onomasia_xwras = address.getCountryName();

        String kwdikos_xwras = address.getCountryCode();

    }

public void reverseGeoLocate(View v) throws IOException {

        Geocoder gc = new Geocoder(this);
```

```
        double lat = 37.673508;
        double lng = 21.439893;

        List<Address> list = gc.getFromLocation(lat, lng, 1);
        Address address = list.get(0);
        String onomasia_perioxis = address.getLocality();
        String taxidromikos_kwdikas = address.getPostalCode();
        String onomasia_xwras = address.getCountryName();
        String kwdikos_xwras = address.getCountryCode();
}
```

2 Σχεδιασμός και Υλοποίηση εφαρμογής

Στο παρών κεφάλαιο αναλύονται οι προδιαφραφές της εφαρμογής android που δημιουργήσαμε, της KalymnosGuide. Πριν όμως περάσουμε στην περαιτέρω περιγραφή θα πρέπει να περιγράψουμε, από μία αφαιρετική σκοπιά, την ιδέα που προυπήρξε της δημιουργίας.

2.1 Περιγραφή απαιτήσεων – χαρακτηριστικών εφαρμογής

Στο παρών κεφάλαιο αναλύονται οι προδιαφραφές της εφαρμογής android που δημιουργήσαμε, της KalymnosGuide. Πριν όμως περάσουμε στην περαιτέρω περιγραφή θα πρέπει να περιγράψουμε, από μία αφαιρετική σκοπιά, την ιδέα που προυπήρξε της δημιουργίας.

Η ιδέα πίσω από την KalymnosGuide

Η εφαρμογή KalymnosGuide δημιουργήθηκε με σκοπό να αντιπροσωπεύσει έναν ταξιδιωτικό οδηγό της νήσου Καλύμνου, πλαισιωμένο στην πλατφόρμα του Android. Ακριβώς όπως ένας ολιγοσέλιδος ταξιδιωτικός οδηγός που έχει αποκτηθεί δωρεάν από ένα ταξιδιωτικό πρακτορείο, έτσι και η KalymnosGuide απευθύνεται στους επισκέπτες του νησιού των σφουγγαράδων.

Σκοπός της θα είναι η ενημέρωση του επισκέπτη σχετικά με το νησί, όσον αφορά τη διαμονή, τα μέρη που θα δειπνήσει, τις δραστηριότητες που του παρέχονται, τις όμορφες παραλίες, τις παραδόσεις-έθιμα και πολλά άλλα χρήσιμα στοιχεία τα οποία θα κάνουν τη ζωή του ευκολότερη κατά τη διάρκεια της διαμονής του.

Σε αντίθεση με την τυπωμένη έκδοση του ταξιδιωτικού οδηγού, τα παραπάνω στοιχεία θα παρουσιάζονται δυναμικά μέσα από τη συσκευή του κινητού τηλεφώνου, κάνοντας την πλοήγηση του επισκέπτη πιο ευχάριστη και φυσικά την αναζήτηση πιο γρήγορη, χάρη σε μία σειρά ευέλικτων μενού της εφαρμογής.

Επί παραδείγματι, υποθέτουμε πως ο επισκέπτης ψάχνει μία ταβέρνα για να γευτεί τους τοπικούς μεζέδες. Από το κυρίως μενού θα επιλέξει την κατηγορία “Eat and Drink” και στη συνέχεια από τη λίστα την κατηγορία“Greek Taverns”, ώστε να εμφανίσει όλες οι διαθέσιμες ταβέρνες της Καλυμνου. Από εκεί και πέρα μένει να επιλέξει την εκάστοτε ταβέρνα για την προβολή της προεπισκόπισης, η οποία περιλαμβάνει πληροφορίες τηλεφώνου, διεύθυνσης, πλοήγηση σε χάρτη και άλλα.

Οπότε αυτό που ακολουθεί είναι η υλοποίηση της περιγραφούσας ιδέας σε εφαρμογή Android.

Από την ιδέα στην υλοποίηση

Σε προηγούμενο κεφάλαιο είχαμε ορίσει μία εφαρμογή android ως συλλογή εργασιών, οι οποίες καλούνται δραστηριότητες ή activities. Μία δραστηριότητα είναι μία κλάση της προγραμματισμού Java, ένα αρχείο .java.

Θα φαινόταν απολύτως λογικό να ξεκινήσουμε την ανάλυση της εφαρμογής από τις κλάσεις των δραστηριοτήτων, ωστόσο θα πρέπει να υπογραμμισθεί πως για την υλοποίηση της ιδέας της εφαρμογής ήταν απαραίτητη η δημιουργία κλάσεων, οι οποίες δεν αποτελούν δραστηριότητες της εφαρμογής.

Επομένως ο φάκελος /src του έργου περιέχει δύο πακέτα. Το πακέτο εφαρμογής com.Skemelio.kalymnosguide, το οποίο δηλώνεται στο αρχείο manifest, και το πακέτο utilities.

Το πρώτο πακέτο περιέχει τις κλάσεις των δραστηριοτήτων, ενώ το πακέτο utilities περιέχει τις κλάσεις και τα interface τα οποία υλοποιούν την ιδέα της εφαρμογής. Η ανάλυση μας θα αρχίσει με το πακέτο utilities, μιας και η ιδέα προυπάρχει των δραστηριοτήτων.

2.1.1.1 KalymnosGuide/src/utilities

Το πακέτο utilities περιλαμβάνει όλες τις κλάσεις και τα interface τα οποία υλοποιούν την ιδέα της εφαρμογής.

Κύρια κλάση είναι η GeneralClass.java η οποία αντιπροσωπεύει γενικά ένα αντικείμενο της εφαρμογής μας. Για παράδειγμα ένα εστιατόρειο θα είναι αντικείμενο της κλάσης GeneralClass, ένα ξενοδοχείο θα είναι επίσης αντικείμενο της GeneralClass, μία τοποθεσία κοκ.

Από εκεί και πέρα έχουν δημιουργηθεί κλάσεις οι οποίες κληρονομούν την GeneralClass και αντιπροσωπεύουν αντικείμενα ειδικότερου περιεχομένου Αναλυτικά:

- Η Epixirisi.java κληρονομεί την GeneralClass.java, αλλά αναφέρεται μόνο σε αντικείμενα επιχειρήσεων. Ταβέρνες, ξενοδοχεία, ταξιδιωτικά πρακτορεία, σούπερ μάρκετ κτλ

- Η κλάση Festival κληρονομεί την GeneralClass.java και αναφέρεται σε πολιτιστικές εκδηλώσεις

- Η κλάση Traditions αναφέρεται στις παραδόσεις-έθιμα του νησιού και κληρονομεί την GeneralClass

- Η κλάση Product παράγει στιγμιότυπα τα οποία αντιπροσωπεύουν τα τοπικά προιόντα του νησιού. Κληρονομεί την GeneralClass.java

- Η κλάση Region κληρονομεί την GeneralClass και αντιπροσωπεύει γενικά μία τοποθεσία. Από την κλάση Region παράγονται οι παρακάτω τρεις κλάσεις που περιγράφουν ειδικότερα μία τοποθεσία:

-Beach.java: Κληρονομεί την Region, άρα και την GeneralClass, και παράγει αντικείμενα που αντιπροσωπεύουν παραλίες

-Sight.java: Κληρονομεί την Region και παράγει αντικείμενα που αντιπροσωπεύουν αξιοθέατα

-Village.java: Κληρονομεί την Region και παράγει αντικείμενα που αντιπροσωπεύουν χωριά

Πέραν της κεντρικής κλάσης GeneralClass.java και όλων των απογόνων της, το πακέτο utilities περιέχει διάφορα interface, τα οποία βοηθούν στην βέλτιστη υλοποίηση των παραπάνω κλάσεων. Τα interface αυτά είναι τα εξής:

- Το interface BeachInterface.java υλοποιείται από αντικείμενα της κλάσεις Beach και καθορίζει αν οι παραλίες αυτές έχουν ξαπλώστρες ή δέντρα για σκιά

- Μία κλάση που υλοποιεί το ByTheSea.java έχει τη δυνατότητα να κρίνει αν ένα αντικείμενο είναι παραθαλάσσιο, πχ ένα μπαρ

- Το Delivery.java υποδυκνείει αν ένα αντικείμενο διαθέτει υπηρεσία κατ'οίκον παράδοσης

- Τέλος το WiFi.java υποδυκνείει αν ένα αντικείμενο διαθέτει σύνδεση στο διαδίκτυο μέσω WiFi

Τέλος, το πακέτο utilities συμπεριλαμβάνει με τις παραπάνω και δύο επιπλέον κλάσεις.

Η Favourites.java είναι μία στατική κλάση όπου μέσα από μία διαδικασία μας προμηθέυει με μία λίστα αντικειμένων GeneralClass (List<GeneralClass>) η οποία αποτελεί τα αγαπημένα του χρήστη της εφαρμογής.

Η FavouritesAdapter.java πρόκεται περί βοηθητικής κλάσης η οποία ορίζει την εμφάνιση της κάθε στήλης από την λίστα που περιέχει τις αγαπημένες επιλογές του χρήστη.

2.1.1.2 KalymnosGuide/src/com.Skemelio.kalymnosguide

Το πακέτο της εφαρμογής περιέχει συνολικά έντεκα κλάσεις. Η δέκα από αυτές είναι οι δραστηριότητες της εφαρμογής και με τη σειρά τους κληρονομούν τη μία που περισσεύει. Η περισσευάμενη πρόκειται για κλάση σημείο αναφοράς και κληρονομεί την android.app.Activity.

Μία τούτη την κίνηση, τη δημιουργία δηλαδή μίας έξτρα κλάσης η οποία κληρονομεί την Activity και με τη σειρά τους οι δραστηριότητες θα κληρονομούν αυτή την κλάση, πετυχαίνουμε το εξής: Δυνατότητα να πραγματοποιήσουμε αλλαγές και προσθαφαιρέσεις στην κλάση, αφήνοντας ανέπαφη την android.app.Activity.

Αναλυτικά, ξεκινώντας από την κλάση σημείο αναφοράς και προχωρώντας στις δραστηριότητες:

- MyActivity.java: Κλάση η οποία κληρονομεί την android.app.Activity. Σκοπός της δεν είναι η παραγωγή αντικειμένων, αλλά να κληρονομείται από τις δραστηριότητες. Τεχνικά θα μπορούσαμε να κάνουμε λόγο για στατική κλάση. Επιπροσθέτως η MyActivity συγκεντρώνει πλήθος σταθερών, όπως όλα τα "static final" αντικείμενα της GeneralClass, καθώς επίσης και σταθερές οι οποίες θα χρησιμοποιηθούν από τις δραστηριότητες απογόνους. Τέλος, η MyActivity περιέχει δύο εσωτερικές κλάσεις οι οποίες βοηθούν στην ομαλή λειτουργία των δραστηριοτήτων. Ακολουθούν ονομαστικά οι εσωτερικές κλάσεις καθώς και ο ρόλος τους περιληπτικά.

1)TabHostListViewItem: "Γεμίζει" την ListView της δραστηριότητας AboutList.java

2)SubMainMenuItems: "Γεμίζει" την ListView της δραστηριότητας SubMainMenu.java

- MainMenu.java: Η δραστηριότητα η οποία αποτελεί την αφετηρία της εφαρμογής. Προβάλλει το κυρίως μενού το οποίο είναι χωρισμένο σε δύο οθόνες και η εναλλαγή τους γίνεται με button. Το κυρίως μενού περιλαμβάνεται από οκτώ κατηγορίες

- SubmainMenu.java: Μετά την επιλογή μίας κατηγορίας της MainMenu ακολουθεί η δραστηριότητα SubMainMenu, η οποία προβάλλει μία λίστα σχετική με την κατηγορία που επιλέχθηκε προηγουμένως

- AboutList.java: Δραστηριότητα στην οποία ο χρήστης έχει πρόσβαση μόνο από την SubMainMenu. Η οθόνη ουσιαστικά αποτελείται από έναν TabHost με δύο καρτέλες. Η καρτέλα About περιγράφει την επιλογή που έγινε στην δραστηριότητα SubMainMenu, ενώ η καρτέλα List παρέχει μία λίστα η οποία προβάλλει τα διαθέσιμα αντικείμενα της επιλογής εκείνης

- EpixirisiActivity.java: Δραστηριότητα η οποία προβάλλει αντικείμενα-επιχειρήσεις, δηλαδή στιγμιότυπα της Epixirisi.java

- Epexigisi.java: Δραστηριότητα η οποία προβάλλει άϋλα αντικείμενα. Η μόνη πληροφορία που δίνουν τα αντικείμενα αυτά είναι ένα επεξηγηματικό κείμενο, όπως για παράδειγμα το αντικείμενο "Legends and Myths"

- BeachActivity.java: Δραστηριότητα η οποία προβάλλει τα αντικείμενα-παραλίες της Καλύμνου, δηλαδή στιγμιότυπα της Beach.java

- VillageActivity.java: Δραστηριότητα η οποία προβάλλει αντικείμενα-χωριά του νησιού, δηλαδή στιγμιότυπα της Village.java

- SightActivity.java: Δραστηριότητα η οποία προβάλλει αντικείμενα-αξιοθέατα του νησιού, δηλαδή στιγμιότυπα της Sight.java

- OneMapActivity.java: Δραστηριότητα η οποία υλοποιεί το Google Maps Android API v2. Προβάλλονται όλα τα αντικείμενα της εφαρμογής τα οποία έχουν μία θέση στον χάρτη

- MapActivity.java: Δραστηριότητα η οποία υλοποιεί επίσης το Google Maps Android API v2, ωστόσο προβάλλει ένα συγκεκριμένο αντικείμενο της εφαρμογής στο χάρτη

*2.1.1.3 **Πόροι** – Resources*

Αρχίσαμε την περιγραφή των απαιτήσεων – χαρακτηριστικών της εφαρμογής με τις δραστηριότητες και τις κλάσεις του πακέτου utilities. Ωστόσο και άλλα χαρακτηριστικά απαιτούνται για την δημιουργία της KalymnosGuide και αυτά είναι οι πόροι οι οποίοι αναλύονται σε βάθος στο ζήτημα 1.3 του πρώτου κεφαλαίου.

Η κάθε δραστηριότητα της εφαρμογής χρησιμοποιεί έναν πόρο xml, ο οποίος βρίσκεται στον υποφάκελο /layout του φακέλου /res, ως διεπαφή χρήστη. Οι εικόνες που προβάλλονται από τις δραστηριότητες της εφαρμογής, όπως επίσης το κείμενο και άλλα, είναι πόροι αποθηκευμένοι στο έργο της εφαρμογής ή πόρους του συστήματος, ανάλογα με τις επιλογές του developer.

Η KalymnosGuide εκμεταλλεύεται πλήθος πόρων εφαρμογής. Ακολουθεί αναφορά των πόρων της εφαρμογής: Τα αρχεία γραφικών είναι αποθηκευμένα στους διάφορους φακέλους /res/drawable, ανάλογα με το μέγεθος.

Διαστάσεις, χρώματα και συμβολοσειρές είναι αποθηκευμένες στον φάκελο res/values και συγκεκριμένα στα αρχεία res/values/dimen.xml, res/values/color.xml και res/values/string.xml αντιστοίχως.

Οι πόροι που δίνουν τη διεπαφή χρήστη σε μία δραστηριότητα είναι αποθηκευμένοι σε έναν από τους φακέλους /res/layout και τέλος οι πόροι που χρησιμοποιούνται από τα μενού των δραστηριοτήτων αποθηκεύονται στον φάκελο /res/menu.

Η εφαρμογή χρησιμοποιεί σε μερικές περιπτώσεις πόρους συστήματος για να δώσει μορφή σε διάφορες Views, όπως για παράδειγμα ο πόρος android.R.resources.layout.simple_list_item1 που χρησιμοποιείται ως layout σε μία ListView.

2.1.1.4 Άδειες – Απαιτήσεις

Η KalymnosGuide θα πρέπει να κάνει χρήση αδειών για να χρησιμοποιήσει τους χάρτες της Google.

Αναλυτικά οι άδειες, όπως αυτές δηλώνονται στο αρχείο manifest της εφαρμογής:

```
<permission

android:name="com.Skemelio.kalymnosguide.permission.MAPS_RECEIVE"

android:protectionLevel="signature" />

<uses-permission android:name="com.Skemelio.kalymnosguide.permission.MAPS_RECEIVE"
/>

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

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

<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

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

Τέλος, στο manifest ορίζονται επίσης τα δύο παρακάτω στοιχεία. Το πρώτο για το API-key και το δεύτερο για το google play services version. Απαραίτητη η δήλωση και των δύο για την εμφάνιση του χάρτη.

```
<meta-data
```

android:name="com.google.android.maps.v2.API_KEY"

android:value="AIzaSyD_6G9_OjnNydabEk768dtjwIFDIQR" />

<meta-data

android:name="com.google.android.gms.version"

android:value="@integer/google_play_services_version" />

2.2 Ανάπτυξη εφαρμογής (περιγραφή όλων των βημάτων, Activities, classes, resources)

Πριν τη δημιουργία της εφαρμογής KalymnosGuide στην πλατφόρμα του Android προυπήρχε η ιδέα για την υλοποίηση της και θα ξεκινήσουμε με αυτή. Θα περιγραφεί σε ένα γενικό πλαίσιο το πλάνο της εφαρμογής, ο σχεδιασμός της, πρωτού αυτός μεταφρασθεί σε κώδικα.

Βήμα 1ο: Σχεδίαση του Πλάνου της Εφαρμογής

Σκεφτήκαμε λοιπόν πως η δομή της εφαρμογής KalymnosGuide θα πρέπει να είναι σχετικά απλή και αποφασίσαμε πως θα πρέπει να περιέχει τις εξής οθόνες:

Οθόνη Κυρίως Μενού: Αυτή η οθόνη θα λειτουργήσει ως μία οθόνη εκκίνησης. Ο χρήστης έχει τη δυνατότητα να επιλέξει ανάμεσα σε αρκετές επιλογές του κυρίως μενού όσον αφορά το που θα δειπνήσει, τη διαμονή και άλλες σχετικές κατηγορίες. Συνολικά θα υπάρχουν οκτώ επιλογές στο κυρίως μενού. Σε κάθε οθόνη, στο κάτω μέρος, θα υπάρχει μία γραμμή επιλογών. Όσον αφορά την οθόνη του κυρίως μενού, οι ενέργειες οι οποίες θα τελούνται από την γραμμή επιλογών θα είναι οι εξής:

α) πλοήγηση στο δεύτερο μέρος των επιλογών του κυρίως μενού και αντίστροφα

β) εμφάνιση παράθυρου διαλόγου των αγαπημένων (favorites)

γ) μετάβαση σε οθόνη χάρτη

Σε αυτό το σημείο σημειώνεται η αιτία απόρριψης χρήσης μίας εμβόλιμης οθόνης, η οποία απλώς θα εμφάνιζε μία διάταξη για μερικά δευτερόλεπτα και μετά θα εκκινούσε αυτόματα την οθόνη του κυρίως μενού, κάτι που χρησιμοποιούν αρκετές εφαρμογές της πλατφόρμας, ακόμα και σήμερα. Η αλήθεια είναι πως δεν θα θέλαμε να φλυαρήσουμε με τον χρήστη. Γνωρίζουμε πως ένα άτομο θα χρησιμοποιήσει την “Kalymnos Guide” για την αναζήτηση μιας πληροφορίας σχετικά με την Κάλυμνο, είτε αυτό είναι ένα τηλέφωνο ενός εστιατορείου, είτε μία πληροφορία θέσης για ένα μουσείο, μία παραλία κτλ. Επιθυμούμε, λοιπόν, να δώσουμε αμέσως στον χρήστη αυτό που αναζητάει, χωρίς χρονοτριβές.

Οθόνη Υπομενού: Η συγκεκριμένη οθόνη αποτελεί το υπομενού μίας συγκεκριμένης επιλογής από το κυρίως μενού. Το υπομενού θα φέρει ως τίτλο το όνομα της επιλογής που πραγματοποίησε αρχικά ο χρήστης στο κυρίως μενού. Στο κυρίως μέρος της οθόνης θα εμφανίζεται μία λίστα με επιλογές που αφορούν την αρχική επιλογή και θα καταλείγουμε με την συνήθης γραμμή επιλογών στο κάτω μέρος, με μία μικρή διαφορά. Εδώ αντί της

επιλογής που μας μεταφέρει στο δεύτερο μέρος των επιλογών του κυρίως μενού, απλώς μας πάει πίσω σε αυτό.

Οθόνη Περιγραφής Γενικού Περιεχομένου: Για να γίνει ευκολότερα κατανοητός ο ρόλος της οθόνης περιγραφής γενικού περιεχομένου θα βασίσουμε την εξήγηση μας σε ένα παράδειγμα. Για την ώρα έχουμε αναφέρει πως όταν ο χρήστης "τρέξει" το application, αρχικά, θα βρεθεί στην οθόνη του κυρίως μενού. Υποθέτοντας ότι επιλέγει την πρώτη κατηγορία που βλέπει, ας πούμε την "Eat and Drink", θα οδηγηθεί στην οθόνη του υπομενού όπου εκεί καλείται να επιλέξει μία από τις επιλογές της λίστας, όπως bars, restaurants, fast food κτλ. Εκ νέου, υποθέτουμε πως έχει επιλέξει bars. Σε αυτό το σημείο περνάμε στην οθόνη περιγραφής για την οποία γίνεται ο λόγος. Η οθόνη περιγραφής θα φέρει τον τίτλο της προηγούμενης μας επιλογής, ("bars" όσον αφορά το δικό μας παράδειγμα). Το μεγαλύτερο μέρος της οθόνης θα το καταλαμβάνει ένας TabHost (μία λειτουργία του android με καρτέλες). Η πρώτη καρτέλα περιγράφει αναλυτικά την επιλογή που κάναμε στο υπομενού, ενώ η δεύτερη θα περιέχει τη λίστα με τα διαθέσιμα μπαρ του νησιού. Κάπου εδώ θα πρέπει να υπογραμμισθεί πως υπάρχουν επιλογές στο κυρίως μενού, όπως η "History", όπου η οθόνη περιγραφής είναι περιττή. Σε αυτές τις περιπτώσεις θα παιρνάμε αμέσως σε οθόνη περιγραφής ειδικού περιεχομένου

Οθόνη Περιγραφής Ειδικού Περιεχομένου: Κάπου εδώ φθάνουμε στο σημείο όπου ο χρήστης μετά από πορεία τριών τουλάχιστον οθονών βρήκε αυτό που έψαχνε. Ένα ταξιδιωτικό πρακτορείο ονόματι "Magos Tours", την ταβέρνα του Παντελή, τις όμορφες ακρογυαλιές του Εμπορειού ή μία συγκεκριμένη δραστηριότητα που του προσφέρει το νησί.

Η οθόνη ειδικού περιεχομένου φέρει για τίτλο το όνομα της επιχείρησης, τοποθεσίας ή δραστηριότητας και φυσικά τις αναλυτικές πληροφορίες σχετικά με το αντικείμενο. Οι πληροφορίες αυτές θα καταλαμβάνουν μεγάλο μέρος της οθόνης. Τέλος δεν θα λείπει, ούτε από εδώ, η γραμμή επιλογών στο κάτω μέρος

Οθόνες Χάρτη: Πρόκειται για οθόνες οι οποίες θα υλοποιούν το Google Maps Api V2, άρα και θα εμφανίζουν τους συγκεκριμένους χάρτες ρυθμισμένους στις ανάγκες της εφαρμογής.Υπάρχουν δύο τέτοιες οθόνες. Η πρώτη θα εμφανίζει τον χάρτη της νήσου Καλύμνου. Απάνω στον χάρτη θα διαφαίνονται με markers(δείκτες, σημάδια στον χάρτη) όλα τα αντικείμενα της εφαρμογής που δύναται να έχουν μία γεωγραφική θέση, παραδείγματος χάρην όλες οι επιχειρήσεις, οι τοποθεσίες κτλ. Πρόκειται για μία οθόνη χάρτη η οποία θα συγκεντρώνει όλα τα αντικείμενα της εφαρμογής, τα οποία έχουν μία θέση σε αυτόν.

