Títol: Sistema de gestió i visualització
de mapes cartogràfics per a dispositius mòbils Android.
Volum: 1
Alumne: Ribatallada Torelló, Adrià
Director/Ponent: Vilaplana Pastó, Josep
Codirector: Martí Pino, David
Departament: LSI
Data: 19 de juny de 2013

DADES DEL PROJECTE

Títol del Projecte: Sistema de gestió i visualització de mapes cartogràfics per a dispositius mòbils Android.

Nom de l'Estudiant: Ribatallada Torelló, Adrià

Titulació: Enginyeria Informàtica

Crèdits: 37,5 crèdits

Director/Ponent: Vilaplana Pastó, Josep

Codirector: Martí Pino, David

Departament: LSI

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Soto Riera, Antoni

Vocal: Castro Pérez, Jordi

Secretari: Vilaplana Pastó, Josep

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Agraïments

Ara que aquest projecte arriba a la fi, m'agradaria dedicar uns paràgrafs a donar les gràcies a totes les persones que m'han ajudat a realitzar-lo i que m'han donat suport durant la seva realització.

Aquests agraïments van dirigits especialment a la meva família, pel seu suport incondicional, i també a David Martí Pino i a Josep Vilaplana Pastó, els dos codirectors del projecte, per l'esforç i temps dedicats a ajudar-me.

A tots els que també m'han ajudat però que no he pogut mencionar directament també els dono el meu sincer agraïment.

Sense vosaltres aquest PFC no s'hauria pogut realitzar.

Índex

Capítol 1

Introducció

El propòsit final del projecte és desenvolupar un conjunt de biblioteques de codi lliure per a la visualització i gestió de mapes cartogràfics. Aquestes biblioteques tenen com a finalitat permetre crear aplicacions que requereixin visualitzar informació cartogràfica, i en concret, donar suport a aplicacions de recollida de dades vinculades a la localització geogràfica en la plataforma Android.

El projecte, però, va sorgir d'un altre PFC, el projecte ZamiaDroid, iniciat fa uns anys per David Martí Pino a la mateixa Universitat Politècnica de Catalunya.

Així que, malgrat que les biblioteques pretenen ser prou àmplies i genèriques per a servir a un gran nombre d'aplicacions que en puguin fer us, la majoria dels exemples i funcionalitats han estat pensades tenint en ment ZamiaDroid, i per tant estan força lligades a les seves necessitats concretes.

Però, tot i que aquestes biblioteques són la part central del projecte, com es pot entreveure al capítol d'objectius, la seva implementació no ha estat l'únic i exclusiu interès del projecte.

De fet s'ha posat molt d'èmfasi en cercar programes ja existents i integrar-los per a aconseguir els objectius.

Així també es pretenia, al mateix temps, veure fins a quin nivell el món del GIS ha penetrat a la plataforma Android.

Incrementar el coneixement de la plataforma i quins són els seus límits també era un dels propòsits que feien atractiu aquest plantejament del pro-

jecte.

Així que mentre s'estudiaven i desenvolupaven aquestes biblioteques també s'han explorat aspectes "tangents" de la plataforma Android i el seu entorn de desenvolupament.

S'ha intentat estructurar la present memòria de forma semblant a la del desenvolupament real del projecte per tal d'intentar facilitar-ne la comprensió.

- Objectius
- Estudi tecnològic
- Anàlisi i Disseny
- Implementació
- Conclusions

Així després d'enumerar els objectius generals del projecte es presenta la documentació del procés d'estudi tecnològic que es va realitzar tant a nivell de plataforma de desenvolupament com d'aplicacions existents i similars.

Després s'especifica amb més concreció com es vol abordar els objectius, i en el següent capítol es poden veure alguns detalls interessants del procés d'implementació.

Finalment l'estudi econòmic i les conclusions com a valoració final.

I un últim apartat on es comenten aspectes a millorar del present projecte, i com encarar-ne una possible continuació.

