Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά Συστήματα

Ανάπτυξη συστήματος γεωγραφικού εντοπισμού δυνατοτήτων αξιοποίησης ελεύθερου χρόνου στην πλατφόρμα Android.

Κωνσταντίνος Αθανασόπουλος

Επιβλέπων Καθηγητής
Βασίλης Βασσάλος

Αύγουστος 2012

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Ανάπτυξη συστήματος γεωγραφικού εντοπισμού δυνατοτήτων αξιοποίησης ελεύθερου χρόνου στην πλατφόρμα Android.

Κωνσταντίνος Αθανασόπουλος

Επιβλέπων Καθηγητής
Βασίλης Βασσάλος

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Αύγουστος 2012

Στην Άννα και τη Μαρία

Περίληψη

Σκοπός της παρούσης διατριβής είναι η ανάλυση, σχεδίαση, κατασκευή (προγραμματισμός), εγκατάσταση και λειτουργία ενός σύγχρονου υπολογιστικού συστήματος που αποτελείται από τέσσερα διακριτά αλλά συνεργαζόμενα μεταξύ τους μέρη:

1. Εφαρμογή που εξάγει αυτοματοποιημένα πληροφορίες για τα σημεία ενδιαφέροντος από τον παγκόσμιο ιστό μαζί με τις γεωγραφικές συντεταγμένες αυτών των σημείων, δηλαδή το γεωγραφικό πλάτος (latitude) (φ) και το γεωγραφικό μήκος (longitude) (λ).

2. Σχεσιακή βάση δεδομένων ικανή να διαχειριστεί χωρικά δεδομένα στην οποία καταχωρούνται τα προηγούμενα στοιχεία.

3. Υπηρεσία διαδικτύου που εκτελείται στον διακομιστή (server) και χρησιμοποιεί RESTful web service (web services που βασίζονται στην αρχιτεκτονική REST - Representational state transfer). Η υπηρεσία αυτή καθιστά διαθέσιμα κατάλληλα επεξεργασμένα τα δεδομένα της βάσης ώστε να καταναλωθούν από έξυπνες κινητές συσκευές τύπου Android.

4. Εφαρμογή που εκτελείται στο κινητό Android και εμφανίζει αυτά τα σημεία ενδιαφέροντος σε χάρτη, πληροφορίες δρομολόγησης και άλλες πιο εξειδικευμένες πληροφορίες που εξαρτώνται από τον τύπο του κάθε σημείου.

Λέξεις Κλειδιά: Web Crawler, Web bot, Web spider, γεωχωρικά δεδομένα, χωρικά δεδομένα, διαδικτυακές υπηρεσίες, Restful διαδικτυακές υπηρεσίες, Android, χάρτες, δρομολόγηση.

Summary

In this thesis we analyse, and describe the development, installation and operation of an end-to-end information system which consists of the following modules:

1. An application that automatically gathers information about points of interest (POI) from the WEB, including the point's geographical coordinates, meaning latitude (φ) and longitude (λ).

2. A relational database, able to manage spatial data which are imported from previous module.

3. A RESTfull web service which prepares the data from module 2, in order to be consumed from smart devices i.e. android phones

4. An Android smart phone application which displays all these points of interest in the map, routing information, and other more specific information, depending on the type of each point.

Keywords: Web Crawler, Web bot, Web spider, geospatial data, spatial data, web services, Restful web services, Android, maps, routing.

Ευχαριστίες

Ευχαριστώ τον Καθηγητή Βασίλη Βασσάλο για την βοήθεια του στη σχεδίαση της παρούσης διατριβής, τις πολύτιμες συμβουλές του στη διάρκειά της και κυρίως την υποστήριξη του στις δύσκολες ώρες. Επίσης, ευχαριστώ την Άννα και τη Μαρία για την συμπαράσταση και την υπομονή που επέδειξαν σε όλη την διάρκεια αυτού του μεταπτυχιακού

Περιεχόμενα

Εικόνες

Πίνακες

Κεφάλαιο 1
Εισαγωγή

Η διάρθρωση της παρούσης διατριβής έχει ως εξής:

Το Πρώτο κεφάλαιο αποτελεί η εισαγωγή.

Στο Δεύτερο κεφάλαιο θα εκθέσουμε πώς έγινε η επιλογή των εργαλείων και των ιστοχώρων για την αυτοματοποιημένη εξαγωγή των δεδομένων.

Στο Τρίτο κεφάλαιο θα αναφερθούμε στα κριτήρια επιλογής του Συστήματος Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ, Database Management System), στις ρυθμίσεις του και στις βιβλιοθήκες που χρησιμοποιήσαμε για την επικοινωνία με το πρόγραμμα συλλογής δεδομένων.

Στο Τέταρτο κεφάλαιο θα αναφερθούμε στις βασικές οντότητες της εφαρμογής μας, τις σχέσεις μεταξύ τους και την σχεδίαση της βάσης δεδομένων.

Στο Πέμπτο κεφάλαιο θα αναφέρουμε κάποιες τεχνικές συλλογής δεδομένων.

Στο Έκτο κεφάλαιο θα αναφερθούμε συνοπτικά στα βασικά των web services, στην προδιαγραφή jax-rs, γιατί επιλέξαμε την java βιβλιοθήκη jersey για να τα υλοποιήσουμε, την βασική εγκατάσταση, καθώς και στην σχεδίαση των υπηρεσιών ιστού.

Στο Έβδομο κεφάλαιο θα αναφερθούμε συνοπτικά στο Android, σε κάποια σημεία που θεωρούμε ότι πρέπει να αναφέρουμε για τις δυσκολίες που συναντήσαμε στην υλοποίηση και θα παρουσιάσουμε κάποιες οθόνες της εφαρμογής.

Το Όγδοο κεφάλαιο αναφέρεται σε θέματα σχεδίασης του συστήματος σαν όλον και την ενοποίηση - ολοκλήρωση των προηγούμενων τμημάτων.

Το Ένατο κεφάλαιο αποτελεί τον επίλογο.

Στο Παράρτημα υπάρχουν οδηγίες εγκατάστασης των εργαλείων που εγκαταστήσαμε στη διάρκεια της ανάπτυξης.

Κεφάλαιο 2
Συλλογή Δεδομένων

Στο κεφάλαιο αυτό θα εκθέσουμε πώς έγινε η επιλογή των εργαλείων και των ιστοχώρων για την αυτοματοποιημένη εξαγωγή των δεδομένων.

2.1 Εργαλεία - βιβλιοθήκες εξαγωγής δεδομένων

Οι βιβλιοθήκες που μπορούν να χρησιμοποιηθούν χωρίζονται σε 2 μεγάλες κατηγορίες:

1. Μετατροπείς HTML σε XML (Html to XML converters) και

2. HTML Parsers

2.1.1 Μετατροπείς HTML σε XML (Html to XML converters)

Στην κατηγορία αυτή ανήκουν βιβλιοθήκες λογισμικού που παίρνουν σαν είσοδο HTML οποιασδήποτε ποιότητας και στην έξοδο δίνουν καλά φορμαρισμένο (well form) XML. Από αυτό το XML στη συνέχεια μπορούμε να προσδιορίσουμε τα εσωτερικά στοιχεία (elements ή attributes) που χρειαζόμαστε χρησιμοποιώντας εκφράσεις XPath ή και σε κάποιες περιπτώσεις εκφράσεις Language Integrated Query (LINQ, προφέρεται "link") εφόσον πρόκειται για .net βιβλιοθήκες. Οι πιο σημαντικές βιβλιοθήκες αυτής της κατηγορίας είναι:

1. HTML Tidy Library Project[1]: Το παλαιότερο, καλύτερο και αυτό που χρειάζεται τον μεγαλύτερο χρόνο εκμάθησης. Μπορεί να χρησιμοποιηθεί επίσης για τον έλεγχο της σύνταξης του HTML και την όμορφη εκτύπωση του. Η τελευταία έκδοση χρονολογείται από το 2008. Η έκδοση για windows δεν είναι συμβατή με το .net. Μπορεί, βέβαια, να εισαχθεί σε .net βιβλιοθήκη με μια διαδικασία η οποία εξαρτάται από τον υπολογιστή στον οποίο θα εκτελεστεί και δεν είναι μεταφέρσιμη.

2. Το JTidy είναι μια έκδοση σε Java του HTML Tidy[2]. Μια αντίστοιχη έκδοση για .net είναι το tidy.net [3].

3. Μία ακόμη βιβλιοθήκη σε .net είναι το SgmlReader[4] ενώ ο ανταγωνιστής του το HTML Agility Pack[5] παρότι χρησιμοποιεί XPath σύνταξη και υποστηρίζει LINQ to Objects ανήκει στην επόμενη κατηγορία.

2.1.2 HTML Parsers

Από την κατηγορία αυτή ερευνήθηκαν οι βιβλιοθήκες:

1. HTML Agility Pack για .net.

2. TagSoup[6] για Java και C++.

3. JSoup[7].

4. NSoup[8] μία έκδοση σε .net του JSoup. Δεν καταφέραμε να την κάνουμε να δουλέψει με κωδικοποίηση χαρακτήρων ISO 8859-7 και εγκαταλείψαμε την προσπάθεια.

Υπάρχουν πολλές ακόμη βιβλιοθήκες κυρίως σε γλώσσα Java οι οποίες υστερούν έναντι της JSoup την οποία αποφασίσαμε να χρησιμοποιήσουμε.

[1] http://tidy.sourceforge.net/
[2] http://sourceforge.net/projects/jtidy/
[3] http://sourceforge.net/projects/tidynet/
[4] http://developer.mindtouch.com/Community/SgmlReader_1.7.2
[5] http://www.codeplex.com/htmlagilitypack
[6] http://mercury.ccil.org/~cowan/XML/tagsoup/)
[7] http://jsoup.org/
[8] http://nsoup.codeplex.com/

Γενικότερα η μετατροπή HTML σε XML είναι πιο χρονοβόρα και καταναλώνει περισσότερους υπολογιστικούς πόρους. Επίσης, η ποιότητα του XML που παράγεται είναι ανάλογη της ποιότητας του HTML και οι περισσότεροι μετατροπείς σε XML δεν καθαρίζουν πρώτα το HTML.

2.2 Γιατί JSoup και Java

Η βιβλιοθήκη JSoup παρέχεται με την άδεια «MIT License». Αναφέρεται στον ιστοχώρο της βιβλιοθήκης: «Η JSoup είναι μια βιβλιοθήκη της Java για εργασία με HTML που συναντάμε στον πραγματικό κόσμο. Παρέχει ένα πολύ βολικό API για την εξαγωγή και το χειρισμό των δεδομένων χρησιμοποιώντας τα καλύτερα του DOM, CSS και jquery-like μεθόδους.»

Η χρήση CSS (Cascading Style Sheets) που μας επιτρέπει τον εντοπισμό και κυρίως μορφοποίηση αντικειμένων σε έγραφα HTML, XHTML και XML, η χρήση DOM(Document Object Model) που μας επιτρέπει την αλληλεπίδραση με αυτά τα αντικείμενα και είναι ανεξάρτητη γλώσσας και πλατφόρμας και η χρήση μεθόδων σαν αυτές που χρησιμοποιεί η JavaScript βιβλιοθήκη jQuery μας επιτρέπουν να αξιοποιήσουμε προϋπάρχουσα γνώση την οποία αποκτήσαμε ή εμβαθύναμε σ' αυτήν κατά τη φοίτηση μας στο ΑΠΚυ.

Κατά τη διάρκεια των αναζητήσεων μιας βιβλιοθήκης που θα χρησιμοποιούσαμε για την εξαγωγή των δεδομένων δεν είχαμε περιοριστεί σε βιβλιοθήκες Java παρά είχαμε συμπεριλάβει και αντίστοιχες βιβλιοθήκες σε .net. Η απόφαση για χρήση της JSoup είναι συνδεδεμένη με την απόφαση να χρησιμοποιηθεί ως η μοναδική γλώσσα προγραμματισμού στην παρούσα μεταπτυχιακή διατριβή η γλώσσα Java. Η Java υποχρεωτικά θα ήταν η γλώσσα προγραμματισμού στην πλατφόρμα Android. Δεν ήταν όμως η μοναδική επιλογή για την υλοποίηση των άλλων τριών τμημάτων του υπολογιστικού συστήματος που αναπτύχθηκε στην παρούσα μεταπτυχιακή διατριβή.

Η γλώσσα Java όπως άλλωστε και η C# είναι μια αντικειμενοστραφής γλώσσα. Προσφέρεται για χρήση με το αντικειμενοστραφές στυλ (υπόδειγμα ή παράδειγμα) προγραμματισμού και μας προσέφερε σημαντική βοήθεια στην προσπάθεια μας να το ακολουθήσουμε.

2.3 Αναζήτηση υλικού στον παγκόσμιο ιστό

Το υλικό που έπρεπε να εντοπίσουμε επιλέξαμε να έχει τα παρακάτω χαρακτηριστικά:

1. Έπρεπε για να είναι αξιοποιήσιμο να περιέχει γεωγραφικά στοιχεία για τα διάφορα σημεία ενδιαφέροντος και τα στοιχεία αυτά να μην είναι κλειδωμένα για να μπορούν να διαβαστούν.

2. Σημαντικό είναι επίσης το HTML να είναι καλής σχετικά ποιότητας.

Χρήσιμα αλλά όχι αναγκαία είναι:

1. Το υλικό να είναι δομημένο με κάποιο λογικό τρόπο. Δηλαδή, η αναπαράσταση της πληροφορίας να μας δίνει τη δυνατότητα να ξεχωρίσουμε τα διάφορα αντικείμενα μεταξύ τους.

2. Τα URLs (Uniform Resource Locator) των διαφόρων πόρων να υπολογίζονται εύκολα, συνήθως με την χρήση ενός μοναδικού ακεραίου. Έτσι, μπορούσαμε να εντοπίζουμε εύκολα τις σελίδες με τις λεπτομέρειες των σημείων ενδιαφέροντος.

3. Το υλικό να είναι εύκολα προσβάσιμο. Δηλαδή, ο ιστοχώρος να έχει ένα ικανοποιητικό. εύρος ζώνης (bandwidth).

Μερικοί από τους ιστοχώρους που εντοπίσαμε είναι:

- Ο ιστοχώρος In.gr http://entertainment.in.gr/. Στη διάρκεια της ανάπτυξης άλλαξε κωδικοποίηση χαρακτήρων από ISO-8859-7 σε UTF-8.

- Ο ιστοχώρος http://www.athinorama.gr/ και http://www.athinorama.gr/mobile/. Η μεταπήδηση από τον έναν στον άλλο είναι εύκολη με τη βοήθεια του κωδικού του σημείου.

- Ο ιστοχώρος http://www.cinemanews.gr/ για κινηματογράφους και

- Ο ιστοχώρος http://www.ask4food.gr/ για εστιατόρια.

- Ο ιστοχώρος http://movies.pathfinder.gr/. Μόνο για ταινίες. Έχει αλλάξει πολύ από τότε που άρχισε η παρούσα μεταπτυχιακή διατριβή. Τότε έκανε μια προσπάθεια να ξαναγραφεί σε HTML5.

Αποφασίσαμε να δουλέψουμε με το Αθηνόραμα εκτός από κινηματογράφους όπου υπάρχει μια σύγχυση σχετικά με τα URIs των multiplex και όχι multiplex κινηματογράφων. Κινηματογράφοι

με περισσότερες από μία αίθουσες έχουν δύο τουλάχιστον διαφορετικά URIs κάτι που δεν βοηθά στην ταυτοποίηση τους. Τους κινηματογράφους τους διαβάσαμε από το in.gr. Επίσης, τα εστιατόρια τα διαβάσαμε από το ask4food.gr.

2.3.1 Μια ακόμη χρήση της JSoup. Ενσωμάτωση πληροφοριών από το IMDB

Το Internet Movie Database (IMDb) είναι μια διαδικτυακή βάση δεδομένων που παρέχει πληροφορίες που σχετίζονται με ταινίες, τηλεοπτικά προγράμματα κλπ. Ανακαλύψαμε ότι στον ιστοχώρο http://www.imdbapi.com παρέχεται ένα web service με πληροφορίες από τη διαδικτυακή βάση. Έτσι, ενσωματώσαμε μια υπηρεσία στις ταινίες μας που παραπέμπει στην διαδικτυακή αυτή βάση. Η κλήση γίνεται χρησιμοποιώντας το όνομα της ταινίας και μας επιστρέφει την ηλεκτρονική διεύθυνση της σελίδας της ταινίας. Για να διαβάσουμε την διεύθυνση χρησιμοποιούμε τη βιβλιοθήκη Jsoup.

Σημείωση: Στη διάρκεια συγγραφής της παρούσης πληροφορηθήκαμε ότι ο ιστοχώρος imdbapi.com βρίσκεται σε νομική διαμάχη με την IMDb και πιθανώς να πάψει να παρέχει δεδομένα.

2.3.2 Δεδομένα δήμων του προγράμματος Καλλικράτης

Προκειμένου να μπορούμε να εντοπίζουμε σε ποιον δήμο ανήκει κάθε σημείο ενδιαφέροντος χρειάστηκε να βρούμε από τον ιστότοπο geodata.gov.gr (και πιο συγκεκριμένα από την ιστοσελίδα με τίτλο Όρια Δήμων (Καλλικράτης)[9]) τα όρια των δήμων της Ελλάδος σύμφωνα με το πρόγραμμα Καλλικράτης. Σύμφωνα με όσα αναφέρονται στον ιστότοπο τα αρχεία SHP με γεωχωρικά δεδομένα που προσφέρονται από το geodata.gov.gr ακολουθούν το σύστημα αναφοράς ΕΓΣΑ87 (EPSG:2100) καθώς προσφέρει καλύτερη ακρίβεια για την Ελληνική επικράτεια και χρησιμοποιείται από όλους τους χρήστες GIS στην Ελλάδα. Επίσης, η κωδικοποίηση των χαρακτήρων (charchet encoding) των δεδομένων είναι ISO-8859-7. Σχετικά με το σύστημα αυτό μπορεί κανείς να δει στην διεύθυνση http://spatialreference.org/ref/epsg/2100/.

9 http://geodata.gov.gr/geodata/index.php?option=com_sobi2&sobi2Task=sobi2Details&catid=16&sobi2Id=182&Itemid=

2.4 Στήσιμο περιβάλλοντος εργασίας

Για την ευκολότερη ανάπτυξη προγραμμάτων σε Java αλλά και σε άλλες γλώσσες προγραμματισμού χρησιμοποιούνται τα ονομαζόμενα Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης (Integrated Development Environments) τα οποία βοηθούν τον προγραμματιστή μεταξύ άλλων στην συγγραφή του κώδικα και προσφέρουν βοήθεια στο compilation, την εκτέλεση και στον εντοπισμό σφαλμάτων (debugging) του προγράμματος. Εμείς αποφασίσαμε να εγκαταστήσουμε και να χρησιμοποιήσουμε το Ολοκληρωμένο Περιβάλλον Eclipse.

Πριν την εγκατάσταση του Eclipse πρέπει να εγκατασταθεί το Java JDK το οποίο χρησιμοποιεί το Eclipse για το compilation, την εκτέλεση και το debugging των προγραμμάτων. Η λήψη του μπορεί να γίνει από την ηλεκτρονική διεύθυνση Java SE Downloads[10].

Στη συνέχεια, κατεβάζουμε την βιβλιοθήκη JSoup[11] και την βιβλιοθήκη καταγραφής συμβάντων (logging library) Apache log4j[12].

Ακολούθως, δημιουργούμε ένα καινούργιο project (File > New > Java Project). Το Eclipse δημιουργεί αυτόματα ένα φάκελο με όνομα src όπου αποθηκεύεται όλος ο κώδικας του project. Φτιάχνουμε έναν φάκελο με όνομα lib στον οποίο τοποθετούμε τις βιβλιοθήκες jsoup-1.6.1.jar και log4j-1.2.16.jar που κατεβάσαμε πριν. Κάνοντας δεξί κλικ επάνω τους και επιλέγοντας *Build Path > Add to Build Path* τις προσθέτουμε στο project μας. Η ως τώρα διαμόρφωση (configuration) είναι αρκετή για το διάβασμα των δεδομένων μας. Πιο κάτω θα μιλήσουμε για το τι πρέπει να κάνουμε για να επικοινωνήσουμε με την βάση.

[10] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[11] http://jsoup.org/download
[12] http://logging.apache.org/log4j/1.2/download.html

Κεφάλαιο 3
Αποθήκευση Δεδομένων

Στο κεφάλαιο αυτό θα αναφερθούμε στα κριτήρια επιλογής του Συστήματος Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ, Database Management System), στις ρυθμίσεις του και στις βιβλιοθήκες που χρησιμοποιήσαμε για την επικοινωνία με το πρόγραμμα συλλογής δεδομένων.

3.1 Αναζήτηση του RDBMS

Περιορίσαμε την αναζήτηση μας σε σχεσιακές (relational - RDBMS) βάσεις δεδομένων, λόγω του μικρού μεγέθους των δεδομένων και του χαρακτήρα της εφαρμογής αφήνοντας να εξετάσουμε την περίπτωση χρήσης μιας NoSQL[13] βάσης δεδομένων DBMS μελλοντικά, αν η εφαρμογή ανέβει στο «σύννεφο» (Cloud). Ένα άλλο χαρακτηριστικό που περιόρισε την αναζήτηση ήταν η ικανότητα της βάσης να χειριστεί χωρικά δεδομένα Τέτοιες βάσεις είναι οι PostgreSQL, MySQL, Oracle 10g/11g και SQLServer. Η πρώτη εξ αυτών είναι πλήρως ελεύθερη και με την βοήθεια του PostGIS[14], ενός προγράμματος ανοιχτού κώδικα, έχει υποστήριξη χωρικών δεδομένων, ακολουθώντας τις προδιαγραφές Simple Features for SQL[15] (ISO 19125) της κοινοπραξίας Open

[13] http://en.wikipedia.org/wiki/NoSQL
[14] http://en.wikipedia.org/wiki/Postgis
[15] http://en.wikipedia.org/wiki/Simple_Features.

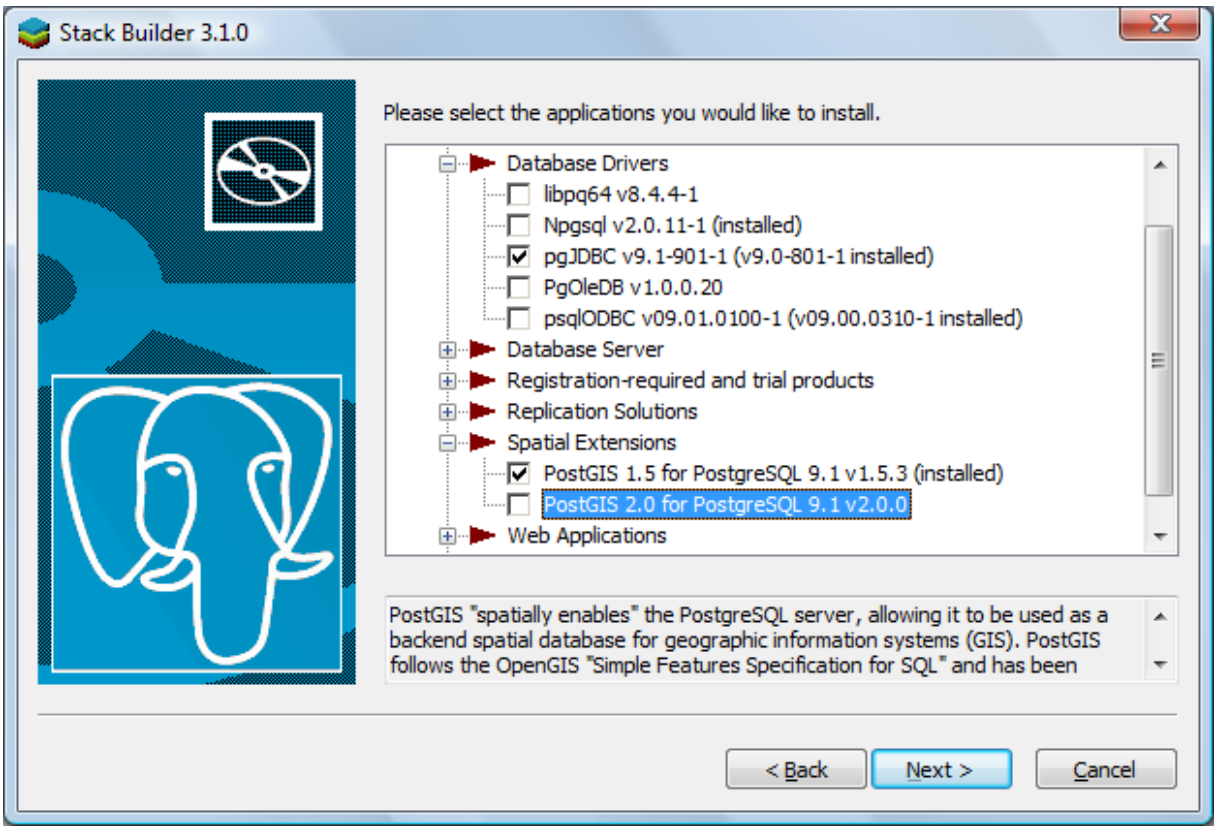
Geospatial Consortium (OGC).Η MySQL έχει και ελεύθερη έκδοση αλλά η υποστήριξη σε χωρικά δεδομένα υπολείπεται σημαντικά της PostgreSQL. Με δεδομένο ότι τα άλλα δύο RDBMS είναι εμπορικά προϊόντα, και οι χωρίς κόστος εκδόσεις τους είναι περιορισμένων δυνατοτήτων, αποφασίσαμε τελικά να χρησιμοποιήσουμε την PostgreSQL.

Προχωρήσαμε λοιπόν στην εγκατάσταση, την βελτιστοποίηση και τελικά τη χρήση της PostgreSQL.

3.2 Εγκατάσταση PostgreSQL και εργαλείων

Κατεβάσαμε την PostgreSQL από την ηλεκτρονική διεύθυνση Postgres Plus Downloads[16]. Διαλέξαμε την έκδοση για λειτουργικό windows. Στη διάρκεια της εγκατάστασης ορίζουμε το path εγκατάστασης, τον κωδικό του χρήστη postgres, τη θύρα στην οποία θα αποκρίνεται (5432) και επιλέγουμε τις τοπικές ρυθμίσεις που θέλουμε (Εμείς επιλέξαμε Greek, Greece). Όταν τελειώσει η εγκατάσταση ξεκινά ο wizard Stack Builder που μας επιτρέπει να εγκαταστήσουμε πρόσθετο λογισμικό. Τον χρησιμοποιήσαμε για να εγκαταστήσουμε τα PostGIS extensions και τους JDBC Drivers:

[16] http://www.enterprisedb.com/downloads/postgres-postgresql-downloads



Εικόνα 3-1 Ο wizard Stack Builder της PostgreSQL

Στο παράρτημα μπορεί κανείς να βρει τη διαδικασία εγκατάστασης των PostgreSQL και PostGIS σε λειτουργικό ubuntu server 12.04.

3.3 Βελτιστοποίηση PostgreSQL

Ο καλύτερος τρόπος για να επεξεργαστούμε τα αρχεία ρυθμίσεων της PostgreSQL είναι χρησιμοποιώντας το οπτικό εργαλείο pgAdminIII. Επιλέγουμε File > Open postgresql.conf και ρυθμίζουμε τις πιο κάτω παραμέτρους σύμφωνα με τις οδηγίες της ιστοσελίδας Tuning Your PostgreSQL Server[17].

max_connections: Αυτή η επιλογή ορίζει τον μέγιστο αριθμό των ανοιχτών συνδέσεων (connections) με τη βάση δεδομένων. Αν ο αριθμός αυτός είναι μεγάλος τότε όλες οι ανοιχτές συνδέσεις καθυστερούν. Πρέπει να αποφασίσουμε μετά από ποιον αριθμό συνδέσεων θα αρνηθούμε την σύνδεση, προκειμένου να εξασφαλίσουμε καλή απόδοση στις ήδη ανοιχτές συνδέσεις. Η προκαθορισμένη τιμή είναι 100. Εμείς θέσαμε την τιμή 20. Χρησιμοποιώντας την

[17] http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

τεχνική connection pooling με χρήση κατάλληλου λογισμικού μπορούμε να ικανοποιούμε πολύ περισσότερες ταυτόχρονες συνδέσεις.

shared_buffers: Η παράμετρος καθορίζει πόση μνήμη είναι αφιερωμένη στην PostgreSQL για την προσωρινή αποθήκευση δεδομένων. Εμπειρικά, η τιμή αυτή πρέπει να τεθεί περίπου στο 25% της μνήμης του συστήματος. Σημειώνουμε ότι στα Windows μεγάλες τιμές της παραμέτρου δεν είναι τόσο αποτελεσματικές. Έχουμε τα καλύτερα αποτελέσματα διατηρώντας τη σε σχετικά χαμηλή τιμή και χρησιμοποιώντας τη μνήμη cache του λειτουργικού. Στα Windows το χρήσιμο εύρος είναι 64MB σε 512MB. Εμείς τη ρυθμίσαμε σε 512MB.

effective_cache_size: Η παράμετρος καθορίζει πόσος κατ' εκτίμηση χώρος για προσωρινή αποθήκευση δεδομένων στο δίσκο είναι αφιερωμένος στην PostgreSQL. Μια εμπειρική τιμή είναι περίπου όσο το 50% της συνολικής μνήμης του συστήματος.

work_mem: Η παράμετρος αυτή χρησιμοποιείται για να ελέγχει την ποσότητα της μνήμης που χρησιμοποιείται στις εργασίες ταξινόμησης και τους πίνακες κατακερματισμού Αυτό το μέγεθος είναι το μέγεθος που χρησιμοποιείται για κάθε χρήστη και όχι το συνολικό. Αν π.χ. το ρυθμίσουμε σε 50MB, και έχουμε 30 ταυτόχρονους χρήστες θα χρειαστούμε 1.5GB πραγματικής μνήμης. Άρα λοιπόν, η τιμή της εξαρτάται και από την παράμετρο **max_connections.** Εμείς την ρυθμίσαμε σε 50MB.

Αν θέλουμε να έχουμε πρόσβαση στο RDBMS και από άλλους υπολογιστές του δικτύου αλλάζουμε την τιμή της μεταβλητής **listen_addresses** από localhost σε '*'. και προσθέτουμε στο τέλος του αρχείου pg_hba.conf την γραμμή:

```
TYPE      DATABASE        USER          ADDRESS              METHOD
host      all             all           192.168.1.1/24       trust
```

3.4 Επικοινωνία με το RDBMS

Για την επικοινωνία του java project με την βάση χρειαζόμαστε και τους ακόλουθους drivers. Αρχικά, κατεβάζουμε τον JDBC driver της PostGIS από τον ιστοχώρο της PostGIS[18]. Επιλέγουμε την τελευταία έκδοση η οποία τη στιγμή που γράφουμε είναι η postgis-2.0.0alpha4SVN.jar. Μαζί

[18] http://postgis.refractions.net/download/

με την εγκατάσταση της PostgreSQL και με την εκτέλεση του Stack Builder έχουμε εγκαταστήσει τον postgresql-9.0-801.jdbc4.jar driver τον οποίο επίσης προσθέτουμε στο project μας.

Για την επικοινωνία επίσης δοκιμάσαμε δύο βιβλιοθήκες:

- Spring JDBC και
- MyBatis

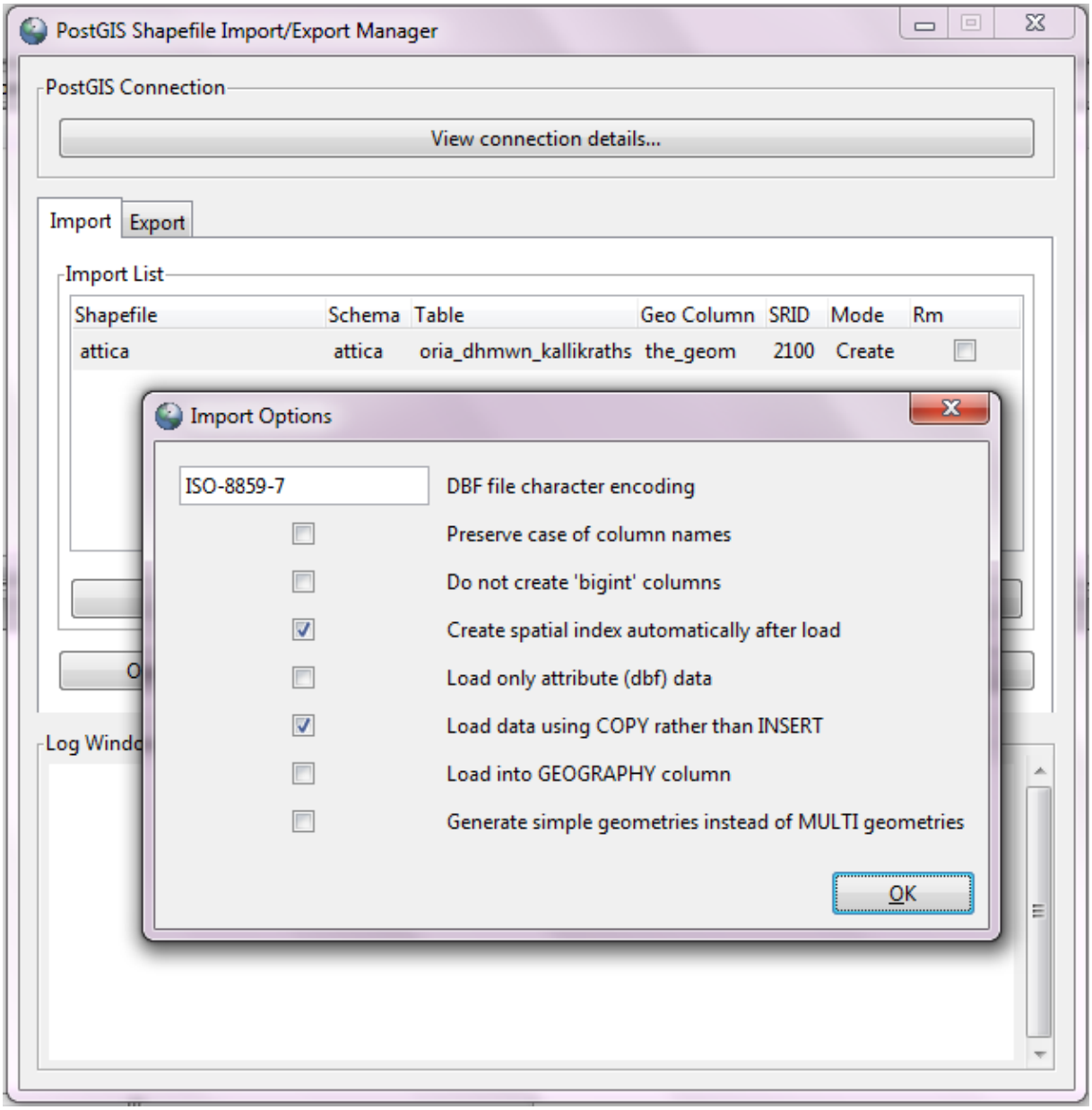
Οι κυριότεροι λόγοι για τους οποίους επελέγησαν οι συγκεκριμένες βιβλιοθήκες είναι ο έλεγχος που επιτρέπουν στον κώδικα SQL που εκτελείται και ότι μας επιστρέφουν αντικείμενα Java βοηθώντας τον αντικειμενοστραφή σχεδιασμό της εφαρμογής.