Εν αντιθέσει η δεύτερη οθόνη χάρτη θα εστιάζει συγκεκριμένα μόνο σε ένα αντικείμενο.

Τέλος οσον αφορά την πρόσβαση στις δύο οθόνες, ο χρήστης θα έχει τη δυνατότητα για πρόσβαση στον συγκεντρωτικό χάρτη από το κυρίως μενού (button της γραμμής επιλογών), ενώ για τον δεύτερο θα υπάρχει επίσης σχετικό button στην οθόνη περιεχομένου ενός αντικειμένου (στις αναλυτικές πληροφορίες του αντικειμένου).

Βήμα 2ο: Δημιουργία Δραστηριοτήτων

Πακέτο com.Skemelio.kalymnosguide

Οι οθόνες που περιγράψαμε παραπάνω θα τις ενσαρκώσουν οι παρακάτω δραστηριότητες οι οποίες εμπεριέχονται στο πακέτο εφαρμογής com.Skemelio.kalymnosguide:

- MainMenu.java: Η activity MainMenu λειτουργεί σαν προεπιλεγμένη δραστηριότητα εκκίνησης και αποτελεί την δραστηριότητα που θα υλοποιήσει την οθόνη του κυρίως μενού. Σκοπός της είναι να εμφανίζει το κυρίως μενού στον χρήστη, με τις διάφορες επιλογές, καθώς και μία γραμμή εργαλείων στο κάτω μέρος όπου ο χρήστης μπορεί να προβεί σε περεταίρω ενέργειες. Όσον αφορά τις επιλογές του κυρίως μενού, αυτές δεν είναι κουμπιά αλλά εικόνες (imageviews), στις οποίες ένα συμβάν onClick( ) εκκινεί την επόμενη δραστηριότητα που περιγράφεται παρακάτω

- SubMainMenu.java: Η SubMainMenu αναλαμβάνει την υλοποίηση της οθόνης του υπομενού. Εκτός από τη γραμμή εργαλείων στο τέλος και την ActionBar που είναι παρόμοια στις περισσότερες οθόνες, το κυρίως μέρος της SubMainMenu καταλαμβάνει μία ListView. Μία λίστα με επιλογές όπου ένα συμβάν onClick( ) θα μας μεταφέρει στις περισσότερες των περιπτώσεων στην δραστηριότητα AboutList.java

- AboutList.java: Η υλοποίηση της οθόνης περιγραφής γενικού περιεχομένου είναι δουλειά της δραστηριότητας AboutList. Η διάταξη όσον αφορά την ActionBar στο πάνω μέρος και την γραμμή εργαλείων στο κάτω μέρος είναι παρόμοια με τις δραστηριότητες SubMainMenu και MainMenu, ωστόσο στο κυρίως μέρος διαφέρει μιας και χρησιμοποιεί έναν TabHost. Πρόκειται για ένα μηχανισμό με καρτέλες, όπου σε κάθε μία από αυτές που επιλέγει ο χρήστης εμφανίζεται διαφορετικό περιεχόμενο. Η καρτέλα “About” περιγράφει την επιλογή που έκανε το άτομο στην SubMainMenu, ενώ η δεξιά καρτέλα, η “List”, περιέχει τη λίστα με επιλογές σχετικές με την καρτέλα About

- EpixirisiActivity.java, BeachActivity.java, VillageActivity.java, SightActivity.java, Epexigisi.java: Εδώ υλοποιείται η ουσιαστικότερη λειτουργία της εφαρμογής, η προβολή του περιεχομένου των αντικειμένων. Εξάλλου αυτός είναι ο απότερος σκοπός της εφαρμογής, αυτή είναι η υπηρεσία που προσφέρει σε αυτόν που θα την κατεβάσει και θα την τρέξει από το κινητό του τηλέφωνο ή το tablet. Η πληροφορία.

Και οι πέντε, λοιπόν, παραπάνω δραστηριότητες αναλαμβάνουν την υλοποίηση της οθόνης περιγραφής ειδικού περιεχομένου. Η κάθε δραστηριότητα προβάλλει περιεχόμενο διαφορετικού είδους.

Η EpixirisiActivity προβάλλει επιχειρήσεις, η BeachActivity προβάλει παραλίες, η VillageActivity τα χωριά, η SightActivity αξιοθέατα, ενώ αυτή που διαφέρει είναι η δραστηριότητα Epexigisi.

Η τελευταία προβάλλει αντικείμενα που είναι απλώς αφηγηματικά, όπως, για παράδειγμα, η ιστορία της Καλύμνου. Η ιστορία θα είναι εμφανίζεται ως ένα κείμενο και μία

αντιπροσωπευτική φωτογραφία. Ο χρήστης δεν θα αναζητήσει κάποιο σχετικό τηλέφωνο ή κάποια θέση στον χάρτη για το συγκεκριμένο ζήτημα.

Όσον αφορά την διάταξη, πλην της Epexigisi.java, έχουμε τα εξής χαρακτηριστικά:

Την γνωστή ActionBar στο πάνω μέρος της οθόνης με τίτλο το όνομα του αντικειμένου. Στο κέντρο της οθόνης, καταλαμβάνοντας περίπου το 80% του συνολικού χώρου, βρίσκουμε τις αναλυτικές πληροφορίες των αντικειμένων όπως τηλέφωνο, διεύθυνση, επιλογή προβολής τοποθεσίας στο χάρτη κτλ.

- OneMapActivity.java: Η OneMapActivity θα είναι μία από τις δύο δραστηριότητες που θα υλοποιήσουν χάρτη (GoogleMaps API V2). Όσον αφορά την διάταξη, προφανώς διαφέρει από τις παραπάνω. Στο κεντρικό μέρος της οθόνης θα εμφανίζεται ο χάρτης με προεπιλεγμένη τοποθεσία την Κάλυμνο. Η ActionBar στο πάνω μέρος της δραστηριότητας δεν θα φέρει κάποιο τίτλο, ωστόσο θα έχει τουλάχιστον μία επιλογή (ανάλογα με το χώρο της οθόνης της εκάστοτε συσκευής που τρέχει την εφαρμογή), η οποία παραπέμπει σε μία ορισμένη ενέργεια του χάρτη. Γραμμή εργαλείων στο κάτω μέρος δεν θα υπάρχει. Κάτω από την ActionBar θα βρίσκεται ένα πλαίσιο Spinner ονόματι "Show markers". Εκεί ο χρήστης θα επιλέγει από αναδυόμενο μενού διάφορες ρυθμίσεις, οι οποίες θα εξηγηθούν στη συνέχεια. Η OneMapActivity είναι η δραστηριότητα που συγκεντρώνει σε έναν χάρτη όλα τα αντικείμενα των οθονών περιγραφής ειδικού περιεχομένου

- MapActivity.java: Για το τέλος μας έμεινε η MapActivity, η δραστηριότητα που θα προβάλλει στους χάρτες της Google τη θέση ενός και μόνο αντικειμένου της οθόνης περιγραφής ειδικού περιεχομένου, δηλαδή ενός αντικειμένου των κλάσεων EpixirisiActivity, BeachActivity, VillageActivity και SightActivity. Η διάταξη της δραστηριότητας θα εξίσου απλή με την OneMapActivity, με μικρές διαφορές. Η MapActivity θα φέρει ως τίτλο το όνομα του αντικειμένου το οποίο φαίνεται η θέση του στον χάρτη συμπληρομένος με τη φράση "on Map". Ακόμη στην ActionBar, δίπλα στον τίτλο, υπάρχει σχετική επιλογή η οποία πραγματοποιεί μία συγκεκριμένη ενέργεια στον χάρτη. Όπως είναι λογικό το κεντρικό μέρος της οθόνης το καταλαμβάνει ο χάρτης.

Τέλος, έχουμε γραμμή επιλογών στο κάτω μέρος της δραστηριότητας, η οποία έχει δύο μόνο επιλογές, την "Home" και την "Google License". Εκτός από την προφανή λειτουργία της πρώτης, η δεύτερη επιλογή προβάλλει ένα παράθυρο διαλόγου με τους όρους χρήσης του Google Maps API.

Εδώ θα πρέπει να σημειωθεί πως όλες οι δραστηριότητες της εφαρμογής κληρονομούν την κλάση MyActivity.java, η οποία βρίσκεται στο ίδιο πακέτο, και με τη σειρά της η τελευταία είναι απόγονος της κλάσης FragmentActivity.

Βήμα 3ο: Δημιουργία Κλάσεων Περιεχομένου

Πέραν των περιγραφόμενων δραστηριοτήτων η εφαρμογή θα πρέπει να προβάλλει κάποιο περιεχόμενο. Το περιεχόμενο αυτό θα είναι διάφορες επιχειρήσεις του νησιού που θα αναζητήσει ο ενδιαφερόμενος, τοποθεσίες, παραλίες, πολιτιστικές εκδηλώσεις κτλ.

Επομένως θα πρέπει να δημιουργηθούν κλάσεις οι οποίες θα παράγουν τέτοιου είδους αντικείμενα. Οι κλάσεις αυτές και διάφορα interface τα οποία βοηθούν στην βέλτιστη υλοποίηση τους, είναι αποθηκευμένες στο πακέτο utilities.

Τέλος, το πακέτο utilities περιέχει δύο επιπλέον κλάσεις οι οποίες χρησιμοποιούνται για την υλοποίηση της λειτουργίας των αγαπημένων επιλογών του χρήστη (favorites). Ακολουθεί η περιγραφή τους.

Πακέτο utilities

- GeneralClass.java: Η βασική κλάση του πακέτου. Η GeneralClass θα είναι αυτή που θα παράγει τα αντικείμενα της εφαρμογής. Συνεπώς κάθε αντικείμενο της KalymnosGuide θα είναι τύπου GeneralClass, ωστόσο δεν θα παράγονται απευθείας από αυτήν, αλλά θα δημιουργηθούν νέες κλάσεις, απογόνοι της GeneralClass, οι οποίες θα παράγουν αντικείμενα ειδικότερου περιεχομένου

- Epixirisi.java: Η πρώτη κλάση που δημιουργήθηκε για την παραγωγή αντικειμένων ειδικότερου περιεχομένου είναι η Epixirisi.java. Η κλάση αυτή κληρονομεί την GeneralClass, όπως και όλες οι κλάσεις παραγωγής αντικειμένων τα οποία θα αποτελούν το προβαλλόμενο περιεχόμενο της εφαρμογής. Ακόμη υλοποιεί τα interface Delivery, WiFi και ByTheSea για τα οποία θα γίνει αναφορά παρακάτω. Η κλάση Epixirisi παράγει αντικείμενα επιχειρήσεις, όπως εστιατόρεια, ταξιδιωτικά πρακτορεία, σούπερ μάρκετ κτλ

- Festival.java: Η κλάση Festival κληρονομεί την GeneralClass και παράγει αντικείμενα τα οποία αποτελούν πολιτιστικές εκδηλώσεις

- Product.java: Η Product.java είναι υπεύθυνη για την παραγωγή αντικειμένων τα οποία αντιπροσωπεύουν τα τοπικά προϊόντα του νησιού

- Region.java: Η κλάση Region κληρονομεί την GeneralClass, ωστόσο δεν δημιουργήθηκε για την παραγωγή αντικειμένων, αλλά με σκοπό την δημιουργία κλάσεων-απογόνων οι οποίες θα παράγουν αντικείμενα ειδικότερου περιεχομένου όσον αφορά κάποια τοποθεσία. Αυτές οι κλάσεις απογόνοι θα μοιράζονται ορισμένα κοινά χαρακτηριστικά που έχουν οι τοποθεσίες (όπως για παράδειγμα γεωγραφικές συντεταγμένες) μέσω της πατρικής κλάσης

- Beach.java: Η κλάση Beach κληρονομεί την Region.java και παράγει αντικείμενα τα οποία εκπροσωπούν παραλίες της Καλύμνου. Τέλος, υλοποιεί το BeachInterface.java

- Sight.java: Και η κλάση Sight κληρονομεί την Region.java και παράγει τα αντικείμενα-αξιοθέατα του νησιού

- Village.java: Η τελευταία κλάση που κληρονομεί την Region.java είναι η κλάση Village. Η κλάση αυτή θα παράγει αντικείμενα τα τα οποία εκπροσωπεύουν τα χωριά του νησιού

- Traditions.java: Η κλάση Traditions κληρονομεί την GeneralClass και παράγει αντικείμενα τα οποία αποτελούν τις παραδόσεις-ήθη-έθιμα της Καλύμνου

- BeachInterface.java: Το πρώτο interface που θα περιγράψουμε είναι το BeachInterface το οποίο υλοποιείται αποκλειστικά από την κλάση Beach. Το BeachInterface.java διασαφηνίζει εάν μία παραλία διαθέτει ξαπλώστρες, δέντρα για σκιά ή και τα δύο

- WiFi.java: Το interface που πιστοποιεί εάν ένα αντικείμενο Epixirisi διαθέτει ασύρματη σύνδεση στο διαδίκτυο

- Delivery.java: Το interface Delivery είναι υπεύθυνο για την πληροφόρηση ενός αντικειμένου Epixirisi για την ύπαρξη υπηρεσίας κατ'οίκου παράδοσης

- ByTheSea.java: Το interface που εξηγεί για ένα αντικείμενο αν είναι παραθαλάσσιο

- Favourites.java: Η κλάση Favourites είναι η πρώτη από τις προαναφερθήσες κλάσεις του πακέτου utilities η οποία δεν κληρονομεί άμεσα ή έμεσα την GeneralClass. Η Favourites.java δεν είναι υπεύθυνη για την παραγωγή αντικειμένων, παρά μόνο για την λήψη σημαντικών λιστών (interface List<E>) οι οποίες χρησιμεύουν στην λειτουργία επιλογής αγαπημένων επιλογών του χρήστη (favorites)

- FavoriteAdapter.java: Η συγκεκριμένη κλάση κληρονομεί την ArrayAdapter<GeneralClass> και είναι υπεύθυνη για να δώσει μορφή σε μία γραμμή μίας ListView. Για την ακρίβεια πρόκειται για την ListView η οποία θα εμφανίζει τις αγαπημένες επιλογές του χρήστη (favorites)

Βήμα 4ο: Πόροι Εφαρμογής– φάκελος /res

Το επόμενο βήμα που θα περιγράψουμε έχει να κάνει με τους πόρους της εφαρμογής.

Σε αυτό το σημείο θα πρέπει να σημειωθεί πως η σειρά περιγραφής των βημάτων δεν συμπίπτει με την πραγματική σειρά της υλοποίησης. Κατά την ώρα της δημιουργίας σε πραγματικό χρόνο υλοποιούμε πολλά από τα αναφερθέντα βήματα ταυτοχρόνως.

Πέραν των κλάσεων-δραστηριοτήτων και των κλάσεων του πακέτου utilities η εφαρμογή KalymnosGuide κάνει χρήση πληθώρας πόρων όπως εικόνες, συμβολοσειρές, μονάδες μεγέθους και άλλα. Όλοι αυτοί οι πόροι είναι αποθηκευμένοι στον φάκελο /res.

Υπάρχουν και περιπτώσεις που χρησιμοποιούνται πόροι συστήματος, ωστόσο αυτές είναι ελάχιστες και πρόκειται κυρίως για προυπάρχοντες πόρους layout οι οποίοι δίνουν μορφή σε μία ListView.

*2.2.1.1 **Αρχεία εικόνων**-/res /drawable*

Τα αρχεία εικόνων της εφαρμογής KalymnosGuide είναι αποθηκευμένα στους διαφόρους φακέλους (ανάλογα με το μέγεθος) drawable.

Δίνοντας ένα παράδειγμα προς χάρη κατανόησης, ένα αντικείμενο της KalymnosGuide, έστω ένα ξενοδοχείο, φέρει μία αντιπροσωπευτική φωτογραφία. Το συγκεκριμένο αρχείο εικόνας αποτελεί πεδίο της κλάσης Epixirisi και θα προβληθεί στην δραστηριότητα EpixirisiActivity (η δραστηριότητα που θα προβάλλει το αντικείμενο-ξενοδοχείο).

*2.2.1.2 **Αρχεία διεπαφών χρήστη**-/res/layout*

Η κάθε δραστηριότητα πρέπει να φέρει μία διεπαφή χρήστη η οποία θα ορίζει την εμφάνιση της.

Οι διεπαφές χρήστη στην πλατφόρμα του Android ορίζονται κατά κύριο λόγω μέσω αρχείων .xml τα οποία είναι αποθηκευμένα στους διαφόρους υποφακέλος layout, του φάκελου /res. Ωστόσο δεν είναι απαραίτητη η ύπαρξη ενός πόρου xml, μιας και μία διεπαφή χρήστη μπορεί να δημιουργηθεί προγραμματιστικά μέσα στη δραστηριότητα.

Οι δραστηριότητες της KalymnosGuide χρησιμοποιούν ένα δικό τους αρχείο .xml για να ορίσουν το layout τους. Αυτά τα αρχεία είναι αποθηκευμένα στον φάκελο /res/layout.

Ακολουθεί αναφορά και σύντομη περιγραφή των σχετικοών πόρων διεπαφής χρήστη:

1. aboutlist.xml: Ο πόρος διεπαφής χρήστη που χρησιμοποιεί η δραστηριότητα AboutList.java (μέσω της μεθόδου Activity.setContentView(R.layout.aboutlist))
2. beachactivity.xml: Πόρος διεπαφής χρήστη της δραστηριότητας BeachActivity
3. dialog_pointup_beachactivity.xml: Πόρος διεπαφής μίας ListView η οποία υλοποιείται μέσα στην BeachActivity.java
4. epexigisi.xml: Πόρος διεπαφής χρήστη της δραστηριότητας Epexigisi
5. epixirisiactivity.xml: Πόρος διεπαφής χρήστη της δραστηριότητας EpixirisiActivity
6. favoritedialog_directions.xml: Πόρος διεπαφής παράθυρου διαλόγου το οποίο υφίσταται μόνο στην που ο χρήστης δεν έχει επιλέξει κάποιο αγαπημένο (favorites)
7. favoritedialog_listview.xml: Πόρος διεπαφής μίας ListView η οποία υλοποιείται μέσα στο παράθυρο διαλόγου των favorites
8. favoritedialog_singlerow.xml: Πόρος ο οποίος δίνει μορφή σε μία μόνο γραμμή της ListView που εμφανίζει τα favorites
9. listview_item_aboutlist.xml: Πόρος διεπαφής μίας ListView η οποία υλοποιείται μέσα στην AboutList.java
10. mainmenu.xml: Πόρος διεπαφής χρήστη της δραστηριότητας MainMenu
11. map_activity.xml: Πόρος διεπαφής χρήστη της δραστηριότητας MapActivity
12. one_map_activity.xml: Πόρος διεπαφής χρήστη της δραστηριότητας OneMapActivity
13. sightactivity.xml: Πόρος διεπαφής χρήστη της δραστηριότητας SightActivity
14. submainmenu.xml: Πόρος διεπαφής χρήστη της δραστηριότητας SubMainMenu
15. villageactivity.xml: Πόρος διεπαφής χρήστη της δραστηριότητας VillageActivity

*2.2.1.3 **Αρχεία πόρων μενού**-/res/menu*

Μία δραστηριότητα αντλεί τα στοιχεία του αναδυόμενου μενού της μέσω πόρων xml οι οποίοι αποθηκεύονται στον υποφάκελο /res/menu.

Στην περίπτωση της KalymnosGuide οι μόνες δραστηριότητες που φέρουν αναδυόμενο μενού, είναι αυτές που υλοποιούν τους χάρτες της Google.

Οπότε έχουμε τον πόρο μενού map.xml που δημιουργεί το αναδυόμενο μενού της MapActivity και τον πόρο onemap.xml που δημιουργεί το αναδυόμενο μενού της OneMapActivity αντίστοιχα.

*2.2.1.4 **Τιμές πόρων**-/res/values*

Ο φάκελος /res/values της KalymnosGuide περιέχει τέσσερα αρχεία .xml, color.xml, dimens.xml, strings.xml και styles.xml.

Οι πόροι χρώματος που χρησιμοποιεί η εφαρμογή αντλούνται από το color.xml , οι συμβολοσειρές από το strings.xml, οι πόροι μεγέθους από το αρχείο dimens.xml και το στυλ από το styles.xml.

3 Περιγραφή εφαρμογής

3.1 Αναλυτική περιγραφή της εφαρμογής (με screenshots)

Πακέτο utilities
GeneralClass.java

```
public class GeneralClass {


	protected String name, address, brief;
	protected int iconResource, imageResource;
	protected boolean isFavorite=false;
	protected LatLng ll;

	public GeneralClass(String name, String address, String brief,
			int iconResource, int imageResource) {
		super();
		this.name = name;
		this.address = address;
		this.brief = brief;
		this.iconResource = iconResource;
		this.imageResource = imageResource;
	}

	public void setLatLng(double latitude, double longitude){
		this.ll=new LatLng(latitude, longitude);
	}

	public LatLng getLatLng(){
		if(ll!=null){
			return this.ll;
		}else{
			return new LatLng(0, 0);
		}
	}

	public boolean getFavoriteResult(){
		return this.isFavorite;
	}

	public void setFavoriteResult(boolean isFavorite){
		this.isFavorite=isFavorite;
	}

	@Override
	public String toString() {
		// TODO Auto-generated method stub
		return name;
	}

	//Getters - Setters
	public String getName() {
		return name;
```

```
	}


	public void setName(String name) {
		this.name = name;
	}


	public String getAddress() {
		return address;
	}


	public void setAddress(String address) {
		this.address = address;
	}


	public String getBrief() {
		return brief;
	}


	public void setBrief(String brief) {
		this.brief = brief;
	}


	public int getIconResource() {
		return iconResource;
	}


	public void setIconResource(int iconResource) {
		this.iconResource = iconResource;
	}


	public int getImageResource() {
		return imageResource;
	}


	public void setImageResource(int imageResource) {
		this.imageResource = imageResource;
	}

}
```

Επεξήγηση κώδικα κλάσης GeneralClass.java: Όλα τα αντικείμενα της εφαρμογής θα έχουν από ένα όνομα, μία διεύθυνση, μία περιγραφή, ένα εικονίδιο, μία αντιπροσωπευτική εικόνα και μία γεωγραφική θέση στο χάρτη.

Γι το σκοπό αυτό δημιουργήθηκαν τα πεδία String name, String address, String brief, int iconResource, int imageResource και LatLng ll. Οι συμβολοσειρές δεν χρειάζονται περαιτέρω εξήγηση, ενώ το εικονίδιο και η εικόνα είναι τύπου integer γιατί αποτελούν αναφορές στους πόρους που θα τους προμηθευτούμε μέσω της κλάσης com.Skemelio.kalymnosguide.R (η διαδικασία έχει περιγραφεί αναλυτικά στην ενότητα 1.3).

Το νέο στοιχείο είναι η αναφορά στην κλάση LatLng. Πρόκειται για κλάση που έχει γίνει import από το Google Android API και δημιουργεί αντικείμενα τα οποία έχουν γεωγραφικό πλάτος και μήκος. Το γεωγραφικό πλάτος και μήκος είναι πεδία της κλάσης LatLng και επομένως μπορούμε να τα εξάγουμε ως εξής: double latitude = ll.latitude; και double longitude = ll.longitude; Με λίγα λόγια σαν να "πακετάρουμε" τις γεωγραφικές συντεταγμένες (γεωγραφικό ύψος δεν συμπεριλαμβάνεται) σε ένα αντικείμενο, το οποίο θα μας χρειαστεί στις δραστηριότητες που υλοποιούν χάρτη.

Όσον αφορά τις μεθόδους της κλάσης, εκτός από τις συνήθης "getters and setters", παρατηρείται μία ενδιαφέρουσα αλλαγή στην μέθοδο toString( ). Η συγκεκριμένη μέθοδος έχει γίνει "override" για να επιστρέφει το πεδίο name του αντικειμένου. Το γεγονός αυτό θα μας φανεί ιδιαίτερα χρήσιμο στην υλοποίηση της εφαρμογής, ιδιαίτερα στις λίστες (ListView).