Capítol 2

Objectius

2.1 Objectius del projecte

El propòsit final del projecte és desenvolupar un conjunt de biblioteques de codi lliure per a la visualització i gestió de mapes cartogràfics. Aquestes biblioteques tenen com a finalitat facilitar la creació d'aplicacions de visualització d'informació cartogràfica, i en concret, donar suport a aplicacions de recollida de dades vinculades a la localització geogràfica en la plataforma Android. Per assolir aquest propòsit final hem plantejat els següents objectius:

1. Desenvolupament d'un conjunt de biblioteques per a Android que permetin visualitzar mapes cartogràfics des de diversos orígens: GML,KML i OSM.

2. Les biblioteques han de cobrir les següents operacions sobre els mapes:

   Visualització i gestió de mapes sense connexió a la xarxa.

   Dibuix d'elements geomètrics associats a dades geogràfiques sobre els mapes.

   Realització de càlculs sobre els elements geomètrics introduïts.

   Proporcionar elements d'interacció gràfica per a facilitar la construcció d'interfícies que realitzin les anteriors funcionalitats.

3. Desenvolupament d'una API "façana" per a fer accessibles les biblioteques a sl4a (que permet programar en llenguatges de scripting sobre la plataforma Android).

   Això ens permetrà facilitarà el desenvolupament de clients que aprofitin les funcionalitats de les nostres biblioteques en altres llenguatges de programació.

4. Desenvolupament d'una aplicació de prova que mostri les capacitats de les biblioteques.

5. Tot el codi generat tindrà llicència lliure GPL v3.

Capítol 3

Estudi Tecnològic

3.1 Introducció de conceptes

Abans de començar amb l'estudi tecnològic pròpiament dit, i per clarificar alguns dels objectius del projecte, veig la necessitat d'introduir alguns conceptes recurrents al llarg del present document.

3.1.1 Els Sistemes GIS

El contingut d'aquest projecte esta íntimament relacionat amb el concepte de GIS. Així que sembla oportú començar definint breument què s'entén per aquest acrònim i com l'aplicarem en la present memòria.

GIS són les sigles en anglès de sistema d'informació geogràfic (geographic information system), i fa referència als sistemes (informàtics o no) dissenyats per adquirir, emmagatzemar, tractar, analitzar i mostrar qualsevol tipus de dades geogràfiques.

Cal tenir present la distinció entre els termes cartografia i GIS, ja que GIS és un terme més ampli que implica no només la cartografia sinó també altres ciències com la estadística, i algun tipus d'emmagatzematge de les dades, i per tant esta vinculat molt prominentment amb la tecnologia que fa possible la implementació del sistema.

Es considera que el primer sistema GIS data de l'any 1854 (el mapa del brot de colera al barri del Soho de Londres fet per John Snow). Això significa que el terme no te perquè implicar necessàriament la presència de tecnologia informàtica per ser aplicat.

Però avui en dia la informàtica i el GIS estan tan íntimament relacionats que semblen gairebé inseparables. Així que al llarg d'aquest treball sempre que es mencioni el terme GIS se sobreentendrà que es tracta d'un sistema informàtic.

Els sistemes GIS tenìen inicialment uns usos importants, però normalment de caire governamental o corporatiu, i no tothom podia accedir a les utilitats d'un sistema GIS.

Per exemple, un dels primers sistemes GIS operacionals com l'entenem avui en dia, que es va desenvolupar al Canadà, servia per monitoritzar els tipus de terra, agricultura, massa forestal, etc. i determinar la capacitat rural del país.

Però en els darrers temps, i sobretot gràcies a "l'obertura" de la tecnologia de posicionament global (GPS) per a usos personals els sistemes GIS s'han obert a un públic més ampli, i no es d'estranyar que la majoria de la gent porti alguna mena de sistema GIS a la butxaca en aquests moments.

Aquest projecte centra bona part dels esforços en el tema de visualització de mapes, però és important tenir present que aquesta només és una part dels sistemes GIS. Naturalment, però, els mapes en són una part molt important.