Η πρώτη εκ των δύο χρειάζεται για να λειτουργήσει πάρα πολλές βιβλιοθήκες (17 συνολικά) και έχει μια σύνθετη διαδικασία στη χρήση transactions. Έτσι, κάποια στιγμή στη διάρκεια της ανάπτυξης την αντικαταστήσαμε με τη βιβλιοθήκη MyBatis (πρώην iBatis). Πρέπει όμως να τονιστεί ότι και με Spring JDBC αν κάποιος προσέξει τις αναφορές (references) και το classpath το πρόγραμμα εκτελείται κανονικά. Επίσης, τονίζεται ότι, το Spring μπορεί να χρησιμοποιηθεί γενικότερα με Java Transaction API (JTA)[19] κάτι που όμως στην παρούσα εφαρμογή δεν είναι απαραίτητο.

3.5 Εισαγωγή των δεδομένα των δήμων.

Χρησιμοποιώντας το εργαλείο pgAdminIII και το εργαλείο εισαγωγής ShapeFile Import/Export Manager εισάγουμε τα δεδομένα που κατεβάσαμε από τον ιστότοπο geodata.gov.gr στην βάση (Εικόνα 3-2).

[19] http://en.wikipedia.org/wiki/Java_Transaction_API



Εικόνα 3-2 Εισαγωγή δεδομένων των δήμων της Αττικής

Στη συνέχεια, από τον πίνακα oria_dhmwn_kallikraths αφού εντοπίσουμε τους κωδικούς των δήμων μεταφέρουμε τα δεδομένα των δήμων της Αττικής σε άλλο πίνακα, μετασχηματίζοντας παράλληλα τα γεωχωρικά δεδομένα από το σύστημα αναφοράς ΕΓΣΑ87 στο σύστημα EPSG:4326 που χρησιμοποιεί η Google στους χάρτες της.

```
CREATE TABLE attica.kallikratis
(
  gid integer NOT NULL,
  name character varying(59),
  kwd_ypes character varying(4)
);
SELECT AddGeometryColumn('attica', 'kallikratis', 'geom', 4326, 'MULTIPOLYGON',
2);
```

```
ALTER TABLE attica.kallikratis OWNER TO postgres;

INSERT INTO attica.kallikratis(gid, name, kwd_ypes, geom)
SELECT gid, name, kwd_ypes, ST_Transform( ST_SetSRID( the_geom, 2100), 4326)
FROM attica.oria_dhmwn_kallikraths
WHERE gid in
(8,12,13,15,18,24,25,27,35,54,56,57,63,68,69,72,76,84,94,95,109,111,114,135,140,15
1,153,161,162,175,181,182,188,189,194,196,199,208,209,211,236,238,240,242,246,247,
248,249,250,265,273,289,307,309,313,316,317,324);
```

Κεφάλαιο 4
Σχεδίαση

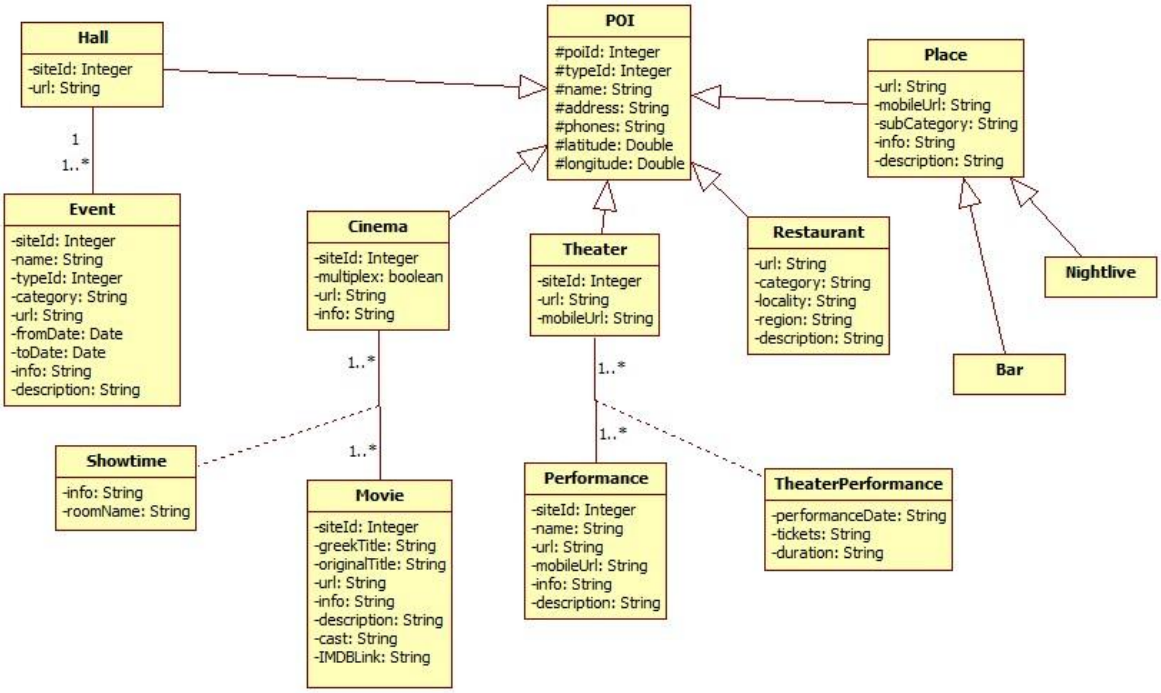
Στο κεφάλαιο αυτό θα αναφερθούμε στις βασικές οντότητες της εφαρμογής μας, τις σχέσεις μεταξύ τους και την σχεδίαση της βάσης δεδομένων.

4.1 Διάγραμμα κλάσεων

Το διάγραμμα κλάσεων των οντοτήτων:

- «Θέατρο», «Παράσταση», «Ώρες Παραστάσεων»,
- «Κινηματογράφος», «Ταινία», «Προβολή»,
- «Εκδήλωση», «Χώρος Εκδήλωσης»,
- «Εστιατόριο», «Κέντρο διασκέδασης» και «Μπαρ ή Κλαμπ»

φαίνεται παρακάτω (Σχήμα 4-1):



Σχήμα 4-1 Διάγραμμα κλάσεων

Η κλάση «Σημείο ενδιαφέροντος» POI(Point of Interest) είναι η βασική κλάση του συστήματος. Από αυτήν κληρονομούν οι κλάσεις «Αίθουσα» Hall, «Κινηματογράφος» Cinema,«Θέατρο» Theater,«Εστιατόριο» Restaurant και η κλάση «Place» που αποτελεί γενίκευση των κλάσεων «Μπαρ ή Κλαμπ» Bar και «Κέντρο διασκέδασης» Nightlive.

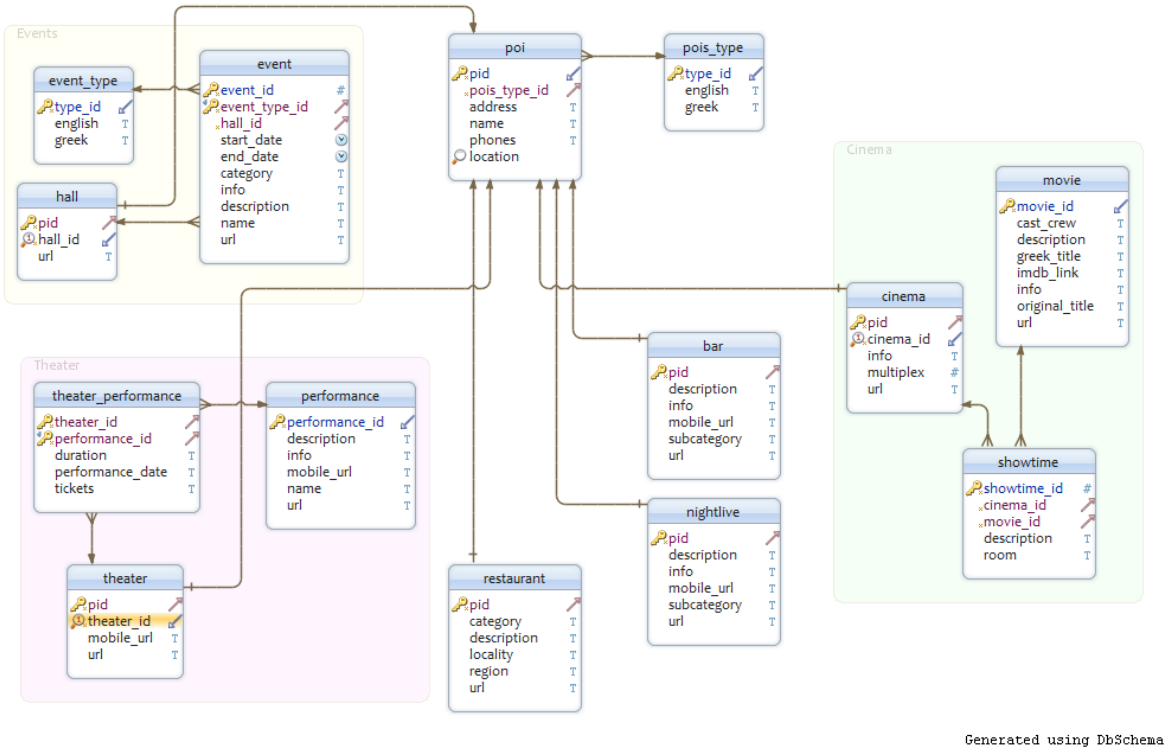
Η κλάση Cinema σχετίζεται με την κλάση «Ταινία» Movie μέσω της κλάσης Showtime με πληθικότητα πολλά προς πολλά. Δηλαδή, μια ταινία παίζεται σε πολλούς κινηματογράφους και ένας κινηματογράφος μπορεί να παίζει πολλές ταινίες.

Η κλάση Theater σχετίζεται με την κλάση «Παράσταση» Performance μέσω της κλάσης TheaterPerformance με πληθικότητα πολλά προς πολλά. Αυτό σημαίνει πως σε ένα θέατρο παίζονται πολλές παραστάσεις και μία παράσταση μπορεί να παίζεται σε πολλά θέατρα. Αυτό συμβαίνει κυρίως τους καλοκαιρινούς μήνες όπου διάφοροι θίασοι περιοδεύουν σε πολλά θέατρα.

Η κλάση Hall σχετίζεται με την κλάση «Εκδήλωση» Event με πληθικότητα ένα προς πολλά. Αυτό σημαίνει πως σε ένα χώρο μπορούν να λαμβάνουν χώρα πολλές εκδηλώσεις.

Οι κλάσεις «Μπαρ ή Κλαμπ» Bar και «Κέντρο διασκέδασης» Nightlive έχουν τα ίδια πεδία και γι' αυτό, τις γενικεύσαμε στην κλάση Place.

4.2 Entity–relationship διάγραμμα της βάσης.



Σχήμα 4-2 Entity–relationship διάγραμμα της βάσης

Δύο σημεία αξίζουν αναφοράς:

Η απόφαση να φτιάξουμε δύο πίνακες «bar» και «nightlive» αντί έναν «place». Θα μπορούσαμε να ακολουθήσουμε την δεύτερη προσέγγιση και να προσθέσουμε ένα επιπλέον πεδίο που να χαρακτηρίζει τον τύπο του σημείου. Στη συνέχεια θα χρησιμοποιούσαμε δυο views για να δούμε τα «Μπαρ ή Κλαμπ» και τα «Κέντρο διασκέδασης». Επιλέξαμε την πρώτη προσέγγιση.

Η δεύτερη απόφαση ήταν να χρησιμοποιήσουμε σαν κλειδί στον πίνακα showtime ένα αυτόματα αυξανόμενο ακέραιο αντί για τον συνδυασμό των πεδίων cinema_id, movie_id και room. Η επιλογή αυτή μας επιτρέπει να χρησιμοποιούμε null τιμές στο πεδίο room (Κινηματογραφική αίθουσα). Στην PostgreSQL για να υλοποιήσουμε το αυτόματο κλειδί χρησιμοποιούμε ένα sequence.

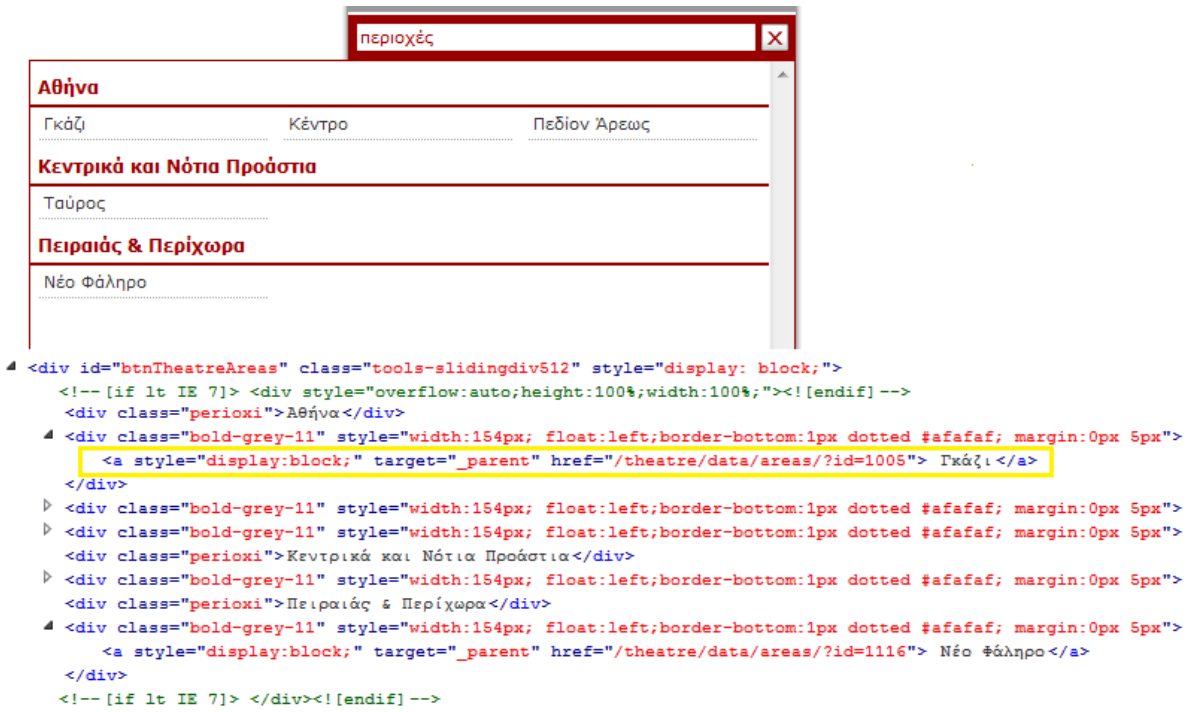
Επίσης, ο πίνακας poi περιέχει το πεδίο pois_type_id που μας επιτρέπει να γνωρίζουμε τον τύπο του σημείου ενδιαφέροντος. Για τις τιμές του πεδίου χρησιμοποιήσαμε το lookup table pois_type.

Κεφάλαιο 5
Τεχνικές Συλλογής Δεδομένων

Στο κεφάλαιο αυτό θα αναφέρουμε κάποιες τεχνικές συλλογής δεδομένων. Παρακάτω θα περιγράψουμε πως συλλέξαμε τα δεδομένα δείχνοντας μόνο κάποιες χαρακτηριστικές περιπτώσεις. Οι περιπτώσεις αυτές είναι οι ιδανικές και αυτές που αξίζει να προβάλλει κανείς, γιατί υπάρχουν κι άλλες που λόγω τις πολυπλοκότητας τους συσκοτίζουν αντί να αναδεικνύουν τις τεχνικές που χρησιμοποιήσαμε. Για το σκοπό αυτό θα χρησιμοποιήσουμε τα θέατρα, που συλλέξαμε από τον ιστοχώρο του Αθηνοράματος.

5.1 Πρώτο βήμα

Ξεκινάμε από τις περιοχές (βλ. Εικόνα 5-1)



Εικόνα 5-1 Περιοχές

Ακολουθώντας την διαδρομή «div#btnTheaterAreas a» δηλ. τα tags a που είναι παιδιά του div με id = btnTheaterAreas συγκεντρώνουμε τα URL των περιοχών της Αθήνας όπου λειτουργούν θέατρα.

5.2 Δεύτερο βήμα

Το επόμενο βήμα είναι να διαβάσουμε πληροφορίες της παράστασης. Πιο κάτω φαίνεται μία παράσταση από την λίστα (Εικόνα 5-2):

Εικόνα 5-2 Πληροφορίες θεάτρου και παράστασης

Διαβάζοντας την παραπάνω λίστα παραστάσεων έχουμε πληροφορίες για το όνομα και το URL του θεάτρου καθώς και το όνομα και το URL της παράστασης. Από τα URL μπορούμε να βρούμε τα id που χρησιμοποιεί ο ιστοχώρος για να προσδιορίσει τα αντικείμενα. Επομένως έχουμε τα βασικά στοιχεία των τριών πινάκων της βάσης που αφορούν τα θέατρα. Μπορούμε επίσης χρησιμοποιώντας το id του ιστοχώρου να μεταφερθούμε στην αντίστοιχη ιστοσελίδα του ιστοχώρου athinorama mobile από όπου μπορούμε ευκολότερα να εντοπίσουμε τα γεωχωρικά δεδομένα.

5.3 Τρίτο βήμα. Threads

Έχοντας λοιπόν διαβάσει όλες τις λίστες των περιοχών και έχοντας αποθηκεύσει τα δεδομένα έχουμε όλα τα θέατρα και όλες τις θεατρικές παραστάσεις με ελλιπή όμως στοιχεία. Πρέπει να προσπελάσουμε κάθε ένα URL που αντιστοιχεί σε κάποιο θέατρο και κάθε URL που αντιστοιχεί σε μια παράσταση για να συγκεντρώσουμε τα στοιχεία που λείπουν.

Για να επιταχύνουμε την διαδικασία χρησιμοποιούμε threads διπλάσια στον αριθμό από τους πυρήνες του επεξεργαστή. Παραδείγματος χάριν, αν ο επεξεργαστής του υπολογιστή μας είναι διπύρηνος τότε ξεκινάμε τέσσερα ταυτόχρονα threads που διαβάζουν τις υπόλοιπες πληροφορίες για τα θέατρα. Όταν τελειώσουμε με τα θέατρα κάνουμε το ίδιο για τις παραστάσεις.

Η χρήση των threads σύμφωνα με τον νόμο του Amdrahl (Άμνταλ) επηρεάζει την ταχύτητα του προγράμματος σύμφωνα με τον τύπο $\frac{1}{(1-P)+\frac{P}{N}}$. όπου P δηλώνει το ποσοστό των υπολογισμών του προγράμματος οι οποίοι μπορούν να εκτελεστούν παράλληλα και N είναι το πλήθος των επεξεργαστών. Σύμφωνα με κάποιους εμπειρικούς υπολογισμούς το P στο συγκεκριμένο πρόβλημα δεν είναι πάνω από 10% ενώ το υπόλοιπο καταναλώνεται στη λήψη των δεδομένων. Το ποσοστό αυτό εξαρτάται μεταξύ άλλων από την δική μας ταχύτητα σύνδεσης στο διαδίκτυο, την διαθεσιμότητα του διακομιστή και το μέγεθος της ιστοσελίδας που κατεβάζουμε. Η επιτάχυνση του προγράμματος είναι της τάξης του 8%.

Κεφάλαιο 6
Restful Web Services

Στο κεφάλαιο αυτό θα αναφερθούμε συνοπτικά στα βασικά των web services, στην προδιαγραφή jax-rs, γιατί επιλέξαμε την java βιβλιοθήκη jersey για να τα υλοποιήσουμε, την βασική εγκατάσταση, καθώς και στην σχεδίαση των υπηρεσιών ιστού.

6.1 Υπηρεσίες διαδικτύου (Web Services)

Μια υπηρεσία διαδικτύου (Web service) είναι μια μέθοδος επικοινωνίας μεταξύ δύο ηλεκτρονικών συσκευών ή εφαρμογών μέσω του διαδικτύου ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού.

Η Κοινοπραξία W3C (World Wide Web Consortium) ορίζει ένα «Web service» ως ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίξει διαλειτουργική[20] αλληλεπίδραση μηχανής-προς-μηχανή μέσω ενός δικτύου.

[20] Διαλειτουργικότητα - interoperability- είναι η ικανότητα διαφορετικών συστημάτων και οργανισμών να εργαστούν μαζί.

Υπάρχουν δύο είδη υπηρεσιών διαδικτύου. Το SOAP (Simple Object Access Protocol) και τα RESTful web services.

6.2 SOAP (Simple Object Access Protocol)

Το SOAP στηρίζεται στην XML (eXtensible Markup Language) για την μορφοποίηση των δεδομένων και χρησιμοποιεί τα πρωτόκολλα Hypertext Transfer Protocol (HTTP) και Simple Mail Transfer Protocol (SMTP) για τη μετάδοση μηνυμάτων.

Η WSDL είναι μια γλώσσα βασισμένη σε XML που χρησιμοποιείται για την περιγραφή της λειτουργικότητας που προσφέρεται από ένα SOAP web service. Μια περιγραφή WSDL μιας υπηρεσίας διαδικτύου (που αναφέρεται επίσης ως ένα αρχείο WSDL) παρέχει μια αναγνώσιμη από μηχάνημα περιγραφή, την ονομασία της υπηρεσίας, τι παραμέτρους αναμένει, και τι δομές δεδομένων επιστρέφει. Εξυπηρετεί επομένως. ένα παρόμοιο σκοπό με την υπογραφή μιας μεθόδου σε μια γλώσσα προγραμματισμού.

Απαιτεί ισχυρούς ορισμούς των τύπων δεδομένων, πράγμα το οποίο επιτυγχάνεται με την ύπαρξη ενός DTD (Document Type Definition)[21] ή ενός σχήματος XSD (XML Schema Definition)[22] που να ορίζουν την δομή των XML αρχείων (documents).

Σημαντικό βήμα στην εξάπλωση των SOAP web services αποτέλεσε η ανάπτυξη από τον οργανισμό OASIS (Organization for the Advancement of Structured Information Standards) ενός μητρώου (registry) με τίτλο UDDI (Universal Description, Discovery and Integration)[23]. Έτσι, μπορούσαν οι δυνητικοί χρήστες να βρίσκουν πληροφορίες ικανές ώστε να εντοπίζουν και να εκτελούν web services της αρεσκείας τους. Μπορούσαν επίσης οι διάφοροι οργανισμοί να γνωστοποιούν τις υπηρεσίες που διέθεταν δημόσια, δωρεάν ή με αμοιβή, αλλά και αυτές που εκτίθενται μόνο εσωτερικά του οργανισμού.

Υποστηρίχτηκε κυρίως από τις εταιρείες Microsoft και IBM.

[21] http://www.w3.org/XML/1998/06/xmlspec-report-v21.htm
[22] http://www.w3.org/XML/Schema
[23] . http://uddi.xml.org/uddi-org

6.3 REST (Representational State Transfer)

Τα RESTful web services έχουν κερδίσει την ευρεία αποδοχή στον παγκόσμιο ιστό ως μία απλούστερη εναλλακτική λύση στο SOAP και τα βασισμένα στην γλώσσα WSDL web services. Κύριοι λόγοι είναι η ευκολότερη ενσωμάτωση του σε υπηρεσίες web 2.0 / AJAX[24], λόγω της ευρείας χρήσης της JSON (JavaScript Object Notation) και η υιοθέτηση του από ισχυρούς παρόχους υπηρεσιών όπως Google, Yahoo και Facebook. Η πιο σύγχρονη αρχιτεκτονική REST έχει αναδειχθεί τα τελευταία χρόνια ως κυρίαρχο μοντέλο σχεδιασμού υπηρεσιών διαδικτύου. Ο όρος «representational state transfer» εισήχθη και ορίστηκε το 2000 από τον Roy Fielding στην διδακτορική του διατριβή. Ο Fielding είναι ένας από τους κύριους συντάκτες των προδιαγραφών του πρωτοκόλλου HTTP για τις εκδόσεις 1.0 και 1.1. Η αρχιτεκτονική αυτή αναφέρεται συνήθως σαν RESTful.

Η RESTful αρχιτεκτονική/ φιλοσοφία περιλαμβάνει πελάτες (clients) και διακομιστές (servers). Οι πελάτες στέλνουν αιτήματα προς στους διακομιστές. Οι διακομιστές επεξεργάζονται τα αιτήματα και επιστρέφουν τις κατάλληλες απαντήσεις. Οι αιτήσεις και οι απαντήσεις αφορούν την μεταφορά κάποιων πόρων (resources). Πόρος θεωρείται οτιδήποτε μπορεί να έχει μια διεύθυνση URI (Universal Resource Identifier ελλ. Ενιαίο Αναγνωριστικό Πόρων) στον παγκόσμιο ιστό. Μια αναπαράσταση ενός πόρου είναι συνήθως ένα έγγραφο που καταγράφει την τρέχουσα ή προβλεπόμενη κατάσταση του. και αντιστοιχεί σε ένα URL (Uniform Resource Locator ελλ. Ενιαίος Εντοπιστής Πόρων). Η RESTful αρχιτεκτονική δεν απαιτεί ισχυρούς ορισμούς των τύπων δεδομένων ούτε απαιτεί την ύπαρξη ενός DTD ή ενός σχήματος XSD που να ορίζουν την δομή του XML αρχείου (document) όπως απαιτεί το SOAP, πράγμα που χρησιμοποιεί λιγότερο εύρος ζώνης (bandwidth). Ο έλεγχος λαθών από την αρχιτεκτονική REST είναι επίσης

[24] «Το AJAX που προέρχεται από τα αρχικά των λέξεων Asynchronous JavaScript and XML, είναι ένα σύνολο αλληλένδετων τεχνικών ανάπτυξης ιστοσελίδων που χρησιμοποιούνται στην πλευρά του πελάτη για να δημιουργηθούν ασύγχρονες εφαρμογές διαδικτύου. Με χρήση AJAX, διαδικτυακές εφαρμογές μπορούν να στέλνουν και να ανακτούν δεδομένα από ένα διακομιστή ασύγχρονα (στο παρασκήνιο), χωρίς να επηρεάζει την συνολική εμφάνιση και συμπεριφορά της υπάρχουσας σελίδας. Τα δεδομένα μπορούν να ανακτηθούν χρησιμοποιώντας το αντικείμενο XMLHttpRequest. Παρά το όνομα, η χρήση της XML δεν απαιτείται (συχνά χρησιμοποιείται JSON), και οι αιτήσεις δεν χρειάζεται να είναι ασύγχρονες» (βλ. http://en.wikipedia.org/wiki/Ajax_%28programming%29).

διαφορετικός από το SOAP. Το SOAP μπορεί να χρησιμοποιεί «ορισμένα από το χρήστη» μηνύματα λάθους ενώ το REST χρησιμοποιεί τα ορισμένα από τις προδιαγραφές του πρωτοκόλλου HTTP λάθη.

Η εφαρμογή έχει αναπτυχθεί με την χρήση των RESTful web services και την γλώσσα Java. Η κοινότητα της Java έχει αναπτύξει τις δικές της προδιαγραφές (API, Application Programming Interface ελλ. Διεπαφή Προγραμματισμού Εφαρμογών)[25] που υποστηρίζουν την αρχιτεκτονική-φιλοσοφία REST Η προδιαγραφή αυτή ονομάζεται JAX-RS (Java API for RESTful Web Services), χρησιμοποιεί σημάνσεις (annotations) και αποτελεί από την έκδοση 1.1 επίσημο μέρος της Java EE 6.

6.4 Java API for RESTful Web Services (JAX-RS)

Στην ενότητα αυτή θα αναφερθούμε στις κυριότερες σημάνσεις (annotations), της προδιαγραφής JAX-RS, στις κυριότερες υλοποιήσεις της προδιαγραφής καθώς και στους κυριότερους τύπους δεδομένων (Internet media types) που επιστρέφονται.

6.4.1 Σημάνσεις (Annotations)

Οι σημαντικότερες εξ αυτών είναι[26]:

- **Consumes**

Ορίζει τους μορφές δεδομένων που μπορεί ένας τύπος ή μία μέθοδος να δεχθεί σαν είσοδο (να καταναλώσει). Μερικές από τις μορφές δεδομένων που αναφέρονται στις προδιαγραφές είναι: "application/atom+xml", "application/json", "application/octet-stream", "application/svg+xml", "application/x-www-form-urlencoded", "application/xhtml+xml", "application/xml", "multipart/form-data", "text/html", "text/plain", "text/xml".

- **Produces**

Ορίζει τις μορφές δεδομένων που παράγει ένας τύπος ή μία μέθοδος στην έξοδο. Θα αναφερθούμε αναλυτικότερα στους μορφές δεδομένων XML και JSON παρακάτω.

[25] Η ιστοσελίδα http://el.wikipedia.org/wiki/API αναφέρει:"Καλούμε Διεπαφή Προγραμματισμού Εφαρμογών, γνωστή και ως Διασύνδεση Προγραμματισμού Εφαρμογών (για συντομία διεπαφή ή διασύνδεση), τη διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή / και ανταλλαγή δεδομένων"

[26] βλ. http://jsr311.java.net/nonav/releases/1.1/spec/spec3.html

- **GET, POST, PUT, DELETE**

Ορίζει ότι η σημασμένη μέθοδος χειρίζεται αιτήσεις αντίστοιχου τύπου HTTP. Δηλαδή, HTTP GET, HTTP POST, HTTP PUT και HTTP DELETE αντίστοιχα.

- **HEAD**

Ορίζει ότι η σημασμένη μέθοδος χειρίζεται αιτήσεις τύπου HTTP HEAD.

- **Path**

Ορίζει τη σχετική διαδρομή για έναν πόρο. Αν χρησιμοποιείται σε μία κλάση ορίζει την ρίζα της διαδρομής. Όταν χρησιμοποιείται σε μία μέθοδο κλάσης, ορίζει είτε ολόκληρη την διαδρομή, εφόσον δεν υπάρχει σήμανση στην κλάση, είτε την διαδρομή μετά την ρίζα.

- **PathParam**

Ορίζει ότι η τιμή μιας παραμέτρου μιας μεθόδου θα εξαχθεί από τη διαδρομή URI της αίτησης. Η παράμετρος αυτού του τύπου αποτελεί τμήμα της διαδρομής URI του πόρου και ορίζεται στη σήμανση Path. Π.χ.

```
@Path("/cinema")
public class CinemaResource {
    @GET
    @Path("/{id}")
    @Produces({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML })
    public Response getCinema(@PathParam("id") Integer id) {
 ...
    }
}
```

Αν η αίτηση έχει URI /cinema/1 στην μέθοδο getCinema θα περάσει ως παράμετρος ο ακέραιος 1.

- **QueryParam**

Η QueryParam είναι μια παράμετρος που θα εξαχθεί από το query string του URI της αίτησης. Το τμήμα αυτό χωρίζεται από το υπόλοιπο URI με ένα αγγλικό ερωτηματικό (?) και αποτελείται από μια ακολουθία ζευγών της μορφής key=value που χωρίζονται με ampersand (&). Παράδειγμα:

```
@Path("/pois")
public class POIsResource {
    @GET
    @Produces({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML })
    public Response getPOIsByName(
        @QueryParam("name") String name,
        @QueryParam("types") String types)
    {...}
}
```

Αν η αίτηση έχει URI /pois/?name=%βαρκ%&types=1,2,3 τότε στην μέθοδο getPOIsByName θα περάσουν οι παράμετροι "%βαρκ%" και "1,2,3" αντίστοιχα.

- **FormParam**

Η FormParam είναι άλλη μια παράμετρος που χρησιμοποιείται μόνο σε μεθόδους και που θα εξαχθεί από μία form parameter αίτησης τύπου POST. Παράδειγμα:

```
@Path("/auth")
public class AuthenticationResource {
    @Path("login")
    @POST
    @Produces({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML })
    public Response getUser(
        @FormParam("email") String email,
        @FormParam("password") String password)
    {...}
}
```

- **MatrixParam**

Η MatrixParam είναι άλλη μια παράμετρος που θα εξαχθεί από το URI της αίτησης. Χρησιμοποιείται σπάνια και δεν υπάρχουν ικανοποιητικά παραδείγματα χρήσης της στον παγκόσμιο ιστό. Ένα σημείο για να αρχίσει κάποιος είναι οι διευθύνσεις http://www.w3.org/DesignIssues/MatrixURIs.html και http://brettdargan.com/blog/2009/01/16/query-vs-matrix-params/.

- **CookieParam**

Η CookieParam είναι άλλη μια παράμετρος που θα εξαχθεί από ένα HTTP cookie. Στην σήμανση Path ορίζουμε το όνομα του cookie.

- **HeaderParam**

Η συγκεκριμένη παράμετρος περιέχεται σε μια HTTP κεφαλίδα. Και εδώ στη σήμανση Path ορίζουμε το όνομα της κεφαλίδας.

- **Encoded**

Η Encoded εμποδίζει την αυτόματη αποκρυπτογράφηση των παραμέτρων.

- **DefaultValue**

Η DefaultValue ορίζει μία εξ ορισμού τιμή για ένα πεδίο, ιδιότητα ή μέθοδο με σήμανση @QueryParam, @MatrixParam, @CookieParam, @FormParam ή @HeaderParam. Η εξ ορισμού τιμή θα χρησιμοποιηθεί αν δεν υπάρχει η αντίστοιχη παράμετρος στην αίτηση HTTP.

- **Context**

Η Context χρησιμοποιείται για να αποκτήσουμε πρόσβαση μέσω αυτής σε αντικείμενα όπως Request, Response, UriInfo, ServletContext και άλλα.

6.4.2 Υλοποιήσεις

Οι κύριες υλοποιήσεις της προδιαγραφής JAX-RS είναι:

- **Apache CXF**, ένα ανοιχτού κώδικα πλαίσιο (framework) υπηρεσιών διαδικτύου.

- **Jersey**, η εφαρμογή αναφοράς από την Sun (τώρα Oracle). Σύμφωνα με το διαδικτυακό μάθημα της Java EE 6 (http://docs.oracle.com/javaee/6/tutorial/doc/giqsx.html) το έργο Jersey «είναι μια έτοιμη για παραγωγή εφαρμογή αναφοράς για τις προδιαγραφές JAX-RS. Το έργο Jersey υλοποιεί πλήρη υποστήριξη για τις σημάνσεις (annotations) που ορίζονται στις προδιαγραφές JAX-RS, καθιστώντας εύκολο για τους προγραμματιστές να αναπτύξουν RESTful web services με την Java και την Java Virtual Machine (JVM) ελλ. Εικονική Μηχανή Java».

- **RESTeasy**, η υλοποίηση της JBoss.

- **Restlet**, που δημιουργήθηκε από τον πρωτοπόρο σε πλαίσια (frameworks) REST Jerome Louvel.

- **Apache Wink**. Ένα έργο του «Apache Software Foundation Incubator», που το τμήμα που εκτελείται στο διακομιστή, υλοποιεί την προδιαγραφή JAX-RS.

Στην εφαρμογή μας χρησιμοποιούμε την υλοποίηση της Sun. Πρέπει όμως να επισημανθεί ότι λόγω των κοινών προδιαγραφών που ακολουθούν οι διάφορες υλοποιήσεις, η μεταφορά της εφαρμογής σε άλλη πλατφόρμα είναι εύκολη υπόθεση.