Κλείνοντας, θα πρέπει να σημειώσουμε πως η κλάση GeneralClass περιγράφει αφαιρετικά ένα αντικείμενο της εφαρμογής KalymnosGuide, ωστόσο δεν είναι υπεύθυνη για την απευθείας παραγωγή αντικειμένων. Τον σκοπό αυτόν θα τον αναλάβουν οι κλάσεις απογόνοι της GeneralClass. Οι απευθείας γόνοι είναι οι Epixirisi.java, Product.java, Traditions.java, Festival.java και Region. Η κάθε μία από τις προηγούμενες κλάσεις παράγει αντικείμενα σύμφωνα με αυτό που δηλώνει με το όνομα της. Η Epixirisi.java θα παράγει αντικείμενα που υποδηλώνουν φυσικές επιχειρήσεις της Καλύμνου κτλ.

Η εξαίρεση βρίσκεται στην κλάση Region, η οποία υποδηλώνει αντικείμενο-περιοχή του νησιού. Όπως και η GeneralClass έτσι και η Region δεν θα παράγει απευθείας αντικείμενα περιοχές, μιας και την εργασία αυτή την έχουν αναλάβει οι κλάσεις απογόνοι της Region Sight.java, Beach.java και Village.java.

Είναι προφανές πως στις κλάσεις γόνους της GeneralClass έχουν δημιουργηθεί νέα πεδία και μεθόδοι, τα οποία εξυπηρετούν τα ειδικότερα χαρακτηριστικά των αντικειμένων που θα παραχθούν, εφόσον αυτό έχει κριθεί απαραίτητο.

Favourites.java

```
public class Favourites {

	private static List<GeneralClass> favouritesList = new
ArrayList<GeneralClass>();
	private static List<GeneralClass> allList = new
ArrayList<GeneralClass>();
```

```
	public static List<GeneralClass> getFavouritesList(){
		return Favourites.favouritesList;
	}

	public static void setFavouritesList(List<GeneralClass> list){
		Favourites.favouritesList=list;
	}

	public static void addObjectToList(GeneralClass object){
		Favourites.favouritesList.add(object);
	}

	public static void removeObjectFromList(GeneralClass object){
		Favourites.favouritesList.remove(object);
	}

	public static List<GeneralClass> getAllList(){

	Favourites.allList.add(com.Skemelio.kalymnosguide.MyActivity.BAR_CAN
TINA);

	Favourites.allList.add(com.Skemelio.kalymnosguide.MyActivity.BAR_DOM
US);

	Favourites.allList.add(com.Skemelio.kalymnosguide.MyActivity.BAR_SCO
RPION);

	Favourites.allList.add(com.Skemelio.kalymnosguide.MyActivity.OUZERI_
GIANNIS);

		return Favourites.allList;
	}

}
```

Επεξήγηση κώδικα κλάσης Favourites.java (δεν προβάλλεται ολόκληρος ο κώδικας): Η κλάση Favourites αναλαμβάνει να συγκεντρώσει μία λίστα με τις αγαπημένες επιλογές (αντικειμένων GeneralClass) του χρήστη στην εφαρμογή KalymnosGuide.

Περιέχει δύο πεδία, το ένα είναι η λίστα favorites List<GeneralClass> favouritesList, ενώ το άλλο μία λίστα, ίδιου τύπου, όπου συγκεντρώνονται όλα τα αντικείμενα της εφαρμογής KalymnosGuide τα οποία δύναται να χαρακτηριστούν ως favorites.

Όταν ο χρήστης χρήσει ως favorite ένα αντικείμενο GeneralClass, κατά τη διάρκεια πλοήγησης του στην εφαρμογή, αυτό θα προστίθεται στο πεδίο Favourites.favouritesList. Κάθε δραστηριότητα θα έχει τη δυνατότητα ανάκτησης της λίστας των αγαπημένων μέσω της Favourites.getFavouritesList( ).

Την ίδια ώρα που η χρησιμότητα της λίστας των αγαπημένων είναι προφανής στον αναγνώστη, γεννάται η απορία σχετικά με την λίστα Favourites.allList, η οποία συγκεντρώνει όλα τα αντικείμενα που δύναται να χαρακτηρισθούν αγαπημένα. Η εξήγηση έχει ως εξής: όταν ο χρήστης κλείσει την εφαρμογή, αποθηκεύονται οι επιλογές που έχει χαρακτηρίσει ως αγαπημένες. Την στιγμή που θα την επανεκκινίσει ο χρήστης,

συγκεκριμένα στην δραστηριότητα MainMenu.java, η εφαρμογή "σκανάρει" όλα τα αντικείμενα της εφαρμογής (μέσω της λίστας Favourites.allList) και χαρακτηρίζει ως αγαπημένα τα αντικείμενα σύμφωνα με τις αποθηκευμένες αλλαγές. Πιο συγκεκριμένα θέτει το πεδίο boolean isFavorite σε true, εφόσον το αντικείμενο έχει χαρακτηριστεί από τον χρήστη ως αγαπημένο.

FavoriteAdapter.java

```
public class FavoriteAdapter extends ArrayAdapter<GeneralClass> {
	private Context context;
	private List<GeneralClass> list;

	public FavoriteAdapter(Context context, int resource,
			List<GeneralClass> objects) {
		super(context, resource, objects);
		this.context=context;
		this.list=objects;
	}


	@Override
	public View getView(final int position, View convertView, ViewGroup
parent) {

		LayoutInflater inflater = (LayoutInflater)
getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
		View singleRow =
inflater.inflate(R.layout.favoritedialog_single_row, parent, false);

		ImageView icon = (ImageView)
singleRow.findViewById(R.id.imageview_icon_favorite_singlerow);

	icon.setImageResource(this.list.get(position).getIconResource());

		TextView name = (TextView)
singleRow.findViewById(R.id.textview_favorite_singlerow);
		name.setText(this.list.get(position).getName());


		ImageView heart = (ImageView)
singleRow.findViewById(R.id.imageview_heart_favorite_singlerow);
		heart.setImageResource(R.drawable.ic_action_favorite_filled);

		return singleRow;

	}
}
```

Επεξήγηση κώδικα κλάσης FavoriteAdapter.java: Η κλάση FavoriteAdapter κληρονομεί την ArrayAdapter<GeneralClass>, οπότε ο σκοπός της είναι να δώσει μορφή στις γραμμές της ListView που θα προβάλει τις αγαπημένες επιλογές του χρήστη.

Η μόνη εργασία που πραγματοποιήθηκε στην συγκεκριμένη κλάση είναι το «override» της getView( ), μέθοδος η οποία επιστρέφει την View (android.View) που θα δίνει μορφή σε κάθε γραμμή αυτής της ListView.

Η διαδικασία έχει ως εξής: αφού ανακτηθεί μέσα στην getView( ) ένα αντικείμενο LayoutInflater μπορούμε να δημιουργούμε την View που θα αντιπροσωπεύει την κάθε γραμμή της ListView, μέσω της μεθόδου inflate().

View singleRow = inflater.inflate(R.layout.*favoritedialog_single_row*, parent, false);

Η μέθοδος inflate επιστρέφει την ανάλογη View από τον πόρο layout που της δώσαμε ως παράμετρο, τον favoritedialog_single_row. Το αρχείο favoritedialog_single_row.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/root_favorite_singlerow"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_light"
    android:isScrollContainer="true">

    <ImageView
        android:id="@+id/imageview_icon_favorite_singlerow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:layout_alignParentLeft="true"
        android:padding="10dp" />

    <TextView
        android:id="@+id/textview_favorite_singlerow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:typeface="serif"
        android:padding="10dp"
        android:layout_toRightOf="@+id/imageview_icon_favorite_singlerow"/>

    <ImageView
        android:id="@+id/imageview_heart_favorite_singlerow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:padding="10dp"
        android:layout_alignParentRight="true" />

</RelativeLayout>
```

Όπως παρατηρούμε ο πόρος περιέχει μία ImageView και δύο TextView, οι οποίες θα αποκτήσουν περιεχόμενο κατά τη διάρκεια του χρόνου εκτέλεσης. Το περιεχόμενο αυτό έχει καθοριστεί στην getView( ). Ακολουθεί η διαδικασία μόνο για την ImageView μιας και η υλοποίηση είναι παρόμοια για τις άλλες δύο View.

ImageView icon = (ImageView) singleRow.findViewById(R.id.*imageview_icon_favorite_singlerow*);

Αφού έχουμε εντοπίσει την ImageView μέσω της μεθόδου View.findViewById( ) της δίνουμε περιεχόμενο.

icon.setImageResource(this.list.get(position).getIconResource());

Η λίστα "list" που έχει μπει ως παράμετρος είναι η λίστα με τις αγαπημένες επιλογές, οπότε this.list.get(position) μας δίνει το αντικείμενο-favorite το οποίο βρίσκεται στην θέση με αριθμό «position». Αφού ανακτηθεί αυτό το αντικείμενο τότε καλούμε την.getIconResource() ώστε να ανακτηθεί ο πόρος της εικόνας που εν τέλει θα δώσει στο αντικείμενο το εικονίδιο που του αναλογεί πλάι στη λίστα, κατά το χρόνο εκτέλεσης.

Πακέτο com.Skemelio.kalymnosguide
MyActivity.java

```
public class MyActivity extends FragmentActivity {

	//Intent Extras
	public static final String IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU="image
resource";
	protected static final String
TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU="text resource";
	protected static final String
TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST="text listview item aboutlist";


	//SharedPrefences file names
	protected static final String SHARED_PREFENCES_FAVORITE =
"SharedPrefs for GeneralClass.this.isFavourite";
	protected static final String SHARED_PREFENCES_FAVORITE_KEY =
"SharedPrefs for GeneralClass.this.isFavourite key for: ";

	// GeneralClass Objects: Bars, Restaurants, Places and shit...
	public static final Epixirisi BAR_DOMUS= new Epixirisi("Domus",
"Kantouni", "This is Domus Brief", R.drawable.ic_icon_cocktail,
R.drawable.bars);
	public static final Epixirisi BAR_CANTINA= new Epixirisi("Cantina",
"Kantouni", "This is Cantinas Brief", R.drawable.ic_icon_beer_mug,
R.drawable.bars);


	//Επεξηγήσεις (fills the epexigisi.xml)
	protected static final GeneralClass EPEXIGISI_ANCIENT_KALYMNOS= new
GeneralClass("Ancient Kalymnos", null, "This is Ancient Kalymnos Brief", 0,
R.drawable.ancient_kalymnos);
	protected static final GeneralClass EPEXIGISI_MEDIEVAL_KALYMNOS= new
GeneralClass("Medieval Kalymnos", null, "This is Medieval Kalymnos Brief",
0, R.drawable.medieval_kalymnos);

}
```

Επεξήγηση κώδικα κλάσης MyActivity.java (δεν προβάλλεται ολόκληρος ο κώδικας): Η κλάση MyActivity κληρονομεί την FragmentActivity και είναι η πατρική κλάση όλων των δραστηριοτήτων της εφαρμογής, με σκοπό οι τελευταίες να κληρονομούν κοινά χαρακτηριστικά από την κλάση πρόγονο.

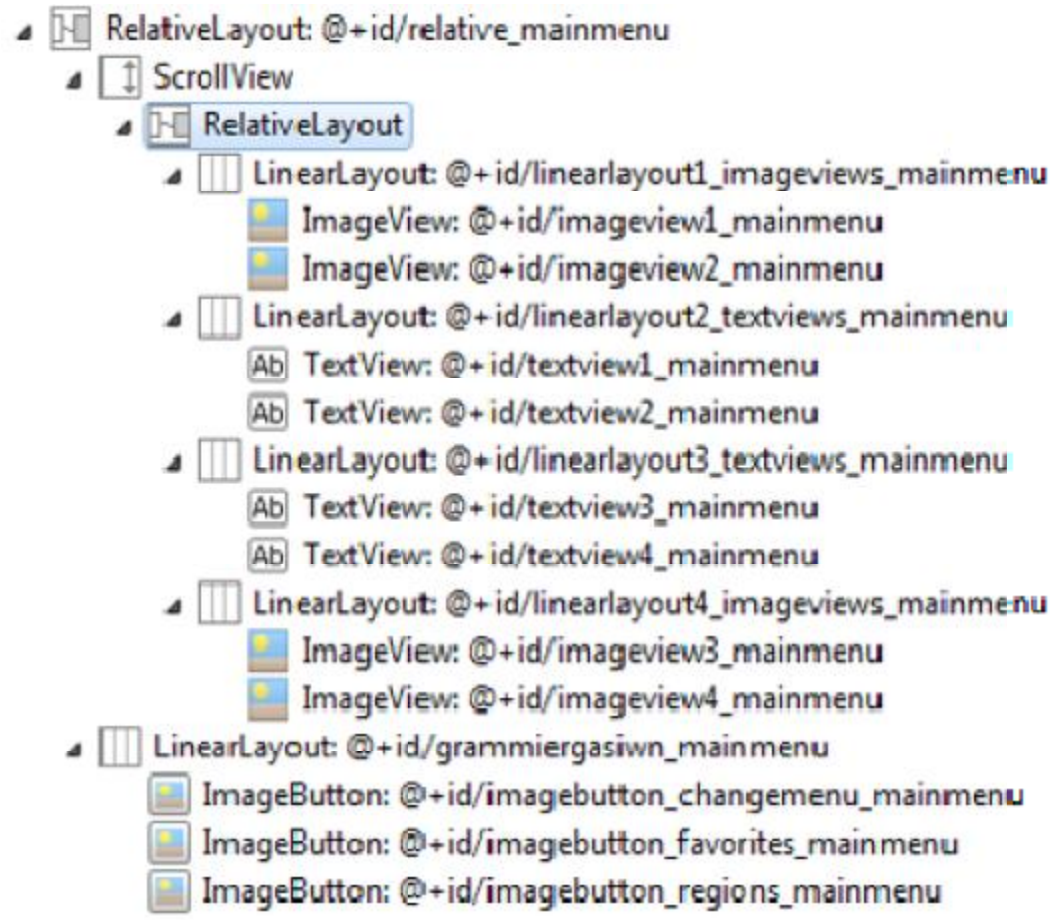
Στην MyActivity θα βρούμε όλες τις χρήσιμες σταθερές της εφαρμογής, ξεκινώντας από τα αντικείμενα GeneralClass της εφαρμογής τα οποία δημιουργούνται εδώ, και στη συνέχεια περνώντας σε σταθερές οι οποίες βοηθούν στην υλοποίηση χαρακτηριστικών της εφαρμογής, όπως τα Intent Extras και τα SharedPreferences file names.

Η κλάση MyActivity περιέχει δύο επιπλέον εσωτερικές κλάσεις, την TabHostListViewItem και την SubMainMenuItems, οι οποίες δεν προβάλλονται στο συγκεκριμένο τμήμα κώδικα. Οι δύο αυτές εσωτερικές κλάσης βοηθούν στην υλοποίηση ορισμένων ListView των δραστηριοτήτων της AboutList.java και της SubMainMenu.java αντιστοίχως.

Και οι δύο inner classes υπό ορισμένη συνθήκη αρχικοποιούν μία λίστα List<String> η οποία μπορεί να ανακτηθεί μέσω μεθόδων των αντικειμένων τους.

Σημείωση*: Στη συνέχεια θα προβληθούν, χάρην συντομίας, οι δραστηριότητες οι οποίες εκπληρούν την κυρίως διαδρομή της εφαρμογής KalymnosGuide. Ποια είναι αυτή; Από την πλοήγηση των κυρίως μενού και υπομενού ως την λεπτομερή εμφάνιση των πληροφοριών ενός αντικειμένου GeneralClass. Επί παραδείγματι, από το μενού "Eat and Drink" μέχρι την προβολή του εστιατορείου «Ο Παντελής». Τέλος στην επεξήγηση των δραστηριοτήτων θα ξεκινάμε πρώτα από το αρχείο πόρου διεπαφής χρήστη και κατόπιν θα αναλύεται το αρχείο της δραστηριότητας*

MainMenu.java(activity) + mainmenu.xml(layout resource)



Εικόνα 3-1: **Το** outline **του πόρου** mainmenu.xml

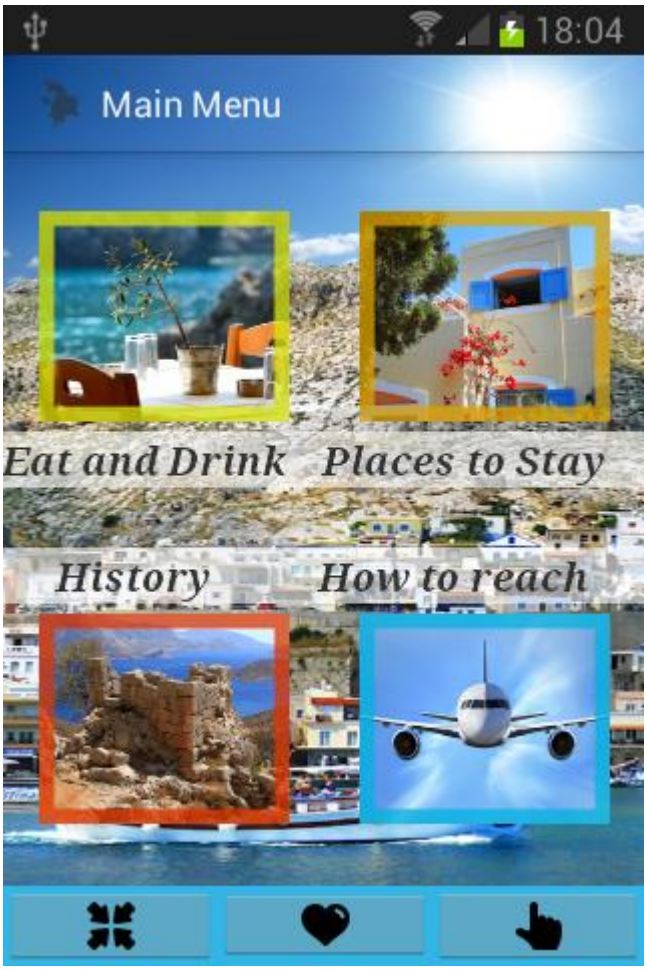
Επεξήγηση mainmenu.xml: Στην περίπτωση της επεξήγησης των δραστηριοτήτων θα ξεκινάμε πρώτα από το αρχείο πόρου διεπαφής χρήστη και κατόπην θα αναλύουμε την δραστηριότητα.

Στην πρώτη εικόνα φαίνεται η δραστηριότητα MainMenu η μορφή της οποία δίνεται από τον πόρο διεπαφής χρήστη mainmenu.xml.

Η εικόνα ακριβώς από πάνω εμφανίζει το σχετικό outline του πόρου mainmenu.xml. Το πατρικό layout του πόρου είναι ένα RelativeLayout το οποίο χωρίζεται σε δύο μέρη. Μία ScrollView για το κυρίως περιεχόμενο της δραστηριότητας και μία LinearLayout για την γραμμή εργαλείων στο κάτω μέρος. Ως αποτέλεσμα αν η δραστηριότητα προβληθεί σε συσκευή με μικρότερη οθόνη από την παρούσα ο χρήστης θα έχει τη δυνατότητα scrolling.

Το scrollView περιέχει μία RelativeLayout η οποία χωρίζεται σε τέσσερις οριζόντιες LinearLayout (android:orientation:"horizontal"). Η πρώτη LinearLayout περιέχει δύο ImageView, η επόμενη LinearLayout περιέχει δύο TextView οι οποίες είναι οι φέρουν κείμενο περιγραφής των αντίστοιχων ImageView της πρώτης LinearLayout. Αυτή η διαδικασία επαναλαμβάνεται ακόμα μία φορά, για δύο ακόμα LinearLayout. Το περιεχόμενο αυτών των View (ImageView-TextView) θα δοθεί κατά τον χρόνο εκτέλεσης, προγραμματιστικά.

Τέλος, η LinearLayout με id="grammiergasiwn_mainmenu", η γραμμή εργασιών δηλαδή που υπάρχει σχεδόν σε όλες τις δραστηριότητες πλην αυτές που προβάλλουν χάρτη, περιέχει τρία ImageButton. Ο καθορισμός του ρόλου των τελευταίων είναι δουλειά της εκάστοτε δραστηριότητας.



Εικόνα 3-2: Screenshot **της δραστηριότητας** MainMenu.java

```
public class MainMenu extends MyActivity {

	private ImageView imageView1, imageView2,imageView3,imageView4;
	private TextView textView1,textView2,textView3,textView4;
	private RelativeLayout background;
	private LinearLayout grammiErgasiwn;
	private ImageButton changeMenu, favorites, regions;
	private static int metritis = 0;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		this.setContentView(R.layout.mainmenu);
		this.loadAllFavoriteResults(Favourites.getAllList());
		this.initActionBar();
		this.initViews();
		this.initViewListeners();
		this.initFavoriteListListener();
	}
```

```
	@Override
	public void onBackPressed() {
		// TODO Auto-generated method stub
		super.onBackPressed();
	}

	private void loadAllFavoriteResults(List<GeneralClass> list) {
		//open the sharedpreferences to load the boolean values
		SharedPreferences sharedPrefs =
getSharedPreferences(MyActivity.SHARED_PREFENCES_FAVORITE,
Context.MODE_PRIVATE);

		//iterate the collection and set the favourite results
		for (Iterator iterator = list.iterator(); iterator.hasNext();)
{
			GeneralClass antikeimeno = (GeneralClass)
iterator.next();

	antikeimeno.setFavoriteResult(sharedPrefs.getBoolean(MyActivity.SHAR
ED_PREFENCES_FAVORITE_KEY+antikeimeno.getName(), false));
			if(antikeimeno.getFavoriteResult()){

	if(!Favourites.getFavouritesList().contains(antikeimeno)){

	Favourites.getFavouritesList().add(antikeimeno);
					Toast.makeText(this,
antikeimeno.getName()+" loaded", Toast.LENGTH_SHORT).show();
				}
			}
		}
	}

	private void initFavoriteListListener() {


		this.favorites.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {

				Dialog dialog = new Dialog(MainMenu.this);
				dialog.setTitle(R.string.favorites);

				if(Favourites.getFavouritesList().size()==0){

	dialog.setContentView(R.layout.favoritedialog_directions);
					dialog.show();
				}else{

	dialog.setContentView(R.layout.favoritedialog_listview);
					ListView listview = (ListView)
dialog.findViewById(R.id.listview_favorites);
					listview.setAdapter(new
FavoriteAdapter(MainMenu.this, R.layout.favoritedialog_listview,
Favourites.getFavouritesList()));
					listview.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
```

```
                                    @Override
                                    public                          void
onItemClick(AdapterView<?> parent, View view,
                                                    int position, long id)
{

                                                TextView text = (TextView)
view.findViewById(R.id.textview_favorite_singlerow);
                                                String          name          =
text.getText().toString();

                                                for (Iterator iterator =
Favourites.getAllList().iterator(); iterator
                                                        .hasNext();) {

                                                GeneralClass
antikeimeno = (GeneralClass) iterator.next();


        if(name.equals(antikeimeno.getName())){



                                                        Intent intent =
new Intent();

        intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
name);
                                                        //add the flag
so all of the other activities on top of it will be closed and this Intent
will be delivered to the (now on top) old activity as a new Intent.