Existeixen diversos proveïdors de mapes (o dades geogràfiques) pels sistemes GIS, i en aquest projecte treballarem amb OpenStreetMap.

3.1.1.1 OpenStreetMap

Un dels proveïdors de mapes més popular actualment és el Google, amb el seu Google maps. També hi han molts altres proveïdors de serveis similars, però majoritàriament aquests serveis no són lliures (tot i que a vegades sí gratuïts).

Aquest fet posa de manifest una realitat, que és que la informació geogràfica no és lliure en moltes parts del planeta. A vegades esta controlada per agències governamentals o fins i tot empreses privades que es lucren permetent l'us d'aquestes dades.

Molts cops existeix la il·lusió que aquestes dades són lliures per ser gratuïtes, o que realment si que són lliures, ja que en algunes ocasions fins i tot és possible que les dades crues en sí siguin de domini públic, però que el tractament i presentació d'aquestes no ho sigui.

En el cas particular de Google es pot comprovar a què ens referim si us adreceu als seus termes de servei[69].

Deixant de banda els aspectes morals de que la informació estigui controlada per certes persones i s'aprofitin d'aquest control per lucrar-se, el cert és que el mateix fet que les dades siguin controlades per interessos comercials és un perill per la mateixa integritat i veracitat d'aquestes.

Per posar un exemple pràctic, a vegades les dades que proporcionen aquests serveis de mapes comercials contenen petites inconsistències posades expressament per detectar-ne l'us de persones no autoritzades (el que els d'OpenStreetMap anomenen "Copyright Easter Eggs").

Això significa incorreccions, o altrament dit dades invàlides. A més de les incorreccions que hi pugui haver per manca d'actualitzacions (que la empresa o agència governamental s'ha d'encarregar d'anar fent) o per errors diversos.

A tot això es sumen restriccions d'us d'allò més variades que s'apliquen sobre les dades per temes de copyright, que condicionen què pots fer o deixar de fer amb les dades que proveeixen. A vegades això significa que el que pots fer amb les dades és limitar-te a observar-les.

Normalment no es pot corregir el nom d'algun carrer, ni Afegir establiments o llocs d'interès o descarregar-te les dades per fer-les servir en el teu ordinador personal per a qualsevol cosa (almenys no lliurement).

I a més, quan les dades són propietat d'algú altre, sempre existeix el risc de que en un futur els termes de servei canviïn i apliquin noves i divertides restriccions que converteixin en il·legal alguna pràctica que fins ara realitzaves assíduament.

La idea d'OpenStreenMap és aprofitar els avenços tecnològics (Internet i el GPS) per permetre a la gent crear els seus propis mapes i, col·laborant amb gent d'arreu del món crear un servei que proporcioni mapes a tothom sense les restriccions abans mencionades. És a dir, un servei de mapes col·laboratiu i lliure.

Actualment el projecte OpenStreetMap està emparat per una fundació sense anim de lucre basada al regne unit anomenada OpenStreetMap Foundation.

Tal i com ells mateixos la defineixen:
La Fundació OpenStreetMap és una organització sense ànim de lucre dedicada a impulsar el creixement, el desenvolupament i la distribució de dades geoespacials lliures i a proporcionar dades geoespacials per l'ús i la compartició de tothom.
Els objectius de la fundació són:

- Custodiar els servidors i els serveis necessaris per mantenir el projecte OpenStreetMap. Concretament gestiona els servidors emplaçats a UCL (University College London) i el domini `www.openstreetmap.org`.
- Protegir el projecte en la mesura del possible d'atacs legals relacionats amb temes de copyright.
- Adquirir els fons necessaris per mantenir el projecte.

Però la fundació no és en cap manera propietària de les dades.

A més de la "base de dades geogràfica", i la plataforma per contribuir-hi OpenStreetMap també ha aportat formats de transmissió de les seves dades geogràfiques. Un d'aquests formats és una gramàtica de XML anomenada osm, que serveix per fer transaccions de les dades d'OpenStreetMap per la xarxa. També existeix el format binari pbf ("Protocolbuffer Binary Format") que aporta més compressió, i facilitat d'ús.