6.4.3 Internet Media Types

Μερικές από τις μορφές δεδομένων που υποστηρίζουν οι υλοποιήσεις της προδιαγραφής JAX-RS είναι:

- **Atom** (Atom Syndication Format),

- **JSON** (JavaScript Object Notation),

- **RSS** (αρχικά RDF Site Summary, συχνά αποκαλείται Really Simple Syndication) 27,
- **APP** (Atom Publishing Protocol),
- **CSV** (Comma-separated values),
- **XML** (eXtensible Markup Language)
- **XHTML** (eXtensible HyperText Markup Language)

Διαδεδομένες είναι οι μορφές XHTML και RSS / Atom καθώς υπάρχουν πολλές εφαρμογές που τις αναγνωρίζουν και τις προβάλλουν αυτόματα.

Η εφαρμογή μας χρησιμοποιεί τις μορφές XML και JSON για τη μετάδοση των δεδομένων. Σχετικά με την μορφοποίηση των δεδομένων υπάρχει πιο κάτω ειδική αναφορά.

6.4.4 XML

Για να μετατρέψουμε ένα αντικείμενο σε μορφή XML αρκεί να συμπεριλάβουμε την βιβλιοθήκη jersey-server.jar στο classpath του προγράμματος μας. Αυτό μας δίνει τη δυνατότητα να χρησιμοποιήσουμε την αρχιτεκτονική JAXB (Java Architecture for XML Binding). Το πρότυπο JAXB μας επιτρέπει να αναπαραστήσουμε αντικείμενα της Java σε μορφή XML και αντίστροφα. Η αρχιτεκτονική μας επιτρέπει να αποθηκεύσουμε στην μνήμη σε μορφή XML δεδομένα, χωρίς να χρειαστεί να παρέμβουμε για τις μετατροπές από και σε XML. Για να γίνει αυτό χρησιμοποιούμε ένα τυποποιημένο σύνολο σημάνσεων (annotations). Η μόνη πραγματικά αναγκαία σήμανση είναι η @XmlRootElement που χρησιμοποιείται στη ρίζα μίας κλάσης. Ένα παράδειγμα είναι το ακόλουθο:

```
package model;

import java.util.List;
```

27 To Resource Description Framework (RDF) είναι μια οικογένεια προδιαγραφών του World Wide Web Consortium (W3C) που αρχικά σχεδιάστηκε για την προτυποποίηση μεταδεδομένων. Έχει δημιουργηθεί για να χρησιμοποιηθεί ως γενική μέθοδο για την εννοιολογική περιγραφή ή μοντελοποίηση των πληροφοριών που υλοποιούνται σε δικτυακούς πόρους, χρησιμοποιώντας μια ποικιλία από μορφές σύνταξης.

```
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "cinema")
@XmlType(propOrder = { "siteId", "poiId", "multiplex", "url", "name",
        "address", "phones", "info", "latitude", "longitude", "showtimes" })
public class Cinema {
    @XmlElement
    private Integer siteId;
    private Integer poiId;
    private boolean multiplex;
    private String url;
    private String name;
    private String address;
    private String phones;
    private String info;
    private String latitude;
    private String longitude;
    // XmLElementWrapper generates a wrapper element around XML representation
    @XmlElementWrapper(name = "showtimes")
    // XmlElement sets the name of the entities
    @XmlElement(name = "showtime")
    private List<Showtime> showtimes;

    .
    .
    .

}
```

Ένα αντικείμενο αυτής της κλάσης αναπαριστάται σε XML με την μορφή:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cinema>
    <siteId>32668</siteId>
    <poiId>2681</poiId>
    <multiplex>false</multiplex>
    <name>BAPKIZA 2</name>
    <address>Θάσου 22</address>
    <phones>2108973926</phones>
    <info>Χειμερινός, Parking</info>
    <latitude>37.822141000000002</latitude>
    <longitude>23.798092</longitude>
    <showtimes>
        <showtime> ... </showtime>
        <showtime> ...</showtime>
    </showtimes>
</cinema>
```

6.4.5 JSON

Το jersey χρησιμοποιεί δύο βιβλιοθήκες για την αναπαράσταση κλάσεων της Java σε JSON. Τις βιβλιοθήκες **Jackson** (http://jackson.codehaus.org/) και **Jettison** (http://jettison.codehaus.org/), όχι όμως και τις δύο ταυτόχρονα. Το Jettison αντιστοιχίζει XML σε και από JSON και είναι η παλαιότερη υλοποίηση εκ των δύο. Στη διάρκεια της ανάπτυξης αποφασίσαμε να χρησιμοποιήσουμε τη βιβλιοθήκη Jackson. Η βιβλιοθήκη Jackson φαίνεται να είναι ταχύτερη και σύμφωνα με την τεκμηρίωση της βιβλιοθήκης jersey (βλ. jersey.java.net/nonav/documentation/latest/json.html) είναι σχετικά απλούστερη. Συγκεκριμένα, «Η βιβλιοθήκη Jackson αποτελεί τον ευκολότερο τρόπο να μετατρέψετε αντικείμενα Java σε JSON και αντίστροφα. Για να χρησιμοποιήσετε αυτή την προσέγγιση, θα πρέπει να ενεργοποιήσετε το χαρακτηριστικό JSONConfiguration.FEATURE_POJO_MAPPING. Αυτό μπορεί να γίνει στο αρχείο web.xml αρχικοποιώντας την παράμετρο POJOMappingFeature»:

```
<init-param>
    <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>
    <param-value>true</param-value>
</init-param>
```

Συνιστούμε τη χρήση της Jackson, γιατί χωρίς αυτήν παρουσιάστηκαν exceptions όπως: «Expected BEGIN_ARRAY but was BEGIN_OBJECT» και «Expected BEGIN_OBJECT but was BEGIN_ARRAY». Αν κάποιος θέλει να χρησιμοποιήσει τη βιβλιοθήκη Jettison αρκεί να αφαιρέσει ή να σχολιάσει το προηγούμενο xml element.

Παρακάτω φαίνεται ένα αντικείμενο τύπου Cinema σε μορφή JSON:

```
{
  "siteId": 32668,
  "poiId": 2681,
  "multiplex": false,
  "url": null,
  "name": "ΒΑΡΚΙΖΑ 2",
  "address": "Θάσου 22",
  "phones": "2108973926",
  "info": "Χειμερινός, Parking",
  "latitude": "37.822141000000002",
  "longitude": "23.798092",
  "showtimes": [
    { ... },
    { ... }
  ]
}
```

6.5 Βασικές έννοιες HTTP

Η πλατφόρμα Ιστού έρχεται με ένα τυποποιημένο πρωτόκολλο επικοινωνίας - το HTTP - για την αλληλεπίδραση με τους πόρους και τις αναπαραστάσεις (representations) τους. Το πρωτόκολλο HTTP ορίζει ένα σύνολο τυποποιημένων *μεθόδων*, τους *κωδικούς κατάστασης* και τις *κεφαλίδες* για την αλληλεπίδραση με τους πόρους στο παγκόσμιο ιστό.

6.6 Μέθοδοι HTTP

Στον επόμενο πίνακα περιγράφονται οι πιο συχνά χρησιμοποιούμενες μεθόδοι HTTP, η σημασιολογία τους, και εάν η κάθε μέθοδος είναι ασφαλής και **Idempotence**[28].

Η μέθοδος GET μας δίνει τη δυνατότητα να ανακτήσουμε μια αναπαράσταση του πόρου, ενώ η PUT μας επιτρέπει να δημιουργήσουμε ή να ενημερώσουμε έναν πόρο και η DELETE μας δίνει τη δυνατότητα να τον διαγράψουμε. Εν ολίγοις, GET, PUT, DELETE παρέχουν τις βασικές πράξεις CRUD (create, retrieve, update, and delete - δημιουργία, ανάκτηση, ενημέρωση και διαγραφή) για τον παγκόσμιο ιστό. HEAD και OPTIONS, από την άλλη πλευρά, παρέχουν τη δυνατότητα ανάκτησης των *μεταδεδομένων* του πόρου, επιτρέποντάς μας να ανακαλύψουμε τον τρόπο με τον οποίο θα αλληλεπιδράσουμε με τον πόρο κατά το χρόνο εκτέλεσης.

Πίνακας 6-1 Μέθοδοι HTTP

Μέθοδος	**Περιγραφή**	**Ασφαλής**	**Idempotent**
GET	Ζητά μια συγκεκριμένη αναπαράσταση ενός πόρου.	ΝΑΙ	ΝΑΙ
PUT	Δημιουργεί ή ενημερώνει έναν πόρο με την παρεχόμενη αναπαράσταση.	ΟΧΙ	ΝΑΙ
DELETE	Διαγράφει τον καθορισμένο πόρο.	ΟΧΙ	ΝΑΙ

[28] Idempotence είναι η ιδιότητα, κάποιων πράξεων στα μαθηματικά και την επιστήμη των υπολογιστών, να μπορούν να εφαρμοστούν πολλές φορές πάνω σε ένα αντικείμενο χωρίς να αλλάξει το αποτέλεσμα της πράξης από την πρώτη φορά.

POST	Υποβάλει τα δεδομένα προς επεξεργασία από τον πόρο.	ΟΧΙ	ΟΧΙ
HEAD	Παρόμοια με GET αλλά ανακτά μόνο τις κεφαλίδες (headers) και όχι τα δεδομένα (body).	ΝΑΙ	ΝΑΙ
OPTIONS	Επιστρέφει τις μεθόδους που υποστηρίζονται από τον πόρο.	ΝΑΙ	ΝΑΙ

Ο Aaron Skonnard στην ιστοσελίδα με τίτλο A Guide to Designing and Building RESTful Web Services with WCF 3.5[29] αναφέρει τα εξής σχετικά με τις μεθόδους του HTTP:

Παρόλο που το HTTP υποστηρίζει πλήρως CRUD, η HTML 4 υποστηρίζει πλήρως μόνο αιτήσεις GET και POST. Η HTML 5, που βρίσκεται ήδη υπό ανάπτυξη, σχεδιάζει να διορθώσει αυτό το πρόβλημα προσθέτοντας υποστήριξη για τις PUT και DELETE. (…) Συχνά επίσης οι φυλλομετρητές και τα firewalls[30] επιτρέπουν μόνο αιτήσεις τύπου GET and POST. Λόγω αυτού του περιορισμού πολλοί ιστοχώροι υπερφορτώνουν τις GET ή POST με ψεύτικα PUT και DELETE αιτήματα. Αυτό μπορεί να επιτευχθεί με τον καθορισμό της πραγματικής μεθόδου του HTTP σε μια προσαρμοσμένη κεφαλίδα HTTP. Μια κοινή κεφαλίδα HTTP που χρησιμοποιείται για το σκοπό αυτό είναι το *X-HTTP-Method-Override* όπως φαίνεται στο παράδειγμα:

```
POST /cinema/123 HTTP/1.1
X-HTTP-Method-Override: DELETE
```

[29] http://msdn.microsoft.com/en-us/library/dd203052.aspx

30 Στην επιστήμη των υπολογιστών ο όρος firewall ή τείχος προστασίας χρησιμοποιείται για να δηλώσει κάποια συσκευή ή πρόγραμμα που είναι έτσι ρυθμισμένο ούτως ώστε να επιτρέπει ή να απορρίπτει πακέτα δεδομένων που περνούν από ένα δίκτυο υπολογιστών σε ένα άλλο.

6.7 Κωδικοί κατάστασης HTTP

Το HTTP ορίζει επίσης, μια σειρά τυποποιημένων κωδικών κατάστασης που καθορίζουν το αποτέλεσμα της επεξεργασίας της αίτησης. Οι κωδικοί κατάστασης είναι οργανωμένοι σε περιοχές που σημαίνουν διαφορετικά πράγματα. Για παράδειγμα, οι κωδικοί κατάστασης στην περιοχή 200 σημαίνουν "επιτυχημένη", ενώ οι κωδικοί κατάστασης στην περιοχή 400 σημαίνουν ότι ο πελάτης έχει αποστείλει μια εσφαλμένη αίτηση. Παρακάτω περιγράφεται κάθε περιοχή κωδικών κατάστασης και παρέχονται μερικά παραδείγματα των πιο κοινών από αυτούς.

- **100** Ενημερωτικό: Αυτό σημαίνει ότι ο διακομιστής έχει λάβει τις κεφαλίδες της αίτησης, και ότι ο πελάτης πρέπει να προχωρήσει για να στείλει το σώμα της αίτησης. π.χ. 100 Συνέχεια

- **200** Επιτυχής: Τυποποιημένη απάντηση για επιτυχημένες αιτήσεις HTTP. Η πραγματική απόκριση εξαρτάται από τη χρησιμοποιημένη μέθοδο αίτησης. Σε μια αίτηση GET, η απάντηση θα περιέχει μια οντότητα που αντιστοιχεί στον πόρο που ζητήσατε. Σε μια POST αίτηση η απόκριση θα περιλαμβάνει οντότητα που περιγράφει ή που περιέχει το αποτέλεσμα της δράσης. π.χ. 200 OK

- **201** Δημιουργήθηκε: Η αίτηση έχει εκπληρωθεί και είχε ως αποτέλεσμα ένας νέος πόρος να δημιουργηθεί.

- **202** Αποδεκτές: Το αίτημα έχει γίνει δεκτό προς μεταποίηση, αλλά η επεξεργασία δεν έχει ολοκληρωθεί.

- **300** Ανακατεύθυνση

- **301** Μεταφέρονται οριστικά

- **304** Δεν τροποποιήθηκε: Δηλώνει ότι ο πόρος δεν έχει τροποποιηθεί από την τελευταία αίτηση.

- **400** Σφάλμα προγράμματος-πελάτη: Η αίτηση δεν μπορεί να εκτελεσθεί λόγω κακής σύνταξης.

- **401** Μη εξουσιοδοτημένη πρόσβαση: Παρόμοια με 403 Forbidden, αλλά ειδικά για χρήση όταν απαιτείται έλεγχος ταυτότητας. Σημαίνει πως ο έλεγχος απέτυχε ή δεν παρασχέθηκαν διαπιστευτήρια.

- **402** Με πληρωμή: Προορίζεται για μελλοντική χρήση.
- **403** Forbidden: Το αίτημα ήταν ένα νόμιμο αίτημα, αλλά ο διακομιστής αρνείται να απαντήσει σε αυτό.
- **404** Δεν βρέθηκε: Ο ζητούμενος πόρος δεν ήταν δυνατό να βρεθεί.
- **405** Δεν επιτρέπεται η μέθοδος: Μια αίτηση ενός πόρου χρησιμοποιώντας μια μέθοδο αίτησης που δεν υποστηρίζεται από αυτόν τον πόρο. Για παράδειγμα, χρησιμοποιώντας GET σε μια φόρμα, η οποία απαιτεί υποβολή δεδομένων μέσω POST, ή χρησιμοποιώντας PUT σε έναν πόρο μόνο για ανάγνωση.
- **500** Σφάλμα διακομιστή: Ένα γενικό μήνυμα λάθους, όταν δεν είναι δυνατόν να δοθεί ένα πιο κατάλληλο μήνυμα. π.χ. 500 Εσωτερικό σφάλμα διακομιστή.
- **501** Δεν έχει υλοποιηθεί : Ο διακομιστής είτε δεν αναγνωρίζει τη μέθοδο αίτησης, ή δεν έχει την ικανότητα να εκπληρώσει το αίτημα.

6.8 Κεφαλίδες HTTP

Τόσο ο διακομιστής όσο και ο πελάτης για να επικοινωνήσουν χρησιμοποιούν κεφαλίδες. Οι κεφαλίδες χωρίζονται από το σώμα της απόκρισης με μια κενή γραμμή. Εδώ περιγράφουμε τις πιο συνηθισμένες από αυτές.

6.8.1 Κεφαλίδες πελάτη και διακομιστή

Συνηθισμένοι τύποι κεφαλίδων που χρησιμοποιούνται τόσο από τον πελάτη όσο και από τον διακομιστή είναι οι εξής:

- **Content-Type**

Ο τύπος MIME (Multipurpose Internet Mail Extensions) του περιεχομένου. Π.χ. Content-Type: application/xml ή Content-Type: application/json; charset=utf-8.

Χρησιμοποιείται τόσο από τον πελάτη (requests) όσο και από τον διακομιστή (response). Για να προσπελάσουμε την τιμή της αίτησης χρησιμοποιούμε την μέθοδο HttpHeaders

#getMediaType(), ενώ για να θέσουμε την τιμή της απάντησης χρησιμοποιούμε την μέθοδο ResponseBuilder #type() ή την ResponseBuilder #variant() του javax.ws.rs.core.Response.

- **Content-Language**

Η γλώσσα του περιεχομένου. Π.χ. Content-Language: el. Για να προσπελάσουμε την τιμή της αίτησης χρησιμοποιούμε την μέθοδο HttpHeaders #getLanguage (), ενώ για να θέσουμε την τιμή της απάντησης χρησιμοποιούμε την μέθοδο ResponseBuilder #language ().

- **Content-Length**

Το μήκος του περιεχομένου σε bytes. Π.χ. Content-Length: 348. Ο διακομιστής θέτει αυτόματα το μήκος του περιεχομένου.

6.8.2 Κεφαλίδες πελάτη

Οι πιο συνηθισμένοι τύποι κεφαλίδων που χρησιμοποιούνται από τον πελάτη είναι οι εξής:

- **Accept**

Η κεφαλίδα καθορίζει τους τύπους περιεχόμενου που ο πελάτης προτιμά όπως XML αντί JSON και όχι απλό κείμενο. Όταν μία μέθοδος έχει σημανθεί (annotated) με @Produces κάθε κεφαλίδα πρέπει να είναι συμβατή με τους τύπους MIME που η μέθοδος επιστρέφει. Την τιμή της κεφαλίδας μπορούμε να βρούμε με την μέθοδο HttpHeaders #getAcceptableMediaTypes() ή Request #selectVariant().

- **Accept-Language**

Οι προτιμώμενες γλώσσες για την απάντηση Π.χ. Accept-Language: el-GR. Η μέθοδος HttpHeaders #getAcceptableLanguages() επιστρέφει μια λίστα των γλωσσών κατά σειρά προτίμησης.

- **Authorization**

Ο πελάτης αποστέλλει τα διαπιστευτήρια του για τον έλεγχο ταυτότητας. Π.χ. Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==. Το τελευταίο τμήμα είναι κωδικοποιημένο με κωδικοποίηση Base64 Η κωδικοποίηση Base64 μετατρέπει δυαδικά δεδομένα σε συμβολοσειρά ASCII μετατρέποντας τα στο 64αδικό σύστημα αρίθμησης. Η μέθοδος Request #getHeaderValue("Authorization") μας δίνει την τιμή της κεφαλίδας.

- **If-Match and If-None-Match**

Αν μία προηγούμενη απόκριση έχει μία ETag κεφαλίδα, ο πελάτης μπορεί να χρησιμοποιήσει την τιμή της σε μία If-Match or If-None-Match κεφαλίδα για να κάνει αίτηση υπό όρους. (Αν ο διακομιστής υποστηρίζει If-Match/If-None-Match κεφαλίδες). Π.χ. μία αίτηση POST με μία If-Match κεφαλίδα και μία παλαιότερη τιμή ETag θα εκτελεστεί όταν η τιμή της ETag είναι ακόμη έγκυρη. Τις τιμές των κεφαλίδων If-Match και If-None-Match μπορούμε να προσδιορίσουμε με την μέθοδο Request #evaluatePreconditions().

- **If-Modified-Since and If-Unmodified-Since**

Οι κεφαλίδες If-Modified-Since και If-Unmodified-Since βασίζονται στην ημερομηνία τελευταίας τροποποίησης (Last-Modified date). Και οι δύο κεφαλίδες με μία έγκυρη ημερομηνία κάνουν μία αίτηση υπό όρους (Αν ο διακομιστής υποστηρίζει τις κεφαλίδες). Π.χ. μία αίτηση GET με μία If-Modified-Since κεφαλίδα και μια περασμένη ημερομηνία επιτρέπει στον διακομιστή να απαντήσει με έναν κωδικό κατάστασης 304 (Not Modified). Στέλνοντας αυτόν τον κωδικό ο διακομιστής λέει στον πελάτη ότι ο ζητούμενος πόρος δεν άλλαξε και δεν χρειάζεται να τον ξανακατεβάσει εξοικονομώντας του πολύτιμο εύρος ζώνης. Η μέθοδος Request #evaluatePreconditions() μας επιτρέπει να εντοπίσουμε τις τιμές των κεφαλίδων. Οι ημερομηνίες πρέπει να έχουν την προσδιορισμένη από τις προδιαγραφές του HTTP μορφή.

6.8.3 Κεφαλίδες διακομιστή

Οι κεφαλίδες που χρησιμοποιούνται από τον διακομιστή είναι πολύ σημαντικές σε μία απόκριση HTTP. Καθορίζουν διάφορα χαρακτηριστικά των δεδομένων που αποστέλλονται.

- **ETag**

Οι κεφαλίδες αυτές αποστέλλονται στον πελάτη ώστε αυτός να τις χρησιμοποιήσει σε υπό όρους αιτήματα. Δείτε τις κεφαλίδες If-Match and If-None-Match. Η πρόσβαση στις τιμές των κεφαλίδων επιτυγχάνεται με τις μεθόδους ResponseBuilder #tag() και javax.ws.rs.core.EntityTag.

- **Expires**

Η κεφαλίδα δείχνει το χρονικό διάστημα που η απόκριση μπορεί να αποθηκευτεί. Είναι χρήσιμη αν ξέρουμε ότι τα δεδομένα δεν θα αλλάξουν για κάποιο χρονικό διάστημα. Οι φυλλομετρητές χρησιμοποιούν αυτή την τιμή για να διαχειριστούν τα δεδομένα. Η μέθοδος ResponseBuilder #expires() χρησιμοποιείται για να τεθεί η τιμή της κεφαλίδας.

- **Last-Modified**

Ορίζει την ημερομηνία (σε μορφή σύμφωνη με τις προδιαγραφές του πρωτοκόλλου HTTP) κατά την οποία ο πόρος μεταβλήθηκε. Ο πελάτης χρησιμοποιεί αυτή την τιμή σε συνεργασία με τις κεφαλίδες If-Modified-Since και If-Unmodified-Since σε υπό όρους αιτήσεις. Η τιμή της κεφαλίδας μπορεί να ρυθμιστεί με την μέθοδο ResponseBuilder #lastModified().

- **Location**

Χρησιμοποιείται σε περιπτώσεις ανακατεύθυνσης για να δηλώσουν που βρίσκεται ένας πόρος ή το URI ενός νεοδημιουργημένου πόρου (Συνήθως μαζί με ένα κωδικό κατάστασης 201 - δημιουργήθηκε). Θέτουμε την τιμή της κεφαλίδας με την μέθοδο ResponseBuilder #location().

- **WWW-Authenticate**

Υποδεικνύει στον πελάτη το σχήμα ελέγχου ταυτότητας που πρέπει να χρησιμοποιηθεί για την πρόσβαση στον πόρο. Π.χ. WWW-Authenticate: Basic.

6.9 Σχεδίαση των RESTful web services

Παρακάτω θα εκθέσουμε τον τρόπο με τον οποίο σχεδιάσαμε την διαδικτυακή μας υπηρεσία.

6.9.1 Θέματα Ασφάλειας

Για την υπηρεσία μας αποφασίσαμε να ζητήσουμε από τους χρήστες να πιστοποιούνται ώστε αργότερα να μπορούμε να προσθέσουμε χαρακτηριστικά όπως η δημιουργία λίστας με φίλους και αγαπημένα μέρη. Θα μπορούμε επίσης να επιτρέπουμε στους χρήστες να βαθμολογούν τα μέρη τα οποία επισκέπτονται και να συλλέγουμε στοιχεία για τα πιο δημοφιλείς προορισμούς.

Το πρωτόκολλο HTTP έρχεται με μερικούς ενσωματωμένους μηχανισμούς ελέγχου ταυτότητας ο δημοφιλέστερος των οποίων είναι βασική πρόσβαση ταυτότητας (Basic authentication). Αυτό είναι ένα από τα πιο δημοφιλή συστήματα ελέγχου ταυτότητας που χρησιμοποιείται στο Διαδίκτυο σήμερα, γιατί είναι εύκολο και υποστηρίζεται ευρέως αλλά είναι επίσης ένα από τα πιο μη ασφαλή επειδή οι κωδικοί πρόσβασης αποστέλλονται μέσω του διαδικτύου σε μια μορφή απλού κειμένου εύκολη στην αποκωδικοποίηση. Μία λύση είναι να απαιτήσουμε ασφαλή επικοινωνία μεταξύ πελάτη και διακομιστή (secure HTTP - HTTPS) οπότε κρυπτογραφείται το σύνολο των αιτήσεων και απαντήσεων που ανταλλάσσονται μεταξύ των δύο. Σύμφωνα με την Βικιπαίδεια (http://el.wikipedia.org/wiki/HTTPS) «Το HTTPS (Secure HTTP) χρησιμοποιείται στην επιστήμη των υπολογιστών για να δηλώσει μία ασφαλή http σύνδεση. Ένας σύνδεσμος

(URL) που αρχίζει με το πρόθεμα https υποδηλώνει ότι θα χρησιμοποιηθεί κανονικά το πρωτόκολλο HTTP, αλλά η σύνδεση θα γίνει σε διαφορετική πόρτα (443 αντί 80) και τα δεδομένα θα ανταλλάσσονται κρυπτογραφημένα. Το σύστημα αυτό σχεδιάστηκε αρχικά από την εταιρία Netscape Communications Corporation για να χρησιμοποιηθεί σε ιστοχώρους όπου απαιτείται αυθεντικοποίηση χρηστών και κρυπτογραφημένη επικοινωνία. Σήμερα χρησιμοποιείται ευρέως στο διαδίκτυο, όπου χρειάζεται αυξημένη ασφάλεια, διότι διακινούνται ευαίσθητες πληροφορίες (πχ αριθμοί πιστωτικών καρτών, passwords κ.ο.κ)».

Μια άλλη προσέγγιση είναι να χρησιμοποιήσουμε έλεγχο ταυτότητας Digest, ένα άλλο σύστημα ελέγχου ταυτότητας ενσωματωμένο στο πρωτόκολλο HTTP. Ο έλεγχος ταυτότητας Digest αποτρέπει υποκλοπές απαγορεύοντας την αποστολή του κωδικού πρόσβασης μέσω διαδικτύου. Αντί για αυτό, ο αλγόριθμος αποστέλλει τιμές κατακερματισμού (Hash) που υπολογίζονται από τον κωδικό πρόσβασης και άλλες τιμές που είναι γνωστές μόνο στον πελάτη και τον διακομιστή.

Να πώς λειτουργεί. Όταν ένας πελάτης προσπαθεί να αποκτήσει πρόσβαση σε ένα προστατευόμενο πόρο, ο διακομιστής επιστρέφει μια 401 («μη εγκεκριμένη») απάντηση στον πελάτη μαζί με μία "WWW-Authenticate" κεφαλίδα αναφέροντας ότι απαιτεί έλεγχο ταυτότητας Digest μαζί με ορισμένα αποδεικτικά στοιχεία. Μόλις ο πελάτης λάβει την κεφαλίδα, μπορεί να δημιουργήσει μία κεφαλίδα "Authorization" που περιέχει την υπολογισμένη τιμή κατακερματισμού και να την στείλει πίσω στον διακομιστή συμπεριλαμβάνοντας και την κεφαλίδα "WWW-Authenticate". Αυτό καθιστά δυνατό για το διακομιστή να υπολογίσει με την σειρά του την ίδια τιμή κατακερματισμού για να επικυρώσει ότι ο πελάτης έχει στην κατοχή του τον κωδικό πρόσβασης. Αν η τιμή της κεφαλίδας είναι σωστή, ο διακομιστής επιτρέπει την πρόσβαση στον πόρο.

Θα πρέπει να αναφερθεί ότι έχουν αναπτυχθεί λεξικά (βλ. http://en.wikipedia.org/wiki/Rainbow_table) που εντοπίζουν τους κωδικούς πρόσβασης των χρηστών, εκτός αν επιβληθεί κάποια πολιτική ισχυρών κωδικών. Επίσης στην Βικιπαίδεια (http://el.wikipedia.org/wiki/Κρυπτογραφική_Συνάρτηση_Κατατεμαχισμού) αναφέρεται «Διάσημες συναρτήσεις κατατεμαχισμού είναι η MD5 και η SHA-1. Το 2008 βρέθηκαν προβλήματα ασφάλειας στην συνάρτηση MD5 σε επίθεση στο πρωτόκολλο Secure Socket Layer (SSL). Η συνάρτηση SHA-0 και η SHA-1 αναπτύχθηκαν από την Υπηρεσία Εθνικής Ασφάλειας (National Security Agency: NSA) των ΗΠΑ. Τον Φεβρουάριο 2005, μια επιτυχημένη επίθεση στην συνάρτηση SHA-1 δημοσιεύθηκε. Η θεωρητική αδυναμία της συνάρτησης SHA-1 είναι γνωστή και έτσι αναπτύχθηκε η συνάρτηση SHA-2.».

Παρόλα αυτά ο έλεγχος ταυτότητας Digest είναι καλύτερος από τον βασικό, παρότι δεν υποστηρίζεται τόσο ευρέως από web browsers και servers.

Μια άλλη προσέγγιση είναι να αποφευχθεί τόσο ο έλεγχος ταυτότητας Digest όσο και ο βασικός. Μπορούμε να εφαρμόσουμε ένα προσαρμοσμένο (custom) έλεγχο ταυτότητας γύρω από την κεφαλίδα "Authorization". Πολλά από αυτά τα σχήματα χρησιμοποιούν την προσέγγιση Hash Message Authentication Code (HMAC), όπου ο διακομιστής μέσω κάποιου άλλου καναλιού επικοινωνίας, συνηθέστερα μέσω e-mail, παρέχει στο χρήστη ένα όνομα και ένα μυστικό κωδικό. Ο πελάτης χρησιμοποιεί τον παρεχόμενο κωδικό για να υπογράψει όλα τα αιτήματα.

Για να λειτουργήσει αυτή η προσέγγιση πρέπει ο διακομιστής και ο πελάτης να γνωρίζουν τον αλγόριθμο που θα ακολουθηθεί για την υπογραφή των αιτήσεων. Είναι σημαντικό ο πελάτης και ο διακομιστής να ακολουθούν τον ίδιο αλγόριθμο για να δουλέψει η μέθοδος. Μόλις ο πελάτης δημιουργήσει την HMAC τιμή κατακερματισμού, μπορεί να την συμπεριλάβει στην κεφαλίδα "Authorization" μαζί με το αναγνωριστικό χρήστη:

```
Authorization:user@gmail.com:b86XjSLZIwDm72io+TTHcu8PqZY=
```

Όταν η υπηρεσία λάβει την αίτηση αυτή, θα διαβάσει την κεφαλίδα "Authorization" και θα χωρίσει την ταυτότητα του χρήστη και την τιμή κατακερματισμού. Θα βρει τον κωδικό χρήστη και θα εκτελέσει τον ίδιο HMAC αλγόριθμο. Εάν η υπολογιζόμενη τιμή κατακερματισμού ταιριάζει με αυτή στο αίτημα ξέρουμε ότι ο πελάτης έχει στην κατοχή του τον κωδικό και είναι ένας έγκυρος χρήστης. Για να μετριάσουμε τυχών επιθέσεις υποκλοπής μπορούμε να συμπεριλάβουμε στον αλγόριθμο υπολογισμού της τιμής κατακερματισμού που στέλνει ο πελάτης και μια χρονική σήμανση. Έτσι ο διακομιστής μπορεί να απορρίψει αιτήσεις με όχι πρόσφατη ημερομηνία.

Η προσέγγιση HMAC είναι ανώτερη τόσο από την τόσο βασική όσο και από την Digest ειδικά αν οι κωδικοί που δημιουργούνται είναι αρκετά μεγάλοι και τυχαίοι ώστε μην περιλαμβάνονται σε λεξικά. Η τεχνική αυτή είναι αρκετά συνηθισμένη στις υπηρεσίες RESTful.

Για να λειτουργήσει η μέθοδος απαιτείται η φύλαξη στο διακομιστή των κωδικών των χρηστών χωρίς κρυπτογράφηση. Αυτό παραβιάζει έναν κανόνα ασφαλείας του OWASP (Open Web Application Security Project). Στην ιστοσελίδα του οργανισμού με τίτλο Hashing Java[31]

[31] https://www.owasp.org/index.php/Hashing_Java

αναφέρεται: «Οι περισσότερες εφαρμογές σήμερα χρησιμοποιούν ένα ζευγάρι login/password για τον έλεγχο ταυτότητας. Οι χρήστες χρησιμοποιούν συχνά το ίδιο ζευγάρι login/password και για άλλα είδη εφαρμογών. Εάν το ζευγάρι κλαπεί, όλοι μπορούν να έχουν πρόσβαση σε όλες τις εφαρμογές που ο χρήστης έχει πρόσβαση.

Πολύ συχνά οι κωδικοί αποθηκεύονται σαν απλό κείμενο. Έτσι, ο κωδικός πρόσβασης μπορεί να διαβαστεί απευθείας από τον διαχειριστή της βάσης δεδομένων, τους υπερχρήστες (super users) ή με SQL Injection attack κλπ. Τα μέσα αποθήκευσης αντιγράφων ασφαλείας είναι επίσης ευάλωτα. Για να λυθεί αυτό το πρόβλημα, οι κωδικοί πρόσβασης πρέπει να αποθηκεύονται κρυπτογραφημένα. Δύο είδη της κρυπτογράφησης είναι διαθέσιμα:

- Συναρτήσεις κρυπτογράφηση μιας κατεύθυνσης (SHA-256 SHA-1 MD5, ..;), επίσης, γνωστές ως συναρτήσεις κατακερματισμού
- Συναρτήσεις αντιστρέψιμης κρυπτογράφησης (DES, AES, ...).

Ωστόσο, η αντιστρέψιμη κρυπτογράφηση είναι άχρηστη για την αποθήκευση των διαπιστευτηρίων (πρβλ. το OWASP οδηγός v2.0.1):

Οι κωδικοί πρόσβασης είναι μυστικοί. Δεν υπάρχει κανένας λόγος να αποκρυπτογραφηθούν κάτω από οποιεσδήποτε συνθήκες. Το προσωπικό του Γραφείου Υποστήριξης (Helpdesk) πρέπει να είναι σε θέση να εκδώσει νέους κωδικούς πρόσβασης (with an audit trail, obviously), και όχι να διαβάσει τους παλιούς κωδικούς πρόσβασης. Ως εκ τούτου, δεν υπάρχει λόγος να αποθηκεύετε κωδικούς πρόσβασης σε μια αντιστρέψιμη μορφή.»

6.9.2 Υλοποίηση ασφάλειας

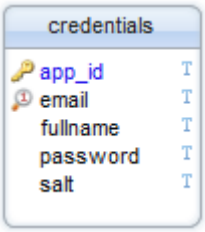
Για την υπηρεσία που σχεδιάζουμε αρκεί ο βασικός έλεγχος ταυτότητας. Αν κάποια στιγμή αποφασίσουμε να υλοποιήσουμε έναν αυστηρότερο έλεγχο μπορούμε εύκολα να προσθέσουμε ασφάλεια SSL (HTTPS).

Στην υλοποίηση μας θα λάβουμε υπ όψιν τις εισηγήσεις του OWASP που παραθέσαμε παραπάνω ώστε να προστατεύσουμε τους χρήστες της εφαρμογής από πιθανή κλοπή των συνθηματικών τους σε περίπτωση επίθεσης στο διακομιστή.

Αρχικά, φτιάχνουμε τον πίνακα credentials που θα φιλοξενήσει τα δεδομένα μας:

```
DROP TABLE pois.credentials;
CREATE TABLE pois.credentials
(
  app_id character varying(48) NOT NULL,
  fullname character varying(128) NOT NULL,
  email character varying(100) NOT NULL,
  password character varying(32) NOT NULL,
  salt character varying(32) NOT NULL,
  CONSTRAINT credential_pkey PRIMARY KEY (app_id )
);
ALTER TABLE pois.credentials OWNER TO postgres;
CREATE UNIQUE INDEX credentials_email_idx ON pois.credentials USING btree (email
COLLATE pg_catalog."default" );
```

Ένα σχήμα του πίνακα φαίνεται πιο κάτω (Εικόνα 6-1)



Εικόνα 6-1 Ο πίνακας credentials

Παρακάτω, φαίνεται το κύριο τμήμα της υλοποίησης της μεθόδου authenticateUser που χρησιμοποιείται για την σύνδεση (login) του χρήστη:

```
    /*
     * Σχετικά με την ασφάλεια δες το άρθρο:
     * https://www.owasp.org/index.php/Hashing_Java
     * Για το OWASP : Open Web Application Security Project
     * μπορείς να δεις στο http://en.wikipedia.org/wiki/OWASP
     **/

    public static synchronized boolean authenticateUser(String email, String
password)
            throws NoSuchAlgorithmException, UnsupportedEncodingException{
        AuthenticationDAO dao = new AuthenticationDAO();
        List<Credentials> cr = dao.selectUser(email);
        if (cr.size() != 1) {
            return false;
        }
        Credentials c = cr.get(0);
        String digest = hashPlainPassword(password, c.getSalt());
        return digest.equals(c.getPassword());
    }

    public static String hashPlainPassword(String password, String salt)
            throws NoSuchAlgorithmException, UnsupportedEncodingException {
        MessageDigest digest = MessageDigest.getInstance("SHA-1");
        digest.reset();
        byte[] bSalt = Base64.decodeBase64(salt);
        digest.update(bSalt);
```

```
        byte[] input = digest.digest(password.getBytes("UTF-8"));
        for (int i = 0; i < ITERATION_NUMBER; i++) {
            digest.reset();
            input = digest.digest(input);
        }
        String pwd = Base64.encodeBase64String(input);
        return pwd;
    }
```

Για να δημιουργήσουμε - καταχωρήσουμε ένα καινούργιο χρήστη στη βάση δεδομένων μας χρησιμοποιούμε την μέθοδο createUser:

```
    public static boolean createUser(String appid, String fname, String email,
String password)
            throws NoSuchAlgorithmException, UnsupportedEncodingException {
        int rows = 0;
        if (email != null && password != null && email.length() <= 100) {
            String salt = generateNewSalt();
            String digest = hashPlainPassword(password, salt);
            AuthenticationDAO dao = new AuthenticationDAO();
            Credentials credentials = new Credentials(email, digest);
            credentials.setSalt(salt);

            credentials.setAppId(appid);
            credentials.setFullName(fname);

            rows = dao.insertUser(credentials);
        }
        return (rows == 1);
    }
```

Ένα ακόμη σημείο που πρέπει να απαντηθεί είναι πού θα γίνεται ο έλεγχος. Αν δηλαδή θα γίνεται σε κάθε μέθοδο που μπορεί να καλέσει ο χρήστης ή σε κάποιο κεντρικό σημείο για κάθε κλήση της υπηρεσίας. Η βιβλιοθήκη Jersey υποστηρίζει φίλτρα που μας παρέχουν χρήσιμες λειτουργίες που είναι κρυμμένες από το επίπεδο της εφαρμογής, τόσο κατά την λήψη αιτήσεων του πελάτη, όσο και την αποστολή απαντήσεων.

Για να ενεργοποιήσουμε τα φίλτρα προσθέτουμε στο web.xml το παρακάτω xml element:

```
<init-param>
    <param-name>com.sun.jersey.spi.container.ResourceFilters</param-name>
    <param-value>filter.RestAuthFilterFactory</param-value>
</init-param>
```

και υλοποιούμε την κλάση **filter.RestAuthFilterFactory**.