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                                        // ελέγχουμε τον
τύπου του antikeimeno και ανοίγουμε την κατάλληλη δραστηριότητα
                                                        //(δεν        μου
δούλεψε με το instanceof)

        if(antikeimeno.getClass().equals(Epixirisi.class)){

        intent.setClass(MainMenu.this, EpixirisiActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
1);

        MainMenu.this.startActivity(intent);
                                                        }else
if(antikeimeno.getClass().equals(Beach.class)){

        intent.setClass(MainMenu.this, BeachActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
1);

        MainMenu.this.startActivity(intent);
                                                        }else
if(antikeimeno.getClass().equals(Sight.class)){
```

```
        intent.setClass(MainMenu.this, SightActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
1);

        MainMenu.this.startActivity(intent);
                                                              }else
if(antikeimeno.getClass().equals(Village.class)){

        intent.setClass(MainMenu.this, VillageActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
1);

        MainMenu.this.startActivity(intent);
                                                              }



                                                        }
                                                    }

                                                }
                                            });
                                            dialog.show();
                                        }

                    }
            });
        }

        private void initActionBar() {
            // TODO Auto-generated method stub
            ActionBar bar = this.getActionBar();
            bar.setTitle(R.string.main_menu);
            bar.setIcon(R.drawable.ic_action_kalymnoslogotransparent);

        bar.setBackgroundDrawable(this.getResources().getDrawable(R.drawable
.sky));
        }


        private void initViews(){

            //Instantiation
            imageView1                      =                     (ImageView)
this.findViewById(R.id.imageview1_mainmenu);
            imageView2                      =                     (ImageView)
this.findViewById(R.id.imageview2_mainmenu);
            imageView3                      =                     (ImageView)
this.findViewById(R.id.imageview3_mainmenu);
            imageView4                      =                     (ImageView)
this.findViewById(R.id.imageview4_mainmenu);


            textView1                       =                      (TextView)
this.findViewById(R.id.textview1_mainmenu);
```

```
            textView2                       =                       (TextView)
this.findViewById(R.id.textview2_mainmenu);
            textView3                       =                       (TextView)
this.findViewById(R.id.textview3_mainmenu);
            textView4                       =                       (TextView)
this.findViewById(R.id.textview4_mainmenu);


            background                  =                   (RelativeLayout)
this.findViewById(R.id.relative_mainmenu);
            grammiErgasiwn                  =                (LinearLayout)
this.findViewById(R.id.grammiergasiwn_mainmenu);

            changeMenu                      =                    (ImageButton)
this.findViewById(R.id.imagebutton_changemenu_mainmenu);
            favorites=                                           (ImageButton)
this.findViewById(R.id.imagebutton_favorites_mainmenu);
            regions                         =                    (ImageButton)
this.findViewById(R.id.imagebutton_regions_mainmenu);
      }


      @SuppressLint("NewApi") private void initViewListeners(){

//            //changeMenu
            changeMenu.setOnClickListener(new View.OnClickListener() {

                  @SuppressLint("NewApi") @Override
                  public void onClick(View v) {
                        metritis++;

                        if(metritis%2==1){


      background.setBackground(MainMenu.this.getResources().getDrawable(R.
drawable.kalymnos_background2));

      grammiErgasiwn.setBackgroundColor(MainMenu.this.getResources().getCo
lor(android.R.color.holo_orange_light));


      imageView1.setImageResource(R.drawable.islandbeauties);

      imageView2.setImageResource(R.drawable.whattodo);

      imageView3.setImageResource(R.drawable.culture);

      imageView4.setImageResource(R.drawable.usefullinfo);


      textView1.setText(R.string.islandbeauties);
                              textView2.setText(R.string.whattodo);
                              textView3.setText(R.string.culture);
                              textView4.setText(R.string.usefullinfo);


      MainMenu.this.getActionBar().setBackgroundDrawable(MainMenu.this.get
Resources().getDrawable(R.drawable.sky2));
```

```
                        }else{


        background.setBackground(MainMenu.this.getResources().getDrawable(R.
drawable.kalymnos_background));

        grammiErgasiwn.setBackgroundColor(MainMenu.this.getResources().getCo
lor(android.R.color.holo_blue_light));


        imageView1.setImageResource(R.drawable.eatanddrink);

        imageView2.setImageResource(R.drawable.placestostay);

        imageView3.setImageResource(R.drawable.history);

        imageView4.setImageResource(R.drawable.howtoreach);

                                textView1.setText(R.string.eatanddrink);
                                textView2.setText(R.string.placestostay);
                                textView3.setText(R.string.history);
                                textView4.setText(R.string.howtoreach);


        MainMenu.this.getActionBar().setBackgroundDrawable(MainMenu.this.get
Resources().getDrawable(R.drawable.sky));

                        }

                }
        });
        //end of changeMenu


        //imageview listeners
        imageView1.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {

                        //Γιατί η imageView1 παίρνει και τις 2 εικόνες
                        //οπότε και το island beauties αν ανοίξεις θα σου
βγάλει το περιεχόμενο του eat and drink

        if(textView1.getText().toString()==MainMenu.this.getResources().getS
tring(R.string.eatanddrink)){

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

        intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.eatanddrink);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);

                        }else{
```

```
                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

      intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.islandbeauties);

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);
                        }

                }
        });


        imageView2.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {


      if(textView2.getText().toString()==MainMenu.this.getResources().getS
tring(R.string.placestostay)){

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

      intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.placestostay);

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);

                        }else{

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

      intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.whattodo);

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);
                        }

                }
        });


        imageView3.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {


      if(textView3.getText().toString()==MainMenu.this.getResources().getS
tring(R.string.history)){

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);
```

```
        intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.history);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);

                            }else{

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

        intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.culture);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);
                            }

                    }
                });


                imageView4.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {


        if(textView4.getText().toString()==MainMenu.this.getResources().getS
tring(R.string.howtoreach)){

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

        intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.howtoreach);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);

                            }else{

                                Intent intent = new Intent(MainMenu.this,
SubMainMenu.class);

        intent.putExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.usefullinfo);

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                MainMenu.this.startActivity(intent);
                            }

                    }
                });



                //regions listener (map)
```

```
            this.regions.setOnClickListener(new View.OnClickListener() {

                  @Override
                  public void onClick(View v) {

                        MainMenu.this.startActivity(new
Intent(MainMenu.this, OneMapActivity.class));

                  }
            });


      }


}//end of class
```

Επεξήγηση κώδικα κλάσης MainMenu.java: Η MainMenu είναι η δραστηριότητα εκκίνησης της εφαρμογής KalymnosGuide. Αυτό μπορούμε να το διαπιστώσουμε και στην δήλωση της στο AndroidManifest.xml, όπου περιέχει ένα Intent-filter το οποίο καθορίζει την παρακάτω δήλωση:

```
<intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Κατά τη διάρκεια της αρχικοποίησης της MainMenu (μέθοδος onCreate( )) ορίζεται ως διεπαφή χρήστη ο πόρος mainmenu.xml μέσω της μεθόδου this.setContentView(R.layout.mainmenu);.

Στη συνέχεια καλείται η μέθοδος loadAllFavoritesResults( ) η οποία «φορτώνει» τις αγαπημένες επιλογές του χρήστη. Η loadAllFavoritesResults λαμβάνει ως παράμετρο μία λίστα , τύπου List<GeneralClass>, η οποία περιλαμβάνει όλα τα αντικείμενα της εφαρμογής τα οποία δύναται να χρηστούν ως favorite.

Η επόμενη εργασία της loadAllFavoritesResults( ) είναι το «ψιλάφισμα» αυτών των αντικειμένων ώστε να ξεκαθαριστεί εάν κάποιο από αυτά είχε καθοριστεί από τον χρήστη ως αγαπημένο (αυτή η εργασία πραγματοποιείται από τον χρήστη στις δραστηριότητες ειδικού περιεχομένου όπως η EpixirisiActivity, BeachActivity κτλ). Στην περίπτωση που κάποιο GeneralClass αντικείμενο είναι “favorite” τότε εάν δεν υπάρχει στην συνολική λίστα των αγαπημένων επιλογών (utilities.Favourites.favoritesList) προστίθεται εκεί. Όταν φορτωθούν τα αποτελέσματα αυτά θα χρησιμοποιούνται στη συνέχεια από τις υπόλοιπες δραστηριότητες ειδικού περιεχομένου.

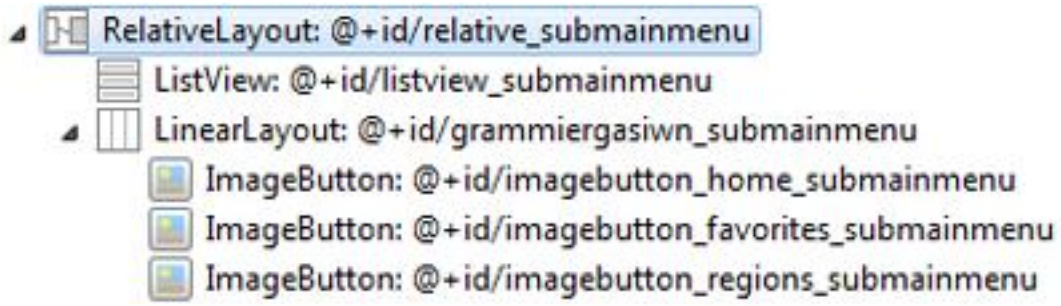
Ακολουθούν αρχικοποίηση πεδίων και γέμισμα των View με σχετικούς πόρους, στυλιζάρισμα της ActionBar κτλ.

Το επόμενο σημαντικό βήμα της MainMenu υλοποιείται στη δραστηριότητα initViewListeners( ). Όπως προϊδεάζει και το όνομα της μεθόδου, εδώ υλοποιούνται οι ακροατές ενεργειών των Views. Αρχικά υλοποιείται ο κώδικας για την ενέργεια του

ImageButton της γραμμής εργαλείων το οποίο αλλάζει το μενού. Στη για τον listener της κάθε ImageView η εφαρμογή εργάζεται ως εξής: ελέγχεται το αντίστοιχο της TextView, δημιουργείται μία πρόθεση Intent στην οποία αποθηκεύεται η αναφορά του πόρου που χρησιμοποιείται ως εικόνα περιεχομένου στην ImageView και η εφαρμογή μεταβαίνει στην SubMainMenu. Στην ίδια μέθοδο καθορίζεται και ο listener του ImageButton region της γραμμής εργασιών, όπου πατώντας το μεταβαίνουμε στην OneMapActivity.

Τέλος, καλείται η initFavoriteListListener( ) η οποία αρχικοποιεί τον listener του ImageButton favorites, όπου ανοίγει το παράθυρο διαλόγου με τη λίστα των αγαπημένων επιλογών του χρήστη. Η μέθοδος δημιουργεί ένα παράθυρο διαλόγου και ελέγχει αν η λίστα με τα αγαπημένα utilities.Favourites.favoritesList περιέχει αντικείμενα. Στην περίπτωση που δεν περιέχει κανένα (είναι null) τότε εμφανίζεται ένα παράθυρο διαλόγου με ένα κείμενο που επεξηγεί την διαδικασία χρήσης ενός αντικειμένου ως favorite. Αντίθετα, αν η utilities.Favourites.favoritesList περιέχει αντικείμενα προβάλλεται ένα παράθυρο διαλόγου το οποίο εμφανίζει μία ListView με τις αγαπημένες επιλογές. Κλικάροντας ο χρήστης σε μία από τις επιλογές αυτές η εφαρμογή θα αντιστοιχίσει το κείμενο της γραμμής από την ListView με το όνομα του αντικειμένου και θα μεταβεί στην κατάλληλη δραστηριότητα η οποία προβάλλει το κείμενο.

SubMainMenu.java(activity) + submainmenu.xml(layout resource)



Εικόνα 3-3: **Το** outline **του πόρου** submainmenu.xml

Επεξήγηση κώδικα submainmenu.xml: Ο πόρος της δραστηριότητας SubMainMenu είναι απλούστερος από αυτόν της MainMenu, όπως φαίνεται και στο σχετικό outline (άνω εικόνα).

Κύριο layout και εδώ είναι μία RelativeLayout η οποία εμπεριέχει μία ListView και την συνήθης LinearLayout που εμφανίζει τα ImageButtons της γραμμής εργαλείων.

Εικόνα 3-4: Screenshot **της δραστηριότητας** SubMainMenu.java

```
public class SubMainMenu extends MyActivity {

	private ListView listView;
	private RelativeLayout background;
	private LinearLayout grammiErgasiwn;
	private ImageButton home, favorites, regions;
	private int imageResourceCode;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		this.setContentView(R.layout.submainmenu);
		this.initViews();
		this.initActionBar(imageResourceCode);
		this.initListView(imageResourceCode);
		this.initBackgroundColor(imageResourceCode);
		this.initGrammiErgasiwnColor(imageResourceCode);
		this.initViewListeners();
		this.initListViewItemListener();
		this.initFavoriteListListener();
	}


	private void initFavoriteListListener() {
```

```
                    this.favorites.setOnClickListener(new
View.OnClickListener() {

                            @Override
                            public void onClick(View v) {

                                    Dialog         dialog         =         new
Dialog(SubMainMenu.this);
                                    dialog.setTitle(R.string.favorites);


       if(Favourites.getFavouritesList().size()==0){

       dialog.setContentView(R.layout.favoritedialog_directions);
                                        dialog.show();
                                    }else{

       dialog.setContentView(R.layout.favoritedialog_listview);
                                        ListView   listview   =   (ListView)
dialog.findViewById(R.id.listview_favorites);
                                        listview.setAdapter(new
FavoriteAdapter(SubMainMenu.this,          R.layout.favoritedialog_listview,
Favourites.getFavouritesList()));
                                        listview.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

                                                @Override
                                                public                    void
onItemClick(AdapterView<?> parent, View view,
                                                                    int     position,
long id) {

                                                    TextView     text     =
(TextView) view.findViewById(R.id.textview_favorite_singlerow);
                                                    String      name      =
text.getText().toString();

                                                    for  (Iterator  iterator
= Favourites.getAllList().iterator(); iterator

       .hasNext();) {

                                                            GeneralClass
antikeimeno = (GeneralClass) iterator.next();


       if(name.equals(antikeimeno.getName())){



                                                                    Intent
intent = new Intent();

       intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
name);
```

```
                                                                    //add the
flag so all of the other activities on top of it will be closed and this
Intent will be delivered to the (now on top) old activity as a new Intent.

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                                                    //
ελέγχουμε τον τύπου του antikeimeno2 και ανοίγουμε την κατάλληλη
δραστηριότητα
                                                                    //(δεν μου
δούλεψε με το instanceof)

        if(antikeimeno.getClass().equals(Epixirisi.class)){

        intent.setClass(SubMainMenu.this, EpixirisiActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
2);

        SubMainMenu.this.startActivity(intent);
                                                                    }else
if(antikeimeno.getClass().equals(Beach.class)){

        intent.setClass(SubMainMenu.this, BeachActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
2);

        SubMainMenu.this.startActivity(intent);
                                                                    }else
if(antikeimeno.getClass().equals(Sight.class)){

        intent.setClass(SubMainMenu.this, SightActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
2);

        SubMainMenu.this.startActivity(intent);
                                                                    }else
if(antikeimeno.getClass().equals(Village.class)){

        intent.setClass(SubMainMenu.this, VillageActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
2);

        SubMainMenu.this.startActivity(intent);
                                                                    }



                                                                }
                                                            }

                                                        }
                                                    });
                                                    dialog.show();
                                                }

                                            }
```

```
                });
        }

    private void initViews(){

        listView                    =                    (ListView)
this.findViewById(R.id.listview_submainmenu);
        background                  =                (RelativeLayout)
this.findViewById(R.id.relative_submainmenu);
        grammiErgasiwn              =                 (LinearLayout)
this.findViewById(R.id.grammiergasiwn_submainmenu);

        home                    =                     (ImageButton)
this.findViewById(R.id.imagebutton_home_submainmenu);
        favorites=                                    (ImageButton)
this.findViewById(R.id.imagebutton_favorites_submainmenu);
        regions                 =                     (ImageButton)
this.findViewById(R.id.imagebutton_regions_submainmenu);

        imageResourceCode                                          =
this.getIntent().getIntExtra(IMAGE_RESOURCE_EXTRA_KEY_MAIN_MENU,
R.drawable.eatanddrink);
    }


    private void initActionBar(int imageResource){

        ActionBar bar = this.getActionBar();
        bar.setDisplayHomeAsUpEnabled(true);
        bar.setIcon(R.drawable.ic_action_kalymnoslogotransparent);

        switch (imageResource) {
        case R.drawable.eatanddrink:

            bar.setTitle(R.string.eatanddrink);

    bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky));

            break;
        case R.drawable.placestostay:

            bar.setTitle(R.string.placestostay);

    bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky));

            break;
        case R.drawable.history:

            bar.setTitle(R.string.history);

    bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky));

            break;
        case R.drawable.howtoreach:

            bar.setTitle(R.string.howtoreach);
```

```
	bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky));

			break;
		case R.drawable.islandbeauties:

			bar.setTitle(R.string.islandbeauties);

	bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky2));

			break;
		case R.drawable.whattodo:

			bar.setTitle(R.string.whattodo);

	bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky2));

			break;
		case R.drawable.culture:

			bar.setTitle(R.string.culture);

	bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky2));

			break;
		case R.drawable.usefullinfo:

			bar.setTitle(R.string.usefullinfo);

	bar.setBackgroundDrawable(SubMainMenu.this.getResources().getDrawabl
e(R.drawable.sky2));

			break;

		default:
			break;
		}

	}


	private void initListView(int imageResource){

		MyActivity.SubMainMenuItems	antikeimeno	=	new
SubMainMenuItems(imageResource);
		List<String> items = antikeimeno.getSubMainMenuItems();
		ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, items);

		if(items!=null){
			listView.setAdapter(adapter);
		}else{
			Toast.makeText(SubMainMenu.this, "Η λίστα είναι null",
Toast.LENGTH_LONG).show();
		}
```

```
            switch (imageResource) {

                  case R.drawable.eatanddrink:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.eatanddrink_transparent40));
                        break;

                  case R.drawable.placestostay:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.placestostay_transparent40));
                        break;

                  case R.drawable.history:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.history_transparent40));
                        break;

                  case R.drawable.howtoreach:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.howtoreach_transparent40));
                        break;

                  case R.drawable.islandbeauties:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.islandbeauties_transparent40));
                        break;

                  case R.drawable.whattodo:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.whattodo_transparent40));
                        break;

                  case R.drawable.culture:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.culture_transparent40));
                        break;

                  case R.drawable.usefullinfo:

      listView.setBackgroundColor(SubMainMenu.this.getResources().getColor
(R.color.usefullinfo_transparent40));
                        break;


                  default:
                        break;
            }

      }
```

```
	private void initListViewItemListener(){

		listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

			@Override
			public void onItemClick(AdapterView<?> parent, View
item, int position,
					long iD) {

				TextView textView = (TextView) item;
				String itemText = textView.getText().toString();
				if(itemText.equals("Bars") ||
itemText.equals("Try Ouzo") || itemText.equals("Enjoy your Coffee") ||
itemText.equals("Greek Taverns")
					|| itemText.equals("Greek Souvlaki")
||
					itemText.equals("Restaurants") ||
itemText.equals("Pizza-Spaghetti") || itemText.equals("Patisseries")
					|| itemText.equals("Find a Hotel")
|| itemText.equals("Rent a Room") || itemText.equals("Travel Agencies") ||
					itemText.equals("Wonderful Beaches")
||	itemText.equals("Breathtaking Sightseeings") ||
itemText.equals("Picturesque Villages")
					|| itemText.equals("Visit Museums")
||
itemText.equals(SubMainMenu.this.getResources().getString(R.string.archaeol
ogical_sites))
					|| itemText.equals("Festivals") ||
itemText.equals("Kalymnian Products") || itemText.equals("Car-Bike
Rentals")){

					Intent intent = new
Intent(SubMainMenu.this, AboutList.class);

	intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
itemText);

	intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
					SubMainMenu.this.startActivity(intent);

				}else if(itemText.equals("ATMs")){

					Toast.makeText(SubMainMenu.this, "Εδώ θα
ξεκινάει η ListActivity με τα ATMs", Toast.LENGTH_LONG).show();

				}else
if(itemText.equals(SubMainMenu.this.getResources().getString(R.string.tradi
tions))){
					Toast.makeText(SubMainMenu.this, "Εδώ θα
ξεκινάει η Tradition Activity", Toast.LENGTH_LONG).show();
				}

				else{

					Intent intent = new
Intent(SubMainMenu.this, Epexigisi.class);
```

```
	intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
itemText);

	intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                SubMainMenu.this.startActivity(intent);

                        }

                }
        });

	}


	private void initBackgroundColor(int imageResource){

		switch (imageResource) {

                case R.drawable.islandbeauties:

	background.setBackgroundResource(R.drawable.kalymnos_background2);
                        break;

                case R.drawable.whattodo:

	background.setBackgroundResource(R.drawable.kalymnos_background2);
                        break;

                case R.drawable.culture:

	background.setBackgroundResource(R.drawable.kalymnos_background2);
                        break;

                case R.drawable.usefullinfo:

	background.setBackgroundResource(R.drawable.kalymnos_background2);
                        break;


                default:
                        break;
	}

	}

	private void initGrammiErgasiwnColor(int imageResource){

		switch (imageResource) {

                case R.drawable.islandbeauties:

	grammiErgasiwn.setBackgroundResource(android.R.color.holo_orange_lig
ht);
                        break;

                case R.drawable.whattodo:
```

```
		grammiErgasiwn.setBackgroundResource(android.R.color.holo_orange_lig
ht);
                        break;

                    case R.drawable.culture:

		grammiErgasiwn.setBackgroundResource(android.R.color.holo_orange_lig
ht);
                        break;

                    case R.drawable.usefullinfo:

		grammiErgasiwn.setBackgroundResource(android.R.color.holo_orange_lig
ht);
                        break;


                    default:
                        break;
                }
            }


            private void initViewListeners(){

                home.setOnClickListener(new OnClickListener() {

                    @Override
                    public void onClick(View v) {

                        SubMainMenu.this.startActivity(new
Intent(SubMainMenu.this, MainMenu.class));
                        SubMainMenu.this.finish();

                    }
                });
            }
}
```

Επεξήγηση κώδικα δραστηριότητας SubMainMenu.java: Η SubMainMenu είναι η δραστηριότητα στην οποία θα μεταβεί η εφαρμογή εφόσον ο χρήστης κλικάρει σε μία από τις επιλογές της MainMenu.

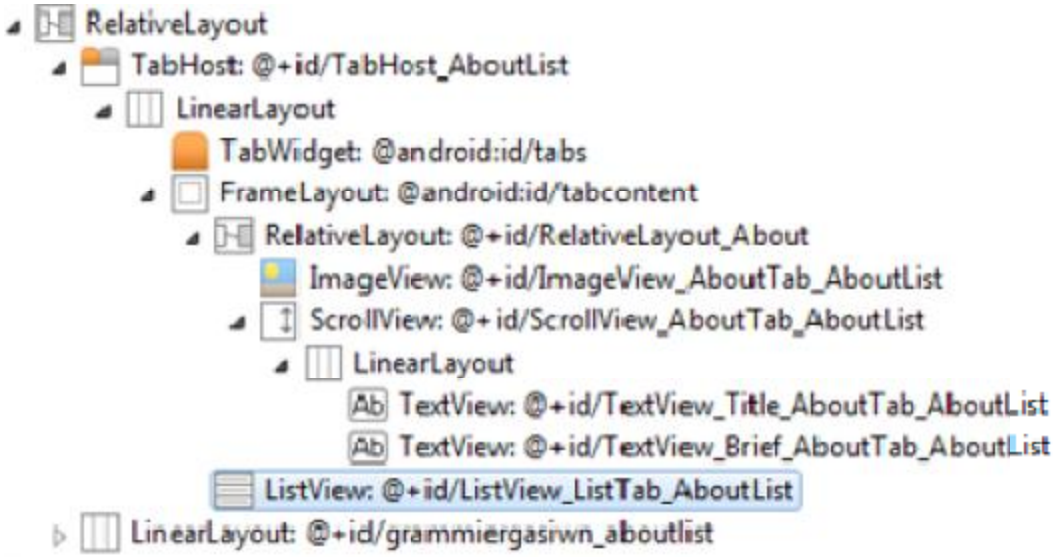
Αρχικά, στην μέθοδο onCreate( ) ορίζεται ως πόρος διεπαφής χρήστη ο submainmenu.xml, μέσω της μεθόδου setLayout( ) και αρχικοποιεί τα πεδία της τύπου View (μέθοδος initViews( ) κάνοντας χρήση της Activity.findViewById( )).