OpenStreetMap treballa i transmet la major part de les dades geogràfiques de que disposa en aquests formats, però aquests no són els únics llenguatges de descripció geogràfica. N'existeixen d'altres com el GML i el KML que a continuació es descriuran.

3.1.1.2 Geography Markup Language

Geography Markup Language (GML) és una gramàtica de XML (descrita amb un XML Schema) definida per l'Open Geospatial Consortium (OGC) per expressar característiques geogràfiques. El seu objectiu és permetre tant emmagatzemar com comunicar i representar aquestes dades. Cal notar que aquestes característiques (una mala traducció de features) són molt abstractes i generals i permeten modelitzar tota mena d'informació geogràfica, com

ara característiques vectorials (com carreteres o vies de ferrocarril), dades topològiques, imatges de terreny o fins i tot dades de sensors com pluviòmetres, etc.

Aquesta propietat d'integrar i aglomerar tot tipus de informació geogràfica és molt important per definir la raó de ser de GML.

Així doncs un dels motius més importants de la seva existència és proporcionar una "lingua franca" entre els diferents sistemes GIS existents, per tal que es pogués transportar informació d'un sistema a l'altre de manera més organitzada.

Cal notar també que GML no defineix explícitament *com* s'han de visualitzar les seves dades.

Com amb la majoria de gramàtiques basades en XML, GML consta de dos parts, un esquema (XML schema) que descriu el document i un document "instància" que conté les dades pròpiament dites.

El fet que un document GML sigui descrit per un XML schema permet als usuaris i als desenvolupadors descriure en els documents dades geogràfiques genèriques, com ara punts, línies o polígons, però també permet definir extensions del llenguatge per aplicacions més especifiques que defineixin conceptes no tant genèrics com ara un edifici, una carretera, etc.[21].

Així, fent servir aquests esquemes més específics també es podran entendre i comunicar mitjançant GML els desenvolupadors de diferents comunitats o sectors concrets.

La "universalitat" d'aquest llenguatge i el fet que sigui un estàndard molt utilitzat justifiquen els esforços per a intentar incloure'l en el projecte actual.

A l'annex podeu trobar alguns exemples de fitxers GML.

3.1.1.3 Keyhole Markup Language

Keyhole Markup Language és un llenguatge dissenyat per Keyhole Inc. per a la representació i visualització de mapes bidimensionals o tridimensionals.

Esta basat en XML i des de l'abril de 2008 és un estàndard de l'OGC (Open Geospatial Consortium)[58] cosa que el ratifica com a estàndard obert per a tots els geo-navegadors.

La seva popularitat rau en el fet que a l'any 2004 Google va comprar Keyhole Inc. i el seu visualitzador es va convertir en el conegut Google Earth.

Google maps també treballa amb el format KML, i així es va convertir ràpidament en un estàndard *de facto* per a la transmissió i representació de mapes per Internet.

La seva estructura és relativament similar a la de GML. Els fitxers KML especifiquen un conjunt de "features" (marques de llocs, imatges, polígons, models tridimensionals, descripcions textuals, etc.) per a ser visualitzats.

Cadascun d'aquests elements determina una longitud i una latitud, però també es dona la possibilitat d'afegir dades addicionals com ara "tilt" (inclinació), "heading" (rotació), altitud, etc. que defineixen una "posició de la càmera".

Un fitxer KML també pot contenir referències a dades addicionals no autocontingudes (per exemple un model tridimensional o una imatge). Això fa que sigui habitual trobar "publicacions" de paquets KML comprimits, anomenats KMZ. Això sovint no es fa només per guanyar compressió de dades, sinó per empaquetar tots aquests diversos fitxers addicionals i poder-los distribuir millor. KML també permet referenciar dades remotes que necessitin connexió a la xarxa.