```
public class RestAuthFilterFactory implements ResourceFilterFactory {
    final static Logger log =
Logger.getLogger(RestAuthFilterFactory.class);
```

```
    @Context private UriInfo uriInfo;

    @Override
    public List<ResourceFilter> create(AbstractMethod method) {
        return Collections.singletonList((ResourceFilter) new Filter());
    }

    private class Filter implements ResourceFilter, ContainerRequestFilter {

        @Override
        public ContainerRequest filter(ContainerRequest request) {
            log.info("Url invoked is {} " + uriInfo.getPath());

            if (uriInfo.getPath().matches("(?i)auth/.*?")) {
                log.info("uriInfo matches!!");
                return request;
            }
            String authHeader = request.getHeaderValue("Authorization");
            if (authHeader != null && authHeader.startsWith("Basic ")) {
                authHeader = authHeader.substring("Basic ".length());
                String[] creds = new
String(Base64.decodeBase64(authHeader)).split(":");
                String email = creds[0];
                String password = creds[1];
                try {
                    if(AuthenticationManager.authenticateUser(email, password)){
                        log.info("όλα καλά!");
                    }else{
                        log.info("UNAUTHORIZED!
AuthenticationManager.authenticateUser = false");
                        throw new
WebApplicationException(Response.status(Status.UNAUTHORIZED).header("WWW-
Authenticate", "Basic realm=\"\"").build());
                    }
                } catch (NoSuchAlgorithmException | UnsupportedEncodingException
e) {
                    throw new
WebApplicationException(Response.Status.INTERNAL_SERVER_ERROR);
                }
            } else {
                log.info("UNAUTHORIZED! case authHeader != null &&
authHeader.startsWith('Basic ')");
                throw new
WebApplicationException(Response.status(Status.UNAUTHORIZED).header("WWW-
Authenticate", "Basic realm=\"\"").build());
            }
            return request;
        }

        @Override
        public ContainerRequestFilter getRequestFilter() {
            return this;
        }

        @Override
        public ContainerResponseFilter getResponseFilter() {
            return null;
        }

    }
```

```
}
```

Τώρα, κάθε αίτημα προς την υπηρεσία μας εκτός από αυτά που απευθύνονται στις διευθύνσεις /auth/login και /auth/register, ελέγχεται και η συνέχεια επιτρέπεται μόνο αν τα διαπιστευτήρια είναι έγκυρα. Σε αντίθετη περίπτωση, ένας κωδικός κατάστασης 401 ("Μη εξουσιοδοτημένη πρόσβαση") αποστέλλεται στο χρήστη.

6.10 Σχεδιασμός των URI σύμφωνα με την φιλοσοφία REST.

Ο Paul Prescod (στην ιστοσελίδα με τίτλο «Common REST Mistakes» την οποία ανακτήσαμε από τις αποθηκευμένες σελίδες της Google[32]) έχει φτιάξει μια λίστα με τα πιο συνηθισμένα λάθη που πρέπει να αποφεύγουν όσοι ασχολούνται για πρώτη φορά με τα RESTful web services. Για την αποφυγή τους προτείνει κάποιες οδηγίες ενδεικτικές των οποίων είναι οι εξής:

1. Συνεδρίες (sessions) δεν υπάρχουν. Δεν πρέπει να υπάρχει ανάγκη για έναν πελάτη να «συνδεθεί» ή «να ξεκινήσει μια σύνδεση». Ο έλεγχος ταυτότητας γίνεται αυτόματα σε κάθε μήνυμα. (HTTP authentication is done automatically on every message).

2. Να μη γίνεται κατάχρηση της μεθόδου POST. Η POST είναι η πιο ευέλικτη από τις μεθόδους του HTTP. Μπορούμε να στέλνουμε και να παίρνουμε πληροφορίες την ίδια στιγμή. Να χρησιμοποιείται POST μόνο όταν δημιουργείται ένα νέο URI.

3. Να μην τοποθετούνται δράσεις (actions) στα URI. π.χ. /?action=delete. Ο σημαντικότερος λόγος είναι ότι χρησιμοποιούμε την GET για κάτι μη ασφαλές.

Στον ιστότοπο του βιβλίου «Ajax Design Patterns» του Michael Mahemoff, η ιστοσελίδα με τίτλο «RESTful Service»33 αναφέρει: «Οι πόροι είναι διευθύνσεις URL. Ένας πόρος είναι κάτι σαν μια «επιχειρηματική οντότητα (business entity)» στη διάλεκτο της μοντελοποίησης. Είναι μια οντότητα που επιθυμούμε να εκθέσουμε ως μέρος ενός API. Σχεδόν πάντα, η οντότητα είναι ένα **ουσιαστικό.(...).**Οι λειτουργίες είναι μέθοδοι HTTP».

[32] Η ιστοσελίδα ανακτήθηκε από το cache της Google http://webcache.googleusercontent.com/search?q=cache:-rIHhiYZwVYJ:www.prescod.net/rest/mistakes/+&cd=1&hl=el&ct=clnk&gl=gr&client=firefox-a γιατί η αρχική διεύθυνση δεν λειτουργεί πια: http://www.prescod.net/rest/mistakes/

[33] http://ajaxpatterns.org/RESTful_Service

Ο Joe Gregorio (στην ιστοσελίδα με τίτλο «How to Create a REST Protocol»34 του ιστοτόπου xml.com) προτείνει την ακόλουθη μεθοδολογία σχεδίασης αποτελούμενη από 4 βήματα.

1. Ποια είναι τα URIs;
2. Ποια είναι η μορφή των δεδομένων;
3. Ποιες μέθοδοι υποστηρίζονται σε κάθε URI;
4. Ποιοι κωδικοί κατάστασης θα μπορούσαν να επιστραφούν;

6.10.1 Ποια είναι τα URIs;

Στο πρώτο βήμα για να προσδιορίσουμε τα URIs βρίσκουμε τα ουσιαστικά της εφαρμογής ή τα business entities. Σύμφωνα με όσα έχουμε δει ως τώρα αυτά είναι:

- Τα σημεία ενδιαφέροντος (POIs).
- Οι κινηματογράφοι, οι ταινίες και οι προβολές.
- Τα θέατρα, οι παραστάσεις και οι ώρες προβολής.
- Οι διαφόρων τύπων εκδηλώσεις και οι αίθουσες στις οποίες φιλοξενούνται.
- Μπαρ και Clubs, Κέντρα και Εστιατόρια.

Ο τρόπος σκέψης είναι αντίστοιχος, σε κάθε μια περίπτωση, οπότε αρκεί να δείξουμε την σχεδίαση των σημείων ενδιαφέροντος. Τα σημεία ενδιαφέροντος είναι συλλογές αντικειμένων και η βασική URL είναι: /pois. (Πιο κάτω ότι υπάρχει μέσα σε άγκιστρα είναι παράμετρος την οποία παρέχει ο πελάτης. Το υπόλοιπο τμήμα είναι το σταθερό τμήμα του URI).

/pois : Επιστρέφει την συλλογή όλων των σημείων ενδιαφέροντος χωρίς κανένα περιορισμό.

/pois/municipality/?mid={mid} : Επιστρέφει τα σημεία που ανήκουν στο δήμο με το συγκεκριμένο id. Η παράμετρος mid είναι υποχρεωτική.

[34] http://www.xml.com/pub/a/2004/12/01/restful-web.html

/pois/distance/?cx={cx}&cy={cy}&radius={radius} : Επιστρέφει τα σημεία που βρίσκονται σε ακτίνα {radius} από το σημείο: ({cx}, {cy}). Όλες οι παράμετροι είναι υποχρεωτικές.

/pois/box/?left={left}&bottom={bottom}&right={right}&top={top} : Επιστρέφει τα σημεία που βρίσκονται μέσα το παραλληλόγραμμο που ορίζεται από τα σημεία {left}, {bottom}, {right} και {top}. Όλες οι παράμετροι είναι υποχρεωτικές.

Ένα από τα προβλήματα που έχουμε να αντιμετωπίσουμε είναι το φιλτράρισμα των δεδομένων λόγω του πλήθους τους. Για να το πετύχουμε χρησιμοποιούμε σε όλα τα προηγούμενα URIs τις ακόλουθες παραμέτρους (Query Parameters):

name={name} : Η παράμετρος name μπορεί να είναι της μορφής 'όνομα%' ή '%όνομα%' ή '%όνομα' που σημαίνει το όνομα «αρχίζει από», «περιέχει» ή «τελειώνει σε» όνομα. Η παράμετρος είναι προαιρετική.

types={types} : Το σημείο ενδιαφέροντος είναι τύπου {types} (π.χ. τα εστιατόρια έχουν types = 4). Πιο σωστά η παράμετρος είναι συμβολοσειρά της μορφής «1,2,3» που σημαίνει ότι τα σημεία ενδιαφέροντος είναι τύπου 1 ή 2 ή 3. Είναι προαιρετική και μπορεί να συνυπάρχει με την {name}.

ux={ux}&uy={uy}: Τα {ux} και {uy} είναι το Longitude και Latitude της τωρινής θέσης του χρήστη. Όταν περιλαμβάνονται στο URL ο διακομιστής επιστρέφει και την απόσταση του σημείου από την θέση του χρήστη. Είναι προαιρετικές αλλά αν υπάρχουν πρέπει να υπάρχουν και οι δυο. Συνδυάζονται με την {name} και την {types}.

6.10.2 Ποια είναι η μορφή των δεδομένων;

Καθορίζουμε την επιστρεφόμενη μορφή χρησιμοποιώντας τις κατάλληλες σημάνσεις (annotations) στην κλάση POI:

```
package model;

import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name = "poi")
@XmlType(propOrder = { "poiId", "poiType",
        "name", "address", "phones",
        "latitude", "longitude", "distance"
        })
public class POI {
```

```
    private Integer poiId;
    private Integer poiType;
    private String name;
    private String address;
    private String phones;
    private String latitude;
    private String longitude;
    private Integer distance;

    public POI() { }
    .
    .
    .
}
```

Η αναπαράσταση σε μορφή XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<pOIs>
    <poi>
        <poiId>2515</poiId>
        <poiType>2</poiType>
        <name>Μουσείο Κυκλαδικής Τέχνης</name>
        <address>Νεοφύτου Δούκα 4, Κολωνάκι</address>
        <phones>2107228321</phones>
        <latitude>37.975717000000003</latitude>
        <longitude>23.742173999999999</longitude>
    </poi>
    <poi>...</poi>
    <poi>...</poi>
</pOIs>
```

Η αναπαράσταση σε μορφή JSON:

```
[
   {
       "poiId": 2515,
       "poiType": 2,
       "name": "Μουσείο Κυκλαδικής Τέχνης",
       "address": "Νεοφύτου Δούκα 4, Κολωνάκι",
       "phones": "2107228321",
       "latitude": "37.975717000000003",
       "longitude": "23.742173999999999",
       "distance": null
   }
   {…}
   {…}
]
```

6.10.3 Ποιες μέθοδοι υποστηρίζονται σε κάθε URI;

Οι υποστηριζόμενες μέθοδοι φαίνονται στον επόμενο πίνακα (Πίνακας 6-2).

Πίνακας 6-2 Ποιες μέθοδοι υποστηρίζονται σε κάθε URI;

URI	Μέθοδος
/pois	GET
/pois/municipality/	GET
/pois/distance/	GET
/pois/box/	GET

6.10.4 Ποιοι κωδικοί κατάστασης θα μπορούσαν να επιστραφούν;

Στον επόμενο πίνακα δείχνουμε τους κωδικούς κατάστασης που μπορούν να επιστραφούν (Πίνακας 6-3).

Πίνακας 6-3 Ποιοι κωδικοί κατάστασης θα μπορούσαν να επιστραφούν;

URI	Κωδικοί κατάστασης
/pois	200 (OK), 401(Unauthorized), 404 (Not Found)
/pois/municipality/	200 (OK), 400 (Bad Request), 401(Unauthorized), 404 (Not Found).
/pois/distance/	200 (OK), 400 (Bad Request), 401(Unauthorized), 404 (Not Found).
/pois/box/	200 (OK), 400 (Bad Request), 401(Unauthorized), 404 (Not Found).

6.11 Επικοινωνία με την βάση

Πιο κάτω (Πίνακας 6-4) βλέπουμε την αντιστοίχιση των πράξεων του αρκτικόλεξου CRUD, που αναφέραμε προηγουμένως, και των πράξεων των RESTful web sevices καθώς και των στάνταρ δηλώσεων SQL (SQL statements):

Πίνακας 6-4 Επικοινωνία με την βάση

Operation	Λειτουργία	RESTful Method	SQL
Create	Δημιουργία	POST	INSERT
Read (Retrieve)	Ανάκτηση	GET	SELECT
Update (Modify)	Ενημέρωση	PUT	UPDATE
Delete (Destroy)	Διαγραφή	DELETE	DELETE

Εύκολα καταλαβαίνουμε λοιπόν ότι σε κάθε μία μέθοδο GET των υπηρεσιών διαδικτύου που αναφέραμε πριν, αντιστοιχεί και ένα SELECT της SQL. Αν συνδυάσουμε όμως τα φίλτρα που

χρησιμοποιούμε για να περιορίσουμε τον αριθμό των δεδομένων με τα SELECT που χρειάζονται για τη συλλογή ανά περίπτωση των δεδομένων ο αριθμός των δηλώσεων SQL είναι αρκετά μεγάλος. Ευτυχώς, η βιβλιοθήκη MyBatis μας δίνει τρόπους να περιορίσουμε αυτό τον αριθμό. Όπως αναφέρεται στην ιστοσελίδα με τίτλο «Mapper XML Files»[35] του ιστοτόπου mybatis.org «Η πραγματική δύναμη του MyBatis είναι τα Mapped Statements. Εκεί συμβαίνει όλη η μαγεία. Παρόλη την ισχύ τους τα Mapper XML αρχεία είναι σχετικά απλά. Αν τα συγκρίνουμε με τον αντίστοιχο κώδικα JDBC, θα δείτε αμέσως μια εξοικονόμηση 95% του κώδικα.». Το δικό μας αρχείο περιλαμβάνει τον κώδικα:

```
<select id="selectByCriteria" resultMap="BaseResultMap"
parameterType="model.POISearchCriteria" useCache="true">
SELECT
    poi.pid, poi.pois_type_id, poi.name, poi.address, poi.phones,
    ST_Y(location) AS latitude,
    ST_X(location) AS longitude
    <if test="criteria.userLocation != null">
      , ST_Distance(location,
ST_SetSRID(ST_MakePoint(#{criteria.userLocation.x,jdbcType=NUMERIC},
#{criteria.userLocation.y,jdbcType=NUMERIC}), 4326)::geography)::integer AS
distance
    </if>
FROM
    <choose>
        <when test="criteria.municipalities != null">
            pois.poi, attica.kallikratis
        </when>
        <otherwise>
            pois.poi
        </otherwise>
    </choose>
    <trim prefix="WHERE" prefixOverrides="AND |OR ">
        <if test="criteria.name != null">
            poi.name iLIKE #{criteria.name}
        </if>
        <if test="criteria.types != null">
            AND poi.pois_type_id IN (${criteria.types})
        </if>
        <if test="criteria.municipalities != null">
            AND kallikratis.gid IN (${criteria.municipalities})
                AND ST_Contains(kallikratis.geom, poi.location)
        </if>
        <choose>
            <when test="criteria.center != null">
                AND ST_DWithin(poi.location,
ST_SetSRID(ST_MakePoint(#{criteria.center.x,jdbcType=NUMERIC},
                    #{criteria.center.y,jdbcType=NUMERIC}), 4326)::geography,
#{criteria.radius,jdbcType=DOUBLE})
            </when>
            <when test="criteria.box != null">
                AND ST_Within(poi.location,
ST_SetSRID(#{criteria.box,jdbcType=OTHER}, 4326))
            </when>
```

[35] http://www.mybatis.org/core/sqlmap-xml.html

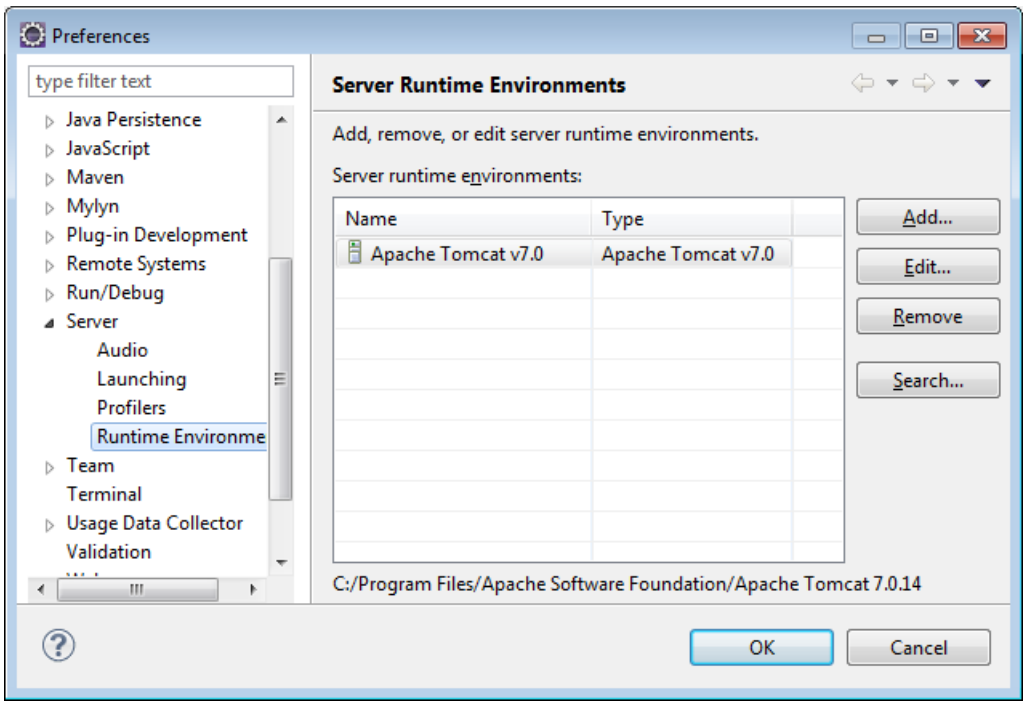
```
        </choose>
    </trim>
     <choose>
         <when test="criteria.userLocation != null">
             ORDER BY distance
         </when>
         <otherwise>
             ORDER BY name
         </otherwise>
     </choose>
</select>
```

Τα κριτήρια ορίζονται στην κλάση POISearchCriteria και αναλόγως την τιμή κάθε παραμέτρου, έχουμε όπως φαίνεται στον κώδικα SQL παραπάνω και διαφορετική δήλωση SELECT περιορίζοντας τον κώδικα σε βαθμό πολύ κοντά στον ισχυρισμό του ιστοτόπου mybatis.org δηλ. στο 95%. Το παραπάνω Mapped Statement επιστρέφει μια λίστα από αντικείμενα POI (ένα List<POI>). Η κλάση POISearchCriteria έχει 6 ανεξάρτητες μεταξύ τους μεταβλητές και έτσι από το πιο πάνω mapper αρχείο μπορούν να παραχθούν 2^6 (= 64) διαφορετικές δηλώσεις SQL. Κάτι επίσης που πρέπει να επισημανθεί είναι η έκφραση useCache="true" η οποία σημαίνει ότι τα δεδομένα αποθηκεύονται στην προσωρινή μνήμη και ανακαλούνται από εκεί και όχι από την βάση όταν η δήλωση ξανακληθεί με τις ίδιες παραμέτρους.

6.12 Ρυθμίσεις Jersey και Tomcat

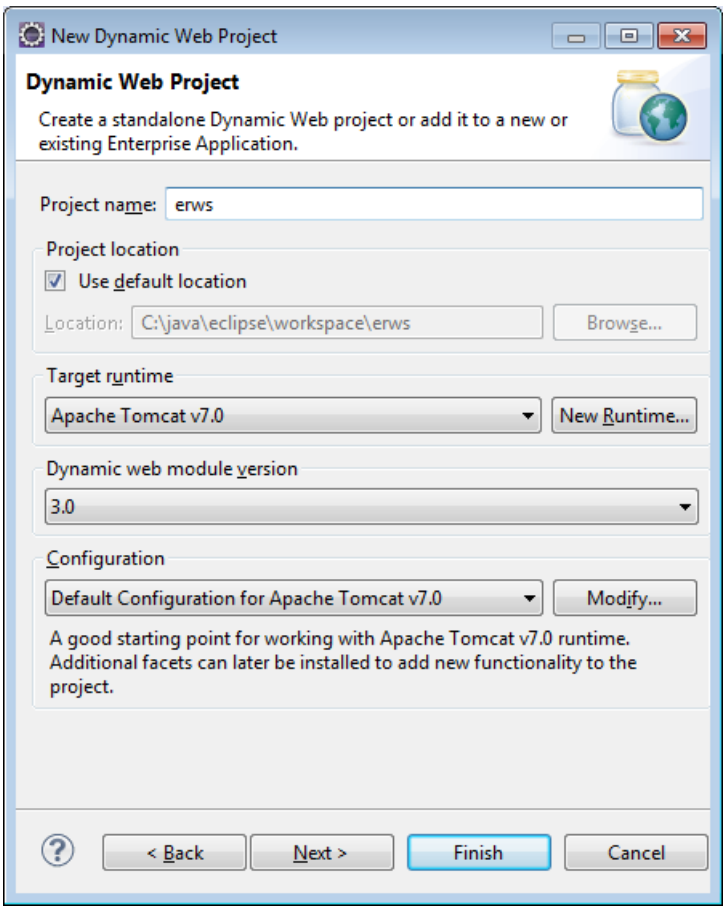
Η βιβλιοθήκη Jersey μπορεί να χρησιμοποιηθεί σε ένα απλό java project ενσωματώνοντας τον διακομιστή ιστοσελίδων (webserver) Grizzly που αποτελεί τμήμα του application server GlassFish. Μπορεί όμως να χρησιμοποιηθεί και σε ένα «Dynamic Web Project» και να διαμοιραστεί χρησιμοποιώντας τον διακομιστή ιστοσελίδων Apache Tomcat 7. Επιλέξαμε την δεύτερη μέθοδο.

Πριν δημιουργήσουμε ένα νέο project πρέπει να εγκαταστήσουμεμε τον Apache Tomcat 7 και να ρυθμίσουμε το Eclipse να χρησιμοποιεί τον Tomcat. Επιλέξαμε Windows > Preferences > Server > Runtime Environments και πατήσαμε Add… Στην επόμενη οθόνη επιλέξαμε Apache Tomcat 7, πιέσαμε Next> και ορίσαμε το φάκελο όπου εγκαταστήσαμε τον Tomcat (Εικόνα 6-2).



Εικόνα 6-2 Ενημέρωση των Servers του Eclipse

Στη συνέχεια δημιουργούμε ένα καινούργιο Dynamic Web Project (Εικόνα 6-3):



Εικόνα 6-3 Νέο Dynamic Web Project

Κατεβάζουμε από τη διεύθυνση http://maven.java.net/ service/ local/ artifact/ maven/ redirect?r=releases&g=com.sun.jersey&a=jersey-archive&v=1.12&e=zip το συμπιεσμένο αρχείο jersey-archive-1.12.zip και το αποσυμπιέζουμε. Αντιγράφουμε τα περιεχόμενα του υποφακέλου lib στο φάκελο /erws/WebContent/WEB-INF/lib του project μας.

Ρυθμίζουμε το αρχείο /erws/WebContent/WEB-INF/web.xml ως εξής:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">
    <display-name>erws</display-name>
    <servlet>
        <servlet-name>restService</servlet-name>
        <servlet-
class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>

        <init-param>
```

```
            <param-name>com.sun.jersey.config.property.packages</param-name>
            <param-value>resources</param-value>
        </init-param>

        <init-param>
            <param-name>com.sun.jersey.spi.container.ResourceFilters</param-name>
            <param-value>filter.RestAuthFilterFactory</param-value>
        </init-param>

        <!-- This says Jersey to use Jackson -->
        <init-param>
            <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>
            <param-value>true</param-value>
        </init-param>

        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>restService</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Κεφάλαιο 7
Android Application

Στο κεφάλαιο αυτό θα αναφερθούμε συνοπτικά στο Android σε κάποια σημεία που θεωρούμε ότι πρέπει να αναφέρουμε για τις δυσκολίες που συναντήσαμε στην υλοποίηση και θα παρουσιάσουμε κάποιες οθόνες της εφαρμογής.

7.1 Η πλατφόρμα Android

Το Android είναι ένα λειτουργικό σύστημα και μία πλατφόρμα λογισμικού πάνω στην οποία αναπτύσσονται εφαρμογές. Ένα βασικό σύνολο από εφαρμογές για καθημερινές εργασίες όπως περιήγηση στο διαδίκτυο και email συμπεριλαμβάνονται στις συσκευές Android.

Το Android είναι μία αναπτυσσόμενη πλατφόρμα ανάπτυξης εφαρμογών κινητών τηλεφώνων. Είναι προϊόν του οράματος της συμμαχίας «Open Handset Alliance» για ένα ανθεκτικό και ανοιχτό περιβάλλον ανάπτυξης για ασύρματες εφαρμογές. Η πλατφόρμα σχεδιάστηκε με σκοπό την ενθάρρυνση μίας ελεύθερης και ανοιχτής αγοράς που θα

μπορούν να απολαμβάνουν όλοι οι χρήστες εφαρμογών κινητών τηλεφώνων και που θα μπορούν να αναπτύσσουν όλοι οι προγραμματιστές λογισμικού.

Τη στιγμή που γράφουμε η τελευταία έκδοση του Android είναι η 4.0 (πρώτη έκδοση 19 Οκτωβρίου, 2011), ενώ έχει αναγγελθεί η κυκλοφορία της έκδοσης 4.1. Οι κωδικές ονομασίες των διαφόρων εκδόσεων έχουν τα ονόματα γλυκών ή επιδορπίων και παρατίθενται παρακάτω[36]:

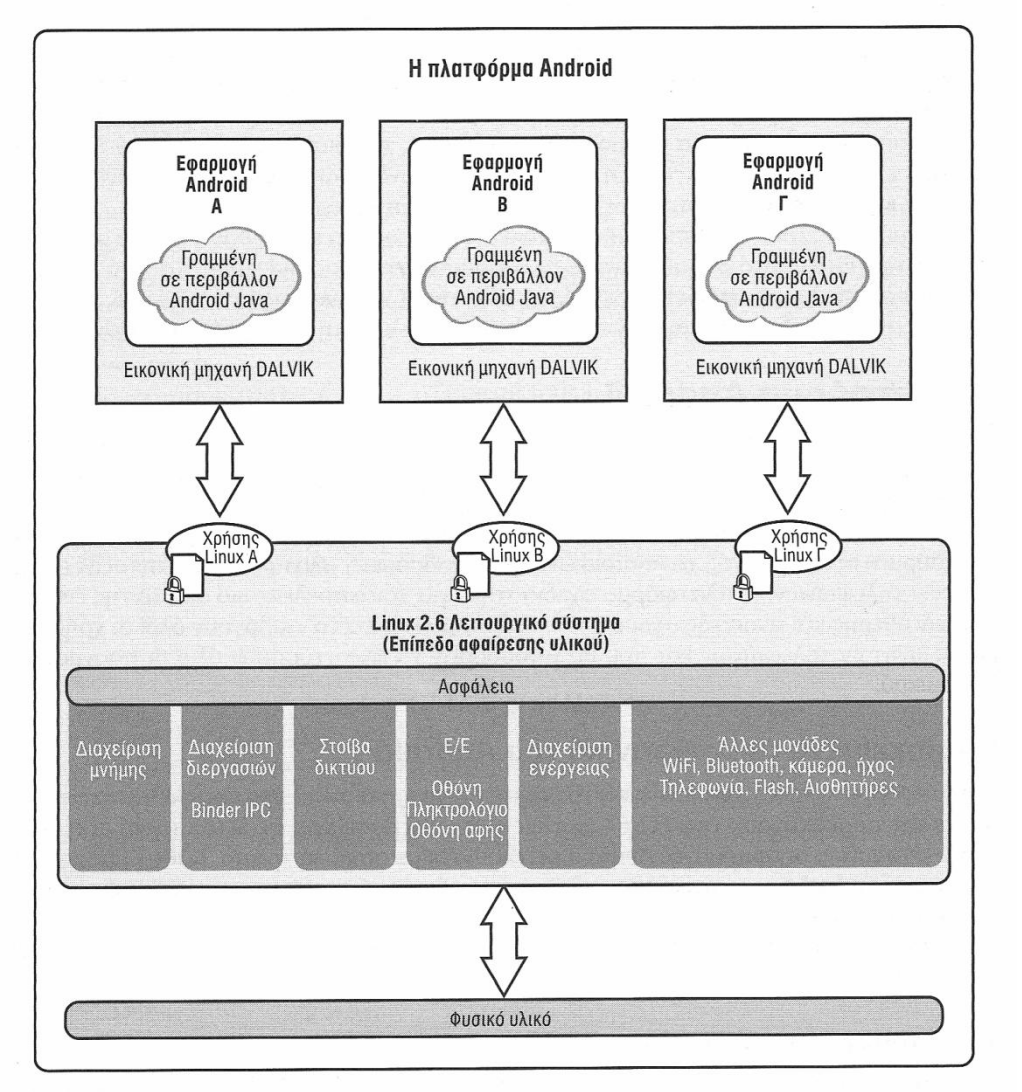
Πίνακας 7-1 Οι κωδικές ονομασίες των εκδόσεων του Android

Έκδοση	Κωδική ονομασία
1.0	Apple Pie
1.1	Banana bread
1.5	Cupcake
1.6	Donut
2.0	Eclair
2.1	Eclair
2.2	Froyo
2.3	Gingerbread
3.0	Honeycomb
3.1	Honeycomb
4.0	Ice Cream Sandwich
4.1	Jelly Bean

7.2 Η αρχιτεκτονική του Android

Η πλατφόρμα Android σχεδιάστηκε έτσι ώστε να είναι πιο ανθεκτική σε σφάλματα από πολλούς από τους προκατόχους της. Η τηλεφωνική συσκευή βασίζεται στο λειτουργικό σύστημα Linux πάνω στο οποίο οι εφαρμογές Android εκτελούνται με ασφαλή τρόπο. Κάθε εφαρμογή Android λειτουργεί στη δική της εικονική μηχανή (δείτε την Εικόνα 7-18). Οι εφαρμογές Android είναι κώδικας υπό διαχείριση που σημαίνει ότι είναι λιγότερο πιθανό να προκαλέσουν το κρασάρισμα του τηλεφώνου και η πιθανότητα καταστροφής της συσκευής («κόλλημα») είναι πολύ μικρότερη.

[36] Περισσότερα μπορεί κανείς να δει στην ιστοσελίδα: http://en.wikipedia.org/wiki/Android_versions



Εικόνα 7-1 Η πλατφόρμα Android

7.3 Το λειτουργικό σύστημα Linux

Το κέλυφος Linux 2.6 χειρίζεται τις βασικές υπηρεσίες του συστήματος και ενεργεί ως επίπεδο αφαίρεσης υλικού (HAL, Hardware Abstraction Layer) ανάμεσα στο φυσικό υλικό της συσκευής και τη στοίβα λογισμικού του Android.

Μερικές απ' τις βασικές λειτουργίες που χειρίζεται το κέλυφος είναι:

- Επιβολή δικαιωμάτων χρήσης και ασφάλειας εφαρμογών.
- Διαχείριση μνήμης χαμηλού επιπέδου.