Το σημαντικό βήμα είναι το επόμενο όπου γίνεται η **αρχικοποίηση της** ListView, η οποία καταλαμβάνει το μεγαλύτερο μέρος της οθόνης της δραστηριότητας. Επειδή κάθε επιλογή από το κυρίως μενού θα οδηγήσει σε ένα υπομενού με διαφορετικές επιλογές, έτσι η ListView της SubMainMenu αρχικοποιείται αναλόγως την αρχική επιλογή. Προγραμματιστικά αυτό επιτεύχθηκε μέσω της μεθόδου initListView( ): η πρόθεση Intent η οποία μας έφερε από την MainMenu στην SubMainMenu είχε αποθηκεύσει τον πόρο της εικόνας του που έγινε η επιλογή στο κυρίως μενού. Σύμφωνα με αυτόν τον πόρο αρχικοποιείται και η ListView. Σε αυτό το σημείο γίνεται η χρήση της εσωτερικής κλάσης

MyActivity.SubMainMenuItems, το αντικείμενο της οποίας θα επιστρέψει την λίστα που αντιστοιχεί στην κάθε περίπτωση μέσω της μεθόδου MyActivity.SubMainMenuItems.getSubMainMenuItems( ). Με παρόμοιο τρόπο (με μεθόδους οι οποίοι δέχονται ως παράμετρο την integer αναφορά πόρου της εικόνας που επελέγει από την MainMenu) η SubMainMenu στυλιζάρει την ActioBar και το χρώμα του Background καθώς επίσης και το χρώμα της γραμμής εργασιών μέσω των μεθόδων initActionBar( ), initBackgroundColor( ) και initGrammiErgasiwnColor( ) αντίστοιχα.

Τώρα που έχουμε αρχικοποιημένη την ListView της δραστηριότητας MainMenu το μόνο που μένει να πράξουμε είναι να θέσουμε έναν listener, ο οποίος θα ακροάζεται τα συμβάντα κλικ ξεχωριστά στην κάθε γραμμή της λίστας και αναλόγως με την επιλογή αυτή θα μεταβεί η εφαρμογή στην ανάλογη δραστηριότητα. Η εργασία αυτή υλοποιείται στην μέθοδο initListViewItemListener( ). Εκεί η εφαρμογή αφού λάβει το κείμενο το οποίο εμφανίζεται στην στην γραμμή της λίστας που έχει κλικάρει ο χρήστης το συγκρίνει με ορισμένα String και με αυτό που ταιριάζει δημιουργείται μία πρόθεση Intent η οποία, αφού αποθηκεύσει αυτό το String, μας στέλνει στην AboutList.java ή στην Epexigisi.java. Ανάλογα με την περίπτωση.

AboutList.java(activity) + aboutlist.xml(layout resource)

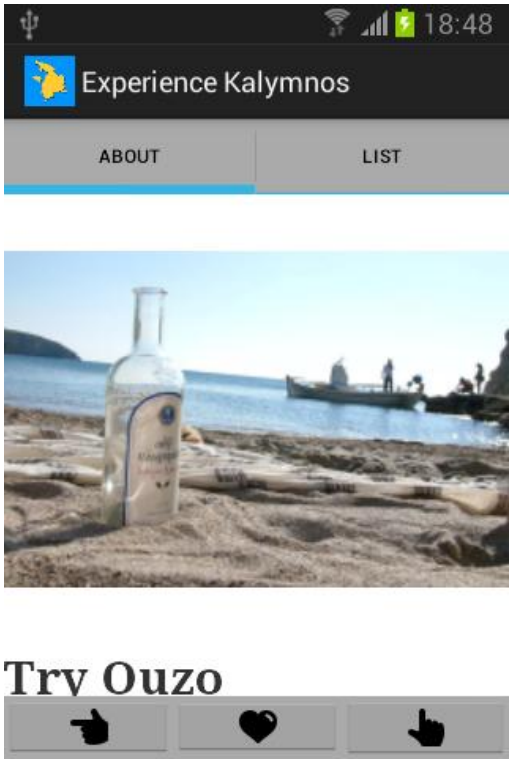


Εικόνα 3-5: **Το** outline **του πόρου** aboutlist.xml

Επεξήγηση κώδικα aboutlist.xml: Το νέο στοιχείο του αρχείου aboutlist.xml είναι ο TabHost που βρίσκεται μέσα στην πατρική RelativeLayout.

Ο TabHost είναι ένα μηχανισμός του Android με καρτέλες. Οι απόγονοι του TabHost, TabWidget και FrameLayout έχουν συγκεκριμένο ρόλο. Το TabWidget είναι υπεύθυνο για τις καρτέλες του TabHost ενώ στο FrameLayout θα μπει το περιεχόμενο της κάθε καρτέλας. Η σύνδεση του περιεχομένου με το ανάλογο Tab θα καθοριστεί προγραμματιστικά μέσα στην δραστηριότητα, κατά την αρχικοποίηση του TabHost.

Τέλος, βρίσκουμε και εδώ την LinearLayout στο τέλος, που αποτελεί την γραμμή εργασιών που βρίσκεται στο κάτω μέρος της εφαρμογής.



Εικόνα 3-6: Screenshot **της δραστηριότητας** AboutList.java

```
public class AboutList extends MyActivity {

	private String incomingText;
	private ImageButton favorites;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		//Έλεγχος του incoming text ώστε η aboutlist.java να πάρει το
ανάλογο layout
		incomingText                                                    =
this.getIntent().getStringExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAI
N_MENU);


	if(incomingText.equals(this.getResources().getString(R.string.bars))
|| incomingText.equals(this.getResources().getString(R.string.try_ouzo)) ||
incomingText.equals(this.getResources().getString(R.string.coffee))       ||
incomingText.equals(this.getResources().getString(R.string.taverns)) ||

	incomingText.equals(this.getResources().getString(R.string.souvlaki)
)                                                                          ||
incomingText.equals(this.getResources().getString(R.string.restaurant))   ||
incomingText.equals(this.getResources().getString(R.string.pizza)) ||

	incomingText.equals(this.getResources().getString(R.string.patisseri
es))  ||  incomingText.equals(this.getResources().getString(R.string.hotel))
|| incomingText.equals(this.getResources().getString(R.string.room)) ||

	incomingText.equals(this.getResources().getString(R.string.travel_ag
encies))                                                                   ||
```

```
incomingText.equals(this.getResources().getString(R.string.beaches))      ||
incomingText.equals(this.getResources().getString(R.string.sightseeings))
||

       incomingText.equals(this.getResources().getString(R.string.villages)
)  ||  incomingText.equals(this.getResources().getString(R.string.museums))
||
incomingText.equals(this.getResources().getString(R.string.archaeological_s
ites))
                           ||
incomingText.equals(this.getResources().getString(R.string.festivals))    ||
incomingText.equals(this.getResources().getString(R.string.kalymnian_produc
ts))                                                                       ||
incomingText.equals(this.getResources().getString(R.string.car_bike_rentals
))){

                    this.setContentView(R.layout.aboutlist);
                    this.initTabHost();

             }else if(incomingText.equals("ATMs")){
                    Toast.makeText(this, "setContentView for AboutList for
ATMs", Toast.LENGTH_LONG).show();
             }else
if(incomingText.equals(this.getResources().getString(R.string.traditions)))
{
                    Toast.makeText(this, "setContentView for AboutList for
Traditions", Toast.LENGTH_LONG).show();
             }

             this.initFavoriteListListener();

       }


       private void initFavoriteListListener() {

             this.favorites                  =                  (ImageButton)
this.findViewById(R.id.imagebutton_favorites_aboutlist);
             this.favorites.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {

                           Dialog dialog = new Dialog(AboutList.this);
                           dialog.setTitle(R.string.favorites);

                           if(Favourites.getFavouritesList().size()==0){

       dialog.setContentView(R.layout.favoritedialog_directions);
                                  dialog.show();
                           }else{

       dialog.setContentView(R.layout.favoritedialog_listview);
                                  ListView     listview     =     (ListView)
dialog.findViewById(R.id.listview_favorites);
                                  listview.setAdapter(new
FavoriteAdapter(AboutList.this,            R.layout.favoritedialog_listview,
Favourites.getFavouritesList()));
```

```
                                listview.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

                                    @Override
                                    public                          void
onItemClick(AdapterView<?> parent, View view,
                                                        int position, long id)
{

                                                TextView  text  =  (TextView)
view.findViewById(R.id.textview_favorite_singlerow);
                                                String         name         =
text.getText().toString();

                                                for   (Iterator   iterator   =
Favourites.getAllList().iterator(); iterator
                                                              .hasNext();) {

                                                        GeneralClass
antikeimeno = (GeneralClass) iterator.next();


      if(name.equals(antikeimeno.getName())){



                                                              Intent  intent  =
new Intent();

      intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
name);
                                                              //add  the  flag
so all of the other activities on top of it will be closed and this Intent
will be delivered to the (now on top) old activity as a new Intent.

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                                              // ελέγχουμε τον
τύπου του antikeimeno2 και ανοίγουμε την κατάλληλη δραστηριότητα
                                                              //(δεν       μου
δούλεψε με το instanceof)

      if(antikeimeno.getClass().equals(Epixirisi.class)){

      intent.setClass(AboutList.this, EpixirisiActivity.class);

      intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
3);
                                                                    //bug
fixed: Φτιάχτηκε το πρόβλημα με την aboutList, τώρα αν μπω μέσα στο Domus
πχ, μπορώ να πατήσω back και
                                                                    //να βρεθώ
στην aboutlist. Η intent στέλνει το incoming text (πχ Bars) στην
επιχείρηση, και η τελευταία με τη
                                                                    //σειρά
της επιστρέφει το Bars στην AboutList, με αποτέλεσμα η εξής δραστηριότητα
να αρχικοποιείται και να θέτεται
                                                                    //ομαλά η
setContentView(), μιας και κάτι τέτοιο θα ήτο αδύνατο χωρίς το incoming
text!
```

```
        intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
AboutList.this.incomingText);

        AboutList.this.startActivity(intent);
                                                                }else
if(antikeimeno.getClass().equals(Beach.class)){

        intent.setClass(AboutList.this, BeachActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
3);

        intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
AboutList.this.incomingText);

        AboutList.this.startActivity(intent);
                                                                }else
if(antikeimeno.getClass().equals(Sight.class)){

        intent.setClass(AboutList.this, SightActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
3);

        intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
AboutList.this.incomingText);

        AboutList.this.startActivity(intent);
                                                                }else
if(antikeimeno.getClass().equals(Village.class)){

        intent.setClass(AboutList.this, VillageActivity.class);

        intent.putExtra(MainMenu.INT_EXTRA_KEY_FOR_BACK_PRESSED_MAIN_MENU,
3);

        intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
AboutList.this.incomingText);

        AboutList.this.startActivity(intent);
                                                                }


                                                        }
                                                    }

                                                }
                                            });
                                            dialog.show();
                                        }

                                }
                            });
                }
```

```
	private void initTabHost(){

		this.initTabHostAboutTab();
		this.initListView();

	}

	private void initListView() {
		ListView listView = (ListView)
this.findViewById(R.id.ListView_ListTab_AboutList);
		List<String> lista = new
MyActivity.TabHostListViewItem(incomingText).getLista();
		ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, lista);
		listView.setAdapter(adapter);

		//setonItemClickListener
		listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

			@Override
			public void onItemClick(AdapterView<?> parent, View
view, int position,
					long id) {
				TextView textView = (TextView) view;
				String textListItem =
textView.getText().toString();

				//όλες τα αντικείμενα Epixirisi
				if(textListItem.equals(BAR_DOMUS.getName()) ||
textListItem.equals(BAR_CANTINA.getName()) ||
textListItem.equals(BAR_SCORPION.getName())
					||
textListItem.equals(OUZERI_GIANNIS.getName()) ||
textListItem.equals(OUZERI_MANIAS.getName()) ||
textListItem.equals(CAFE_ELYSE.getName()) ||

	textListItem.equals(CAFE_SOUL.getName()) ||
textListItem.equals(TAVERNA_O_PANTELIS.getName()) ||

	textListItem.equals(TAVERNA_MAMOUZELOS.getName()) ||
textListItem.equals(HOTEL_PLAZA.getName()) ||
textListItem.equals(HOTEL_ELENA.getName()) ||
textListItem.equals(TRAVEL_AGENCIE_MAGOS.getName()) ||

	textListItem.equals(TRAVEL_AGENCIE_KALDERIS.getName()) ||
textListItem.equals(MUSEUM_SEA_WORLD.getName())
					||
textListItem.equals(MUSEUM_ARCHAEOLOGICAL.getName()) ||
textListItem.equals(CAR_BIKE_RENTALS_MAKIS.getName())
					||
textListItem.equals(CAR_BIKE_RENTALS_NONTAS.getName()) ||
textListItem.equals(AboutList.this.ROOMS_AFRODITI.getName()) ||
textListItem.equals(ROOMS_NIKI.getName())){

				Intent intent = new Intent(AboutList.this,
EpixirisiActivity.class);
```

```
      intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
textListItem);
                                    AboutList.this.startActivity(intent);

                              }else
if(textListItem.equals(BEACH_PLATIS_YALOS.getName()) ||

      textListItem.equals(BEACH_SIKIES.getName())){

                                    Intent intent = new Intent(AboutList.this,
BeachActivity.class);

      intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
textListItem);
                                    AboutList.this.startActivity(intent);

                              }else
if(textListItem.equals(SIGHT_EFTA_PARTHENES.getName())                    ||
textListItem.equals(SIGHT_THERMA.getName())                               ||
textListItem.equals(ARCHAEOLOGICAL_SITE_CASTLE_OF_CHORA.getName())
                                          ||
textListItem.equals(ARCHAEOLOGICAL_SITE_JESUS.getName())){

                                    Intent intent = new Intent(AboutList.this,
SightActivity.class);

      intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
textListItem);
                                    AboutList.this.startActivity(intent);

                              }else
if(textListItem.equals(VILLAGE_ARGINONTA.getName())                       ||
textListItem.equals(VILLAGE_EMPORIOS.getName())){

                                    Intent intent = new Intent(AboutList.this,
VillageActivity.class);

      intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
textListItem);
                                    AboutList.this.startActivity(intent);
                              }


                        }
                  });
      }

      private void initTabHostAboutTab(){    //Μαλακία     ονομασία     το
initTabHost"AboutTab" μιας και εδώ γίνονται initialized και τα 2 Tabs
            TabHost      host                  =                 (TabHost)
this.findViewById(R.id.TabHost_AboutList);
            host.setup();

            TabHost.TabSpec aboutTab = host.newTabSpec("aboutTag");

      aboutTab.setIndicator(this.getResources().getString(R.string.about))
;
            aboutTab.setContent(R.id.RelativeLayout_About);
```

```
            host.addTab(aboutTab);

            TabHost.TabSpec listTab = host.newTabSpec("listTag");

      listTab.setIndicator(this.getResources().getString(R.string.list));
            listTab.setContent(R.id.ListView_ListTab_AboutList);
            host.addTab(listTab);

            host.setCurrentTabByTag("aboutTag");



            ImageView            image            =            (ImageView)
this.findViewById(R.id.ImageView_AboutTab_AboutList);
            image.setImageResource(new
AboutTabFactory().getImageResource());

            TextView             title             =             (TextView)
this.findViewById(R.id.TextView_Title_AboutTab_AboutList);
            title.setText(new AboutTabFactory().getTitleResource());

            TextView             brief             =             (TextView)
this.findViewById(R.id.TextView_Brief_AboutTab_AboutList);
            brief.setText(new AboutTabFactory().getBriefResource());

      }


      // Αντικείμενο της παρακάτω κλάσης θα "γεμίζει" το aboutTab
      private class AboutTabFactory{
            private int titleResource, briefResource;
            private int imageResource;

            public AboutTabFactory() {


      if(incomingText.equals(AboutList.this.getResources().getString(R.str
ing.bars))){

                        this.titleResource = R.string.bars;
                        this.briefResource = R.string.bars_brief;
                        this.imageResource = R.drawable.bars;

                  }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.try
_ouzo))){

                        this.titleResource = R.string.try_ouzo;
                        this.briefResource = R.string.ouzo_brief;
                        this.imageResource = R.drawable.tryouzo;

                  }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.cof
fee))){

                        this.titleResource = R.string.coffee;
                        this.briefResource = R.string.coffee_brief;
                        this.imageResource = R.drawable.enjoyyourcoffee;
```

```
                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.tav
erns))){

                            this.titleResource = R.string.taverns;
                            this.briefResource = R.string.tavern_brief;
                            this.imageResource = R.drawable.greek_tavern;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.sou
vlaki))){

                            this.titleResource = R.string.souvlaki;
                            this.briefResource = R.string.souvlaki_brief;
                            this.imageResource = R.drawable.greek_souvlaki;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.res
taurant))){

                            this.titleResource = R.string.restaurant;
                            this.briefResource = R.string.restaurants_brief;
                            this.imageResource = R.drawable.greekrestaurant;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.piz
za))){

                            this.titleResource = R.string.pizza;
                            this.briefResource = R.string.pizza_brief;
                            this.imageResource = R.drawable.pizza;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.pat
isseries))){

                            this.titleResource = R.string.patisseries;
                            this.briefResource = R.string.patisseries_brief;
                            this.imageResource = R.drawable.patisserie;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.hot
el))){

                            this.titleResource = R.string.hotel;
                            this.briefResource = R.string.hotel_brief;
                            this.imageResource = R.drawable.hotel;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.roo
m))){

                            this.titleResource = R.string.room;
                            this.briefResource = R.string.room_brief;
                            this.imageResource = R.drawable.room;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.tra
vel_agencies))){
```

```
                        this.titleResource = R.string.travel_agencies;
                        this.briefResource                                  =
R.string.travel_agencies_brief;
                        this.imageResource = R.drawable.travel_agencies;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.bea
ches))){

                        this.titleResource = R.string.beaches;
                        this.briefResource = R.string.beaches_brief;
                        this.imageResource = R.drawable.beaches;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.sig
htseeings))){

                        this.titleResource = R.string.sightseeings;
                        this.briefResource = R.string.sightseeings_brief;
                        this.imageResource = R.drawable.sightseeings;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.vil
lages))){

                        this.titleResource = R.string.villages;
                        this.briefResource = R.string.villages_brief;
                        this.imageResource = R.drawable.villages;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.mus
eums))){

                        this.titleResource = R.string.museums;
                        this.briefResource = R.string.museums_brief;
                        this.imageResource = R.drawable.museum;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.arc
haeological_sites))){

                        this.titleResource                                  =
R.string.archaeological_sites;
                        this.briefResource                                  =
R.string.archaological_sites_brief;
                        this.imageResource                                  =
R.drawable.archeological_sites;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.fes
tivals))){

                        this.titleResource = R.string.festivals;
                        this.briefResource = R.string.festivals_brief;
                        this.imageResource = R.drawable.festival;
```

```
                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.kal
ymnian_products))){

                            this.titleResource = R.string.kalymnian_products;
                            this.briefResource                                =
R.string.kalymnian_products_brief;
                            this.imageResource = R.drawable.products;

                    }else
if(incomingText.equals(AboutList.this.getResources().getString(R.string.car
_bike_rentals))){

                            this.titleResource = R.string.car_bike_rentals;
                            this.briefResource                                =
R.string.car_bike_rentals_brief;
                            this.imageResource = R.drawable.car_bike_rentals;

                    }

            }

            public int getTitleResource() {
                    return titleResource;
            }

            public void setTitleResource(int titleResource) {
                    this.titleResource = titleResource;
            }

            public int getBriefResource() {
                    return briefResource;
            }

            public void setBriefResource(int briefResource) {
                    this.briefResource = briefResource;
            }

            public int getImageResource() {
                    return imageResource;
            }

            public void setImageResource(int imageResource) {
                    this.imageResource = imageResource;
            }

      }

}
```

Επεξήγηση κώδικα δραστηριότητας AboutList.java: Η πρόκληση στην δραστηριότητα AboutList.java είναι η αρχικοποίηση και η ορθή λειτουργία του TabHost.

Πριν από κάθε εργασία στην onCreate() της δραστηριότητας αρχικοποιούμε το πεδίο String incomingText της κλάσης, ώστε να δείχνει το κείμενο που έφερε η πρόθεση Intent από την SubMainMenu. Αφού εξετάσουμε αυτό το String, τότε υπό ορισμένες προυποθέσεις (το πρώτο if statement μέσα στην onCreate()) ορίζεται ως πόρος διεπαφής χρήστη το αρχείο aboutlist.xml και καλείται η μέθοδος initTabHost() η οποία αρχικοποιεί τον TabHost.

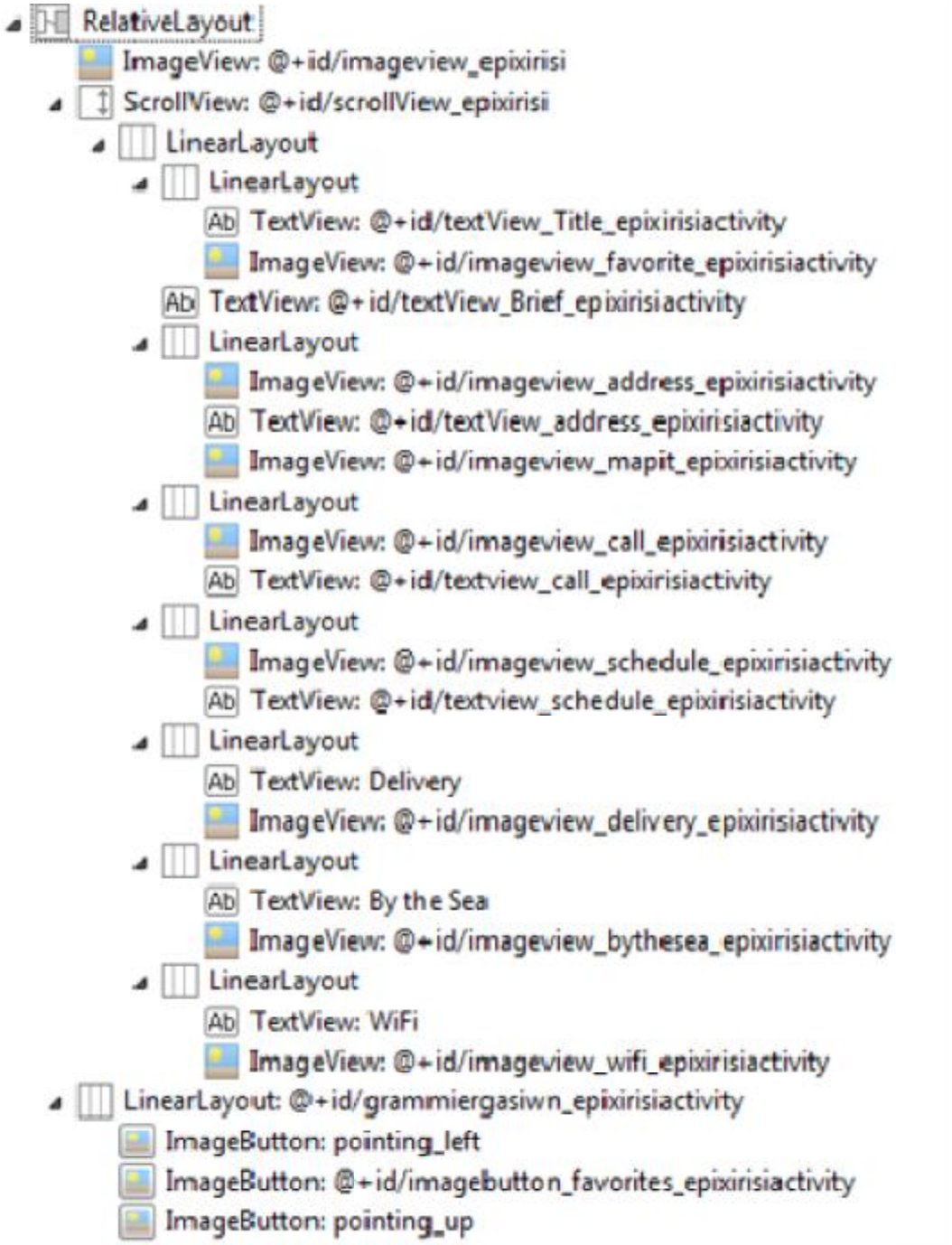
Η μέθοδος initTabHost() μοιράζεται σε δύο μεθόδους, σε δύο σκέλη. Η πρώτη, η initTabHostAboutTab() είναι υπεύθυνη για την αρχικοποίηση των καρτελών (tabs) του TabHost. Αφού αρχικοποιηθεί το πεδίο TabHost της κλάσης και συνδεθεί ουσιαστικά με τον TabHost του aboutlist.xml, δημιουργούνται δύο καρτέλες, η aboutTab και η listTab, οι οποίες είναι αντικείμενα τύπου TabHost.TabSpec. Τα δύο αυτά Tabs για την ώρα δεν έχουν κάποιο περιεχόμενο να προβάλλουν, οπότε ορίζουμε το περιεχόμενο αυτό μέσω της μεθόδου setContent(). Για περιεχόμενο του aboutTab ορίστικε το «R.id.*RelativeLayout_About*» και για το listTab το «`R.id.ListView_ListTab_AboutList`». Πρόκειται για την RelativeLayout και για την ListView που βρίσκονται μέσα στο FrameLayout του αρχείου aboutlist.xml. Στη συνέχεια «γεμίζονται» οι Views που περιέχονται στο AboutTab κάνοντας χρήση μίας εσωτερικής κλάσης, της AboutList.AboutTabFactory.