El fet que permeti tantes capacitats fa que es tracti d'un llenguatge molt potent per a la representació de mapes. I també fa que algunes d'aquestes dades siguin obviades per a alguns clients, per exemple les dades 3D no solen ser considerades per clients que se centren exclusivament en informació de mapes cartogràfics, o que estan pensats per funcionar en dispositius de baix rendiment, com ara telèfons mòbils Android (que es el nostre cas).

A l'annex es poden veure alguns exemples de fitxers KML.

3.1.2 Breu introducció a Android

Android és un sistema operatiu per a dispositius mòbils (smartphones i tablets) basat en Linux. El projecte va ser iniciat per la empresa Android Inc, i actualment esta essent desenvolupat per la "Open Handset Alliance", liderada per Google, però de la qual unes altres 86 companyies (principalment fabricants de hardware) també en formen part.

Google publica part del codi d'Android sota la Llicència Apache, i per tant alguns l'etiqueten com a software-lliure, tot i que només es cert que ho és parcialment.

Al contrari del que passa amb altres sistemes operatius per a smartphones les aplicacions desenvolupades per a Android no han de ser validades per a la companyia que desenvolupa el sistema operatiu, i existeixen diversos canals de publicació d'aplicacions no centralitzats. Tot i que si que és cert que existeix un centre de distribució gestionat per Google, que és el que anteriorment s'anomenava Android Market i que ara es diu Google Play.

Una particularitat interessant relacionada amb la validació d'aplicacions, és que l'aplicació d'Android està molt controlada pel sistema operatiu. Per aconseguir aquest control les aplicacions requereixen de l'existència d'un document XML anomenat manifest, que declari les "intencions de l'aplicació". Això permet a l'usuari que la instal·la saber a quins serveis tindrà accés i decidir si vol instal·lar-la o no. O sigui que malgrat la manca de validació d'aplicacions per part de Google el sistema no està exempt de certs mecanismes de control.

Com es pot entreveure la filosofia d'una aplicació d'Android difereix lleugerament de la dels programes de software convencionals. A part de l'existència del manifest i l'execució "més controlada", el cicle de vida de les aplicacions també és "particular", i esta principalment condicionat per les característiques dels dispositius on s'executa, que tenen una funcionalitat principal de telèfon, bateria limitada, interfície tàctil, etc.

Un programa d'Android consta d'*Activities*, que es poden entendre com a "pantalles" que l'usuari veu i *Services* que són processos que no té perquè veure (poden córrer en segon pla). En primer lloc, les *Activities* estan pensades per poder ser "pausades" en qualsevol moment (per exemple si entra una trucada) i més endavant recuperar-ne l'execució, però sense necessitat de que el procés estigui actiu o consumeixi. Així que no s'espera que les aplicacions siguin "matades" o finalitzades mai, sinó que s'espera que el sistema s'encarregui de la gestió completa de la vida de l'aplicació.

Un conjunt d'Activities poden comunicar-se i cridar-se entre elles mitjançant *Intents*, que són objectes contenidors de dades per a la comunicació.

De en la mateixa manera a través d'Intents es pot controlar l'execució de *Services* en segon terme si és necessari. Aquest conjunt d'Activities, Services i Intents és el que normalment conforma una aplicació d'Android.

Aquestes i d'altres particularitats fa que, en comptes del terme "programes" s'utilitzi "aplicacions" al llarg de tota la memòria, això és així per emfatitzar aquesta subtil diferència de visió.

Cap a finals de 2011, hi havien més de 500,000 aplicacions disponibles per a Android.

Sigui per aquests motius o per alguns d'altres, el fet és que actualment Android és el sistema operatiu de mòbil amb més creixement amb diferència.
Es calcula que el primer trimestre del 2012, Android controlava un 59% del mercat mundial de smartphones, amb 331 milions de dispositius utilitzant-lo.