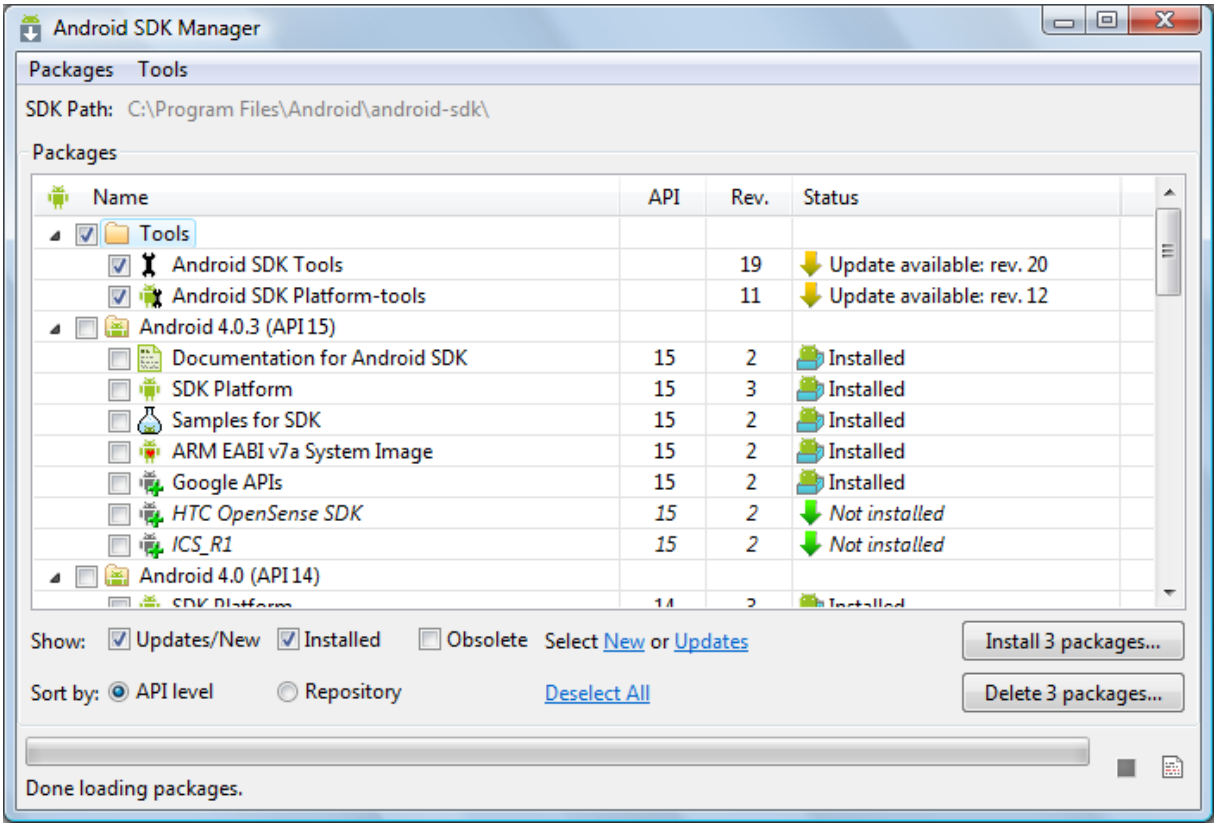
- Διαχείριση διεργασιών και αλληλουχία ενεργειών.
- Η στοίβα δικτύου.
- Οθόνη, είσοδος από πληκτρολόγιο, κάμερα, Wi-Fi, μνήμη Flash, ήχος και πρόσβαση προγράμματος οδήγησης λειτουργίας σύνδεσης (επικοινωνία μεταξύ διεργασιών).

7.4 Περιβάλλον εκτέλεσης εφαρμογών Android

Κάθε εφαρμογή Android εκτελείται σε μία ξεχωριστή διεργασία με το δικό της στιγμιότυπο της εικονικής μηχανής (VM) Dalvik. Η σχεδίαση της Dalvik έχει βελτιστοποιηθεί για φορητές συσκευές. Η Dalvik VM καταλαμβάνει μικρή ποσότητα μνήμης και πολλαπλά στιγμιότυπά της μπορούν να εκτελούνται ταυτόχρονα στη συσκευή. Για την εικονική μηχανή θα επανέλθουμε πιο κάτω.

7.5 Η εγκατάσταση του Android SDK

Το Android SDK (Software Developers Kit) αποτελεί μια συλλογή εργαλείων και βιβλιοθηκών που καθιστούν εφικτή την ανάπτυξη εφαρμογών στο Android. Είναι διαθέσιμο στη διεύθυνση http://developer.android.com/sdk/index.html Το SDK συνοδεύεται από το εργαλείο Android SDK Manager που επιτρέπει την επικαιροποίηση των πακέτων λογισμικού (packages) διαγράφοντας τα ξεπερασμένα και εγκαθιστώντας ενημερώσεις και ad-ons (Εικόνα 7-2).



Εικόνα 7-2 Η εγκατάσταση του Android SDK

Το Android SDK περιλαμβάνει μια ποικιλία από εργαλεία που βοηθούν στην ανάπτυξη mobile εφαρμογών για την πλατφόρμα Android. Τα εργαλεία του κατατάσσονται σε δύο ομάδες: Εργαλεία SDK και εργαλεία πλατφόρμας (platform tools) . Τα εργαλεία SDK είναι ανεξάρτητα πλατφόρμας και απαιτούνται άσχετα σε ποια πλατφόρμα Android κάνουμε ανάπτυξη. Τα εργαλεία πλατφόρμας (platform tools) υποστηρίζουν τα χαρακτηριστικά κάποιας συγκεκριμένης, συνήθως της τελευταίας, πλατφόρμας Android.

Τα πιο σημαντικά εργαλεία είναι : το **Android SDK Manager**, το **AVD Manager** (για την διαχείριση των AVDs - Android Virtual Devices), ο **εξομοιωτής** (**emulator**) που μας επιτρέπει να αναπτύξουμε και να δοκιμάσουμε Android εφαρμογές χωρίς τη χρήση φυσικής συσκευής και το **Dalvik Debug Monitor Server** (DDMS).

Μια σύντομη περίληψη μερικών συχνά χρησιμοποιούμενων εργαλείων SDK παρέχεται παρακάτω.

- **android:** Μας επιτρέπει να διαχειριστούμε AVDs έργα και τα εγκατεστημένα στοιχεία του SDK.

- **Dalvik Debug Monitor Server (DDMS):** Μας βοηθά στον εντοπισμό σφαλμάτων (debugging) της εφαρμογής μας. Το DDMS μας επιτρέπει την διαχείριση των διεργασιών στον εξομοιωτή ή στην συσκευή. Μεταξύ άλλων μας δίνει τη δυνατότητα port-forwarding υπηρεσιών, λήψη screenshots, εμφάνιση πληροφοριών για τον σωρό και τα νήματα, διαχείρισης του logcat (εργαλείο εμφάνισης συμβάντων), εμφάνιση πληροφοριών διεργασιών, προσομοίωση εισερχόμενων κλήσεων και μηνυμάτων, προσομοίωση δεδομένων θέσης κ.ά.

- **dmtracedump:** Δημιουργεί γραφικά διαγράμματα από τα αρχεία καταγραφής του συστήματος (trace log files).

- **Draw 9-patch**: Μας επιτρέπει να δημιουργήσουμε εύκολα NinePatch γραφικά χρησιμοποιώντας ένα πρόγραμμα επεξεργασίας WYSIWYG.(από τα αρχικά What You See Is What You Get.) Τα γραφικά αυτά είναι βελτιστοποιημένα για κινητές συσκευές.

- **Android Emulator (εξομοιωτής):** Μια QEMU (από τα αρχικά «Quick EMUlator») συσκευή. Χρησιμοποιείται για το σχεδιασμό, τον εντοπισμό σφαλμάτων και την δοκιμή της εφαρμογής μας σε ένα πραγματικό περιβάλλον Android.

- **Hierarchy Viewer**: Μας επιτρέπει να διορθώσουμε και να βελτιστοποιήσουμε τη διεπαφή χρήστη (UI) της εφαρμογής του Android. Παρέχει μια μεγεθυμένη οπτική αναπαράσταση της οθόνης του κινητού παρουσιάζοντας τα επιμέρους στοιχεία σε ιεραρχική μορφή.

- **mksdcard**: Μας βοηθάει να δημιουργήσουμε μια εικονική κάρτα SD που μπορούμε να χρησιμοποιήσουμε με τον εξομοιωτή για να προσομοιώσουμε την παρουσία μιας εξωτερικής κάρτας αποθήκευσης.

- **Monkey:** Τρέχει στον εξομοιωτή ή τη συσκευή μας και δημιουργεί τυχαία γεγονότα που θα μπορούσαν να έχουν προέρθει από τον χρήστη όπως κλικ, αγγίγματα, κ.λ.π , καθώς

και μια σειρά από γεγονότα επιπέδου συστήματος. Χρησιμοποιείται στο στρες-τεστ της εφαρμογής μας.

- **monkeyrunner:** Παρέχει ένα API για τη συγγραφή προγραμμάτων, που ελέγχουν μια συσκευή Android ή εξομοιωτή έξω από τον κώδικα της εφαρμογής.

- **Proguard**: Συρρικνώνει και βελτιστοποιεί τον κώδικα μας με την αφαίρεση αχρησιμοποίητων τμημάτων. Μετονομάζει τις κλάσεις, τα πεδία και τις μεθόδους δίνοντας ονόματα χωρίς σημασία με σκοπό να εμποδίσει το reverse-engineering της εφαρμογής. Τέλος, βελτιστοποιεί το bytecode της εφαρμογής καθιστώντας τον κατάλληλο για εκτέλεση σε περιβάλλοντα Java 6 ή για την έκδοση Java Micro Edition.

- **sqlite3:** Μας επιτρέπει πρόσβαση στα αρχεία δεδομένων της ελαφριάς αλλά ισχυρής βάσης δεδομένων SQLite που δημιουργούνται και χρησιμοποιούνται από τις εφαρμογές του Android.

- **traceview:** Παρέχει ένα γραφικό πρόγραμμα προβολής για αρχεία καταγραφής (execution logs) που δημιουργούνται κατά το χρόνο εκτέλεσης της εφαρμογής μας.

- **zipalign:** Βελτιστοποιεί τα .apk αρχεία.

Εκτός των εργαλείων SDK υπάρχουν και τα εργαλεία πλατφόρμας (platform tools)

Τα εργαλεία πλατφόρμας ενημερώνονται συνήθως κάθε φορά που εγκαθιστούμε μια νέα πλατφόρμα SDK. Κάθε ενημερωμένη έκδοση των εργαλείων της πλατφόρμας είναι συμβατή με παλαιότερες πλατφόρμες. Από αυτά τα εργαλεία χρησιμοποιούμε συνήθως άμεσα μόνο ένα το Android Debug Bridge (ADB). Αυτό είναι ένα ευέλικτο εργαλείο που σας επιτρέπει να διαχειριστούμε την κατάσταση του εξομοιωτή ή της συσκευής Android. Μπορούμε επίσης να το χρησιμοποιήσουμε για να εγκαταστήσουμε μια εφαρμογή Android (*.apk) σε μια συσκευή.

Άλλα εργαλεία πλατφόρμας, όπως όπως aidl, aapt, dexdump, και dx , καλούνται από τα build tools του Android ή τα Android Development Tools (ADT) έτσι που σπάνια χρειάζεται να τα χρησιμοποιήσουμε άμεσα. Η Google συνιστά την εγκατάσταση του

Eclipse και του plugin Android Development Tools (ADT) το οποίο έχει σχεδιαστεί για να μας δώσει ένα ισχυρό, ενοποιημένο περιβάλλον οικοδόμησης εφαρμογών Android.

7.6 Η εγκατάσταση των Android Development Tools

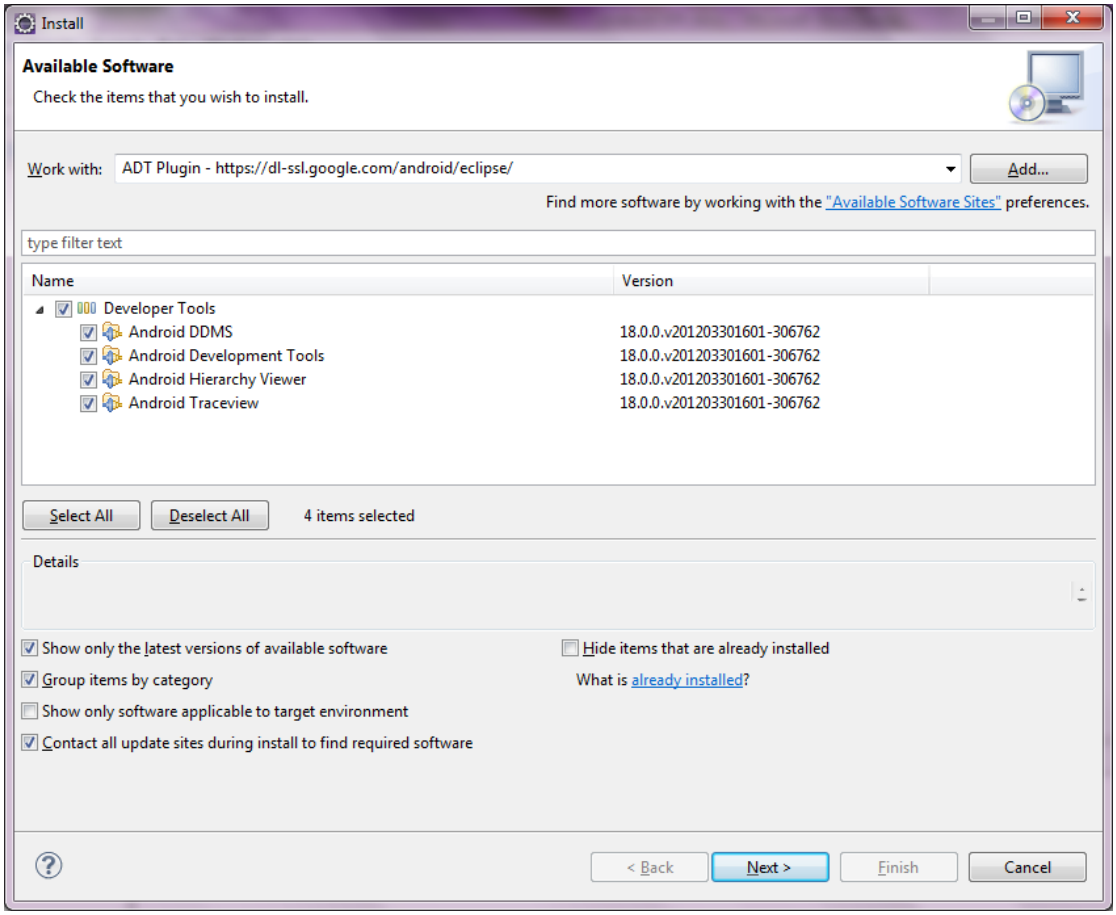
Για να εγκαταστήσουμε τα Android Development Tools ακολουθούμε τα εξής βήματα:

1. Ξεκινάμε το Eclipse και επιλέγουμε *Help > Install New Software....*

2. Κάνουμε κλικ στο κουμπί Add, που υπάρχει στην πάνω αριστερή γωνία.

3. Στη φόρμα που εμφανίζεται συμπληρώνουμε "ADT Plugin" στο πεδίο Name και την ακόλουθη URL στο πεδίο Location: https://dl-ssl.google.com/android/eclipse/ (Αν υπάρξουν προβλήματα η Google συνιστά να χρησιμοποιήσουμε «http» στο URL αντί «https», αν και το δεύτερο είναι προτιμότερο για λόγους ασφαλείας).

4. Στην οθόνη «Available Software» επιλέγουμε το checkbox δίπλα από το Developer Tools και πατάμε Next (Εικόνα 7-3).



Εικόνα 7-3 Εγκατάσταση ADT Plugin

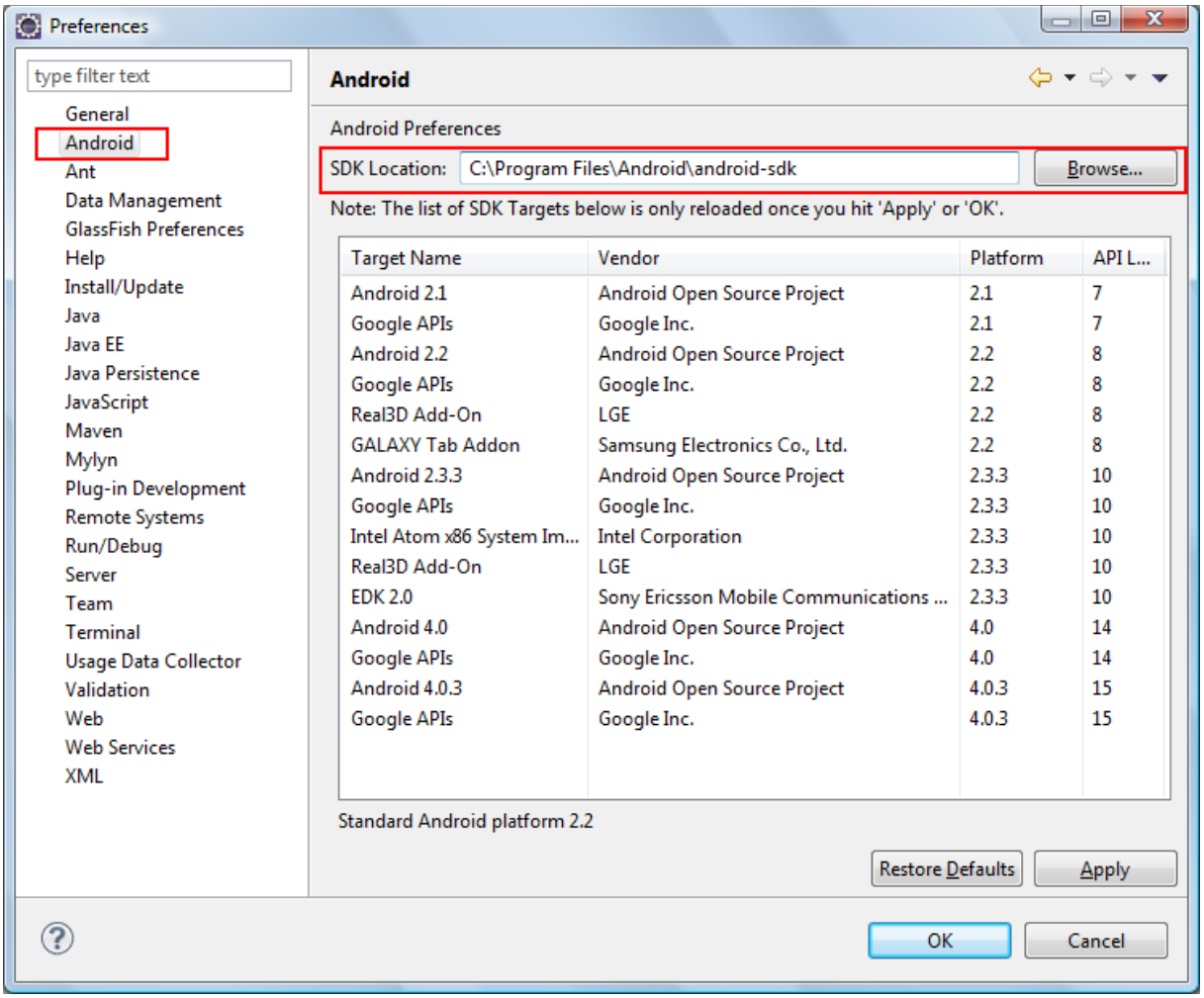
5. Το μόνο που μπορεί να μας προβληματίσει κατά τη διάρκεια της υπόλοιπη διαδικασίας είναι κάποιες προειδοποιήσεις ασφαλείας στις οποίες απαντάμε πατώντας OK.

Στην Εικόνα 7-3 βλέπουμε ότι εκτός των εργαλείων ανάπτυξης εγκαθίστανται και οι προοπτικές (perspectives) του Eclipse DDMS, Traceview και Hierarchy Viewer στα οποία έχουμε αναφερθεί πιο πάνω.

7.7 Δημιουργία εικονικής συσκευής Android

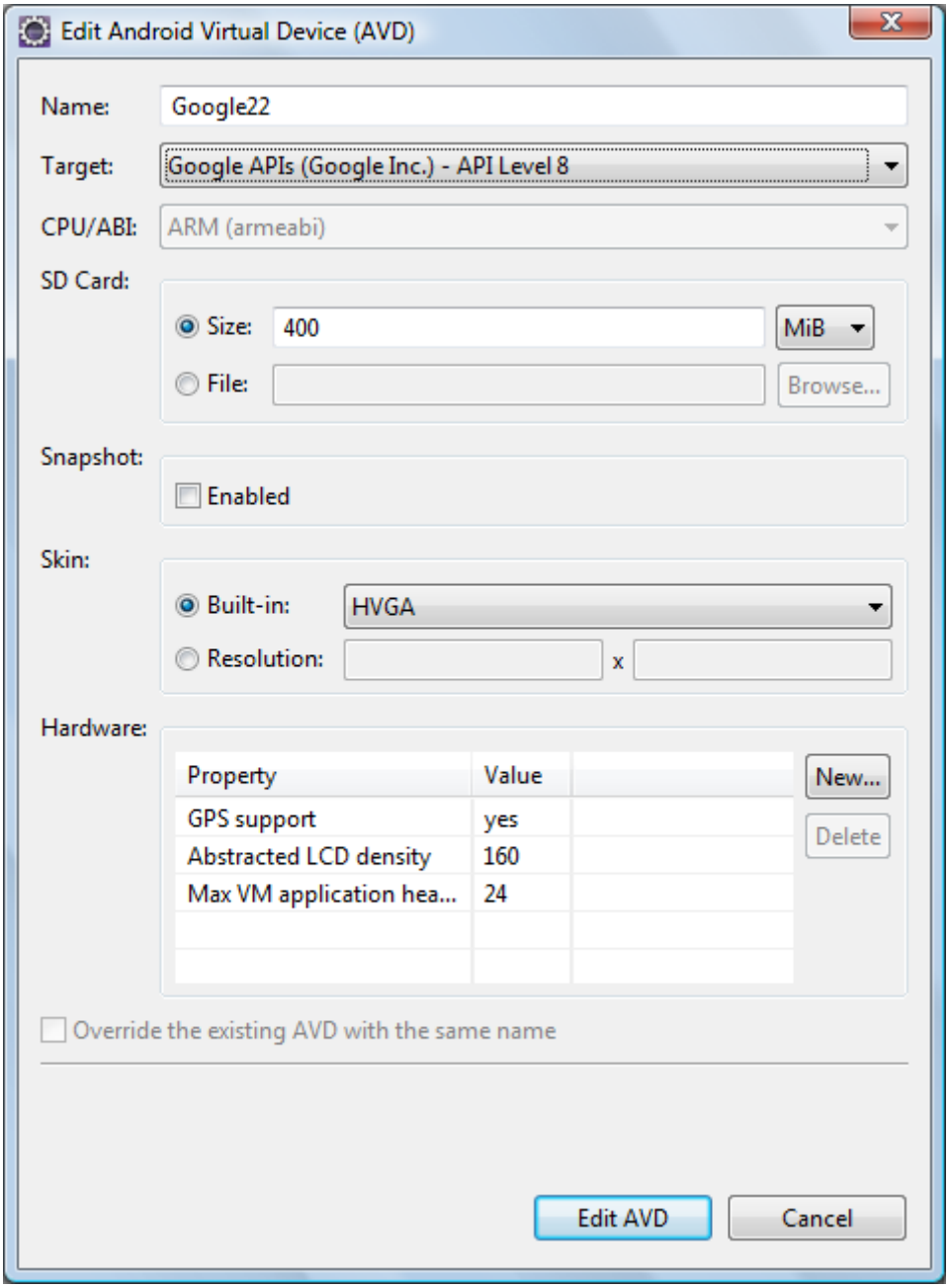
Πριν από τη δημιουργία της εφαρμογής μας, θα δημιουργήσουμε μια ή περισσότερες εικονικές συσκευές Android για να δοκιμάσουμε και εκτελέσουμε το λογισμικό μας. Πριν φτιάξουμε το Android Virtual Device (AVD) θα ενημερώσουμε το Eclipse για την τοποθεσία εγκατάστασης του Android SDK.

Επιλέγουμε *Window >Preferences*, στο παράθυρο που θα ανοίξει πάμε στο Android και εκεί επιλέγουμε την διαδρομή εγκατάστασης του SDK, πατάμε Apply και OK (βλ. Εικόνα 7-4).



Εικόνα 7-4 Τοποθεσία εγκατάστασης του Android SDK

Στη συνέχεια επιλέγουμε *Window > AVD Manager*, πατάμε New και διαλέγουμε τα χαρακτηριστικά που θέλουμε. Επειδή η εφαρμογή μας θα περιέχει και γεωγραφικές πληροφορίες φροντίζουμε στο πεδίο target να διαλέξουμε κάποιο από τα Google APIs (βλ. Εικόνα 7-5).



Εικόνα 7-5 Δημιουργία εικονικής συσκευής Android

Όσο περισσότερα Virtual Devices φτιάξουμε με διαφορετικές πλατφόρμες και διαφορετικές αναλύσεις οθονών, τόσο το καλύτερο για την δοκιμή του λογισμικού μας.

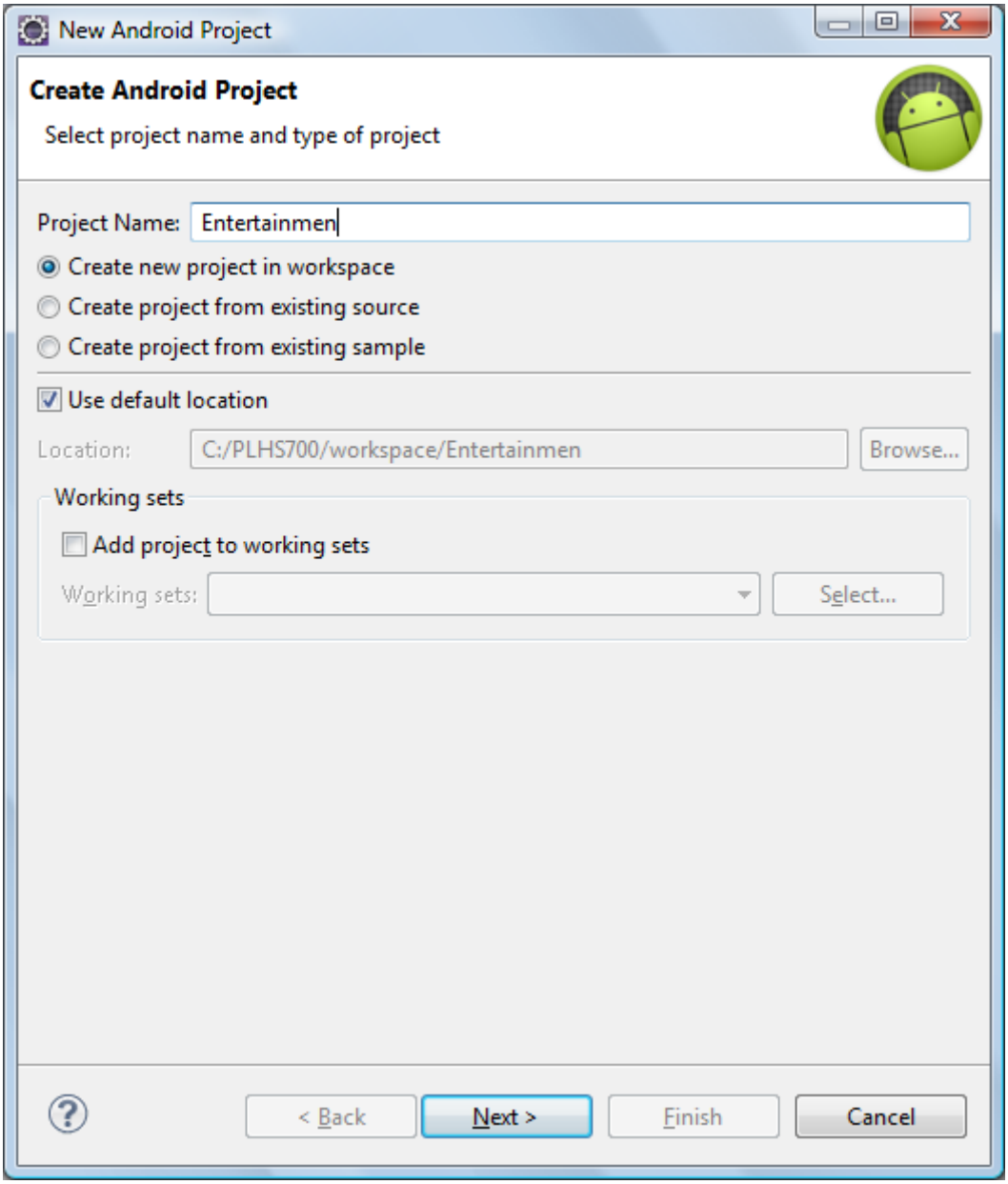
7.8 Δημιουργία και ρύθμιση παραμέτρων ενός νέου έργου Android

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να δημιουργήσουμε ένα νέο έργο στο χώρο εργασίας του Eclipse. Ο οδηγός έργου (wizard) Android δημιουργεί όλα τα απαραίτητα αρχεία για μία εφαρμογή Android. Ακολουθούμε τα παρακάτω βήματα μέσα στο Eclipse, ώστε να δημιουργήσουμε ένα νέο έργο:

1. Επιλέγουμε *File > New > Android Project* ή επιλέγουμε το εικονίδιο δημιουργίας έργου *Android*, το οποίο μοιάζει με φάκελο (με το γράμμα a και ένα σύμβολο συν), στη γραμμή εργαλείων Eclipse.

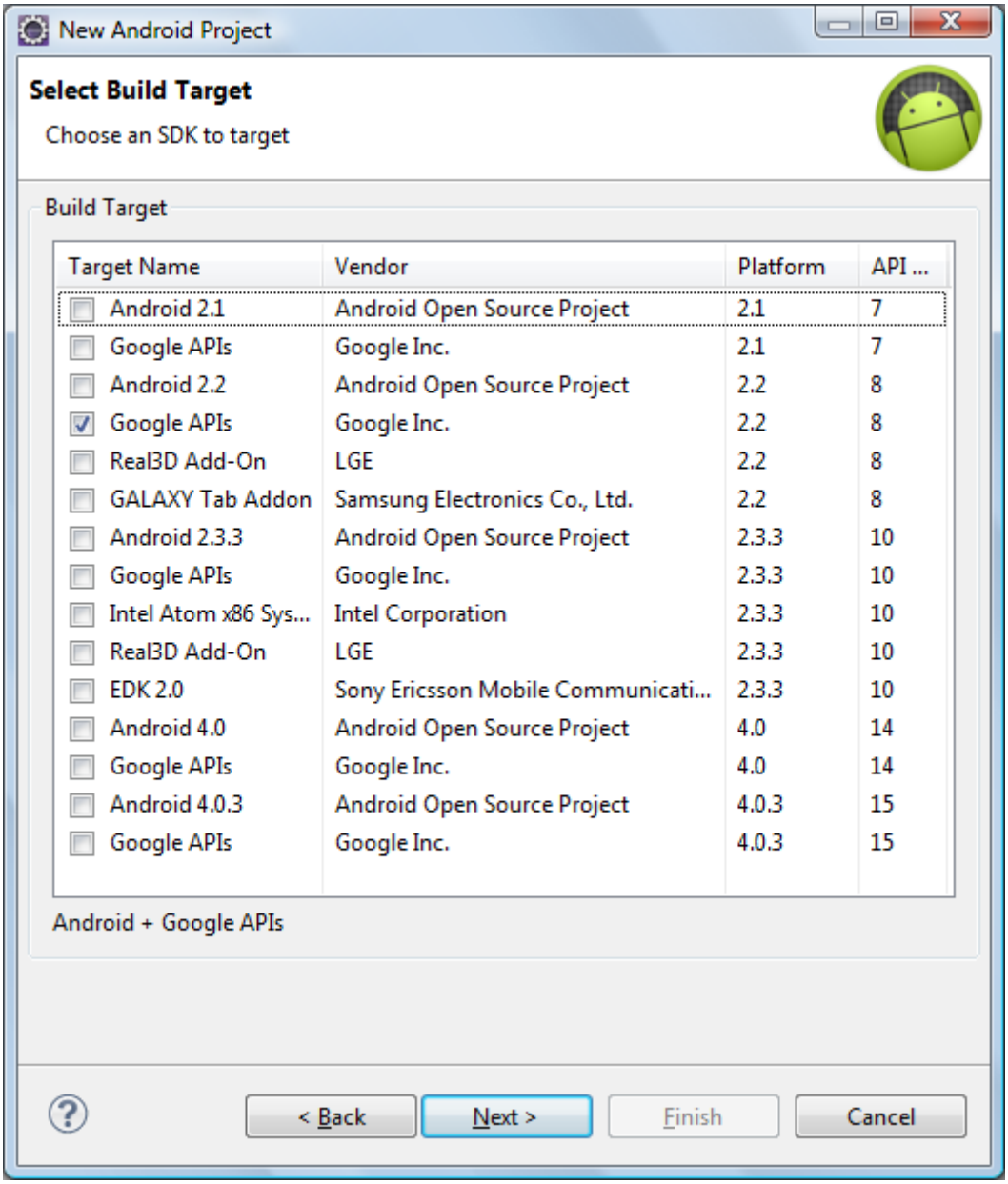
2. Επιλέγουμε ένα όνομα για το έργο (Project Name). Εμείς χρησιμοποιήσαμε το όνομα Entertainment .

3. Επιλέγουμε μία θέση (Location) για τα αρχεία του έργου. Επειδή πρόκειται για ένα νέο έργο, επιλέξτε το κουμπί (radio button) Create New Project in Workspace (Δημιουργία νέου έργου στο χώρο εργασίας). Επιλέξτε το πλαίσιο ελέγχου (Checkbox) Use Default Location (Χρήση προεπιλεγμένης θέσης) ή αλλάζουμε τον κατάλογο με τον κατάλογο όπου θέλουμε να αποθηκεύονται τα αρχεία προέλευσης (βλ. Εικόνα 7-6).



Εικόνα 7-6 Δημιουργία ενός Android Project

4. Επιλέγουμε ένα στόχο κατασκευής για την εφαρμογή μας. Επιλέγουμε έναν στόχο, ο οποίος είναι συμβατός με τις συσκευές Android στις οποίες θέλουμε να τρέχει η εφαρμογή μας. Εμείς επιλέξαμε ως στόχο το Google APIs 2.2 (βλ. Εικόνα 7-7).