Συνεχίζοντας, στο επόμενο σκέλος της μεθόδου initTabHost(), βρίσκουμε την μέθοδο initListView() η οποία αρχικοποιεί την ListView του listTab του TabHost. Η λίστα που θα επιλεχθεί να γεμίσει αυτή την ListView θα παρθεί από την εσωτερική κλάση MyActivity.TabHostListViewItem. Αφού συνδεθεί η παρμένη λίστα με την ListView μέσω ενός ArrayAdapter<String> αντικειμένου (μέθοδος setAdapter()) θέτεται ο ακροατής συμβάντων μίας γραμμής της ListView. Δηλαδή μία επιλογή που θα κάνει ο χρήστης από την ListView (listView.setOnClickListener()). Εκεί συγκρίνεται ο πόρος String της γραμμής που επέλεξε ο χρήστης από την ListView και με βάση αυτό το String θα δημιουργηθεί μία πρόθεση Intent οποία θα μας στείλει στις παρακάτω δραστηριότητες:

1. EpixirisiActivity.java
2. BeachActivity.java
3. SightActivity.java
4. VillageActivity.java

Η πρόθεση θα αποθηκεύσει αυτό το String το οποίο θα χρησιμοποιήσουν οι παραπάνω αριθμημένες δραστηριότητες με σκοπό την αναγνώριση του αντικειμένου GeneralClass τα χαρακτηριστικά του οποίου θα προβάλλουν.

EpixirisiActivity.java(activity) + epixirisiactivity.xml(layout resource)



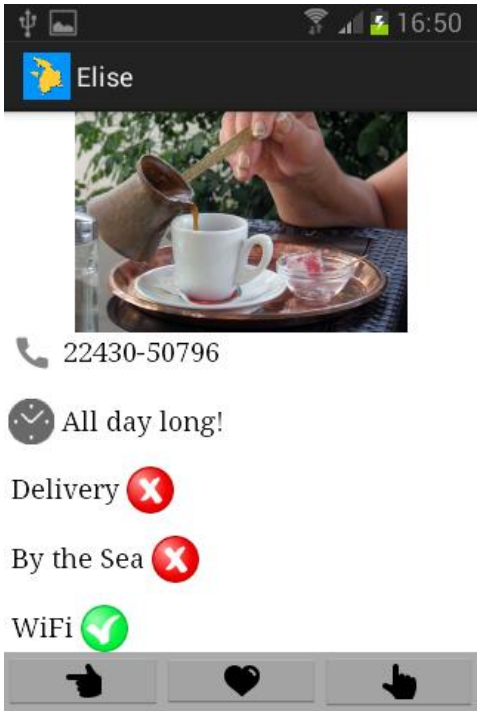
Εικόνα 3-7: **Το** outline **του πόρου** epixirisiactivity.xml

Επεξήγηση κώδικα epixirisiactivity.xml: Το αρχείο epixirisiactivity.xml αποτελείται από μία πατρική RelativeLayout η οποία περιέχει μία ImageView στο πάνω μέρος της οθόνης, μία ScrollView στη μέση και μία LinearLayout στο κάτω μέρος.

Μιας και πρόκειται για τον πόρο διεπαφής χρήστη της δραστηριότητας EpixirisiActivity, η οποία θα προβάλλει αντικείμενα-επιχειρήσεις, η ImageView στο πάνω μέρος θα «γεμίσει» με την **αντιπροσοπευτική εικόνα** του αντικειμένου.

Όσον αφορά την ScrollView, αυτή θα καλλείται να προβάλλει τις σχετικές **πληροφορίες** μίας επιχείρησης, όπως τηλέφωνα, διεύθυνση κτλ, με την δυνατότητα scroll από τον χρήστη. Αναλύοντας περαιτέρω την ScrollView διαπιστώνουμε πως αποτελείται από μία LinearLayout η οποία στοιχειοθετεί τα εσωτερικά της στοιχεία καθέτως. Μέσα σε αυτήν θα βρούμε μία οριζόντια LinearLayout για κάθε πληροφορία που θα προβάλλεται. Στις περισσότερες των περιπτώσεων η LinearLayout θα περιέχει μία ImageView και μία TextView. Στην περίπτωση του τηλεφώνου η ImageView θα προβάλλει ένα εικονίδιο ενός τηλεφώνου ενώ η TextView θα προβάλλει τον αριθμό τηλεφώνου της επιχείρησης.

Τέλος, στο κάτω μέρος, δεσπόζει η κλασική LinearLayout που αποτελεί την γραμμή εργαλείων της δραστηριότητας, με τα σχετικά ImageButtons.



Εικόνα 3-8: Screenshot **της δραστηριότητας** EpixirisiActivity.java

```
public class EpixirisiActivity extends MyActivity {

	private String incomingText;
	private ImageView epixirisiPhoto, favoriteImage, addressImage,
mapImage,
	scheduleImage, callImage, tickDeliveryImage, tickByTheSeaImage,
tickWiFiImage;
	private TextView titleText, briefText, addressText, callText,
scheduleText;
	private Epixirisi epixirisi;
	private ImageButton favorites;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		this.setContentView(R.layout.epixirisiactivity);
```

```
            incomingText                                                    =
this.getIntent().getStringExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_L
IST);
            this.initEpixirisi();
            this.initViews();
            this.initActionBar();
            this.fillViewsWithResources();
            this.initFavoriteListener();
            this.initFavoriteListListener();
            this.initViewListeners();

      }


      private void initViewListeners() {

            this.mapImage.setOnClickListener(new View.OnClickListener() {

                              @Override
                              public void onClick(View arg0) {
                                    Intent      intent      =      new
Intent(EpixirisiActivity.this, MapActivity.class);

      intent.putExtra(MyActivity.INTENT_EXTRA_KEY_FOR_PASSING_GENERALCLASS
_TO_MAPACTIVITY, EpixirisiActivity.this.epixirisi.getName());

      EpixirisiActivity.this.startActivity(intent);
                              }
                        });

      }


      private void initFavoriteListListener() {


                  this.favorites.setOnClickListener(new
View.OnClickListener() {

                        @Override
                        public void onClick(View v) {

                              Dialog         dialog         =         new
Dialog(EpixirisiActivity.this);
                              dialog.setTitle(R.string.favorites);


      if(Favourites.getFavouritesList().size()==0){

      dialog.setContentView(R.layout.favoritedialog_directions);
                                    dialog.show();
                              }else{

      dialog.setContentView(R.layout.favoritedialog_listview);
                                    ListView   listview   =   (ListView)
dialog.findViewById(R.id.listview_favorites);
                                    listview.setAdapter(new
FavoriteAdapter(EpixirisiActivity.this,    R.layout.favoritedialog_listview,
Favourites.getFavouritesList()));
```

```
                                        listview.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

                                            @Override
                                            public                    void
onItemClick(AdapterView<?> parent, View view,
                                                                  int     position,
long id) {

                                                    TextView     text     =
(TextView) view.findViewById(R.id.textview_favorite_singlerow);
                                                    String       name     =
text.getText().toString();

                                                    for  (Iterator  iterator
= Favourites.getAllList().iterator(); iterator

       .hasNext();) {

                                                           GeneralClass
antikeimeno = (GeneralClass) iterator.next();


       if(name.equals(antikeimeno.getName())){



                                                                  Intent
intent = new Intent();

       intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
name);
                                                                  //add  the
flag so all of the other activities on top of it will be closed and this
Intent will be delivered to the (now on top) old activity as a new Intent.

       intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                                                  //
ελέγχουμε  τον  τύπου  του  antikeimeno2  και  ανοίγουμε  την  κατάλληλη
δραστηριότητα
                                                                  //(δεν  μου
δούλεψε  με  το  instanceof)

       if(antikeimeno.getClass().equals(Epixirisi.class)){

       intent.setClass(EpixirisiActivity.this, EpixirisiActivity.class);

       EpixirisiActivity.this.startActivity(intent);
                                                                  }else
if(antikeimeno.getClass().equals(Beach.class)){

       intent.setClass(EpixirisiActivity.this, BeachActivity.class);

       EpixirisiActivity.this.startActivity(intent);
                                                                  }else
if(antikeimeno.getClass().equals(Sight.class)){

       intent.setClass(EpixirisiActivity.this, SightActivity.class);
```

```
	EpixirisiActivity.this.startActivity(intent);
                                                                }else
if(antikeimeno.getClass().equals(Village.class)){

	intent.setClass(EpixirisiActivity.this, VillageActivity.class);

	EpixirisiActivity.this.startActivity(intent);
                                                                }



                                                            }
                                                        }

                                                    }
                                                });
                                                dialog.show();
                                            }

                                        }
                                    });
                            }

	@Override
	public void onBackPressed() {
		int                         anagnoristiko                       =
getIntent().getIntExtra(EpixirisiActivity.INT_EXTRA_KEY_FOR_BACK_PRESSED_MA
IN_MENU, 0);

		if(anagnoristiko==1){     //Εφόσον μπήκαμε στη δραστηριότητα
από την λίστα των Favourites της MainMenu...
			this.finish();
			Intent   intent   =   new   Intent(EpixirisiActivity.this,
MainMenu.class);
			intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
			this.startActivity(intent);
		}else if(anagnoristiko==2){            //Εφόσον   μπήκαμε   στη
δραστηριότητα από την λίστα των Favourites της SubMainMenu...
			this.finish();
			Intent   intent   =   new   Intent(EpixirisiActivity.this,
SubMainMenu.class);
			intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
			this.startActivity(intent);
		}else if(anagnoristiko==3){      //Εφόσον      μπήκαμε      στη
δραστηριότητα από την λίστα των Favourites της AboutList...
			this.finish();
			//bug fixed: Μόνο για την AboutList πρέπει να στείλω το
όνομα της κατηγορίας της listview (πχ Bars, Try Ouzo κτλ) γιατί πρώτα
ελέγχει το όνομα και μετά κάνει setContentView
			//με λίγα λόγια όταν onBackPressed() ξαναγυρίσω στην
AboutList, η τελευταία δεν έχει το όνομα που ζητείται για να ενεργοποιήσει
την contentView της
			Intent   intent   =   new   Intent(EpixirisiActivity.this,
AboutList.class);
			intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

	intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAIN_MENU,
```

```
this.getIntent().getStringExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_SUB_MAI
N_MENU));
                    this.startActivity(intent);
             }else{
                    super.onBackPressed();
             }
      }

      private void initFavoriteListener() {

             this.favoriteImage.setOnClickListener(new
View.OnClickListener() {

                    @Override
                    public void onClick(View v) {

                           ImageView image = (ImageView) v;

                           SharedPreferences          sharedPrefs          =
getSharedPreferences(MyActivity.SHARED_PREFENCES_FAVORITE,
Context.MODE_PRIVATE);
                           SharedPreferences.Editor         editor         =
sharedPrefs.edit();


      if(EpixirisiActivity.this.epixirisi.getFavoriteResult()){


      image.setImageResource(R.drawable.ic_action_favorite);

      EpixirisiActivity.this.epixirisi.setFavoriteResult(false);
                                  //Save the isFavorite result

      editor.putBoolean(MyActivity.SHARED_PREFENCES_FAVORITE_KEY+Epixirisi
Activity.this.epixirisi.getName(),
EpixirisiActivity.this.epixirisi.getFavoriteResult());
                                  editor.commit();

                                  //remove          epixirisi          from
Favourites.favouritesList

      if(Favourites.getFavouritesList().contains(EpixirisiActivity.this.ep
ixirisi)){

      Favourites.getFavouritesList().remove(EpixirisiActivity.this.epixiri
si);
                                  }else{
                                        Toast.makeText(getBaseContext(),
EpixirisiActivity.this.epixirisi.getName()+"   was   not   a   favourite!",
Toast.LENGTH_LONG).show();
                                  }

                           }else{


      image.setImageResource(R.drawable.heart_full);

      EpixirisiActivity.this.epixirisi.setFavoriteResult(true);
                                  //Save the isFavorite result
```

```
	editor.putBoolean(MyActivity.SHARED_PREFENCES_FAVORITE_KEY+Epixirisi
Activity.this.epixirisi.getName(),
EpixirisiActivity.this.epixirisi.getFavoriteResult());
                                editor.commit();

                                //add             epixirisi             to
Favourites.favouritesList

	if(!Favourites.getFavouritesList().contains(EpixirisiActivity.this.e
pixirisi)){

	Favourites.getFavouritesList().add(EpixirisiActivity.this.epixirisi)
;
                                }else{
                                        Toast.makeText(getBaseContext(),
EpixirisiActivity.this.epixirisi.getName()+"   is   allready   favourite!",
Toast.LENGTH_LONG).show();
                                }
                        }

                }
        });

	}



	private void initActionBar(){
		ActionBar bar = this.getActionBar();
		bar.setTitle(epixirisi.getName());
	}

	private void initViews(){
		//imageViews
		epixirisiPhoto                  =                (ImageView)
this.findViewById(R.id.imageview_epixirisi);
		favoriteImage                   =                (ImageView)
this.findViewById(R.id.imageview_favorite_epixirisiactivity);
		addressImage=                                    (ImageView)
this.findViewById(R.id.imageview_address_epixirisiactivity);
		mapImage                        =                (ImageView)
this.findViewById(R.id.imageview_mapit_epixirisiactivity);
		scheduleImage=                                   (ImageView)
this.findViewById(R.id.imageview_schedule_epixirisiactivity);
		callImage                       =                (ImageView)
this.findViewById(R.id.imageview_call_epixirisiactivity);
		tickDeliveryImage=                               (ImageView)
this.findViewById(R.id.imageview_delivery_epixirisiactivity);
		tickByTheSeaImage               =                (ImageView)
this.findViewById(R.id.imageview_bythesea_epixirisiactivity);
		tickWiFiImage                   =                (ImageView)
this.findViewById(R.id.imageview_wifi_epixirisiactivity);
		//textViews
		titleText                       =                 (TextView)
this.findViewById(R.id.textView_Title_epixirisiactivity);
		briefText                       =                 (TextView)
this.findViewById(R.id.textView_Brief_epixirisiactivity);
```

```
            addressText=                                        (TextView)
this.findViewById(R.id.textView_address_epixirisiactivity);
            callText                    =                       (TextView)
this.findViewById(R.id.textview_call_epixirisiactivity);
            scheduleText=                                       (TextView)
this.findViewById(R.id.textview_schedule_epixirisiactivity);
            //imagebuttons
            this.favorites                  =               (ImageButton)
this.findViewById(R.id.imagebutton_favorites_epixirisiactivity);
      }




      private void fillViewsWithResources(){
            //imageviews
            epixirisiPhoto.setImageResource(epixirisi.getImageResource());

      addressImage.setImageResource(epixirisi.ADDRESS_IMAGE_RESOURCE);
            mapImage.setImageResource(epixirisi.MAP_IMAGE_RESOURCE);

      scheduleImage.setImageResource(epixirisi.SCHEDULE_IMAGE_RESOURCE);
            callImage.setImageResource(epixirisi.CALL_IMAGE_RESOURCE);

      tickDeliveryImage.setImageResource(epixirisi.hasDelivery()?epixirisi
.TICK_IMAGE_RESOURCE:epixirisi.TICK_WRONG_IMAGE_RESOURCE);

      tickByTheSeaImage.setImageResource(epixirisi.isByTheSea()?epixirisi.
TICK_IMAGE_RESOURCE:epixirisi.TICK_WRONG_IMAGE_RESOURCE);

      tickWiFiImage.setImageResource(epixirisi.hasWiFi()?epixirisi.TICK_IM
AGE_RESOURCE:epixirisi.TICK_WRONG_IMAGE_RESOURCE);
            //textviews
            titleText.setText(epixirisi.getName());
            briefText.setText(epixirisi.getBrief());
            addressText.setText(epixirisi.getAddress());
            callText.setText(epixirisi.getTelephone());
            scheduleText.setText(epixirisi.getSchedule());

            //Special

      this.favoriteImage.setImageResource(this.epixirisi.getFavoriteResult
()?R.drawable.heart_full:R.drawable.ic_action_favorite);
      }

      private void initEpixirisi(){
            if(incomingText.equals(BAR_DOMUS.getName())){
                  epixirisi = BAR_DOMUS;
                  epixirisi.setDeliveryStatus(true);
                  epixirisi.setByTheSeaStatus(true);
                  epixirisi.setWiFiStatus(true);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");

            }else if(incomingText.equals(BAR_CANTINA.getName())){
                  epixirisi = BAR_CANTINA;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(true);
```

```
            epixirisi.setWiFiStatus(true);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(BAR_SCORPION.getName())){
            epixirisi = BAR_SCORPION;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(false);
            epixirisi.setWiFiStatus(true);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(OUZERI_GIANNIS.getName())){
            epixirisi = OUZERI_GIANNIS;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(false);
            epixirisi.setWiFiStatus(false);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(OUZERI_MANIAS.getName())){
            epixirisi = OUZERI_MANIAS;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(false);
            epixirisi.setWiFiStatus(false);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(CAFE_ELYSE.getName())){
            epixirisi = CAFE_ELYSE;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(false);
            epixirisi.setWiFiStatus(true);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(CAFE_SOUL.getName())){
            epixirisi = CAFE_SOUL;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(false);
            epixirisi.setWiFiStatus(true);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(TAVERNA_MAMOUZELOS.getName())){
            epixirisi = TAVERNA_MAMOUZELOS;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(true);
            epixirisi.setWiFiStatus(false);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(TAVERNA_O_PANTELIS.getName())){
            epixirisi = TAVERNA_O_PANTELIS;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(false);
            epixirisi.setWiFiStatus(false);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
        }else if(incomingText.equals(HOTEL_ELENA.getName())){
            epixirisi = HOTEL_ELENA;
            epixirisi.setDeliveryStatus(false);
            epixirisi.setByTheSeaStatus(true);
            epixirisi.setWiFiStatus(true);
            epixirisi.setTelephone("22430-50796");
            epixirisi.setSchedule("All day long!");
```

```
            }else if(incomingText.equals(HOTEL_PLAZA.getName())){
                  epixirisi = HOTEL_PLAZA;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(true);
                  epixirisi.setWiFiStatus(true);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else
if(incomingText.equals(TRAVEL_AGENCIE_KALDERIS.getName())){
                  epixirisi = TRAVEL_AGENCIE_KALDERIS;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(false);
                  epixirisi.setWiFiStatus(false);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else if(incomingText.equals(TRAVEL_AGENCIE_MAGOS.getName())){
                  epixirisi = TRAVEL_AGENCIE_MAGOS;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(false);
                  epixirisi.setWiFiStatus(true);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else
if(incomingText.equals(MUSEUM_ARCHAEOLOGICAL.getName())){
                  epixirisi = MUSEUM_ARCHAEOLOGICAL;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(false);
                  epixirisi.setWiFiStatus(false);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else if(incomingText.equals(MUSEUM_SEA_WORLD.getName())){
                  epixirisi = MUSEUM_SEA_WORLD;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(true);
                  epixirisi.setWiFiStatus(false);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else
if(incomingText.equals(CAR_BIKE_RENTALS_MAKIS.getName())){
                  epixirisi = CAR_BIKE_RENTALS_MAKIS;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(false);
                  epixirisi.setWiFiStatus(false);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else
if(incomingText.equals(CAR_BIKE_RENTALS_NONTAS.getName())){
                  epixirisi = CAR_BIKE_RENTALS_NONTAS;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(false);
                  epixirisi.setWiFiStatus(false);
                  epixirisi.setTelephone("22430-50796");
                  epixirisi.setSchedule("All day long!");
            }else if(incomingText.equals(ROOMS_AFRODITI.getName())){
                  epixirisi = ROOMS_AFRODITI;
                  epixirisi.setDeliveryStatus(false);
                  epixirisi.setByTheSeaStatus(true);
                  epixirisi.setWiFiStatus(true);
                  epixirisi.setTelephone("22430-50796");
```

```
                epixirisi.setSchedule("All day long!");
            }else if(incomingText.equals(ROOMS_NIKI.getName())){
                epixirisi = ROOMS_NIKI;
                epixirisi.setDeliveryStatus(false);
                epixirisi.setByTheSeaStatus(false);
                epixirisi.setWiFiStatus(true);
                epixirisi.setTelephone("22430-50796");
                epixirisi.setSchedule("All day long!");
            }

        }

}
```

Επεξήγηση κώδικα δραστηριότητας EpixirisiActivity.java: Στην μέθοδο onCreate() της δραστηριότητας ορίζεται ως πόρος διεπαφής χρήστη το αρχείο epixirisiactivity.xml.

Στη συνέχεια αρχικοποιείται το πεδίο String incomingText ώστε να δείχνει στο κείμενο που έφερε μαζί της η πρόθεση Intent από την AboutList.java. Εκμεταλλευόμενη αυτό το αρχείο String η μέθοδος initEpixirisi() αναγνωρίζει ουσιαστικά ποια επιχείρηση επέλεξε ο χρήστης για να προβάλλει επιλέγοντας το σχετικό όνομα από την ListView της δραστηριότητας AboutList. Αφού αναγνωριστεί η επιχείρηση αυτή, αποθηκεύεται στο πεδίο Epixirisi epixirisi της κλάσης.

Στη συνέχεια στυλιζάρεται κατάλληλα η ActionBar της δραστηριότητας, λαβαίνοντας ως τίτλο το όνομα της επιχείρησης (μέθοδος initActionBar()), ενώ πραγματοποιείται η αρχικοποίηση όλων των πεδίων Views της EpixirisiActivity καθώς και το γέμισμα αυτών με πόρους, βάση του αντικειμένου Epixirisi epixirisi (μέθοδοι initActionBar() και fillViewsWithResources()). Παραδείγματος χάρη την κεντρική ImageView όπου θα προβάλλεται η αντιπροσωπευτική εικόνα της επιχείρησης θα τη γεμίσουμε με τον πόρο epixirisi.getImageResource().

Η μέθοδος initFavoriteListener() υλοποιεί μία από τις πιο σημαντικές λειτουργίες της εφαρμογής, αυτήν του ορισμού ως αγαπημένης επιλογής της συγκεκριμένης επιχείρησης που παρουσιάζεται την δεδομένη χρονική στιγμή από την δραστηριότητα που περιγράφουμε.

Στην πραγματικότητα πρόκειται για έναν ακροατή συμβάντων “onClick” στο πεδίο ImageView favorites. Αρχικά θα πρέπει να παρατηρήσουμε τον πόρο που γεμίσαμε την συγκεκριμένη ImageView μέσα στη μέθοδο fillViewsWithResources.

```
this.favoriteImage.setImageResource(this.epixirisi.getFavoriteResult()?R.dr
awable.heart_full:R.drawable.ic_action_favorite);
```

Σε «ελεύθερη μετάφραση» ο παραπάνω κώδικας λέει το εξής. Αν το αντικείμενο Epixirisi έχει χαρακτηριστεί ως favorite γέμισε την ImageView favoriteImage με μία κόκκινη καρδιά, αλλιώς με μία γκρι.