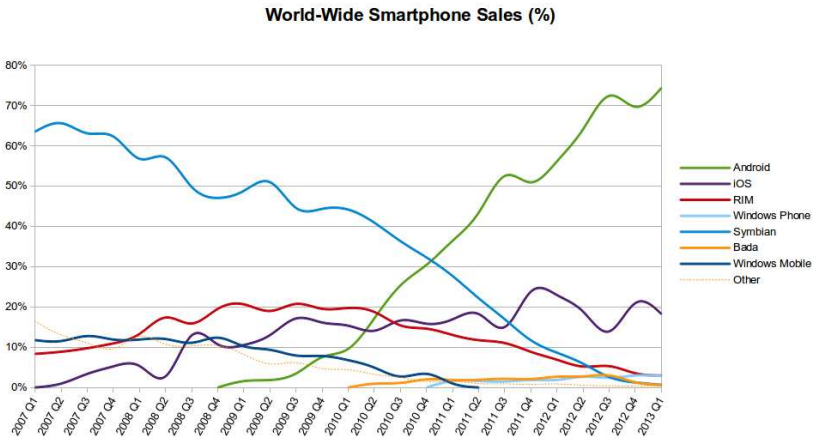

Figura 3.1: Percentatge de vendes de "smartphones" (font wikipedia).

A part d'aquestes impressionants dades, i de la vinculació de Android amb el món del software Open Source, la vinculació del present projecte amb el ZamiaDroid també ha condicionat l'elecció d'aquest sistema operatiu.

3.1.3 ZamiaDroid

El projecte *Sistema GIS per a la gestió d'espais naturals amb dispositius mòbils* (que ha esdevingut ZamiaDroid o Zamia, per simplificar) és un projecte sorgit de la mateixa facultat d'informàtica de Barcelona, realitzat l'any 2010 per David Martí Pino.

Els objectius principals del projecte eren inicialment de desenvolupar una aplicació per a dispositius mòbils per a la recopilació i gestió d'informació geogràfica d'estudis científics, a més de desenvolupar un servidor web per emmagatzemar les mostres científiques rebudes dels dispositius mòbils.

Un cop finalitzat el PFC el projecte va continuar desenvolupant-se (amb notables variacions sobre el plantejament original) en col·laboració amb departament de Biologia de la Universitat de Barcelona.

Zamia es el nom que ha adquirit l'aplicació de recol·lecció i gestió de dades geo-referenciades per a dispositius Android del projecte. I malgrat que el servidor de mostres continua existint la major par de la rellevància de tot projecte és actualment aquesta aplicació, així que a partir d'ara sempre em referiré a Zamia o ZamiaDroid al parlar d'aquest projecte.

Cal notar doncs, que el projecte Zamia es un projecte viu, en constant desenvolupament i utilitzat en aplicacions pràctiques per a la comunitat científica i per a qualsevol que el vulgui usar (ja que aquesta aplicació es troba a disposició de tothom).

Com molts PFCs, però, Zamia tenia certes ampliacions i millores proposades quan es va realitzar, algunes d'elles han estat implementades en aquests anys, d'altres han esdevingut obsoletes o impracticables degut als canvis d'orientació del projecte, i algunes millores o ampliacions romanen encara per implementar.

El present projecte pretén implementar algunes d'aquestes millores i suplir així algunes de les mancances de Zamia, o estendre'n les funcionalitats.

A continuació cito un extracte de l'apartat de *Futures ampliacions i millores* que es pot trobar a la memòria del PFC (que ha esdevingut ZamiaDroid).

Tot seguit es descriuran les possibles millores que es poden introduir al nostre sistema. Aquestes millores no han estat implementades per manca de temps [..]. S'intentaran dur a terme quan el projecte hagi finalitzat.

[..]

Millores

1. *Possibilitat de capturar mostres de figures que són representades per línies o polígons. Ens permetria el càlcul posterior d'àrees o longituds.*

2. *Possibilitat de correcció de la posició de les mostres sobre mapes. Si es pren una mostra i es veu que l'error del GPS ha estat massa alt es pot donar la possibilitat de canviar la posició de la mostra a un altre indret del mapa.*

A part d'aquestes millores i ampliacions proposades a la memòria del PFC, la resta d'altres objectius del present projecte també responen a una voluntat d'ampliar i millorar el Zamia, com ara la utilització de OpenStreetMap, que permetria prescindir del mapes de Google que ara utilitza l'aplicació, o les funcionalitats fora de línia, etc. I han estat definits amb connivència dels responsables de ZamiaDroid[71].