Εικόνα 7-7 Επιλογή στόχου του έργου Android

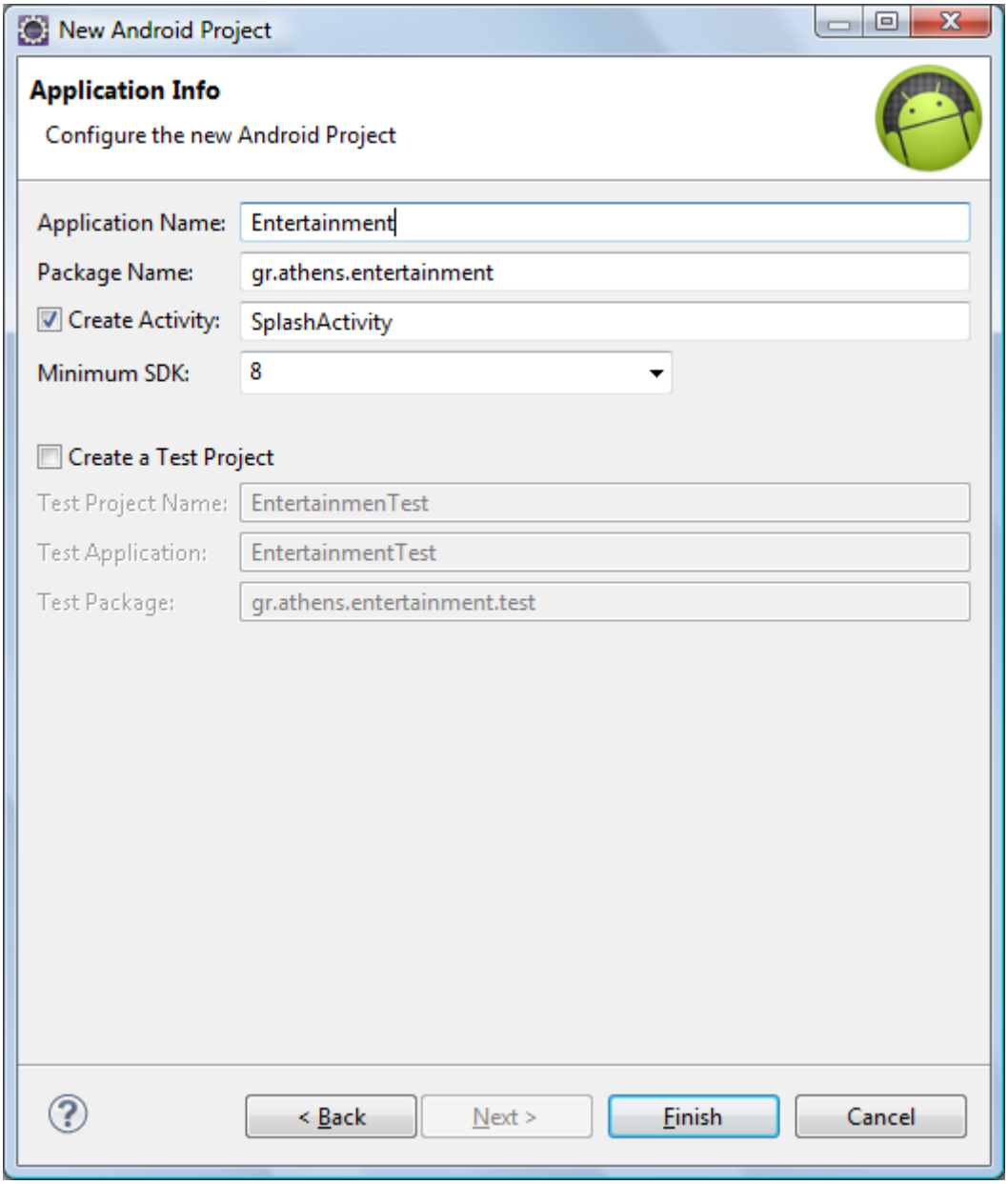
5. Επιλέγουμε ένα όνομα εφαρμογής. Το όνομα εφαρμογής είναι το «φιλικό» όνομα της εφαρμογής και το όνομα που εμφανίζεται με το εικονίδιο στην εκκίνηση της εφαρμογής. Σ' αυτήν την περίπτωση, το όνομα της εφαρμογής είναι "Ψυχαγωγία στην Αθήνα".

6. Επιλέγουμε ένα όνομα πακέτου. Επιλέξαμε το όνομα "gr.athens.entertainment".

7. Επιλέγουμε το πλαίσιο ελέγχου Create Activity (Δημιουργία δραστηριότητας). Δίνουμε έτσι στον οδηγό την εντολή να δημιουργήσει μία προεπιλεγμένη δραστηριότητα

εκκίνησης για την εφαρμογή. Επιλέξαμε για αυτή την κλάση Activity το όνομα SplashActivity

8. Τέλος, κάναμε κλικ στο κουμπί Finish (Τέλος) βλ. Εικόνα 7-8.



Εικόνα 7-8 Ρυθμίσεις του έργου Android

7.9 Σημαντικά αρχεία έργου και κατάλογοι Android

AndroidManifest.xml: Καθολικό αρχείο περιγραφής εφαρμογής. Καθορίζει τις δυνατότητες και τα δικαιώματα της εφαρμογής μας και πώς λειτουργεί.

project.properties: Αρχείο έργου που δημιουργείται αυτόματα. Καθορίζει το στόχο κατασκευής της εφαρμογής μας και άλλες επιλογές συστήματος, όπως απαιτείται.

Φάκελος src : Υποχρεωτικός φάκελος όπου βρίσκεται όλος ο πηγαίος κώδικας για την εφαρμογή.

/Entertainment/src/gr/athens/entertainment/SplashActivity.java Βασικό πηγαίο αρχείο το οποίο καθορίζει το σημείο εισόδου της εφαρμογής. Είναι η δραστηριότητα (activity) που ορίσαμε κατά την δημιουργία του έργου (project).

Φάκελος gen : Υποχρεωτικός φάκελος, όπου βρίσκονται τα αρχεία πόρων για την εφαρμογή τα οποία παράγονται αυτόματα.

/Entertainment/gen/gr/athens/entertainment/R.java: Αρχείο προέλευσης για τη διαχείριση των πόρων εφαρμογών το οποίο παράγεται αυτόματα. Δεν πρέπει να γίνονται τροποποιήσεις σ' αυτό. Αν η εφαρμογή μας «κολλήσει» μπορούμε να δοκιμάσουμε να το διαγράψουμε. Θα ξαναδημιουργηθεί αυτόματα στο επόμενο compilation.

Φάκελος res Υποχρεωτικός φάκελος όπου γίνεται η διαχείριση όλων των πόρων εφαρμογών. Οι πόροι εφαρμογών περιλαμβάνουν animations, εικόνες, αρχεία διάταξης, αρχεία XML, συμβολοσειρές (strings) και ακατέργαστα αρχεία (raw). Κάθε αρχείο του φακέλου παίρνει μια ταυτότητα που είναι προσβάσιμη μέσω του R.id κατά τη διάρκεια συγγραφής του κώδικα.

Entertainment/res/drawable-*/*.png: Φάκελοι πόρων όπου αποθηκεύονται διαφορετικές αναλύσεις των εικονιδίων της εφαρμογής.

/Entertainment/res/layout/main.xml: Αρχείο διάταξης της αρχικής οθόνης.

/Entertainment/res/values/strings.xml: Πόροι συμβολοσειρών της εφαρμογής.

Φάκελος assets: Φάκελος όπου αποθηκεύονται πόροι της εφαρμογής. Αυτοί οι πόροι είναι δεδομένα (αρχεία, κατάλογοι) των οποίων η διαχείριση δεν θέλουμε να γίνεται μέσω του αρχείου R.java. Είναι ευθύνη του προγραμματιστή να προσπελάσει και να αξιοποιήσει τα αρχεία του καταλόγου. Η προσπέλαση γίνεται με κλήση της μεθόδου getAssets() του αντικειμένου context. Πολλοί προγραμματιστές αναφέρουν ότι προκαλούνται λάθη τύπου IOException αν τα αρχεία του καταλόγου ξεπερνούν σε μέγεθος το 1 MB.

7.10 Activities και Intents

Η διεπαφή χρήστη (user interface) αποτελείται από Activities. Μια δραστηριότητα είναι μια οθόνη της εφαρμογής. Κάθε Activity περιλαμβάνει διάφορα Views. Η Activity είναι το αντίστοιχο μιας φόρμας σε μια εφαρμογή windows.

Το layout σχεδιάζεται χρησιμοποιώντας xml. Το xml της activity σύνδεσης με το διακομιστή φαίνεται παρακάτω:

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/username"/>

    <EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/username2">

        <requestFocus />

    </EditText>

    <TextView
```

```
        android:id="@+id/textView2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/password" />

    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/password2" />

    <CheckBox
        android:id="@+id/rememberme"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rememberme" />

    <Button
        android:id="@+id/okbutton"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="OKClick"
        android:text="@string/login" />
          <!-- Link to Registration Screen -->
          <TextView android:id="@+id/link_to_register"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dip"
                android:text="@string/link_to_register"
                android:gravity="center"
                android:textSize="20dip"
                android:textColor="#0b84aa"/>

</LinearLayout>
```

Όπως μπορούμε να δούμε η activity περιέχει ένα LinearLayout που περιλαμβάνει views τύπου TextView, EditText, CheckBox και Button. Οι συμβολοσειρές έχουν επίσης αποθηκευτεί στο αρχείο /res/values/strings.xml για καλύτερη διαχείριση.

Η πλοήγηση από οθόνη σε οθόνη γίνεται με Intents. Για να πλοηγηθούμε ανάμεσα στις διάφορες δραστηριότητες καλούμε την μέθοδο startActivity(intent). Ένα Intent μπορεί να περιέχει πληροφορίες χρήσιμες για την αρχικοποίηση μιας δραστηριότητας. Αυτές αποθηκεύονται στο Intent με την μέθοδο intent.putExtra(name, value) και ανακαλούνται με τη μέθοδο getIntent().getExtras().

7.11 Το αρχείο AndroidManifest.xml και οι ρυθμίσεις του.

Είναι το βασικότερο αρχείο της εφαρμογής. Όπως όλα τα αρχεία xml περιέχει elements και attributes κάποια από τα οποία περιγράφουμε παρακάτω:

Το root element ονομάζεται <manifest>. Περιλαμβάνει όλα τα άλλα elements.

- <uses-sdk> εδώ ορίζουμε την ελάχιστη version του android sdk που χρησιμοποιούμε π.χ. <uses-sdk android:minSdkVersion="8" />

- <uses-permission> Εδώ ορίζουμε τα δικαιώματα που χρειάζεται η εφαρμογή μας για να εκτελεστεί. Εμείς ζητάμε δικαιώματα για android.permission.INTERNET, android.permission.ACCESS_FINE_LOCATION και, android.permission.CALL_PHONE.

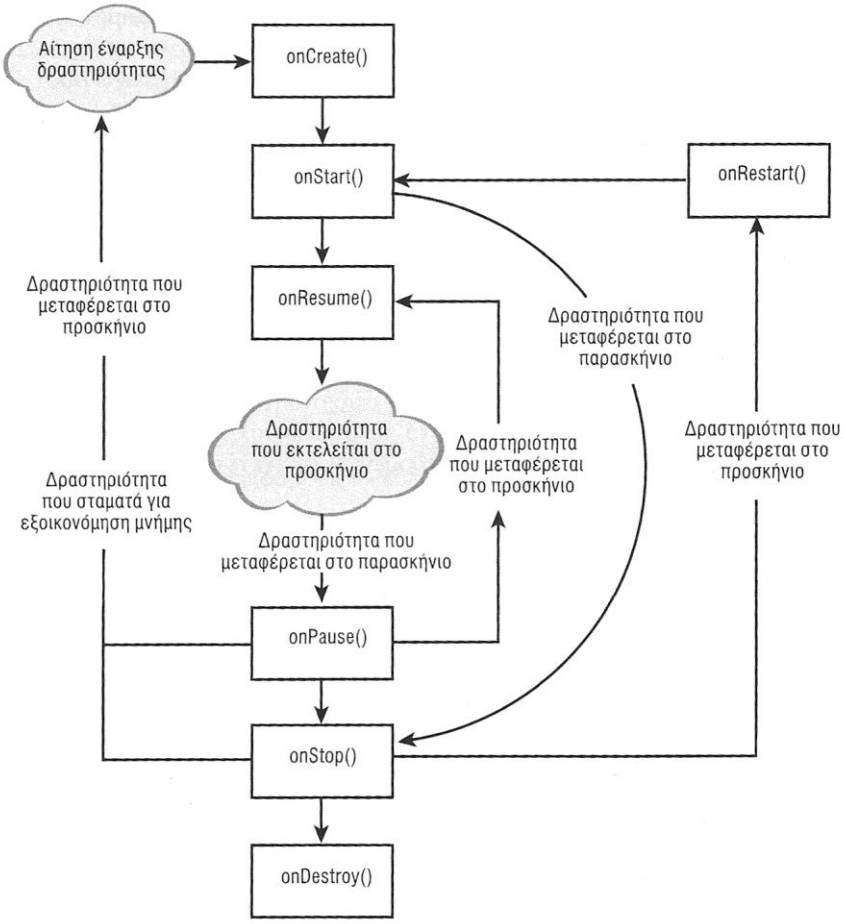
- Το element <application> περιέχει το όνομα της εφαρμογής και τα elements:

    - <uses-library> Εδώ μπαίνουν όλες οι βιβλιοθήκες που χρησιμοποιεί η εφαρμογή μας πχ. com.google.android.maps.

    - <activity> Εδώ πρέπει να δηλωθούν όλα τα activities της εφαρμογής μας. Π.χ. <activity android:name=".base.MyWebViewActivity" />. Αν κάποια activity δεν δηλωθεί προκαλείται σφάλμα.

7.12 Ο κύκλος ζωής μιας activity

Όπως θα δούμε παρακάτω η γνώση του κύκλου ζωής μιας δραστηριότητας είναι σημαντικός για τον προγραμματισμό εφαρμογών Android. Ο κύκλος ζωής μιας activity φαίνεται παραστατικά στην Εικόνα 7-9.



Εικόνα 7-9 Ο κύκλος ζωής μιας activity

Στην onCreate() γίνεται η αρχικοποίηση των στατικών δεδομένων μιας δραστηριότητας. Η μέθοδος onCreate() έχει μία παράμετρο, το Bundle που έχει τιμή null αν η Activity μόλις ξεκίνησε, ενώ άν η Activity σταμάτησε για λόγους απελευθέρωσης μνήμης και τώρα ξαναξεκινά, το Bundle περιέχει τις πληροφορίες της προηγούμενης κατάστασης. Μπορούμε να χρησιμοποιήσουμε τη μέθοδο για τη λήψη δεδομένων και για να καλέσουμε τη μέθοδο setContentView().

Όταν η Activity φτάσει στην κορυφή της στοίβας δραστηριοτήτων, γίνεται η διεργασία προσκηνίου και καλείται η μέθοδος onResume(). Η μέθοδος είναι το κατάλληλο σημείο για έναρξη ήχου, βίντεο και animations.

Όταν η ενεργή Activity πρόκειται να μεταφερθεί πιο κάτω στη στοίβα δραστηριοτήτων επειδή μια άλλη Activity μετακινείται στην κορυφή της στοίβας καλείται η μέθοδος

onPause(). Εδώ, η Activity θα πρέπει να διακόψει οποιονδήποτε ήχο, βίντεο ή animations που έχει ξεκινήσει στη μέθοδο onResume(). Ειναι πιθανό η Activity να καταργηθεί μετά την onPause() και οι μέθοδοι onStop() και onDestroy() να μην κληθούν. Εδώ είναι ίσως η τελευταία ευκαιρία να αποθηκεύσουμε όσα δεδομένα δεν έχουν ακόμη αποθηκευτεί. Όμως όσο περισσότεροι πόροι απελευθερωθούν στη μέθοδο onPause(), τόσο λιγότερο πιθανό είναι η Activity να καταργηθεί όσο βρίσκεται στο παρασκήνιο.

Ένα παράδειγμα εφαρμογής των παραπάνω φαίνεται στο τμήμα κώδικα που ακολουθεί:

```
public class GoogleMapsDirectionsActivity extends MapActivity implements
LocationListener {

    /** Καλείται αν η Activity μόλις ξεκίνησε ή άν η Activity σταμάτησε για
λόγους απελευθέρωσης μνήμης και τώρα ξαναξεκινά. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_button);
        //
        locationManager = (LocationManager)
this.getSystemService(LOCATION_SERVICE);
       …
       …
    }

    /** Σταματάμε τον LocationListener. */
    @Override
    protected void onPause() {
        super.onPause();
        locationManager.removeUpdates(this);
    }

    /** Ξεκινάμε τον LocationListener. */
    @Override
    protected void onResume() {
        super.onResume();

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000,
10, this);
    }

}
```

7.13 Η χρήση του TabActivity

Η εφαρμογή χρησιμοποιεί το view TabActivity για να εμφανίσει δύο καρτέλες (tabs) που περιέχουν ένα ListActivity όπου παραθέτουμε σε μορφή λίστας τα σημεία ενδιαφέροντος και ένα MapView για να εμφανίσουμε τα σημεία πάνω στον χάρτη.

Η επικοινωνία με τον διακομιστή και η λήψη των δεδομένων γίνεται στη μέθοδο onCreate(Bundle savedInstanceState) της δραστηριότητας με την κλήση ενός AsyncTask (που θα δούμε πιο κάτω). Έτσι αποφεύγεται η κλήση του διακομιστή από κάθε καρτέλα χωριστά. Τα δεδομένα αποθηκεύονται στο Intent μέσω του οποίου καλούμε τις δυο καρτέλες.

Κάθε καρτέλα επεξεργάζεται και εμφανίζει τα δεδομένα με την κατάλληλη μορφή κατά την εκτέλεση της μεθόδου onStart(). Αυτό είναι αναγκαίο γιατί αν χρησιμοποιήσουμε τη μέθοδο onCreate τα tabs δεν εμφανίζονται σωστά.

Εδώ πρέπει να σημειωθεί ότι στο Android 3.0 (Honeycomb) η Google προτείνει την χρήση των Fragments και το TabActivity έχει ξεπεραστεί (is deprecated). Εμείς δουλέψαμε με το Android 2.2 (Froyo) και χρησιμοποιήσαμε το TabActivity για λόγους συμβατότητας.

Στην Εικόνα 7-21 στη δεύτερη και τρίτη οθόνη βλέπουμε τις δύο καρτέλες, να εμφανίζουν δεδομένα από τους κινηματογράφους.

7.14 Το MapView και το κλειδί για τους χάρτες της Google

Για την εμφάνιση των χαρτών χρησιμοποιούμε το αντικείμενο MapView. Πριν ξεκινήσουμε να το χρησιμοποιούμε όμως πρέπει αποκτήσουμε ένα κλειδί από την Google. Η διαδικασία που ακολουθήσαμε είναι η εξής:[37]

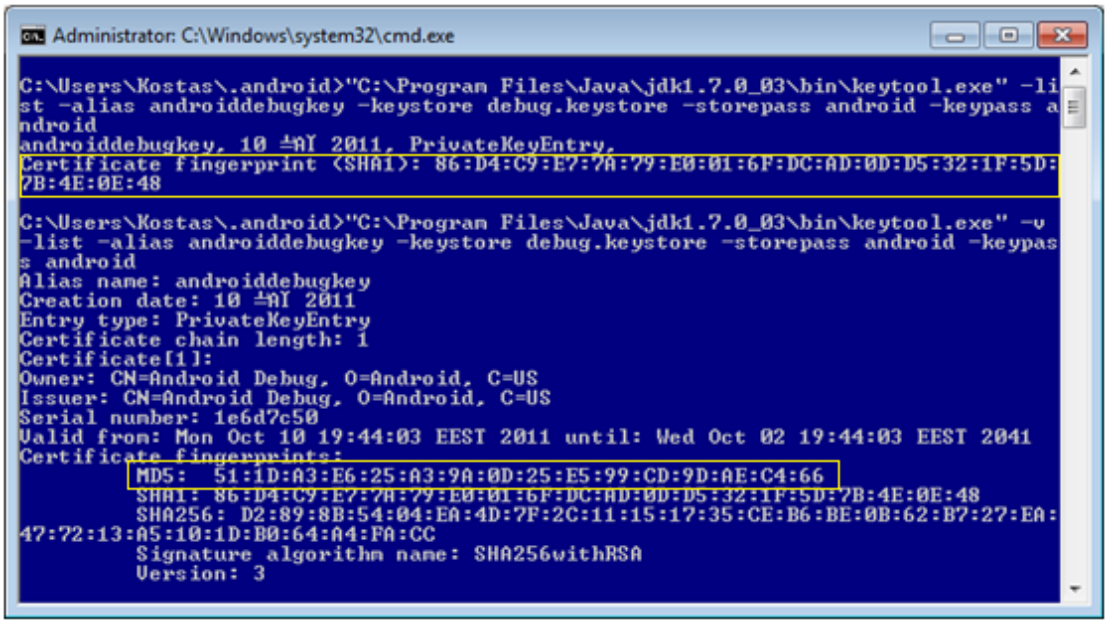
[37] βλ. https://developers.google.com/maps/documentation/android/mapkey

- Μέσα από το Eclipse επιλέγουμε *Windows* > *Preferences* > *Android* > *Build* και αντιγράφουμε το πλήρες path για το αρχείο «debug.keystore» που έχει δημιουργηθεί μόλις φτιάξαμε το πρώτο android project.

- Πηγαίνουμε στον φάκελο που περιέχει το αρχείο και εκτελούμε το ακόλουθο command:

```
keytool -v -list -alias androiddebugkey \
-keystore <path_to_debug_keystore>.keystore \
-storepass android -keypass android
```

Στο σημείο αυτό να σημειώσουμε ότι η εκτέλεση της εντολής χωρίς τον διακόπτη –v επέστρεφε μόνο την τιμή SHA1 certificate fingerprint.



Εικόνα 7-10 Το MD5 certificate fingerprint

- Αντιγράφουμε την τιμή MD5 certificate fingerprint, στην προκειμένη περίπτωση: D9:25:FC:6F:A6:AE:51:E5:DC:6F:3A:C3:80:3D:48:F8.

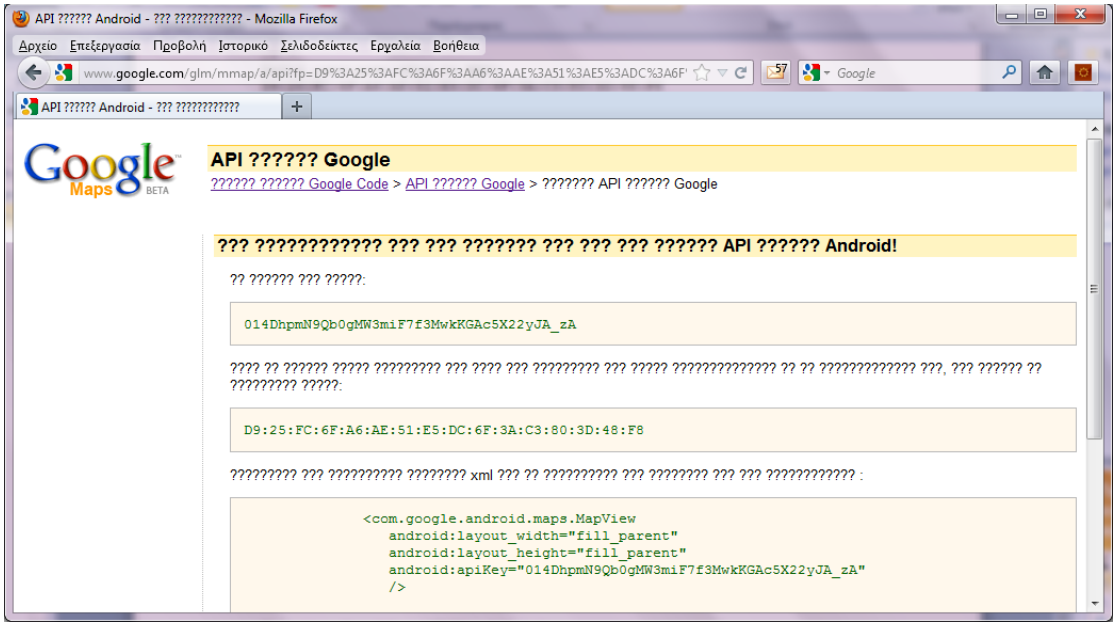
- Κατευθυνόμαστε στην διεύθυνση: http://code.google.com/android/maps-api-signup.html

- Χρησιμοποιούμε ένα υπάρχοντα λογαριασμό Google ή αν δεν έχουμε φτιάχνουμε έναν.

- Διαβάζουμε τους όρους χρήσης της Υπηρεσίας προσεκτικά και τσεκάρουμε το checkbox.

- Επικολλάμε το MD5 certificate fingerprint στο κατάλληλο πεδίο και,

- Επιλέγουμε "Generate API Key".



Εικόνα 7-11 Το κλειδί για τους χάρτες της Google

Στην Εικόνα 7-11 η πρώτη τιμή που διαβάζεται είναι το κλειδί. Η δεύτερη είναι η τιμή MD5 certificate fingerprint Υπάρχουν επίσης οδηγίες μέσα στο πλαίσιο για την χρήση του σε ένα MapView.

Μετά την απόκτηση του κλειδιού θα πρέπει στο αρχείο AndroidManifest.xml να προσθέσουμε το element *<uses-library android:name="com.google.android.maps"/>* σαν παιδί (child) του element *<application>*. Θα πρέπει επίσης να ζητήσουμε να μας επιτραπεί η πρόσβαση στο INTERNET. Αυτό επιτυγχάνεται προσθέτοντας σαν παιδί του element <manifest> το element *<uses-permission android:name =*

"android.permission.INTERNET"/>. Φυσικά, αν πρόκειται για καινούργια δραστηριότητα (Activity) δεν πρέπει να ξεχάσουμε να την δηλώσουμε προσθέτοντας ένα ακόμη παιδί στο element <*application*>. Για παράδειγμα <*activity android:name = ".PoisTabsMapActivity" />.*

Εικόνες δραστηριοτήτων που περιέχουν MapView είναι η δεύτερη οθόνη της Εικόνα 7-20, η τρίτη οθόνη της Εικόνα 7-21 και οι δυο πρώτες οθόνες της Εικόνα 7-24.

7.15 Τα events του MapView.

Το αντικείμενο MapView παρότι εμφανίζει ενσωματωμένα τα ZoomControls δεν μας ενημερώνει όταν ο χρήστης μετακινεί ή αλλάζει την μεγέθυνση του χάρτη. Για να ενημερωθούμε για τυχόν αλλαγές μεγέθους και μετακίνησης του χάρτη πρέπει να ακροαστούμε πότε το αντικείμενο ζωγραφίζει τον εαυτό του. Αυτό μπορούμε να το πετύχουμε δημιουργώντας μια υποκλάση που κάνει extends την κλάση MapView και να κάνουμε override την μέθοδο dispatchDraw.Για το σκοπό αυτό δημιουργήσαμε την κλάση MyMapView που παραθέτουμε στο παράρτημα.

7.16 Διατάξεις (Layouts)

Το layout (διάταξη) είναι η αρχιτεκτονική που χρησιμοποιεί το Android για να δημιουργήσει τη διεπαφή χρήστη. Η διάταξη μπορεί να οριστεί είτε σε αρχεία XML, είτε προγραμματιστικά κατά το χρόνο εκτέλεσης. Η Google προτείνει την χρήση XML για το διαχωρισμό του κώδικα από τη διεπαφή χρήστη.

Οι πιο συνηθισμένες διατάξεις είναι:

LinearLayout είναι ένα view group που στοιχίζει όλα τα children views, είτε κάθετα, είτε οριζόντια. Μπορούμε να ορίσουμε την κατεύθυνση ορίζοντας την ιδιότητα android:orientation σε vertical ή horizontal.

RelativeLayout είναι ένα view group που εμφανίζει τα views που περιέχει σε σχετικές θέσεις.

WebView εμφανίζει ιστοσελίδες**.**

Όταν το περιεχόμενο είναι δυναμικό μπορούμε να χρησιμοποιήσουμε μια διάταξη που κληρονομεί από ένα AdapterView για να δημιουργήσουμε τη διάταξη κατά το χρόνο εκτέλεσης. Μια τέτοια κλάση χρησιμοποιεί έναν Adapter για να ελέγξει την παρουσίαση των δεδομένων. Ο Adapter μεσολαβεί ανάμεσα στην πηγή των δεδομένων και στη διάταξη μετατρέποντας τα δεδομένα σε views που προστίθενται στην διάταξη.

Διατάξεις που χρησιμοποιούν Adapters είναι τα **ListView** και **GridView**. Περισσότερα μπορεί κανείς να δει στην ιστοσελίδα με τίτλο «Layouts» του ιστοτόπου developer.android.com[38]

7.17 Ασύγχρονη επεξεργασία

Για να έχουν οι εφαρμογές καλή απόκριση δεν θα πρέπει οι χρονοβόρες ενέργειες, όπως η κλήση web services να μπλοκάρουν το κύριο νήμα της διεπαφής χρήστη (για συντομία κύριο νήμα). Επεξεργασία μεγάλων ποσοτήτων δεδομένων ή μεγάλη ποσότητα επεξεργασίας θα πρέπει να μεταφέρονται έξω απ' το κύριο νήμα.

Το Android SDK παρέχει δύο εύκολους τρόπους ελάττωσης της επιβάρυνσης στην επεξεργασία απ' το κύριο νήμα. Την τυπική Java κλάση thread και την κλάση AsyncTask.

7.17.1 Η κλάση AsyncTask

Η κλάση AsyncTask είναι μία ειδική κλάση για ανάπτυξη με το Android η οποία παρέχει επεξεργασία στο παρασκήνιο και διευκολύνει την επικοινωνία με το κύριο νήμα, ενώ η διαχείριση του κύκλου ζωής της γίνεται μέσα στο πλαίσιο του κύκλου ζωής της δραστηριότητας (activity).

Το AsyncTask είναι μία αφηρημένη (abstract class) βοηθητική κλάση για τη διαχείριση ενεργειών παρασκηνίου οι οποίες τελικά επιστρέφουν στο κύριο νήμα. Αντί να δημιουργήσουμε νήματα, όπως θα περιγράψουμε πιο κάτω, μπορούμε να

[38] http://developer.android.com/guide/topics/ui/declaring-layout.html

δημιουργήσουμε μία υποκλάση της κλάσης AsyncTask και να υλοποιήσουμε τις κατάλληλες μεθόδους.

Η μέθοδος *onPreExecute()* εκτελείται στο κύριο νήμα πριν ξεκινήσει η επεξεργασία παρασκηνίου και είναι κατάλληλη για την αρχικοποίηση της επεξεργασίας. Η μέθοδος *doInBackground()* χειρίζεται την επεξεργασία παρασκηνίου, ενώ η *publishProgress()* ενημερώνει το κύριο νήμα κατά διαστήματα σχετικά με την πρόοδο της επεξεργασίας. Όταν η επεξεργασία ολοκληρωθεί, η μέθοδος *onPostExecute()* εκτελείται στο κύριο νήμα, και προσφέρεται για την τελική ενημέρωση του νήματος.

Ασύγχρονη επεξεργασία με χρήση της κλάσης AsyncTask χρησιμοποιούμε σε κάθε επικοινωνία με το διακομιστή.

7.17.2 Threads και Handlers

Το Android επίσης, για να επιτύχει ασύγχρονη επεξεργασία, υποστηρίζει την χρήση Threads. Υποστηρίζει το *package java.util.concurrent* για την εκτέλεση εργασιών στο παρασκήνιο με την χρήση των κλάσεων *ThreadPool* και *Executor*. Αν χρειάζεται να ενημερώσουμε ή να συγχρονίσουμε το κύριο νήμα μπορούμε να χρησιμοποιήσουμε είτε την κλάση AsyncTask που αναφέραμε είτε την κλάση android.os.Handler.

Η βοηθητική κλάση Handler μπορεί να ενημερώνει το κύριο νήμα. Ένας Handler υποστηρίζει μεθόδους που δέχονται αντικείμενα των κλάσεων Message ή Runnable. Μπορούμε να κάνουμε override την μέθοδο handleMessage() για να επεξεργαστούμε τα μηνύματα ή να χρησιμοποιήσουμε την μέθοδο post() για να στείλουμε ένα Runnable στην εκτελούμενη δραστηριότητα. Το νήμα μπορεί επίσης να στείλει μηνύματα με την χρήση της μεθόδου sendMessage(Message msg) ή μέσω της sendEmptyMessage(). Χρειάζεται μόνο να ορίσουμε έναν Handler στην εκτελούμενη δραστηριότητα και να τον χρησιμοποιήσουμε.

Την τεχνική αυτή χρησιμοποιήσαμε για την λήψη πληροφοριών από το Google Directions API.

7.18 Google Directions API

Για να δρομολογήσουμε τον χρήστη της υπηρεσίας χρησιμοποιούμε τα web services του Google. Μια αίτηση προς τα web services έχει την μορφή:

```
http://maps.googleapis.com/maps/api/directions/output?parameters
```

Η παράμετρος output μπορεί να πάρει μία από τις μορφές:

- **json** που συνιστάται από την Google και μας επιστρέφει την πληροφορία σε JavaScript Object Notation μορφή και

- **xml** και μας επιστρέφει την πληροφορία σε XML

Οι παράμετροι (parameters) μπορεί να είναι υποχρεωτικές ή προαιρετικές. Υποχρεωτικές είναι οι παράμετροι:

- **origin:** Η αρχική διεύθυνση απ' όπου θέλουμε να υπολογίσουμε το δρομολόγιο. Μπορεί να είναι είτε συμβολοσειρά που είναι μια διεύθυνση, είτε οι συντεταγμένες του σημείου. Αν είναι συμβολοσειρά το web service θα το μετατρέψει σε συντεταγμένες.

- **destination:** Η τελική διεύθυνση προς την οποία θέλουμε να υπολογίσουμε το δρομολόγιο. Ισχύει ότι και πριν. Μπορεί δηλ. να είναι διεύθυνση ή συντεταγμένες.

- **sensor:** Δηλώνει εάν η αίτηση προέρχεται από τη συσκευή με έναν αισθητήρα θέσης. Πρέπει να είναι είτε true είτε false.

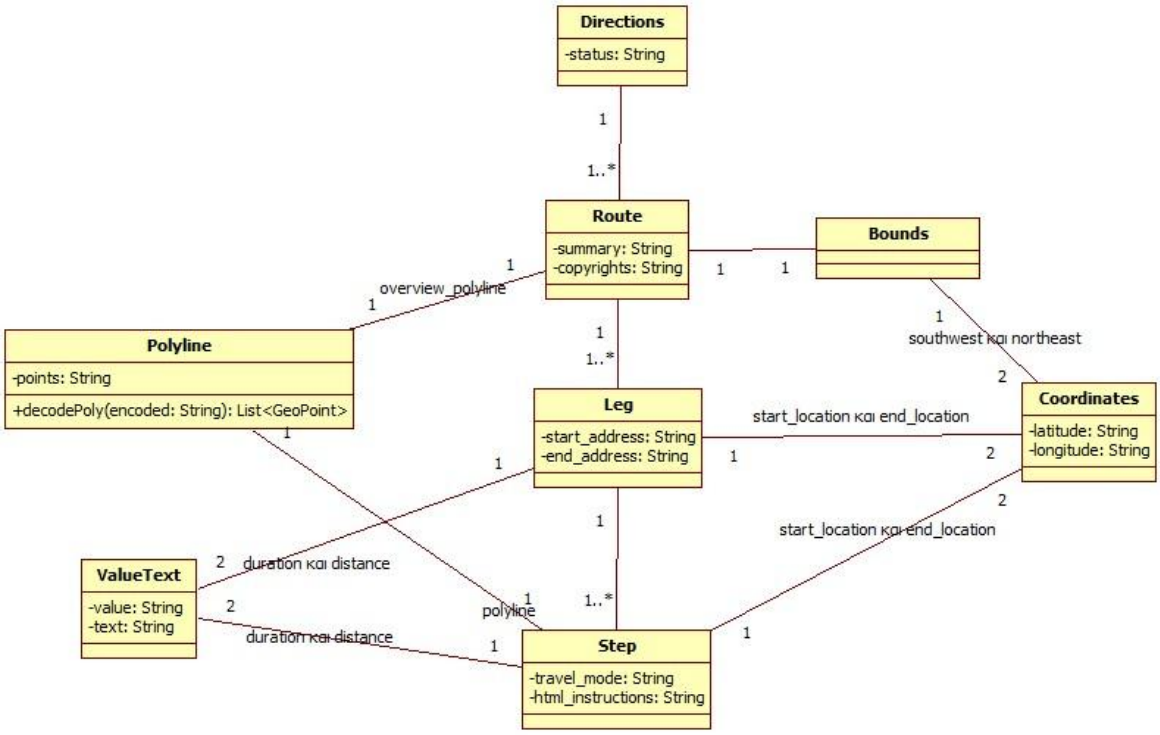
Από τις προαιρετικές παραμέτρους ενδιαφέρον παρουσιάζουν οι:

mode: Δηλώνει το μέσο μεταφοράς που θα χρησιμοποιήσουμε. Μπορεί να πάρει μία από τις τιμές: driving, walking, bicycling, transit. Δηλαδή, είτε οδηγώντας, που είναι και η προεπιλεγμένη τιμή (default), είτε περπατώντας είτε με ποδήλατο είτε χρησιμοποιώντας τα μέσα μαζικής μεταφοράς.

region: Ο κωδικός της περιοχής. Για την εφαρμογή έχει την τιμή gr.

Παρότι δεν αναφέρεται στις οδηγίες του web service της Google, μία ακόμη παράμετρος που μπορεί να χρησιμοποιηθεί είναι η **language.** Για την εφαρμογή μας μπορεί να πάρει τις τιμές **el** (Ελληνικά) ή **en** (Αγγλικά). Επηρεάζει τη γλώσσα περιγραφής των οδηγιών δρομολόγησης. Η δεύτερη δουλεύει καλύτερα με την ενσωματωμένη εφαρμογή text to speech του Android. Η λειτουργία αυτή είναι διαθέσιμη στα αγγλικά δωρεάν, ενώ για τα ελληνικά πρέπει να αναζητήσει κανείς εφαρμογές στο Google Play.

Αναλύοντας το web service καταλήξαμε ότι μπορεί να αναπαρασταθεί από το πιο κάτω διάγραμμα κλάσεων (Εικόνα 7-12).



Εικόνα 7-12 Οι κλάσεις του web service της Google

Τις κλάσεις του διαγράμματος θα χρησιμοποιήσουμε για να μετατρέψουμε σε αντικείμενα τις μορφές XML ή JSON που μας επιστρέφει το service. Η διαδικασία αυτή είναι γνωστή ως deserialization, unmarshalling ή inflating. Πώς γίνεται αυτό θα το δούμε σε επόμενη παράγραφο.

Το web service επιστρέφει τα points της κλάσης Polyline κωδικοποιημένα. Μια τέτοια κωδικοποιημένη συμβολοσειρά μπορούμε να δούμε στην Εικόνα 7-13:

```
<overview_polyline>
<points>
a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\|tNtqf@l{Yd_Fblh@rxo@b}@xxSfytAblk@xxaBeJxlcBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@
vrSnc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznmFfwMbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\~mbDzeVh_W
r|Efc\x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiA
p|gEicy@`omFvaErfo@igQxnlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hL
xiZd~XtaBndgC</points>
</overview_polyline>
```

Εικόνα 7-13 κωδικοποιημένα Polyline

Ο τρόπος αποκωδικοποίησης υπάρχει στην διεύθυνση http://stackoverflow.com/questions/2964982/android-get-and-parse-google-directions και φαίνεται πως έχει προέρθει από reverse engineering αντίστοιχου προγράμματος της Google σε Javascript.

Πληροφορίες δρομολόγησης μπορούμε να δούμε στην Εικόνα 7-24 στη δεύτερη και τρίτη οθόνη.

7.19 Εναλλακτική μέθοδος δρομολόγησης και εμφάνισης χάρτη.

Μια απλούστερη εναλλακτική λύση για την δρομολόγηση η οποία δεν προκρίθηκε, είναι το άνοιγμα του φυλλομετρητή με φορτωμένο τον ιστοχώρο maps της Google να εμφανίζει την δρομολόγηση από το σημείο source στο σημείο destination:

```
    Intent intent = new Intent(Intent.ACTION_VIEW,
        Uri.parse("http://maps.google.com/maps?saddr="
            + source + "&daddr=" + destination));

//https://maps.google.com/maps?saddr=37.992200,23.656970&daddr=37.947028,23
.645029

    startActivity(intent);
```

όπου π.χ. είναι source="37.992200,23.656970" και destination="37.947028,23.645029".

Με την ευκαιρία να αναφέρουμε ότι παρόμοια είναι και η εμφάνιση στον ιστοχώρο maps της Google ενός σημείου:

```
    Intent intent = new Intent(Intent.ACTION_VIEW,
        Uri.parse("https://maps.google.com/maps?q=" + poi + "&z=15"));

//https://maps.google.com/maps?q=37.992200,23.656970&z=15

    startActivity(intent);
```

όπου π.χ. είναι poi="37.992200,23.656970".

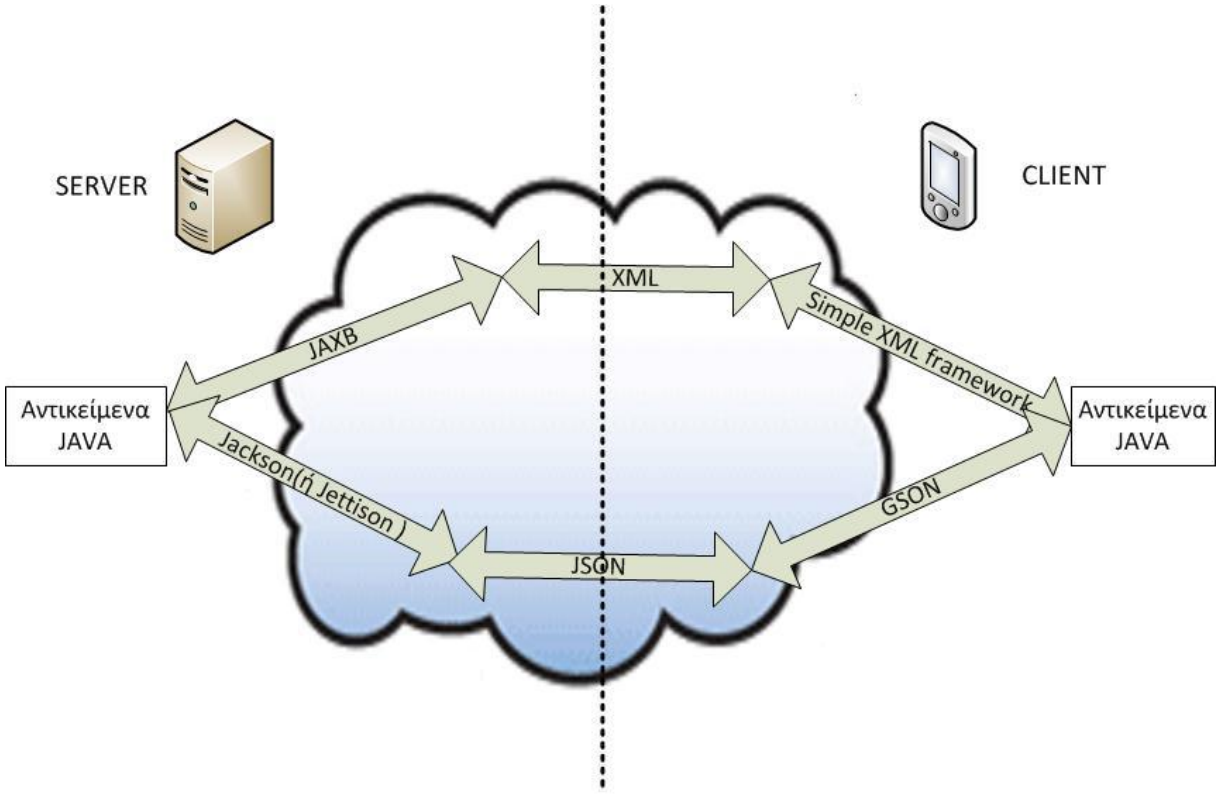
7.20 Εξωτερικές βιβλιοθήκες

Για την επικοινωνία μεταξύ διακομιστή και κινητού android χρησιμοποιήσαμε τις βιβλιοθήκες

1. Simple XML framework (http://simple.sourceforge.net/) και

2. Gson (http://code.google.com/p/google-gson/)

Μια οπτική σχέση των βιβλιοθηκών αυτών και των βιβλιοθηκών που χρησιμοποιούμε στην πλευρά του διακομιστή φαίνεται παρακάτω (Εικόνα 7-14):



Εικόνα 7-14 Σχέσεις μεταξύ βιβλιοθηκών

Η χρήση των βιβλιοθηκών δεν είναι αναγκαία. Η χρήση τους όμως, απλοποιεί τον κώδικα και κυρίως μας επιτρέπει να σκεφτόμαστε με αντικείμενα.

Αρχικά και οι δύο έρχονται με την άδεια Apache License, Version 2.0 που σημαίνει πως μπορούμε να τις χρησιμοποιήσουμε και σε επαγγελματικές εφαρμογές. Και οι δύο έχουν δοκιμαστεί και δουλεύουν σε περιβάλλον Android, συνεργάζονται δηλ. καλά με την Dalvik VM. Σημαντικό ρόλο έπαιξε επίσης το γεγονός ότι καταφέραμε να τις κάνουμε να δουλέψουν με λίγες αλλαγές στον κώδικα.

Ο κώδικας που θα παρατεθεί χρησιμοποιείται για την κλήση των Google Directions APIs και επιστρέφει την δρομολόγηση από ένα σημείο σε ένα άλλο χρησιμοποιώντας τις κλάσεις στις οποίες αναφερθήκαμε μιλώντας για το Google Directions API (βλ. Εικόνα 7-12).

```
01 public abstract class DirectionsProvider {
02
```

```
03  protected String dataType = "json";
04
05  abstract public Directions getDirections(String url)  throws Exception;
06
07  public String readData() {
08      HttpClient httpClient = new DefaultHttpClient();
09
10      HttpHost httpHost = new HttpHost("maps.googleapis.com", 80,
"http");
11      String targetURL = "/maps/api/directions/" + dataType;
12
13      ...
14
15      return result;
16  }
17
18 }
```

Όπως μπορούμε να δούμε η κλάση είναι αφηρημένη (abstract) και για να χρησιμοποιηθεί πρέπει να γίνει extends. Επίσης από την τιμή του πεδίου dataType εξαρτάται αν η συμβολοσειρά που επιστρέφει η μέθοδος readData, της οποίας παραθέτουμε ένα μέρος, είναι JSON ή XML (γραμμή 11). Θα την κάνουμε extends με δύο τρόπους όπως φαίνεται πιο κάτω.

```
public class SimpleXMLDirectionsProvider extends DirectionsProvider {

    public SimpleXMLDirectionsProvider(GeoPoint origin, GeoPoint
destination) {
        super(origin, destination);
        this.dataType = "xml";
    }

    @Override
    public Directions getDirections(String data) throws Exception {
        Serializer serializer = new Persister();
        Directions directions =  serializer.read(Directions.class, data,
false);
        return directions;
    }

}

public class GSONDirectionsProvider extends DirectionsProvider {

    public GSONDirectionsProvider(GeoPoint origin, GeoPoint destination) {
        super(origin, destination);
        this.dataType = "json";
    }

    @Override
    public Directions getDirections(String data) throws Exception {
        Gson gson = new Gson();
        Directions directions = gson.fromJson(data, Directions.class);
        return directions;
```

```
    }

}
```

Το παράδειγμα δείχνει ότι αρκούν δύο διαφορετικές γραμμές κώδικα για να επεξεργαστούμε , είτε XML, είτε JSON.

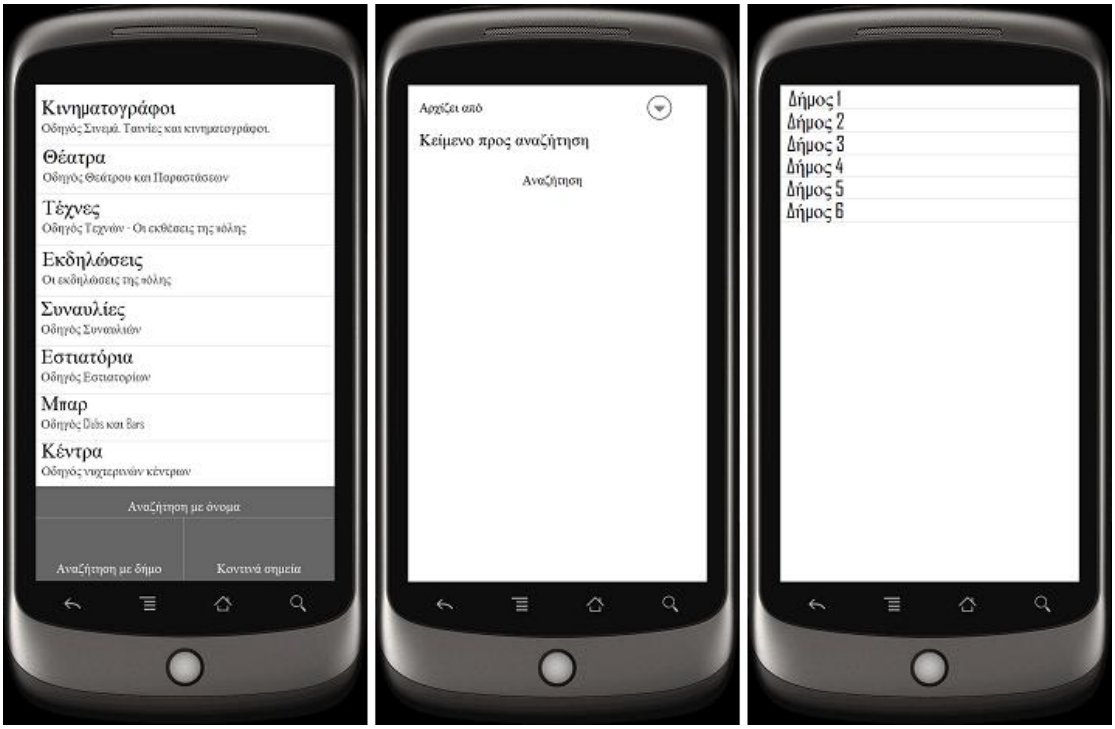
Οι προαναφερόμενες κλάσεις πρέπει να έχουν σημανθεί με τα κατάλληλα annotations. Οι σημάνσεις της κλάσης Directions φαίνονται εδώ:

```
import org.simpleframework.xml.Element;
import org.simpleframework.xml.ElementList;
import com.google.gson.annotations.SerializedName;

public class Directions {
    @SerializedName("status")
    @Element(name = "status")
    private String status;

    @SerializedName("routes")
    @ElementList(name = "route", type = Route.class, inline = true)
    private List<Route> routes;

    ...

}
```

Τα annotations @SerializedName είναι της βιβλιοθήκης gson, ενώ τα @Element και @ElementList της βιβλιοθήκης Simple XML framework (όπως φαίνεται και από τα imports).

7.21 Θέματα σχεδίασης της εφαρμογής

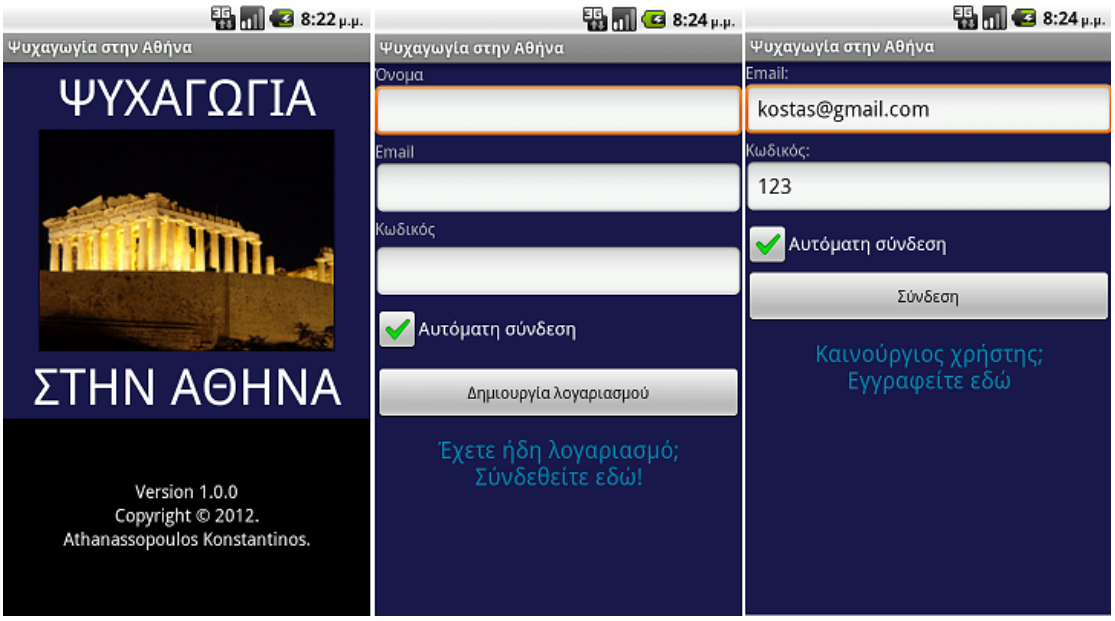
Κατά τη διάρκεια ανάπτυξης της εφαρμογής ανακαλύψαμε ότι αποτελεί καλή πρακτική η πρωτοτυποποίηση των οθονών. Βοηθά στο να αποκτήσουμε μια οπτική αντίληψη του τι θέλουμε να φτιάξουμε και να τακτοποιήσουμε τις ιδέες μας, σχεδιάζοντας την μετάβαση από οθόνη σε οθόνη. Στη συνέχεια σχεδιάζουμε το xml layout των activities και στη συνέχεια προσθέτουμε τον κώδικα. Ένα παράδειγμα οθόνης που έχει κατασκευαστεί με το πρόσθετο Pencil του Firefox βλέπουμε στην Εικόνα 7-15



Εικόνα 7-15 Πρωτοτυποποίηση οθονών της εφαρμογής Android

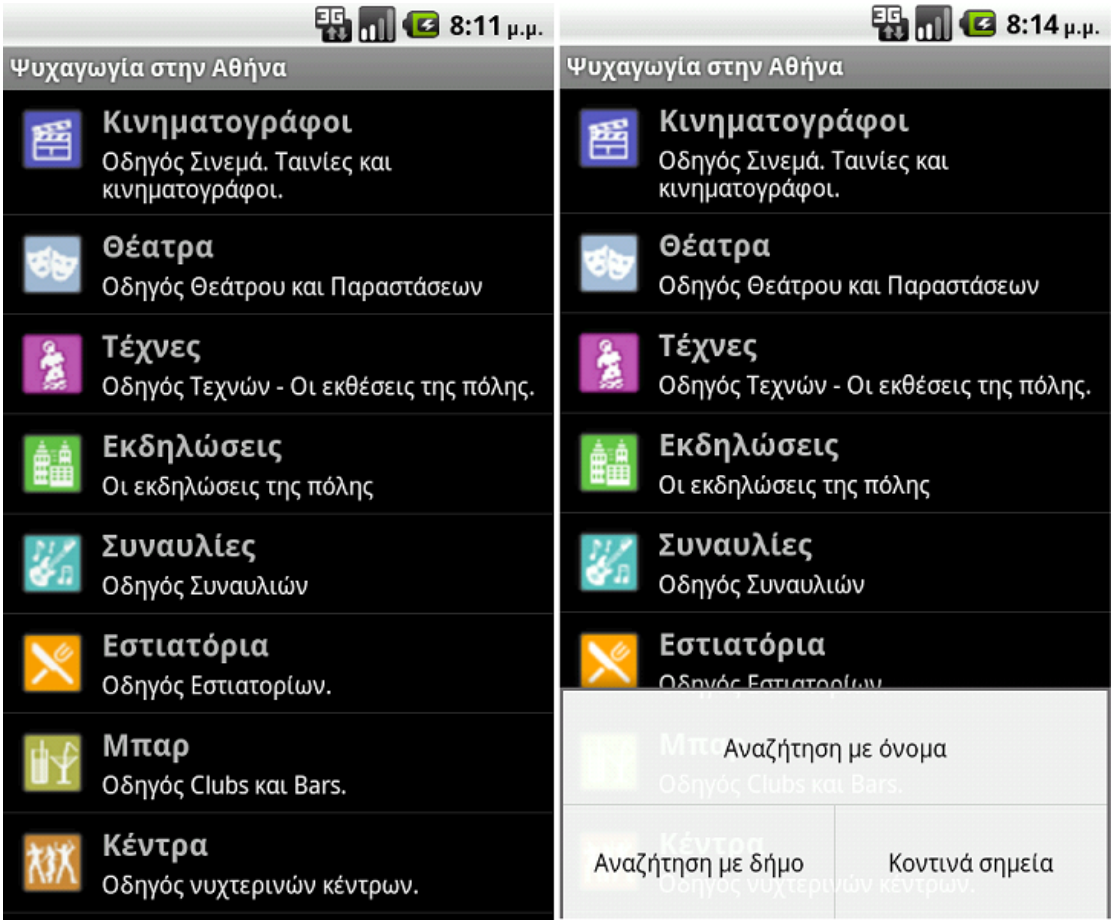
7.22 Οθόνες της εφαρμογής.

Στην Εικόνα 7-16 που ακολουθεί βλέπουμε την αρχική οθόνη της εφαρμογής όπου εμφανίζεται ένα animation. Αν είμαστε νέοι χρήστες ακολουθεί η προτροπή να δημιουργήσουμε ένα νέο λογαριασμό. Αν ο λογαριασμός έχει ήδη δημιουργηθεί εμφανίζεται συμπληρωμένη η οθόνη σύνδεσης όπου αρκεί να πατήσουμε το κουμπί «Σύνδεση».



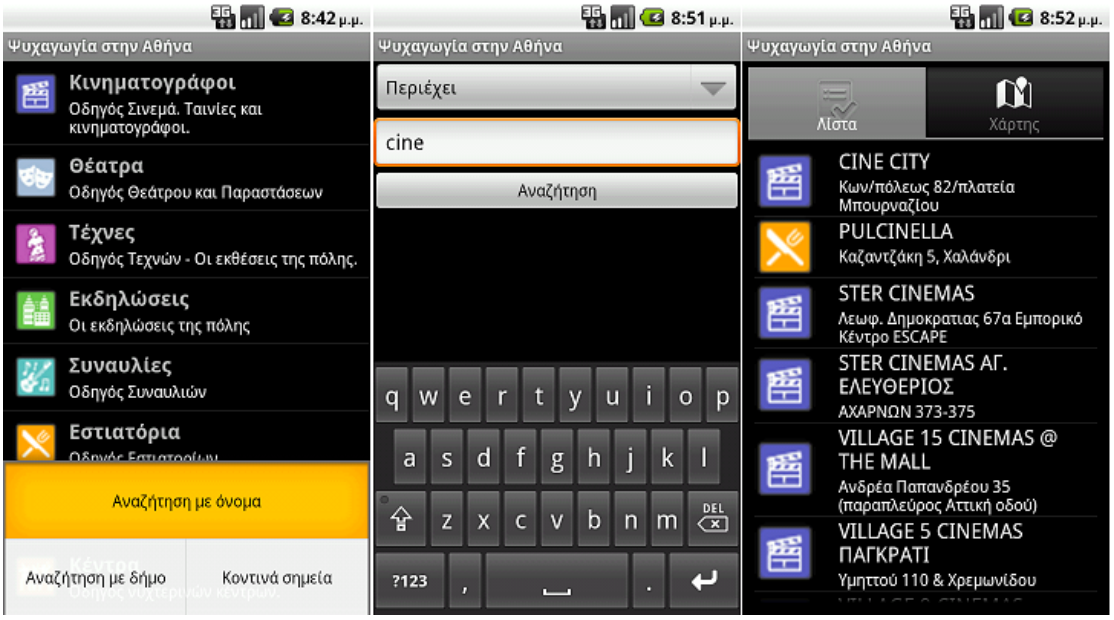
Εικόνα 7-16 Οθόνη 1

Στην Εικόνα 7-17 βλέπουμε την αρχική οθόνη της εφαρμογής. Μπορούμε να διαλέξουμε κατηγορία σημείων ενδιαφέροντος για να πλοηγηθούμε. Αν πατήσουμε το μενού της οθόνης θα μας δοθούν οι επιλογές «Αναζήτηση με όνομα», «Αναζήτηση με δήμο», «Κοντινά σημεία».



Εικόνα 7-17 Οθόνη 2

Στην Εικόνα 7-18 επιλέγουμε «Αναζήτηση με όνομα». Στην δεύτερη οθόνη μπορούμε να διαλέξουμε από το view Spinner μια από τρεις επιλογές: «Αρχίζει από», «Περιέχει» και «Τελειώνει σε» και να συμπληρώσουμε το TextView με το κείμενο που θέλουμε να αναζητήσουμε. Η τρίτη οθόνη της εικόνας μας δείχνει το αποτέλεσμα μιας τέτοιας αναζήτησης.



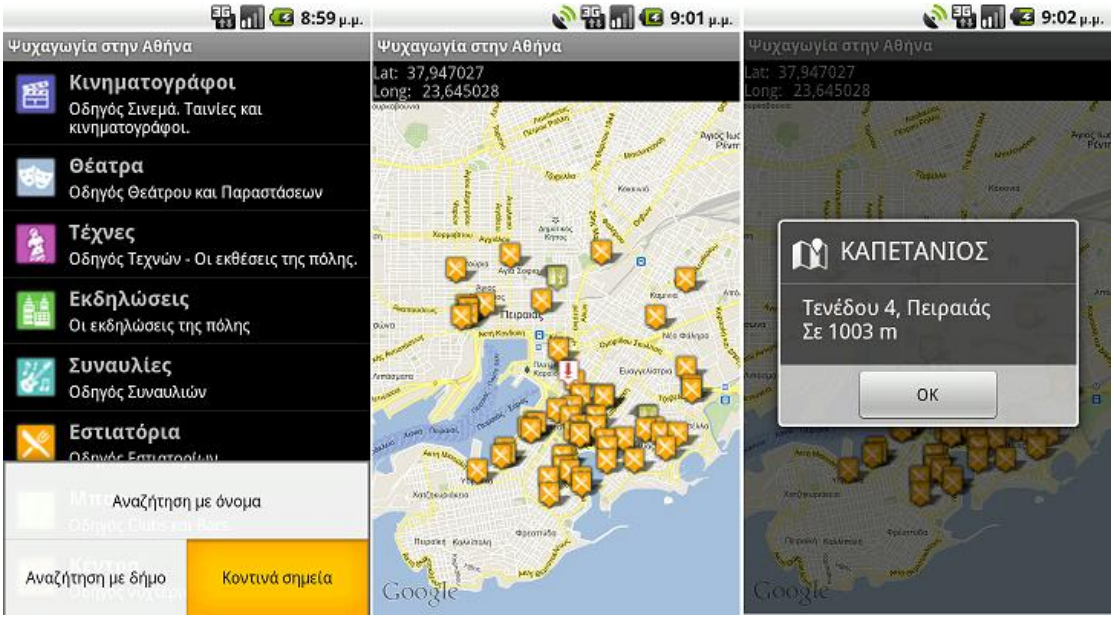
Εικόνα 7-18 Οθόνη 3

Στην Εικόνα 7-19 επιλέγουμε «Αναζήτηση με δήμο». Στην δεύτερη οθόνη μπορούμε να διαλέξουμε σε ποιο δήμο θα γίνει η αναζήτηση. Στην τρίτη οθόνη της εικόνας φαίνονται τα αποτέλεσμα της αναζήτησης. Τα αποτελέσματα αυτά είναι τα σημεία ενδιαφέροντος που υπάρχουν μέσα στα όρια του δήμου. Η δήλωση SQL που επιστρέφει τα δεδομένα από τη βάση έχει τη μορφή:

```
SELECT
    poi.pid, poi.pois_type_id, poi.name, poi.address, poi.phones,
    ST_Y(poi.location) AS latitude,
    ST_X(poi.location) AS longitude
FROM pois.poi, attica.kallikratis
WHERE kallikratis.gid = #{gid,jdbcType=INTEGER}
    AND ST_Contains(kallikratis.geom, poi.location);
--ή ST_Within(poi.location, kallikratis.geom)
```

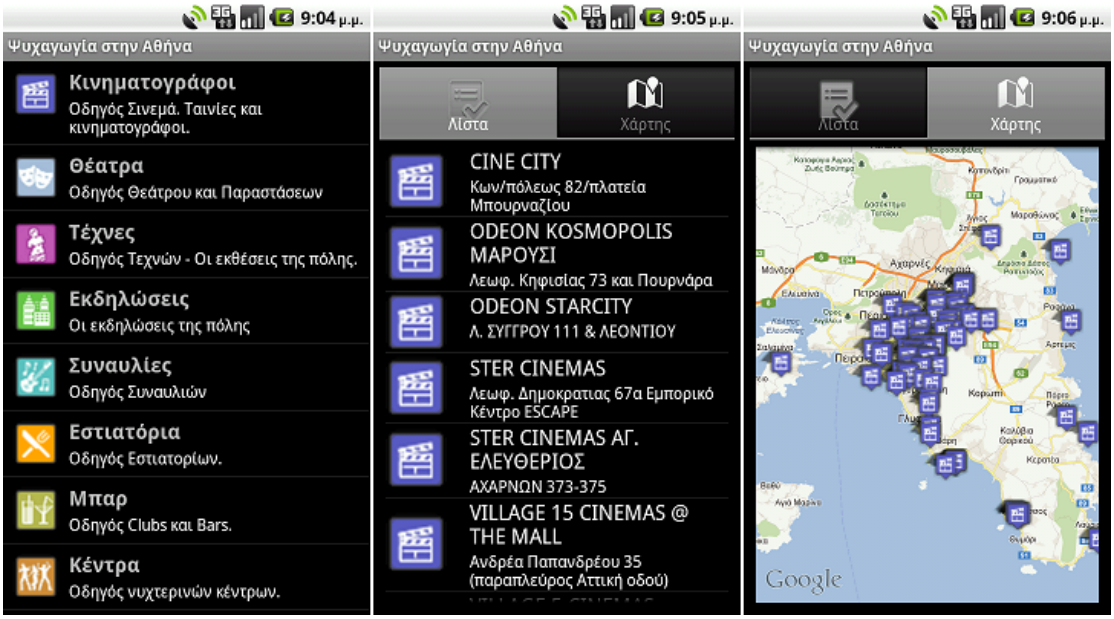
Εικόνα 7-19 Οθόνη 4

Στην Εικόνα 7-20 επιλέγουμε «Κοντινά σημεία». Η δεύτερη οθόνη μας δείχνει τα σημεία που βρίσκονται σε ακτίνα 1500 μέτρων από το σημείο που βρισκόμαστε. Αν πατήσουμε πάνω ένα τέτοιο σημείο θα εμφανιστεί η τρίτη οθόνη όπου αναφέρει το όνομα τη διεύθυνση και την απόσταση του σημείου ενδιαφέροντος.



Εικόνα 7-20 Οθόνη 5

Στην Εικόνα 7-21 βλέπουμε πάλι την αρχική οθόνη της εφαρμογής. Αν επιλέξουμε ένα στοιχείο της λίστας (ListView item) θα δούμε την δεύτερη οθόνη. Η οθόνη αυτή έχει δύο καρτέλες (tabs). Στην πρώτη καρτέλα έχουμε μια λίστα με σημεία ενδιαφέροντος και στην δεύτερη ένα χάρτη που παρουσιάζει αυτά τα σημεία τα οποία φαίνονται στην τρίτη οθόνη.

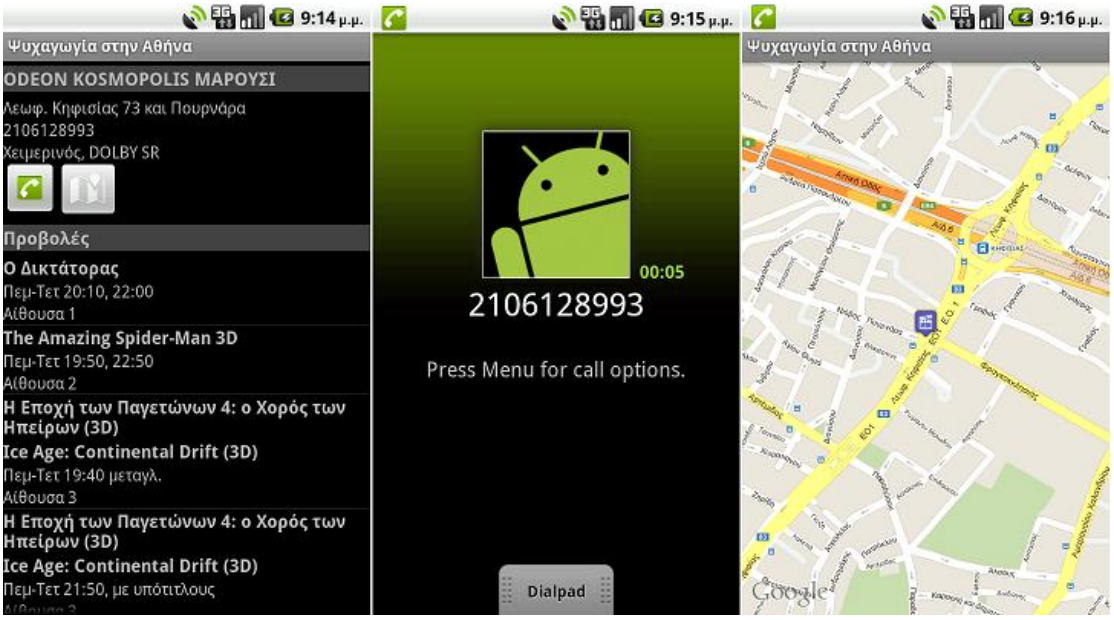


Εικόνα 7-21 Οθόνη 6

Τη λίστα που είδαμε προηγουμένως μπορούμε να τη φιλτράρουμε ώστε να περιορίσουμε τα σημεία. Στην Εικόνα 7-22 η δεύτερη οθόνη δείχνει ένα παράθυρο διαλόγου (Custom Dialog) όπου συμπληρώνουμε το κείμενο σύμφωνα με το οποίο θέλουμε να φιλτράρουμε τη λίστα. Το φίλτρο αυτό εφαρμόζεται τοπικά στη συσκευή Android και μπορεί να καταργηθεί. Αυτό γίνεται με την εμφάνιση στο μενού της επιλογής «Καθαρισμός φίλτρου» που αν πατηθεί καταργεί το φίλτρο. Το αποτέλεσμα του φίλτρου φαίνεται στην τρίτη οθόνη. Το φίλτρο είναι διαφορετικό από την επιλογή «Αναζήτηση με όνομα» που είδαμε στην Εικόνα 7-18. Η αναζήτηση γίνεται στη βάση και τα στοιχεία αποστέλλονται μέσω δικτύου, ενώ το φίλτρο εφαρμόζεται τοπικά. Μπορούμε να φιλτράρουμε και στοιχεία που έχουν προέλθει από αναζήτηση.

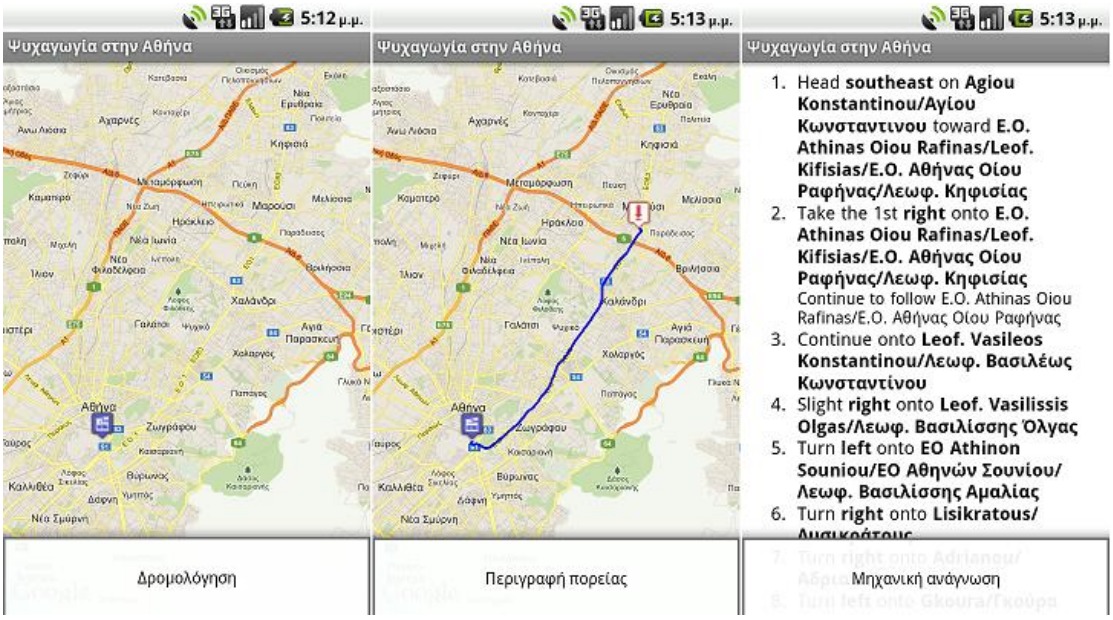
Εικόνα 7-22 Οθόνη 7

Στην Εικόνα 7-23 στην πρώτη οθόνη έχουμε επιλέξει έναν κινηματογράφο. Στο επάνω μέρος της οθόνης έχουμε πληροφορίες για τον κινηματογράφο και δυο κουμπιά. Το ένα μοιάζει με τηλέφωνο και αν πατηθεί καλεί τον αριθμό τηλεφώνου του κινηματογράφου. Το αποτέλεσμα το βλέπει κανείς στην δεύτερη οθόνη. Στην τρίτη οθόνη βλέπουμε το αποτέλεσμα που προκάλεσε το πάτημα του δεύτερου κουμπιού και αυτό είναι η εμφάνιση του σημείου ενδιαφέροντος στο χάρτη.



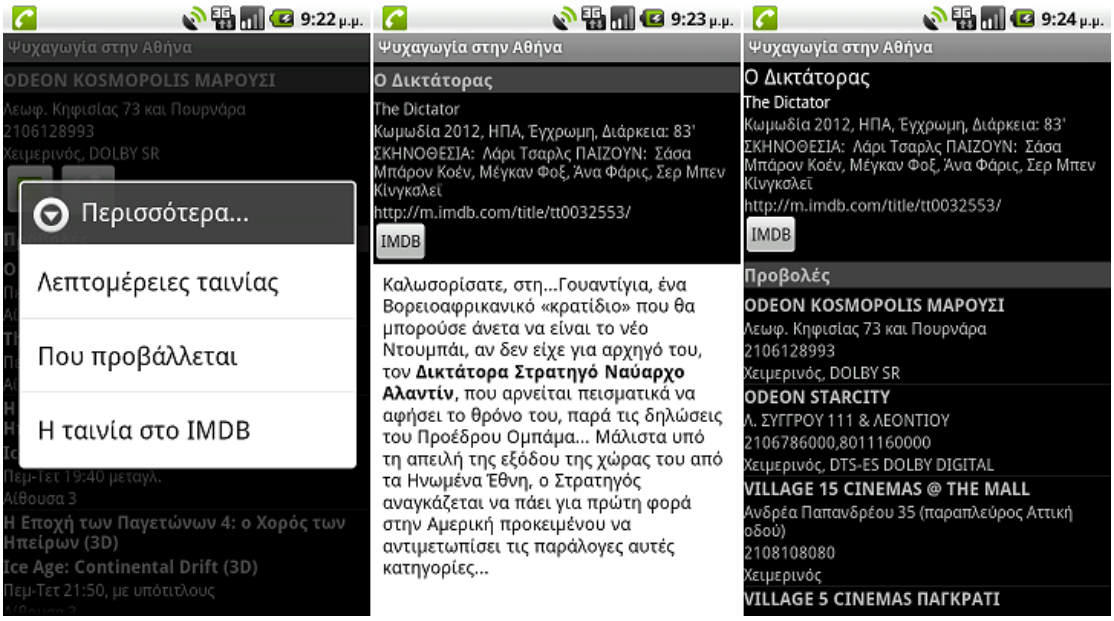
Εικόνα 7-23 Οθόνη 8

Η Εικόνα 7-24 είναι συνέχεια της προηγούμενης. Έχουμε πατήσει το μενού και μοναδική επιλογή μας είναι η δρομολόγηση στο σημείο ενδιαφέροντος. Αυτό οδηγεί στην δεύτερη οθόνη όπου είναι σημειωμένη η διαδρομή από το σημείο που βρισκόμαστε στον κινηματογράφο. Η «Περιγραφή πορείας» μας δίνει γραπτές οδηγίες για τη πορεία που θα ακολουθήσουμε. Η «Μηχανική ανάγνωση» χρησιμοποιεί τη λειτουργία Text-to-Speech του Android για να απαγγείλει το κείμενο. Στη διάρκεια της απαγγελίας το μενού αλλάζει σε «Τέλος ανάγνωσης», που αν πατηθεί σταματά τη λειτουργία Text-to-Speech.



Εικόνα 7-24 Οθόνη 9

Από την Εικόνα 7-23 ακολουθήσαμε μόνο τις επιλογές που αφορούν το σημείο ενδιαφέροντος. Στην Εικόνα 7-25 βλέπουμε τις επιλογές που έχουμε σχετικά με τις ταινίες που παίζονται στον κινηματογράφο. Στη δεύτερη οθόνη βλέπουμε τις λεπτομέρειες της ταινίας και στην τρίτη πού παίζεται η ταινία. Αν επιλέξουμε «Η ταινία στο IMDB» ή το κουμπί «IMDB» θα ανοίξει μια Activity με ενσωματωμένο ένα WebView που θα μας παρουσιάσει την σελίδα της ταινίας στη διαδικτυακή βάση ταινιών http://www.imdb.com/.



Εικόνα 7-25 Οθόνη 10

Κεφάλαιο 8
Θέματα σχεδίασης και λειτουργίας του Συστήματος

Σ΄ αυτό το κεφάλαιο θα συζητήσουμε τα προβλήματα που αντιμετωπίσαμε στην τελική φάση λειτουργίας του συστήματος για να κάνουμε τα διαφορετικά τμήματα της εφαρμογής να συνεργαστούν μεταξύ τους.

8.1 Το πεδίο του προβλήματος (domain)

Μια καλή πρακτική θα ήταν να έχουμε τις κλάσεις του πεδίου του προβλήματος σε ένα χωριστό java project και να αναφερόμαστε σε αυτό τόσο από το project συλλογής δεδομένων, όσο και από τα web services. Θα το ενσωματώναμε επίσης στο Android project. Έτσι κάθε αλλαγή στο πεδίο του προβλήματος θα είχε άμεση εφαρμογή σε ολόκληρο το σύστημα.

Δυστυχώς, η συμπερίληψη των διαφορετικών annotations σε διαφορετικά project δεν μας επέτρεψε να πραγματοποιήσουμε την ιδέα.

Ένα παράδειγμα είναι η κλάση Ταινία (Movie). Στο project συλλογής δεδομένων η κλάση δεν χρειάζεται καμία σήμανση, ενώ στην εφαρμογή των web services, έχει σημανθεί έτσι:

```
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "movie")
@XmlType(propOrder = { "siteId", "greekTitle", "originalTitle", "url",
"info", "description", "cast", "IMDBLink", "showtimes" })
public class Movie {
    private int siteId;
    private String greekTitle;
    private String originalTitle;
    private String url;
    private String info;
    private String description;
    private String cast;
    private String IMDBLink;

    // XmLElementWrapper generates a wrapper element around XML
representation
    @XmlElementWrapper(name = "showtimes")
    // XmlElement sets the name of the entities
    @XmlElement(name = "showtime")
    private List<Showtime> showtimes;
    ...
}
```

Οι αναφορές (import) δείχνουν ότι οι χρησιμοποιημένες σημάνσεις αφορούν την βιβλιοθήκη JAXB. Δεν έχουμε χρησιμοποιήσει σημάνσεις για το JSON και γιατί δεν χρειάζονται, αλλά και για λόγους συμβατότητας με τις δυο βιβλιοθήκες Jackson and Jettison.

Η ίδια κλάση στο Android project έχει σημανθεί διαφορετικά:

```
import org.simpleframework.xml.Element;
import org.simpleframework.xml.ElementList;

import com.google.gson.annotations.SerializedName;

public class Movie {
    @Element
```

```
    private int siteId;
    @Element
    private String greekTitle;
    @Element(required = false)
    private String originalTitle;
    @Element(required = false)
    private String url;
    @Element(required = false)
    private String info;
    @Element(required = false)
    private String description;
    @Element(required = false)
    private String cast;
    @Element(required = false, name = "IMDBLink")
    @SerializedName("IMDBLink")
    private String imdblink;

    @ElementList(name = "showtimes", type = Showtime.class, entry =
"showtime", inline = false, required = false)
    private List<Showtime> showtimes;
    ...
}
```

Εδώ οι αναφορές (import) αφορούν τις βιβλιοθήκες simpleframework.xml και gson. Η διαφορά στην αναπαράσταση του πεδίου imdblink διορθώνεται με τη χρήση σημάνσεων.

Στο πρόβλημα αυτό αναφέρεται στο άρθρο το με τίτλο «A POJO with annotations is not Plain» (POJO προέρχεται από τα αρχικά Plain Old Java Object.) και ο Vincent Partington, στο blog xebia[39] προσεγγίζοντάς το από μια παρόμοια σκοπιά.

8.2 Κάθε πότε ενημερώνουμε τα δεδομένα.

Ένα πρόβλημα που αφορά την συλλογή των δεδομένων είναι η απόφαση για το χρονοδιάγραμμα εκτέλεσής της. Το project έχει υλοποιηθεί ώστε να μπορούμε να συλλέγουμε τα δεδομένα, αν θέλουμε, ανά κατηγορία. Για το σκοπό αυτό υποστηρίζει τους παρακάτω διακόπτες:

[39] http://blog.xebia.com/2007/01/28/a-pojo-with-annotations-is-not-plain/

- **-initialize:** Όταν ο διακόπτης υπάρχει στην γραμμή εντολών τότε το λογισμικό αρχικοποιεί την βάση δεδομένων σβήνοντας όλα τα δεδομένα που αυτή περιέχει εκτός από τα δεδομένα των πινάκων των δήμων και των διαπιστευτηρίων.
- **-cinemas:** Ενημερώνει μόνο τους κινηματογράφους, τις ταινίες και τις προβολές.
- **-events:** Ενημερώνει τις εκδηλώσεις διαφόρων ειδών και εκθέσεις, μουσικές εκδηλώσεις όπως και εκθέσεις σε μουσεία, ιδρύματα και γκαλερί. Ενημερώνει επίσης και τις αίθουσες στις οποίες λαμβάνουν χώρα.
- **-places:** Ενημερώνει τα μπαρ και τα κλαμπ καθώς και τα νυχτερινά κέντρα.
- **-restaurants:** Ενημερώνει τα εστιατόρια.
- **-theaters:** Ενημερώνει τα θέατρα και τις παραστάσεις καθώς και τις ώρες των παραστάσεων.
- **-all:** Εκτελεί όλες τις παραπάνω λειτουργίες.

Ένα λογικό χρονοδιάγραμμα είναι να συλλέγονται τα δεδομένα που αφορούν τους κινηματογράφους και θέατρα, τουλάχιστον μία φορά κάθε εβδομάδα, κατά προτίμηση μέρα παρά μέρα. Οι εκδηλώσεις πρέπει να ενημερώνονται κάθε μέρα ή το πολύ κάθε δεύτερη. Οι πληροφορίες που αφορούν τα μπαρ και τα κλαμπ καθώς και τα εστιατόρια, δεν αλλάζουν συχνά και αρκεί να ανανεώνονται μια φορά κάθε 15 ημέρες.

Για να πετύχουμε την εκτέλεση του χρονοδιαγράμματος χρησιμοποιούμε το Task Scheduler των Windows ή το Crontab στο Ubuntu ή κάποιο αντίστοιχο λογισμικό σε άλλα λειτουργικά.

Κεφάλαιο 9
Επίλογος

Στη μεταπτυχιακή αυτή διατριβή ασχολήθηκαμε με τη δημιουργία ενός υπολογιστικού συστήματος που αποτελείται από τέσσερα τμήματα. Σε κάθε τμήμα αντιμετωπίσαμε διαφορετικές προκλήσεις ενώ σημαντική είναι η ενοποίηση των τεσσάρων αυτών τμημάτων σε ένα όλον.

Για την υλοποίηση του τμήματος «Συλλογή Δεδομένων», αναζητήσαμε και βρήκαμε, τόσο τα κατάλληλα εργαλεία, όσο και τους κατάλληλους ιστοχώρους απ' όπου αντλήσαμε δεδομένα.

Για την υλοποίηση της «Αποθήκευσης Δεδομένων» αναζητήσαμε το κατάλληλο RDBMS και τα εργαλεία που μας επέτρεπαν την επικοινωνία μαζί του. Χρησιμοποιήσαμε PostgreSQL, PostGIS και τη βιβλιοθήκη MyBatis. Μια πρόκληση στο σημείο αυτό θα ήταν η χρήση μιας NoSQL βάσης δεδομένων που θα μας επέτρεπε να ανεβάσουμε την εφαρμογή μας στο Google App Engine.

Στην υλοποίηση των «Restful Web Services» χρησιμοποιήσαμε την βιβλιοθήκη Jersey που ακολουθεί τις προδιαγραφές JAX-RS και εγκαταστήσαμε το service στον web server tomcat 7.

Για την ανάπτυξη του τμήματος «Android Application» χρειάστηκε να γνωρίσουμε τον κόσμο του λειτουργικού Android, ένα κόσμο που εξελίσσεται ραγδαία. Όταν ξεκινήσαμε την ανάπτυξη το 90% των τηλεφώνων Android έτρεχαν τις εκδόσεις Eclair και Froyo (2.1 και 2.2) ενώ μόλις είχε κυκλοφορήσει η έκδοση Ice Cream Sandwich (4.0.x) για ταμπλέτες. Σήμερα όμως ο κώδικας είναι ήδη ξεπερασμένος ή χρειάζεται σε πολλά σημεία διόρθωση για να λειτουργήσει με τις καινούργιες εκδόσεις του Android.

Με την έκδοση Honeycomb (3.x) στις 22 Φεβρουαρίου του 2011 το Android παύει να εξαρτάται από φυσικά κουμπιά και εισάγεται η κλάση ActionBar. Αυτό σημαίνει ότι ο κώδικας πρέπει να ξαναγραφεί, ώστε να μην χρησιμοποιεί το φυσικό κουμπί «Menu». Επίσης, από την ίδια έκδοση η TabActivity θεωρείται ξεπερασμένη (deprecated) και προτείνεται η αντικατάστασή της από Fragments.

Η μεγαλύτερη δυσκολία ήταν να κάνουμε το σύστημα να δουλέψει σαν ένα «όλον» Κάθε νέο τμήμα που προσθέταμε οδηγούσε σε μερική αλλαγή των προηγούμενων. Ειδικά, το τμήμα «Android Application» προκάλεσε πολλές αλλαγές στα άλλα τμήματα που κάποιος θα θεωρούσε εκείνη τη στιγμή τελειωμένα αναδείχνοντας την ανάγκη για καλύτερη σχεδίαση του «όλου».

Βιβλιογραφία

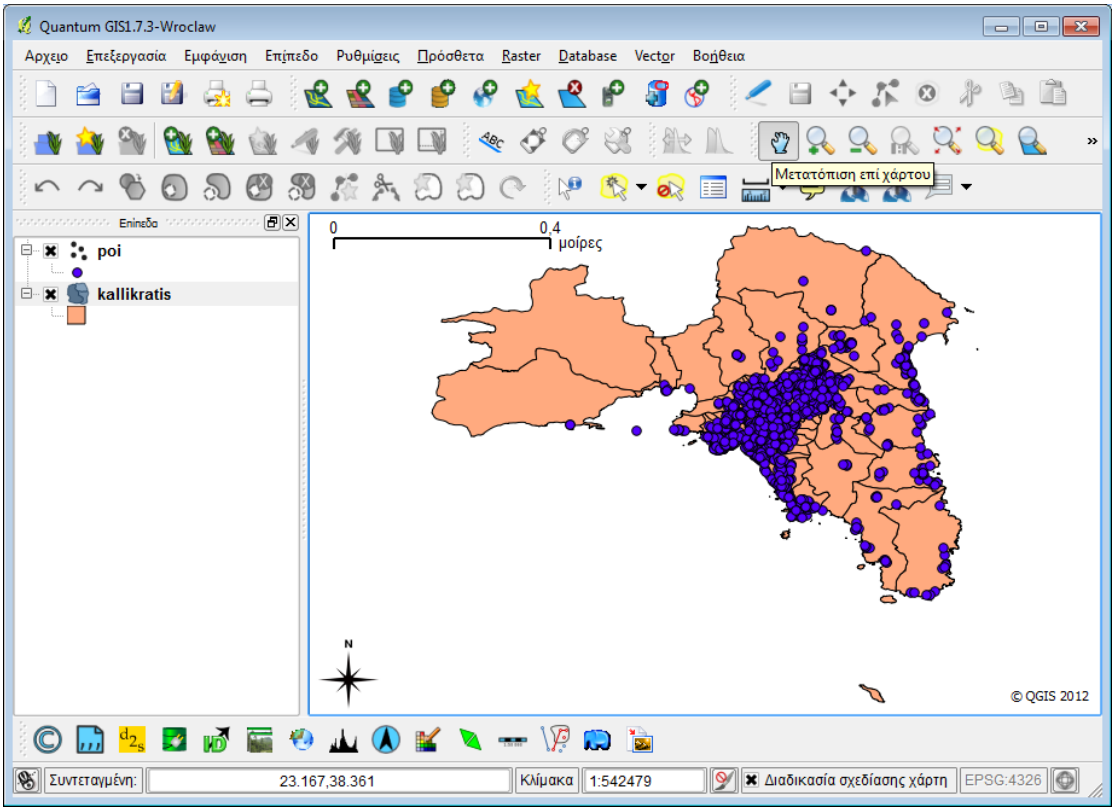
[01] Lauren Darcey; Shane Conder (2011): «Ανάπτυξη Εφαρμογών με το Android 2η Έκδοση», *Εκδόσεις Μ.Γκιούρδας*.

[02] Lauren Darcey; Shane Conder (2010): «Sams Teach Yourself Android™ Application Development in 24 Hours» , *Pearson Education (US).*

Παράρτημα Α
Χρήσιμα λογισμικά

Α-1 Quantum GIS

Στη διάρκεια της ανάπτυξης της εφαρμογής συλλογής δεδομένων υπήρξε η ανάγκη να προβάλουμε τα σημεία ενδιαφέροντος πάνω σε χάρτη, ώστε να ελέγξουμε την ορθότητα και τη διασπορά των σημείων. Χρησιμοποιήσαμε το ελεύθερο λογισμικό Quantum GIS. Αυτό φαίνεται στην εικόνα Α-1.



Εικόνα Α-9-1. Το ελεύθερο λογισμικό Quantum GIS.

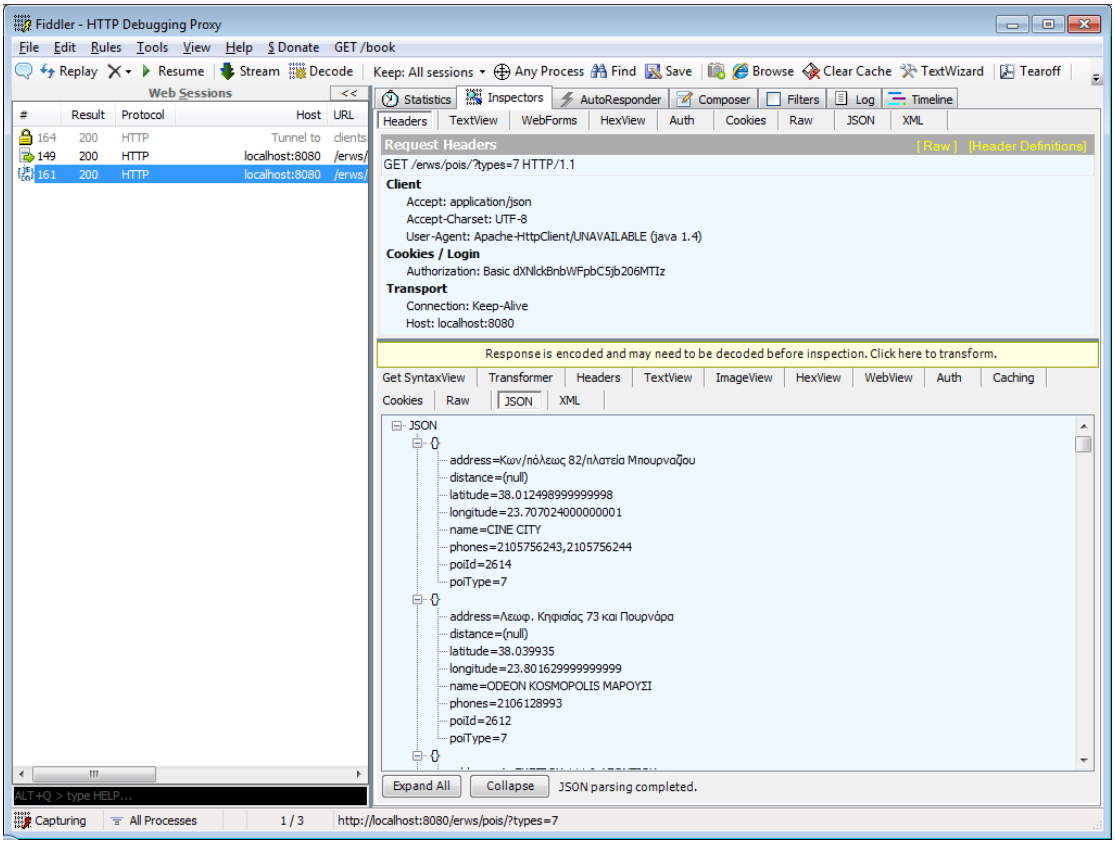
Α-2 Fiddler

Ένα ιδιαίτερα χρήσιμο εργαλείο που χρησιμοποιήσαμε για να οπτικοποιήσουμε την ανταλλαγή δεδομένων μεταξύ συσκευής Android και εξυπερετητή είναι το Fiddler.

Το Fiddler είναι ένας διαμεσολαβητής (proxy) που καταγράφει όλη την HTTP(S) κυκλοφορία μεταξύ του υπολογιστή μας και το Internet και μας δίνει τη δυνατότητα να επιθεωρήσουμε τα εισερχόμενα και εξερχόμενα δεδομένα. Το Fiddler είναι δωρεάν και μπορεί να ρυθμιστεί, να παρακολουθεί την κίνηση από σχεδόν οποιαδήποτε εφαρμογή υποστηρίζει ένα διαμεσολαβητή (proxy), συμπεριλαμβανομένων των συσκευών Android.

Ρυθμίσαμε το Fiddler να ανακατευθύνει την κίνηση από την πόρτα 8888 στην πόρτα 8080 που αποκρίνεται ο tomcat web server, και με τη συσκευή Android ή τον emulator

συνδεθήκαμε στην πόρτα 8888. Το Fiddler μας παρουσίασε όλη την κυκλοφορία δεδομένων όπως βλέπουμε στην εικόνα Α-2.



Εικόνα Α-2.

Παράρτημα Β
Εγκατάσταση σε Ubuntu

Β-1 Εγκατάσταση της Java

Αρχικά εγκαθιστούμε την Java της Oracle[40].

```
sudo apt-get -y install python-software-properties
```

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java7-installer
```

Και ελέγχουμε την σωστή εγκατάσταση:

```
java -version
```

Σε περίπτωση σωστής εγκατάστασης θα δούμε κάτι τέτοιο:

[40] (βλ. http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-ubuntu-via.html)

```
java version "1.7.0_05"
```

Β-2 Εγκατάσταση του tomcat[41]

Στη συνέχεια από την σελίδα http://tomcat.apache.org/download-70.cgi και αφού διαλέξουμε την έκδοση (version) του λογισμικού και τον διακομιστή από όπου θα κατεβάσουμε το λογισμικό (εμείς διαλέξαμε http://apache.forthnet.gr/) αντιγράφουμε την διεύθυνση της έκδοσης που θέλουμε π.χ.

```
http://apache.forthnet.gr/tomcat/tomcat-7/v7.0.28/bin/apache-tomcat-
7.0.28.tar.gz
```

Στη συνέχεια, στη γραμμή εντολών εκτελούμε:

```
wget http://apache.forthnet.gr/tomcat/tomcat-7/v7.0.28/bin/apache-tomcat-
7.0.28.tar.gz
tar xvzf apache-tomcat-7.0.28.tar.gz
sudo mv apache-tomcat-7.0.28/ /usr/share/tomcat7
```

Ακολούθως, επεξεργαζόμαστε το αρχείο tomcat-users.xml του tomcat, ώστε να δώσουμε δικαιώματα των ρόλων «manager» και «admin» στον χρήστη root.

```
sudo nano /usr/share/tomcat7/conf/tomcat-users.xml
```

Το αρχείο μετά την επεξεργασία θα έχει την εξής μορφή:

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

<tomcat-users>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<role rolename="admin"/>

<user username="root" password="******" roles="manager-gui, admin-gui,
manager, admin, manager-script, admin-script"/>
</tomcat-users>
```

[41] (βλ. http://diegobenna.blogspot.gr/2011/01/install-tomcat-7-in-ubuntu-1010.html)

Για να ξεκινά ο tomcat με την εκκίνηση του υπολογιστή, δημιουργούμε το ακόλουθο script:

```
sudo nano /etc/init.d/tomcat7
```

```
# Tomcat auto-start
#
# description: Auto-starts tomcat
# processname: tomcat
# pidfile: /var/run/tomcat.pid

case $1 in
start)
sh /usr/share/tomcat7/bin/startup.sh
;;
stop)
sh /usr/share/tomcat7/bin/shutdown.sh
;;
restart)
sh /usr/share/tomcat7/bin/shutdown.sh
sh /usr/share/tomcat7/bin/startup.sh
;;
esac
exit 0
```

Και το μετατρέπουμε σε εκτελέσιμο τρέχοντας την εντολή:

```
sudo chmod 755 /etc/init.d/tomcat7
```

Στο τέλος, συνδέουμε αυτό το script με τους startup folders με ένα συμβολικό σύνδεσμο, ώστε ο tomcat να επανεκκινήσει σε περίπτωση αλλαγής της κατάστασης του συστήματος:

```
sudo ln -s /etc/init.d/tomcat7 /etc/rc1.d/K99tomcat
sudo ln -s /etc/init.d/tomcat7 /etc/rc2.d/S99tomcat
```

Ξεκινάμε τον tomcat:

```
sudo service tomcat7 start
```

Β-3 Εγκατάσταση των Postgres 9.1 και PostGIS 2.0.1

Τα παρακάτω scripts ακολουθούν το script https://gist.github.com/2846196 με μικρές παραλλαγές π.χ. για την έκδοση (version) του λογισμικού.

Αρχικά εγκαθιστούμε την Postgres 9.1 και PostGIS 2.0.1 και τα σχετικά πακέτα από το repository ppa:ubuntugis/ubuntugis-unstable:

```
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt-get update
sudo apt-get -y install postgis postgresql-9.1 postgresql-server-dev-9.1
postgresql-contrib-9.1 postgis gdal-bin binutils libgeos-3.2.2 libgeos-c1
libgeos-dev libgdal1-dev libxml2 libxml2-dev libxml2-dev checkinstall proj
libpq-dev
```

Στη συνέχεια, εγκαθιστούμε την τελευταία έκδοση της βιβλιοθήκης GEOS για τοπολογική υποστήριξη.

```
wget http://download.osgeo.org/geos/geos-3.3.5.tar.bz2
tar -jxvf geos-3.3.5.tar.bz2
cd geos-3.3.5
sudo ./configure && sudo make
```

Ακολούθως, εγκαθιστούμε την PostGIS 2.0.1

```
sudo mkdir -p '/usr/share/postgresql/9.1/contrib/postgis-2.0.1'

# fetch, compile and install PostGIS
wget http://postgis.refractions.net/download/postgis-2.0.1.tar.gz
tar zxvf postgis-2.0.1.tar.gz && cd postgis-2.0.1/
sudo ./configure && sudo make && sudo checkinstall --pkgname postgis-2.0.1
--pkgversion 2.0.1-src --default
```

Μετά το τέλος της εγκατάστασης βλέπουμε το μήνυμα:

```
**********************************************************************

 Done. The new package has been installed and saved to

 /home/kostas/geos-3.3.5/postgis-2.0.1/postgis-2.0.1_2.0.1-src-1_i386.deb

 You can remove it from your system anytime using:

      dpkg -r postgis-2.0.1

**********************************************************************
```

Στη συνέχεια, δημιουργούμε την βάση **template_postgis** που χρησιμοποιούμε σαν πρότυπο για να δημιουργήσουμε κάθε καινούργια βάση.

```
# now create the template_postgis database template
sudo su postgres -c'createdb -E UTF8 -U postgres template_postgis'
sudo su postgres -c'createlang -d template_postgis plpgsql;'
sudo su postgres -c'psql -U postgres -d template_postgis -c"CREATE
EXTENSION hstore;"'
sudo su postgres -c'psql -U postgres -d template_postgis -f
/usr/share/postgresql/9.1/contrib/postgis-2.0/postgis.sql'
sudo su postgres -c'psql -U postgres -d template_postgis -f
/usr/share/postgresql/9.1/contrib/postgis-2.0/spatial_ref_sys.sql'
sudo su postgres -c'psql -U postgres -d template_postgis -c"select
postgis_lib_version();"'
sudo su postgres -c'psql -U postgres -d template_postgis -c "GRANT ALL ON
geometry_columns TO PUBLIC;"'
sudo su postgres -c'psql -U postgres -d template_postgis -c "GRANT ALL ON
spatial_ref_sys TO PUBLIC;"'
sudo su postgres -c'psql -U postgres -d template_postgis -c "GRANT ALL ON
geography_columns TO PUBLIC;"'
echo "Done!"
```

Β-4 Ρύθμιση της Postgres

Αλλάζουμε το password του χρήστη postgres:

```
sudo su postgres -c'psql -d template1'
template1=# ALTER USER postgres WITH PASSWORD '123';
ALTER ROLE
template1=# \q
```

Για να δώσουμε δικαιώματα πρόσβασης στην βάση από άλλον υπολογιστή του ίδιου δικτύου, επεξεργαζόμαστε το αρχείο pg_hba.conf

```
cd /etc/postgresql/9.1/main/
sudo nano pg_hba.conf
```

Προσθέτουμε στο τέλος την γραμμή:

```
TYPE      DATABASE        USER          ADDRESS             METHOD
host      all             all           192.168.1.1/24      trust
```

Επεξεργαζόμαστε επίσης το αρχείο postgresql.conf

```
sudo nano postgresql.conf
```

και θέτουμε την τιμή της μεταβλητής listen_addresses από localhost σε '*'

```
listen_addresses = '*'
```

Για να εφαρμοστούν οι αλλαγές επανεκκινούμε την βάση.

```
sudo service postgresql restart
```

Παράρτημα Γ

Η κλάση MyMapView

```
package gr.athens.entertainment.base;

import android.content.Context;
import android.graphics.Canvas;
import android.location.Location;
import android.os.Handler;
import android.util.AttributeSet;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;

public class MyMapView extends MapView {

    private Integer DELAY_TASK_MILISECONDS = 500;
    private int zoomLevel = -1;

    private GeoPoint center;
    private Double radius;
    private GeoPoint lowerLeftBottom;
    private GeoPoint upperRightTop;


    private MyMapListener mListener;

    private Handler handler = new Handler() {};

    public MyMapView(Context context, AttributeSet attributes) {
        super(context, attributes);
```

```
    }

    public MyMapView(Context context, AttributeSet attributes, int
defStyle) {
        super(context, attributes, defStyle);
    }

    public MyMapView(Context context, String apiKey) {
        super(context, apiKey);
    }

    public void setOnChangeListener(MyMapListener listener) {
        mListener = listener;
    }

    public void removeOnChangeListener(MyMapListener listener) {
        if (mListener.equals(listener)) {
            mListener = null;
        }
    }

    @Override
    protected void dispatchDraw(Canvas canvas) {
        super.dispatchDraw(canvas);
        if (mListener == null) {
            return;
        }
        int oldZoomLevel = zoomLevel;
        GeoPoint oldCenter = center;

        zoomLevel = this.getZoomLevel();
        center = this.getMapCenter();

        System.out.println("oldZoomLevel: " + oldZoomLevel);
        System.out.println("zoomLevel" + zoomLevel);

        if ((oldCenter != null)
                && (oldCenter.equals(center))
                && (oldZoomLevel <= zoomLevel)) {
            return;
        }

        GeoPoint oldLowerLeftBottom = lowerLeftBottom;
        GeoPoint oldUpperRightTop = upperRightTop;


        lowerLeftBottom = this.getProjection().fromPixels(0,
this.getHeight());
        upperRightTop = this.getProjection().fromPixels(this.getWidth(),
0);

        //
        Location locLowerLeftBottom = new Location("Lower Left Bottom");

        locLowerLeftBottom.setLatitude(lowerLeftBottom.getLatitudeE6() /
1E6);
        locLowerLeftBottom.setLongitude(lowerLeftBottom.getLongitudeE6() /
1E6);
```

```
        Location locUpperRightTop = new Location("Upper Right Top");

        locUpperRightTop.setLatitude(upperRightTop.getLatitudeE6() / 1E6);
        locUpperRightTop.setLongitude(upperRightTop.getLongitudeE6() /
1E6);

        double distance = locLowerLeftBottom.distanceTo(locUpperRightTop);
        radius = distance * 0.5;
        System.out.println("radius: " + radius);
        //

        if (oldLowerLeftBottom == null
                || oldUpperRightTop == null
                || !oldLowerLeftBottom.equals(lowerLeftBottom)
                || !oldUpperRightTop.equals(upperRightTop))
        {
            handler.removeCallbacks(mUpdateTimeTask);
            System.out.println("--------------------");
            handler.postDelayed(mUpdateTimeTask, DELAY_TASK_MILISECONDS);
        }

    }

    private Runnable mUpdateTimeTask = new Runnable() {
           public void run() {
               mListener.onChange(lowerLeftBottom.getLongitudeE6(),
                       lowerLeftBottom.getLatitudeE6(),
                       upperRightTop.getLongitudeE6(),
                       upperRightTop.getLatitudeE6());
           }
        };

}
```