Όταν ο χρήστης εγκαταστήσει την εφαρμογή για πρώτη φορά, είναι φυσικό το αντικείμενο epixirisi να μην έχει χαρακτηριστεί ποτέ ως favorite, κάτι που σημαίνει πως εάν

boolean favoriteResult = new Epixirisi(...).getFavoriteResult();, τότε favoriteResult==false;.

Όταν όμως ο χρήστης κλικάρει την ImageView favoriteImage (η οποία για την ώρα φαίνεται με το εικονίδιο μιας γκρι καρδιάς) τότε αυτό θα σημαίνει πως θέλει να χαρακτηρίσει το αντικείμενο Epixirisi ως αγαπημένο. Εκείνη την ώρα καλείται η initFavoriteListener() που θα πράξει το εξής: Θα δημιουργήσει ένα αρχείο SharedPreferences για να προετοιμαστεί για την **αποθήκευση του** boolean **αποτελέσματος αγαπημένης επιλογής**, όταν αυτό καθοριστεί. Στη συνέχεια θα ελέγξει την τρέχουσα κατάσταση της επιχείρησης όσον αφορά τον χαρακτηρισμό ως αγαπημένη επιλογή ή όχι.

Στην περίπτωση που το αντικείμενο Epixirisi **ΔΕΝ ΕΙΝΑΙ** τη δεδομένη χρονική στιγμή χαρακτηρίσιμο ως favorite τότε πράττει τα εξής:

1. Αλλάζει την εικόνα της ImageView favoriteImage και από μία γκρι καρδιά την αντικαθιστά με μία μεγαλύτερη κόκκινη καρδιά! Ένδειξη για τον χρήστη ότι μετά το κλικ η επιχείρηση χαρακτηρίστηκε από μη αγαπημένη, ως αγαπημένη. Η εν λόγω εργασία στην πραγματικότητα υλοποιείται στο επόμενο βήμα

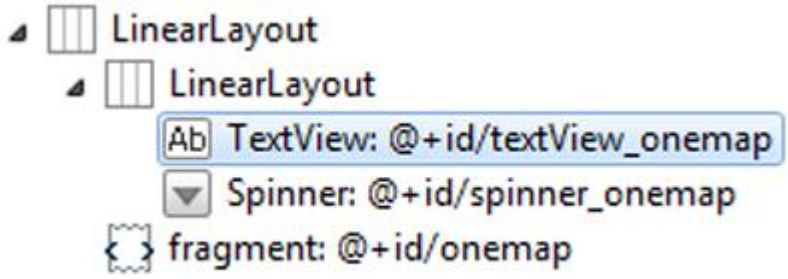
2. Θέτει την τρέχουσα κατάσταση της επιχείρησης όσον αφορά τον χαρακτηρισμό ως αγαπημένη επιλογή ή όχι σε "true" και το συγκεκριμένο boolean αποτέλεσμα αποθηκεύεται στις SharedPreferences. Επομένως η επιχείρηση από εκεί που δεν ήταν αγαπημένη επιλογή, μετά το κλικ έγινε και... επίσημα αγαπημένη επιλογή

3. Αναζητεί το αντικείμενο Epixirisi στην λίστα της υπεύθυνης κλάσης για τις αγαπημένες επιλογές, της utilities.Favourites. Εάν αυτό δεν υπάρχει μέσα στη λίστα τότε το **προσθέτει** στη λίστα.

Με το τρίτο βήμα πετύχαμε την ενημέρωση της λίστας των αγαπημένων επιλογών. Σε αυτό το σημείο λύνεται και ο γρίφος της μεθόδου loadAllFavoritesResults() που βρίσκεται μέσα στην δραστηριότητα εκκίνησης MainMenu. Όταν τερματιστεί η εφαρμογή και επανεκκινίσει από τον χρήστη, η loadAllFavoritesResults() θα ανοίξει το αρχείο SharedPreferences που έχει αποθηκεύσει τα boolean αποτελέσματα των αντικειμένων που έχουν χαρακτηριστεί ως αγαπημένα ή όχι (true για όσα είναι αγαπημένα, false για όσα δεν είναι).

Αφού «σκανάρει» ένα-ένα όλα τα αντικείμενα που δύναται να χαρακτηριστούν ως αγαπημένη επιλογή, θα θέτει το πεδίο boolean isFavorite των αντικειμένων GeneralClass στην σωστή τιμή (true στα αγαπημένα, false στα μη).

OneMapActivity.java(activity) + one_map_activity.xml(layout resource)

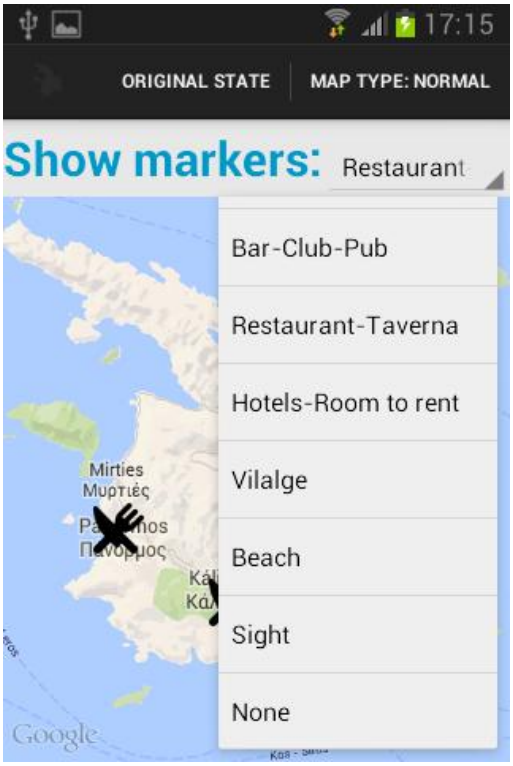


Εικόνα 3-9: **Το** outline **του πόρου** one_map_activity.xml

Επεξήγηση κώδικα αρχείου one_map_activity.xml: Όπως παρατηρούμε η δομή του αρχείου one_map_activity.xml είναι πολύ απλή, μιας και η πατρική LinearLayout περιέχει μόνο δύο γόνους, μία ακόμα LinearLayout και ένα fragment.

Παρόλα αυτά, στο αρχείο βρίσκουμε δύο νέα στοιχεία. Το ένα βρίσκεται εντός της απογόνου LinearLayout και είναι το Spinner. Το Spinner είναι το κυλιόμενο μενού, όπως φαίνεται σε λειτουργία στην δεύτερη εικόνα της δραστηριότητας OneMapActivity.

Το δεύτερο στοιχείο είναι το fragment με id "onemap". Το fragment, όπως παρατηρούμε και στον κώδικα παραπάνω, φέρει στην ιδιότητα "android:name" το όνομα της κλάσης που θα υλοποιήσει. Εφόσον η κλάση αυτή είναι η com.google.android.gms.maps.SupportMapFragment τότε καταλαβαίνουμε πως το συγκεκριμένο fragment θα είναι αυτό που θα υλοποιήσει τον Google Map στην δραστηριότητα OneMapActivity.



Εικόνα 3-10: Screenshots **της δραστηριότητας** OneMapActivity.java

```
public class OneMapActivity extends FragmentActivity {

	//map related fields
	private GoogleMap mMap;
	private static final int GPS_ERRORDIALOG_REQUEST = 9000;
	private Marker markerCantina, markerDomus, markerScorpion,
markerPlatiYalos, markerSikies, markerEftaParthenes, markerTherma,
markerArginonta, markerEmporios, markerMamouzelos, markerPantelis,
markerElyse, markerSoul, markerElena, markerPlaza;
	private Marker[] allMarkers = {markerCantina, markerDomus,
markerScorpion, markerPlatiYalos, markerSikies, markerEftaParthenes,
markerTherma, markerArginonta, markerEmporios, markerMamouzelos,
markerPantelis, markerElyse, markerSoul, markerElena, markerPlaza};
	private CameraPosition originalCameraPosition;

	private Spinner spinner;

	//SharedPrefences file names and keys
	private final static String SHAREDPREFENCES_MAPTYPE_FILE_NAME="Save
Google Map Type";
	private final static String SHAREDPREFENCES_MAPTYPE_KEY_NAME="Save
Google Map Type Key";
	private final static String
SHAREDPREFENCES_MAP_CAMERAPOSITION_FILE_NAME="Save Google Map Camera
Position";
	private final static String
SHAREDPREFENCES_MAP_CAMERAPOSITION_TILT_KEY="Save Google Map Tilt";
	private final static String
SHAREDPREFENCES_MAP_CAMERAPOSITION_BEARING_KEY="Save Google Map Bearing";
	private final static String
SHAREDPREFENCES_MAP_CAMERAPOSITION_ZOOM_KEY="Save Google Map Zoom";
	private final static String
SHAREDPREFENCES_MAP_CAMERAPOSITION_LATITUDE_KEY="Save Google Map Latitude";
	private final static String
SHAREDPREFENCES_MAP_CAMERAPOSITION_LONGITUDE_KEY="Save Google Map
Longitude";


	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		if(this.servicesOK()){
			this.setContentView(R.layout.one_map_activity);
			this.initActionBar(this.getActionBar());

			if(this.initMap()){

				// the default location of Kalymnos island
				this.goToLocation(36.988653, 26.964342, 11);

				//set the original camera position for the
"Original State" menu option
				this.originalCameraPosition=new
CameraPosition(this.mMap.getCameraPosition().target, 11, 0, 0);

				if(this.mMap!=null){
```

```
                                FixObjectsManager      mgr      =      new
FixObjectsManager();
                                final     List<GeneralClass>     lista     =
mgr.getLista();
                                //add all markers
                                for (int i = 0; i < lista.size(); i++) {

       this.allMarkers[i]=this.mMap.addMarker(new MarkerOptions().

       icon(BitmapDescriptorFactory.fromResource(lista.get(i).getIconResour
ce())).

       position(lista.get(i).getLatLng()).

       title(lista.get(i).getName()).

       snippet(lista.get(i).getAddress()).
                                                      anchor(.5f, .5f).
                                                      flat(true));
                                }

                                this.mMap.setOnInfoWindowClickListener(new
GoogleMap.OnInfoWindowClickListener() {

                                      @Override
                                      public void onInfoWindowClick(Marker
marker) {

                                            String        title        =
marker.getTitle();

                                            for   (int   i   =   0;   i   <
lista.size(); i++) {


       if(title.equals(lista.get(i).getName())){

                                                                  Intent  intent  =
new Intent();

       intent.putExtra(MyActivity.TEXT_LIST_ITEM_EXTRA_KEY_ABOUT_LIST,
title);


       if(lista.get(i).getClass()==Epixirisi.class){

       intent.setClass(OneMapActivity.this, EpixirisiActivity.class);

       OneMapActivity.this.startActivity(intent);
                                                                  }else
if(lista.get(i).getClass()==Sight.class){

       intent.setClass(OneMapActivity.this, SightActivity.class);

       OneMapActivity.this.startActivity(intent);
                                                                  }else
if(lista.get(i).getClass()==Village.class){

       intent.setClass(OneMapActivity.this, VillageActivity.class);
```

```
	OneMapActivity.this.startActivity(intent);
                                                        }else
if(lista.get(i).getClass()==Beach.class){

	intent.setClass(OneMapActivity.this, BeachActivity.class);

	OneMapActivity.this.startActivity(intent);
                                                        }else{

	Toast.makeText(OneMapActivity.this,    "No    preview    available!",
Toast.LENGTH_LONG).show();
                                                        }
                                                    }
                                                }

                                        }
                                    });

                                    this.initSpinner();


                                }

                            }
                        }
	}

	private void initSpinner(){
		this.spinner=(Spinner) this.findViewById(R.id.spinner_onemap);
		ArrayAdapter  adapter  =  ArrayAdapter.createFromResource(this,
R.array.spinner_items, android.R.layout.simple_spinner_item);

	adapter.setDropDownViewResource(android.R.layout.simple_spinner_drop
down_item);
		this.spinner.setAdapter(adapter);
		this.spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {

			@Override
			public void onItemSelected(AdapterView<?> parrent, View
view,
                                    int position, long id) {
                                TextView textview = (TextView) view;
                                String keimeno = textview.getText().toString();

                                if(keimeno.equalsIgnoreCase("All")){
                                    for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

	OneMapActivity.this.allMarkers[i].setVisible(true);
                                    }
                                }else if(keimeno.equalsIgnoreCase("Cafe")){
                                    for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

	OneMapActivity.this.showAndHideMarkers(12,11);
                                    }
```

```
                        }else      if(keimeno.equalsIgnoreCase("Bar-Club-
Pub")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.showAndHideMarkers(0,1,2);
                                }
                        }else    if(keimeno.equalsIgnoreCase("Restaurant-
Taverna")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.showAndHideMarkers(9,10);
                                }
                        }else if(keimeno.equalsIgnoreCase("Hotels-Room to
rent")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.showAndHideMarkers(13,14);
                                }
                        }else if(keimeno.equalsIgnoreCase("Village")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.showAndHideMarkers(7,8);
                                }
                        }else if(keimeno.equalsIgnoreCase("Beach")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.showAndHideMarkers(3,4);
                                }
                        }else if(keimeno.equalsIgnoreCase("Sight")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.showAndHideMarkers(5,6);
                                }
                        }else if(keimeno.equalsIgnoreCase("None")){
                                for     (int     i     =     0;     i     <
OneMapActivity.this.allMarkers.length; i++) {

      OneMapActivity.this.allMarkers[i].setVisible(false);
                                }
                        }

                }

                @Override
                public void onNothingSelected(AdapterView<?> arg0) {
                        // TODO Auto-generated method stub

                }
        });
      }

      private void initActionBar(ActionBar bar){
            bar.setTitle("Kalymnos");
```

```
            bar.setIcon(R.drawable.ic_action_kalymnoslogotransparent);
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            MenuInflater inflater = this.getMenuInflater();
            inflater.inflate(R.menu.onemap, menu);
            return super.onCreateOptionsMenu(menu);
      }

      @Override
      protected void onStop() {
            super.onStop();
            this.saveMapType();
            this.saveCameraPosition(this.mMap.getCameraPosition());
      }

      @Override
      protected void onResume() {
            super.onResume();
            //Load the map-type
            this.mMap.setMapType(this.loadMapType());

      this.mMap.moveCamera(CameraUpdateFactory.newCameraPosition(this.getS
avedCameraPosition()));
      }

      private int loadMapType(){
            SharedPreferences                sharedPrefs                =
this.getSharedPreferences(this.SHAREDPREFENCES_MAPTYPE_FILE_NAME,
Context.MODE_PRIVATE);
            return
sharedPrefs.getInt(this.SHAREDPREFENCES_MAPTYPE_KEY_NAME,
GoogleMap.MAP_TYPE_NORMAL);
      }

      private void saveMapType(){
            SharedPreferences                sharedPrefs                =
this.getSharedPreferences(this.SHAREDPREFENCES_MAPTYPE_FILE_NAME,
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor=sharedPrefs.edit();
            editor.putInt(this.SHAREDPREFENCES_MAPTYPE_KEY_NAME,
this.mMap.getMapType());
            editor.commit();
      }

      private void saveCameraPosition(CameraPosition position){
            //break the CameraPosition of the map
            float bearing = position.bearing;
            double lat = position.target.latitude;
            double lng = position.target.longitude;
            float tilt = position.tilt;
            float zoom= position.zoom;

            //Save each one of the Camera's position characteristics
            SharedPreferences                prefs                =
this.getSharedPreferences(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_FILE_NAME
, Context.MODE_PRIVATE);
            SharedPreferences.Editor editor=prefs.edit();
```

```
		editor.putFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_BEARING_KEY,
bearing);

		editor.putFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_LATITUDE_KEY
, (float)lat);

		editor.putFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_LONGITUDE_KE
Y, (float)lng);

		editor.putFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_TILT_KEY,
tilt);

		editor.putFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_ZOOM_KEY,
zoom);
			editor.commit();
		}

		private CameraPosition getSavedCameraPosition(){
			SharedPreferences prefs =
this.getSharedPreferences(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_FILE_NAME
, Context.MODE_PRIVATE);
			//get the values
			float bearing =
prefs.getFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_BEARING_KEY, 0);
			float tilt =
prefs.getFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_TILT_KEY, 0);
			float zoom =
prefs.getFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_ZOOM_KEY, 0);
			double
lat=prefs.getFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_LATITUDE_KEY,
0);
			double
lng=prefs.getFloat(this.SHAREDPREFENCES_MAP_CAMERAPOSITION_LONGITUDE_KEY,
0);

			if(bearing!=0 || tilt!=0 || zoom!=0 || lat!=0 || lng!=0){
				CameraPosition position = new CameraPosition(new
LatLng(lat, lng), zoom, tilt, bearing);
				return position;
			}else{
				return this.originalCameraPosition;
			}
		}

		@Override
		public boolean onOptionsItemSelected(MenuItem item) {
			switch (item.getItemId()) {
			case R.id.mapTypeNormal:
				this.mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
				break;
			case R.id.mapTypeSatelite:
				this.mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
				break;
			case R.id.mapTypeHybrid:
				this.mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
				break;
			case R.id.mapTypeTerrain:
				this.mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

```
                    break;
              case R.id.originalState:

       this.mMap.animateCamera(CameraUpdateFactory.newCameraPosition(origin
alCameraPosition));
                    break;

              default:
                    break;
              }
              return super.onOptionsItemSelected(item);
       }

       private void goToLocation(double lat, double lon){
              LatLng ll = new LatLng(lat, lon);
              CameraUpdate update = CameraUpdateFactory.newLatLng(ll);
              this.mMap.moveCamera(update);
       }


       private void goToLocation(double lat, double lon, float zoom){
              LatLng ll = new LatLng(lat, lon);
              CameraUpdate  update  =  CameraUpdateFactory.newLatLngZoom(ll,
zoom);
              this.mMap.moveCamera(update);
       }

       public boolean servicesOK() {
              int                         isAvailable                         =
GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);

              if (isAvailable == ConnectionResult.SUCCESS) {
                    return true;
              }
              else                                                          if
(GooglePlayServicesUtil.isUserRecoverableError(isAvailable)) {
                    Dialog                      dialog                       =
GooglePlayServicesUtil.getErrorDialog(isAvailable,                    this,
GPS_ERRORDIALOG_REQUEST);
                    dialog.show();
              }
              else {
                    Toast.makeText(this,  "Can't  connect  to  Google  Play
services", Toast.LENGTH_SHORT).show();
              }
              return false;
       }

       private boolean initMap() {
              if (mMap == null) {
                    SupportMapFragment mapFrag =
                                 (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.onemap);
                    mMap = mapFrag.getMap();
              }
              return (mMap != null);
       }
```

```
	//Κλάση η οποία θα είναι υπεύθυνη για να ορίζει LatLng στα αντικείμενα μας
	//, να φτιάξει σε όλα μία λίστα κτλ...
	private static final class FixObjectsManager{

		private List<GeneralClass> lista;

	public FixObjectsManager(){

			//set objects LatLng
			MyActivity.BAR_CANTINA.setLatLng(36.971540, 26.934140);
			MyActivity.BAR_DOMUS.setLatLng(36.971440, 26.934140);
			MyActivity.BAR_SCORPION.setLatLng(36.995276,
26.934593);
			MyActivity.BEACH_PLATIS_YALOS.setLatLng(36.974923,
26.928274);
			MyActivity.BEACH_SIKIES.setLatLng(36.993443,
26.931943);
			MyActivity.SIGHT_EFTA_PARTHENES.setLatLng(36.964740,
26.965609);
			MyActivity.SIGHT_THERMA.setLatLng(36.938921,
26.987180);
			MyActivity.VILLAGE_ARGINONTA.setLatLng(37.014891,
26.970534);
			MyActivity.VILLAGE_EMPORIOS.setLatLng(37.046243,
26.929029);
			MyActivity.CAFE_ELYSE.setLatLng(36.948058, 26.982606);
			MyActivity.CAFE_SOUL.setLatLng(36.948729, 26.982782);
			MyActivity.HOTEL_ELENA.setLatLng(37.005800, 26.941329);
			MyActivity.HOTEL_PLAZA.setLatLng(36.989646, 26.931594);
			MyActivity.TAVERNA_MAMOUZELOS.setLatLng(36.972906,
26.932710);
			MyActivity.TAVERNA_O_PANTELIS.setLatLng(36.947600,
26.982911);

			//ΣΗΜΕΙΩΣΗ: Το schedule και το latlng των αντικειμένων τα έχω ορίσει σε άλλες activities
			//με αποτέλεσμα όταν ανοίγω για πρώτη φορά τον χάρτη οι τιμές να μην υπάρχουν, γιατί δεν
			//έχουν ακόμα οριστεί (κάτι τέτοιο θα γίνει αφού επισκεφθώ τις άλλες activities και
			//επιστρέψω στον χάρτη)

			//set objects schedule
			MyActivity.BAR_CANTINA.setSchedule("All day long!");
			MyActivity.BAR_DOMUS.setSchedule("All day long!");
			MyActivity.BAR_SCORPION.setSchedule("All day long!");
			MyActivity.CAFE_ELYSE.setSchedule("All day long!");
			MyActivity.CAFE_SOUL.setSchedule("All day long!");
			MyActivity.HOTEL_ELENA.setSchedule("All day long!");
			MyActivity.HOTEL_PLAZA.setSchedule("All day long!");
			MyActivity.TAVERNA_MAMOUZELOS.setSchedule("All      day
long!");
			MyActivity.TAVERNA_O_PANTELIS.setSchedule("All      day
long!");


			//create the list
			this.lista=new ArrayList<GeneralClass>();
			this.lista.add(MyActivity.BAR_CANTINA);
```

```
                this.lista.add(MyActivity.BAR_DOMUS);
                this.lista.add(MyActivity.BAR_SCORPION);
                this.lista.add(MyActivity.BEACH_PLATIS_YALOS);
                this.lista.add(MyActivity.BEACH_SIKIES);
                this.lista.add(MyActivity.SIGHT_EFTA_PARTHENES);
                this.lista.add(MyActivity.SIGHT_THERMA);
                this.lista.add(MyActivity.VILLAGE_ARGINONTA);
                this.lista.add(MyActivity.VILLAGE_EMPORIOS);
                this.lista.add(MyActivity.TAVERNA_MAMOUZELOS);
                this.lista.add(MyActivity.TAVERNA_O_PANTELIS);
                this.lista.add(MyActivity.CAFE_ELYSE);
                this.lista.add(MyActivity.CAFE_SOUL);
                this.lista.add(MyActivity.HOTEL_ELENA);
                this.lista.add(MyActivity.HOTEL_PLAZA);
        }

        public List<GeneralClass> getLista(){
                return this.lista;
        }

    }

    private void showAndHideMarkers(int...markers){

        for (int i = 0; i < this.allMarkers.length; i++) {
                this.allMarkers[i].setVisible(false);
        }

        for (int i = 0; i < markers.length; i++) {
                this.allMarkers[markers[i]].setVisible(true);
        }