3.1.4 Erdapfel

Erdapfel és el nom amb el qual he batejat l'aplicació autònoma de demostració de les funcionalitats de la biblioteca (i el nom que faré servir d'ara en endavant per referir m'hi).

Es tracta d'una aplicació d'Android desenvolupada íntegrament en Java que permet visualitzar mapes de diversos orígens al dispositiu on s'executa, així com generar noves dades, realitzar càlculs simples sobre els mapes i gestionar els mapes "offline" locals del dispositiu.

Tot i que inicialment aquesta aplicació de demostració havia de ser molt menys rellevant en el projecte, i només havia de mostrar que les funcions implementades en les biblioteques eren operatives, ha acabat guanyant importància ja que la major part de la integració amb el projecte Zamia en fa ús.

Així doncs, tot i que l'objectiu d'erdapfel continua essent demostrar les capacitats de les biblioteques de mapes implementades, en realitat també realitza la funció d'ampliar les capacitats de Zamia (o qualsevol altre aplicació que en faci us) de manera senzilla.

Més endavant s'explicaran els motius d'haver acabat utilitzant erdapfel per gestionar i coordinar aquesta integració amb Zamia, i no algun altre mitjà.

Com a curiositat, he triat el mot erdapfel per batejar el programa, ja que em va semblar un nom atractiu i a l'hora rellevant en l'àmbit cartogràfic.

Prové de l'alemany, i la seva traducció literal seria "poma de la terra" o "poma del món". Vaig pensar que seria un nom idoni, ja que és el nom d'una famosa bola del món. De fet, es el nom amb el que es coneix el globus terraqüi més antic preservat actualment. Va ser fabricat per Martin Behaim al 1492, i, per tant, el continent americà encara no hi és present, cosa que no deixa de ser divertida.

3.2 Introducció a l'estudi tecnològic

La cartografia és un camp amb un llarg recorregut en els mitjans informàtics, i les capacitats GPS dels dispositius mòbils els han fet especialment atractius i propicis al desenvolupament d'aquesta ciència.

Així que no es d'estranyar que existeixin moltes alternatives per permetre mostrar i interactuar amb mapes cartogràfics per als dispositius mòbils. I menys encara si tenim en compte que Google, empresa que ha desenvolupat el sistema Android, també ha dedicat una part important dels seus recursos a desenvolupar aplicacions de mapes, i que una de les funcionalitats de Google més populars és el maps.

Però a part de les pròpies solucions de Google moltes d'altres han aparegut. Algunes basades en els mapes de Google i d'altres en alternatives lliures com OpenStreetMap.

El següent estudi pretén valorar algunes d'aquestes alternatives existents, tant aplicacions com biblioteques de mapes. Tenint sempre present des d'un bon començament que la nostra proposta havia de complir els objectius marcats anteriorment i per tant havia de ser software lliure, no s'ha estudiat cap tecnologia existent de software privatiu.

Això ha exclòs un bon nombre de possibilitats, però tot i així el nombre de solucions analitzades és considerable, tal i com es podrà veure a continuació.

A més d'estudiar solucions de mapes, també s'ha dedicat part de l'esforça a estudiar la plataforma Android, i més concretament a estudiar la possibilitat de treballar-hi amb llenguatges de programació alternatius a Java (concretament Python).

3.3 Aplicacions existents i solucions que utilitzen

A continuació segueix un estudi a nivell funcional d'algunes aplicacions de mapes existents.

Malgrat que no s'ha pogut trobar cap aplicació que satisfés tots els nostres objectius, cadascuna de les aplicacions analitzades ha aportat valuosa informació que més endavant ha servit per confeccionar el producte resultant d'aquest PFC.