    }

}
```

Επεξήγηση δραστηριότητας OneMapActivity.java: Η OneMapActivity.java είναι η πρώτη δραστηριότητα προς ανάλυση που υλοποιεί τους Google Maps.

Αρχίζουμε με την εξήγηση των μεθόδων servicesOK() και initMap(). Η μέθοδος servicesOK() ελέγχει τη συνδεσιμότητα της εφαρμογής στο Google Play Services. Σε περίπτωση μη σύνδεσης θα εμφανιστεί ένα σχετικό παράθυρο διαλόγου ή ένα μήνυμα Toast το οποίο θα ενημερώνει ανάλογα τον χρήστη. Η initMap() αρχικοποιεί το πεδίο GoogleMap map της δραστηριότητας, μέσω της μεθόδου getMap() του αντικειμένου της κλάσης SupportMapFragment. Η διαδικασία αυτή, η υλοποίηση των δύο μεθόδων καθώς και η θέση τους στην μέθοδο OneMapActivity.onCreate() είναι αμετάβλητη και θα χρησιμοποιείται σε κάθε κλάση που υλοποιεί χάρτη, για παράδειγμα στην MapActivity.

Η μέθοδος goToLocation() δέχεται ως παραμέτρους το γεωγραφικό μήκος, το γεωγραφικό πλάτος και το ζουμ που θέλουμε να προσδιορίσουμε την κάμερα που εμφανίζει τον GoogleMap. Εμείς μέσα στην onCreate() και μέσα στο statement if(this.initMap()) (δηλαδή οι εργασίες θα υλοποιηθούν εφόσον υπάρχει αντικείμενο GoogleMap) μεταφέρουμε την κάμερα του χάρτη στο ώστε αυτός να δείχνει το νησί της Καλύμνου,

this.goToLocation(36.988653, 26.964342, 11);

έτσι με το που «τρέξει» η δραστηριότητα OneMapActivity θα εμφανιστεί στον χρήστη το νησί της Καλύμνου. Αυτή την θέση της κάμερας του χάρτη την αποθηκεύουμε σε ένα αντικείμενο τύπου CameraPosition, με σκοπό την ανάκτηση της όταν εμείς το επιθυμήσουμε. Στην πραγματικότητα η εφαρμογή θα την χρησιμοποιήσει ως default location και κλικάροντας ένα κουμπί ο χρήστης θα προσδιορίζει εκ νέου την κάμερα του αντικειμένου GoogleMap στην default θέση.

Τώρα ήρθε η ώρα να θέσουμε τους **δείκτες** των αντικειμένων GeneralClass που θα εμφανίζονται στον χάρτη. Οι δείκτες αυτοί θα θα είναι τύπου Marker. Ακολουθούν τα παρακάτω βήματα για την προσθήκη των Markers στο αντικείμενο GoogleMap:

- Δημιουργία αντικειμένου της εσωτερικής κλάσης FixObjectsManager από όπου θα λάβουμε τη λίστα με τα αντικείμενα τα οποία θα εμφανίσουν Marker στον χάρτη
- Προσπέλαση της λίστας και για κάθε αντικείμενο GeneralClass προσθήκη ενός Marker στο αντικείμενο GoogleMap

Ένας Marker προστίθεται στον χάρτη ως εξής:

```
Marker marker1 = this.mMap.addMarker(new MarkerOptions());
```

Ωστόσο μπορούμε να χρησιμοποιήσουμε απευθείας τις μεθόδους του αντικειμένου MarkerOptions ταυτοχρόνως με την αρχικοποίηση του αντικειμένου marker. Για παράδειγμα…

```
Marker marker1 = mMap.addMarker(new MarkerOptions()
. position()
. title()
. snippet());
```

Ο marker1 προστέθηκε στον χάρτη mMap μέσω της μεθόδου addMarker(). Η θέση του στον χάρτη θα είναι το αντικείμενο της κλάσης LatLng (γεωγραφικό μήκος και γεωγραφικό πλάτος σε ένα – Latitude Longitude) που θα μπει ως παράμετρος στην μέθοδο position της κλάσης MarkerOptions. Ο τίτλος του παράθυρου του marker θα είναι μία παράμετρος String στην μέθοδο title κτλ.

Όταν ο χρήστης κάνει κλικ σε έναν δείκτη, αυτόματα η κάμερα του χάρτη κεντράρει σε αυτόν, ενώ εμφανίζειται ένα παραθυράκι το οποίο αποκαλείται Info Window. Εμείς αυτό που ζητάμε είναι το κλικ του χρήστη στο Info Window να ανοίγει την κατάλληλη δραστηριότητα του αντικειμένου το οποίο προβάλλεται στον δείκτη. Για παράδειγμα αν ο χρήστης κάνει κλικ στον δείκτη του εστιατορείου «Ο Παντελής» η κάμερα του χάρτη θα κεντράρει σε αυτόν. Αν στη συνέχεια κλικάρει το σχετικό Info Window τότε θα πρέπει να ανοίξει η EpixirisiActivity η οποία θα προβάλει το αντικείμενο Epixirisi MyActivity.TAVERNA_O_PANTELIS.

Η επόμενη εργασία που λαμβάνει μέρος στην OneMapActivity.onCreate() είναι η ρύθμιση του Spinner, του κυλιόμενου μενού όπως φαίνεται στην δεύτερη εικόνα προεπισκόπισης της δραστηριότητας. Η εργασία αυτή υλοποιείται μέσα στην μέθοδο initSpinner(), όπου μετά την αρχικοποίηση του ο Spinner συνδέεται με έναν ArrayAdapter (δημιουργία ενός ArrayAdapter αντικειμένου και κάλεσμα της μεθόδου setDropDownViewResource()).

Στη συνέχεια ορίζεται ένας **ακροατής** για κάθε ένα item του Spinner ξεχωριστά. Όταν ο χρήστης κλικάρει το Spinner αυτό θα εμφανίσει το κυλιόμενο μενού με επιλογές που θα αποτελούν ομάδες των αντικειμένων GeneralClass, όπως "Bar-Club-Pub", "Cafe", "Hotels-Room to rent" και άλλα. Επιλέγοντας μία κατηγορία από το Spinner, θα φιλτραριστούν οι δείκτες του χάρτη, γιατί μην ξεχνάμε πως στην παρούσα φάση ο χάρτης εμφανίζει όλους τους δείκτες των αντικειμένων GeneralClass. Επομένως αν ο χρήστης επιλέξει "Cafe", θα εμφανίζονται μόνο οι δείκτες των καφέ κοκ (μέθοδος showAndHideMarkers()).

Αντίθετα με τις άλλες δραστηριότητες, η OneMapActivity διαθέτει δικό της **μενού επιλογών**, το οποίο επιτρέπει διάφορες ρυθμίσεις όσον αφορά τον χάρτη. Η πρώτη επιλογή φαίνεται πάντα στην ActionBar και είναι η "Original State". Κάνοντας κλικ ο χρήστης θα δει την κάμερα του χάρτη να μετακινείται προς το προεπιλεγμένο σημείο με τις συντεταγμένες της Καλύμνου. Οι άλλες επιλογές αφορούν τον τύπο του αντικειμένου GoogleMap, του χάρτη δηλαδή. Για παράδειγμα ο τύπος του χάρτη μπορεί να επιλεχθεί ως Normal, Satellite, Terrain, Hybrid.

Όσον αφορά την **εγκατάσταση** του μενού στην δραστηριότητα, αυτή επιτυγχάνεται μέσω των onCreateOptionsMenu() και onOptionsItemSelected(). Η πρώτη αρχικοποιεί το μενού μέσω ενός MenuInflater, ενώ στη δεύτερη καθορίζονται οι εργασίες που θα γίνουν εφόσον γίνει κλικ για το κάθε MenuItem ξεχωριστά.

Κλείνοντας, το μεγάλο ζήτημα που αποκαλύπτεται με αυτές τις ρυθμίσεις είναι αυτό της **αποθήκευσης**. Από την στιγμή που καταστραφεί το αντικείμενο της δραστηριότητας OneMapActivity τότε όλες οι ρυθμίσεις θα χαθούν. Για αυτό το σκοπό έχουν δημιουργηθεί οι μέθοδοι:

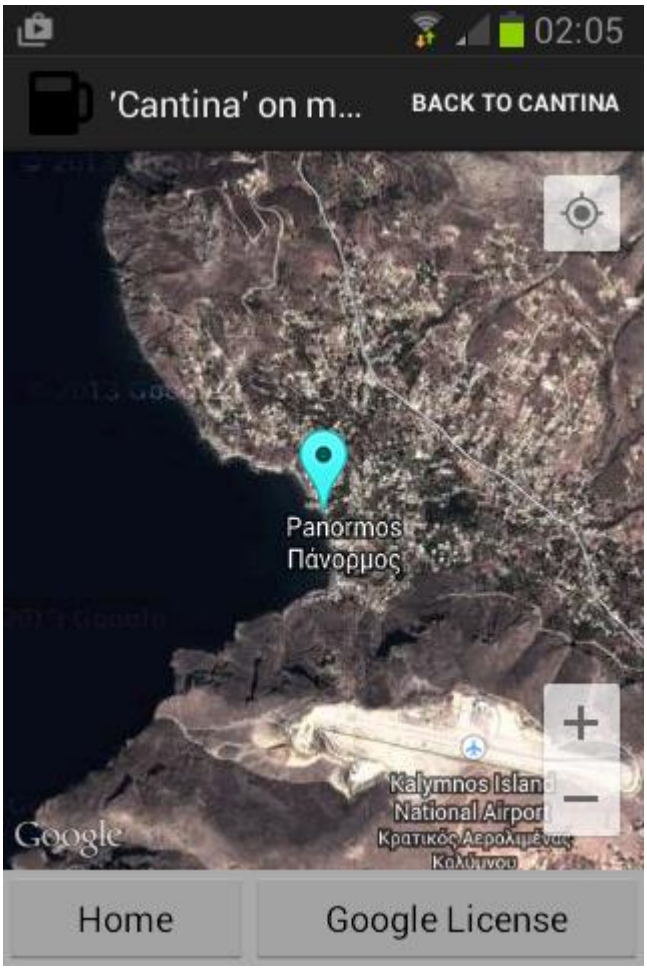
- saveMapType(), αποθηκεύει τον τύπο του χάρτη
- loadMapType() φορτώνει τον τύπο του χάρτη
- saveCameraPosition(CameraPosition) αποθηκεύει την θέση της κάμερας του χάρτη
- getSavedCameraPosition() φορτώνει την θέση της κάμερας του χάρτη

Με αυτές τις τέσσερις μεθόδους είμαστε σε θέσει να αποθηκεύσουμε τις ρυθμίσεις και να τις φορτώσουμε χρησιμοποιώντας φυσικά το interface SharedPreferences του Android SDK. Για να πάρουμε όμως σωστό αποτέλεσμα θα πρέπει το κάλεσμα αυτών των μεθόδων να πραγματοποιηθεί την κατάλληλη στιγμή. Αποθηκεύουμε τις ρυθμίσεις στην onStop() και τις φορτώνουμε στην onResume().

MapActivity.java(activity) + map_activity.xml(layout resource)

Εικόνα 3-11: **Το** outline **του πόρου** map_activity.xml

Επεξήγηση κώδικα map_activity.xml: Η διάταξη του αρχείου πόρου διεπαφής χρήστη της MapActivity είναι απλούστερη από τον σχετικό πόρο της OneMapActivity. Όπως παρατηρούμε στην εικόνα του outline, μέσα στην πατρική RelativeLayout εμπεριέχονται το fragment που θα εμφανίσει τον χάρτη και η μία γραμμή εργαλείων στο κάτω μέρος με δύο ImageButton.



Εικόνα 3-12: Screenshot **της δραστηριότητας** MapActivity.java

```
public class MapActivity extends FragmentActivity {


	private GoogleMap mMap;
	private Marker mMarker;
	private static final int GPS_ERRORDIALOG_REQUEST = 9001;
	private GeneralClass antikeimeno;
	private Button home;
```

```
	private final static String SHAREDPREFENCES_MAPTYPE_FILE_NAME="Save
Google Map Type";
	private final static String SHAREDPREFENCES_MAPTYPE_KEY_NAME="Save
Google Map Type Key";


	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		this.initGeneralClassObject();
		this.initActionBar(this.antikeimeno);
		if(this.servicesOK()){
			this.setContentView(R.layout.map_activity);
			if(this.initMap()){
				this.mMap.setMyLocationEnabled(true);
				if(this.antikeimeno!=null){

	this.goToLocation(this.antikeimeno.getLatLng(), 14);
				}else{
					CameraUpdate update =
CameraUpdateFactory.newLatLngZoom(new LatLng(0, 0), 0);
					this.mMap.moveCamera(update);
					Toast.makeText(this, "GeneralClass object
== null", Toast.LENGTH_LONG).show();
				}
			}
		}
	}


	public void onHomeButtonClick(View v){
		//H  Intent.FLAG_ACTIVITY_CLEAR_TOP  "καθαρίζει"  όλες  τις
προηγούμενες ανοικτές activities
		this.startActivity(new Intent(this,
MainMenu.class).addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
		this.finish();
	}


	@Override
	protected void onStop() {
		super.onStop();
		this.saveMapType();
	}


	@Override
	protected void onResume() {
		super.onResume();
		this.mMap.setMapType(this.loadMapType());
	}

	private int loadMapType(){
		SharedPreferences sharedPrefs =
this.getSharedPreferences(this.SHAREDPREFENCES_MAPTYPE_FILE_NAME,
Context.MODE_PRIVATE);
		return
sharedPrefs.getInt(this.SHAREDPREFENCES_MAPTYPE_KEY_NAME,
GoogleMap.MAP_TYPE_NORMAL);
```

```
	}

	private void saveMapType(){
		SharedPreferences sharedPrefs =
this.getSharedPreferences(this.SHAREDPREFENCES_MAPTYPE_FILE_NAME,
Context.MODE_PRIVATE);
		SharedPreferences.Editor editor=sharedPrefs.edit();
		editor.putInt(this.SHAREDPREFENCES_MAPTYPE_KEY_NAME,
this.mMap!=null ? this.mMap.getMapType() : GoogleMap.MAP_TYPE_NORMAL);
		editor.commit();
	}

	private void initActionBar(GeneralClass antikeimeno) {
		ActionBar bar = this.getActionBar();
		bar.setTitle(" '"+this.antikeimeno.getName()+"' on map");
		bar.setIcon(this.antikeimeno.getIconResource());

	}


	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		 MenuInflater inflater = getMenuInflater();
		 inflater.inflate(R.menu.map, menu);
		return true;
	}


	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		switch (item.getItemId()) {
		case R.id.goBackToGeneralClass:
			item.setTitle("Back to "+this.antikeimeno.getName());

	this.goToLocationViaAnimation(this.antikeimeno.getLatLng(), 14);
			break;
		case R.id.mapTypeHybrid:
			this.mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
			break;
		case R.id.mapTypeNormal:
			this.mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
			break;
		case R.id.mapTypeSatelite:
			this.mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
			break;
		case R.id.mapTypeTerrain:
			this.mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);

			break;

		default:
			break;
		}
		return super.onOptionsItemSelected(item);
	}


	private void initGeneralClassObject(){
		Intent intent = this.getIntent();
```

```
		String name =
intent.getStringExtra(MyActivity.INTENT_EXTRA_KEY_FOR_PASSING_GENERALCLASS_
TO_MAPACTIVITY);
		if(name!=null){

	if(name.equals(this.getResources().getString(R.string.bar_cantina)))
{
				this.antikeimeno=MyActivity.BAR_CANTINA;
				this.antikeimeno.setLatLng(36.971440, 26.934140);
			}else
if(name.equals(this.getResources().getString(R.string.bar_domus))){
				this.antikeimeno=MyActivity.BAR_DOMUS;
				this.antikeimeno.setLatLng(36.971440, 26.934140);
			}else
if(name.equals(this.getResources().getString(R.string.bar_scorpion))){
				this.antikeimeno=MyActivity.BAR_SCORPION;
				this.antikeimeno.setLatLng(36.995276, 26.934593);
			}else
if(name.equals(MyActivity.BEACH_PLATIS_YALOS.getName())){
				this.antikeimeno=MyActivity.BEACH_PLATIS_YALOS;
				this.antikeimeno.setLatLng(36.974923, 26.928274);
			}else
if(name.equals(MyActivity.BEACH_SIKIES.getName())){
				this.antikeimeno=MyActivity.BEACH_SIKIES;
				this.antikeimeno.setLatLng(36.993443, 26.931943);
			}else
if(name.equals(MyActivity.SIGHT_EFTA_PARTHENES.getName())){
				this.antikeimeno=MyActivity.SIGHT_EFTA_PARTHENES;
				this.antikeimeno.setLatLng(36.964740, 26.965609);
			}else
if(name.equals(MyActivity.SIGHT_THERMA.getName())){
				this.antikeimeno=MyActivity.SIGHT_THERMA;
				this.antikeimeno.setLatLng(36.938921, 26.987180);
			}else
if(name.equals(MyActivity.VILLAGE_ARGINONTA.getName())){
				this.antikeimeno=MyActivity.VILLAGE_ARGINONTA;
				this.antikeimeno.setLatLng(37.014891, 26.970534);
			}else
if(name.equals(MyActivity.VILLAGE_EMPORIOS.getName())){
				this.antikeimeno=MyActivity.VILLAGE_EMPORIOS;
				this.antikeimeno.setLatLng(37.046243, 26.929029);
			}
		}
	}

	private void goToLocation(LatLng ll, float zoom){
		CameraUpdate update = CameraUpdateFactory.newLatLngZoom(ll,
zoom);
		this.mMap.moveCamera(update);

		if(this.mMarker!=null){
			this.mMarker.remove();
		}
		MarkerOptions options = new MarkerOptions().
				title(this.antikeimeno.getName()).
				position(ll).

	icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.H
UE_CYAN));
```

```
            this.mMarker=this.mMap.addMarker(options);
      }


      private void goToLocationViaAnimation(LatLng ll, float zoom){
            Toast.makeText(this,          "Moving          back          to
"+this.antikeimeno.getName(), Toast.LENGTH_SHORT).show();
            CameraUpdate  update  =  CameraUpdateFactory.newLatLngZoom(ll,
zoom);
            this.mMap.animateCamera(update);

            if(this.mMarker!=null){
                  this.mMarker.remove();
            }
            MarkerOptions options = new MarkerOptions().
                        title(this.antikeimeno.getName()).
                        position(ll).

      icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.H
UE_CYAN));
            this.mMarker=this.mMap.addMarker(options);
      }


      public boolean servicesOK() {
            int                         isAvailable                         =
GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);

            if (isAvailable == ConnectionResult.SUCCESS) {
                  return true;
            }
            else                                                          if
(GooglePlayServicesUtil.isUserRecoverableError(isAvailable)) {
                  Dialog                        dialog                      =
GooglePlayServicesUtil.getErrorDialog(isAvailable,                    this,
GPS_ERRORDIALOG_REQUEST);
                  dialog.show();
            }
            else {
                  Toast.makeText(this,  "Can't  connect  to  Google  Play
services", Toast.LENGTH_SHORT).show();
            }
            return false;
      }

      private boolean initMap() {
            if (mMap == null) {
                  SupportMapFragment mapFrag =
                              (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
                  mMap = mapFrag.getMap();
            }
            return (mMap != null);
      }



}
```

Επεξήγηση κώδικα δραστηριότητας MapActivity: Φθάσαμε στην τελευταία δραστηριότητα – κλάση προς ανάλυση. Η MapActivity, εφόσον χρησιμοποιείται από την εφαρμογή για την προβολή ενός και μόνο αντικειμένου GeneralClass, καλείται από σχετική ImageView των δραστηριοτήτων προβολής ειδικού περιεχομένου, όπως η EpixirisiActivity, BeachActivity, SightActivity και VillageActivity.

Οι δραστηριότητες προβολής ειδικού περιεχομένου θα μεταβούν στην MapActivity μέσω μίας πρόθεσης Intent η οποία «κουβαλάει» μαζί της και το όνομα του αντικειμένου (GeneralClass.getName()). Αυτό συμβάλλει στην αρχικοποίηση του πεδίου GeneralClass antikeimeno της MapActivity, έτσι ώστε μέσω της initGeneralClassObject() να γνωστοποιηθεί το αντικείμενο του οποίου η θέση θα προβληθεί στον χάρτη.

Όταν γνωστοποιηθεί το αντικείμενο, η μέθοδος goToLocation() θα μετατοπίσει την κάμερα του χάρτη σύμφωνα με τις γεωγραφικές συντεταγμένες του αντικειμένου και θα προσθέσει έναν δείκτη (Marker).

Ο χάρτης που προβάλλεται από την MapActivity φέρει το σχετικό button το οποίο μεταφέρει την κάμερα του χάρτη στην **πραγματική θέση** χρήστη. Σε συνδιασμό με button της ActionBar το οποίο μεταφέρει την κάμερα του χάρτη πίσω στο αντικείμενο GeneralClass, ο χρήστης μπορεί να εκτιμίσει την απόσταση του από τον τελικό προορισμό.

Κλείνοντας, όπως και στην OneMapActivity, έτσι και στην MapActivity βρίσκουμε το μενού της δραστηριότητας με τις ίδιες επιλογές, αυτές που αλλάζουν τον τύπου του χάρτη (Map type).

4 Συμπεράσματα

4.1 Προβλήματα, Περιορισμοί

Κατά τη διαδικασία παραγωγής της εφαρμογής KalymnosGuide βρεθήκαμε αντιμέτωποι με τα εξής προβλήματα-περιορισμούς. Πρώτον, το IDE που χρησιμοποιήθηκε για την εγγραφή του κώδικα της εφαρμογής, το Eclipse, φέρει πάρα πολλά bugs κάνοντας δύσκολη τη ζωή του developer ο οποίος πρέπει να ανατρέχει συνεχώς online προς εύρεση λύσης. Να 'ναι καλά η διαδικτυακή κοινότητα του stackoverflow!

Δεύτερον, εφόσον ένας developer δεν έχει πραγματική συσκευή, θα αναγκαστεί να κάνει χρήση των ανάλογων εξομοιωτών που προσφέρει το Eclipse. Ωστόσο θα πρέπει να γνωρίζει πως οι εξομοιωτές αυτοί τρώνε αρκετούς πόρους από τον ηλεκτρονικό του υπολογιστή, ιδιαίτερα μνήμη RAM. Επομένως ο developer θα πρέπει να διαθέτει και ένα αξιοπρεπές σύστημα για να τρέχει τους εξομοιωτές χωρίς μεγάλες χρονοτριβές.

Ωστόσο, και να πληρεί την παραπάνω προϋπόθεση ένας developer θα πρέπει να κατέχει πραγματική συσκευή που τρέχει το λειτουργικό Android, ώστε να δει πως αντιδράει η εφαρμογή του σε μία πραγματική συσκευή. Εξάλλου οι εξομοιωτές δεν διαθέτουν GPS και άλλα χαρακτηριστικά που διαθέτει μία πραγματική συσκευή, ώστε να γίνουν οι απαραίτητες δοκιμές.

4.2 Μελλοντικές επεκτάσεις

Η εφαρμογή KalymnosGuide θα μπορούσε να ενημερωθεί μελλοντικά με τις εξής επεκτάσεις:

1. Να προσθέσουμε ολόκληρους τους καταλόγους των καταστημάτων και των ξενοδοχείων με τα μενού, τις προσφορές , τις δραστηριότητες και τις φωτογραφίες από τον χώρο τους.

2. Να ανανεώνει τις βαθμολογίες των καταστημάτων που δίνει ο χρήστης μας με κάθε σύνδεση που κάνει στο ίντερνετ.

3. Να μπορεί ο χρήστης να γράφει κριτικές για τα καταστήματα που επισκέπτεται.

4. Να προστεθούν καλύτερα γραφικά

5. Να εμφανίζει ότι event μπορεί να έχει εκείνη τη μέρα στην τοποθεσία που βρίσκεται ο χρήστης.

6. Να προστεθεί άλλη μια κατηγορία στο μενού, δραστηριότητες που θα μπορούσε να κάνει ο χρήστης και που μπορεί να τις βρει.

7. Επιπλέον μια κατηγορία με τα χρήσιμα τηλέφωνα, όπως νοσοκομεία, κτελ, ταξί κτλ

8. Στην κατηγορία με τα καταλύματα θα μπορεί ο χρήστης να βλέπει την διαθεσιμότητα τους.

Αναφορές

1. Deitel, Android Προγραμματισμός, Γκιούρδας (2014)
2. Marko Gargenta, Learning Android, O Reilly, (2011)
3. W-M Lee, Beginning Android Application Development, Wrox programmer to programmer, (2011)
4. Επίσημη ιστοσελίδα Android development http://developer.android.com/sdk/
5. Google Maps Android API v2, https://developers.google.com/maps/documentation/android/
6. Ο επίσημος ιστότοπος της Oracle Java, https://www.oracle.com/java/index.html