Com podreu comprovar no s'han valorat tan sols aplicacions d'Android, sinó que també s'han valorat algunes aplicacions d'escriptori. Això ha estat

degut a que, d'haver cobert els objectius del projecte, s'hauria plantejat la possibilitat de portar-les a Android.

3.3.1 ZamiaDroid

ZamiaDroid és una aplicació de gestió de projectes GIS que ha estat alhora inspiració i punt de partida del present projecte. És l'aplicació que ha de fer un ús més directe de les capacitats de la biblioteca, així que per estudiar-la es volen valorar les seves "limitacions" o coses a millorar, però també es vol entendre la filosofa de l'aplicació i la raó de ser de les seves funcionalitats per no obstruir-ne el bon us futur.

La visió que mostro en aquest estudi és a nivell purament funcional, però tenint en compte que he pogut estudiar l'aplicació des d'una posició privilegiada, ja que he pogut debatre'n les característiques i la evolució amb el seu creador i principal desenvolupador, i també considerant que la integració amb ZamiaDroid és un dels objectius del projecte, l'estudi resultant és més extens que el de la resta de solucions observades.

Com s'ha dit a l'apartat introductori l'aplicació ZamiaDroid sorgeix d'un PFC i evoluciona des dels primers estadis podríem dir-ne acadèmics més genèrics, a l'aplicació força més definida que és avui en dia. Aquesta evolució ha estat constant i a vegades sobtada i/o precipitada, fruit de les necessitats dels projectes o estudis científics que havia d'ajudar a portar a terme.

Ara mateix ZamiaDroid permet a grans trets:

- Crear i gestionar projectes
- Prendre Citacions
- Editar Citacions
- Veure Citacions sobre un mapa
- Veure fotografies de citacions com a galeria

La creació i gestió de projectes és una funcionalitat important dins del ZamiaDroid, però no te relació directa amb el camp concret que el present projecte pretenen tocar (visualització i edició de mapes cartogràfics).

Un projecte de ZamiaDroid és una unitat lògica per agrupar dades d'estudis científics i és molt versàtil. Però a grans trets, i per les finalitats d'aquest estudi, entendrem un projecte del ZamiaDroid com la definició de formulari d'un estudi científic concret, és a dir, d'una especificació de quins camps han de tenir les mostres de l'estudi que prendrem.

Una citació, en l'àmbit de ZamiaDroid, és una d'aquestes mostres. Una instància concreta d'aquests camps o formulari. I a més de les dades definides per l'usuari al crear el projecte (que poden ser una gran varietat de camps i informació de molts tipus) pot contenir informació geogràfica (de fet, aquesta és una de les dades més importants de la citació).

Les citacions es prenen omplint un formulari amb camps definits per l'usuari que poden ser textos, fotografies, etc. i en el moment de prendre-les es te l'opció de que mitjançant el GPS del dispositiu s'incorporin les dades geogràfiques de la ubicació física on ens trobem automàticament.

Això vol dir que en un cas d'us usual del Zamia no fa falta visualitzar un mapa i ubicar les citacions en ell, sinó que simplement es prenen les dades de la mostra (qualsevulla que siguin aquestes) i la citació queda enregistrada amb les dades geogràfiques pertinents.

Evidentment, després aquestes dades poden ser "pintades" a sobre d'un mapa i l'usuari de Zamia les pot observar i fins i tot modificar des d'allà.

De fet hi han diversos casos d'us per generar citacions. Per exemple des de la mateixa pantalla de "veure citacions" es poden crear també noves citacions, i editar les existents. Però al capdavall el que ens interessa és que existeix la possibilitat de mostrar les dades sobre un mapa, i que aquest mapa és proporcionat per GoogleMaps.

La utilització de la solució de mapes de Google ve des del començament del desenvolupament del projecte, i de fet es pot considerar l'opció natural si es volen desenvolupar aplicacions que facin servir mapes en Android, ja que està molt ben integrada, és força versàtil i molt ben documentada.

Tot seguit esmentaré en termes generals el procediment que segueix ZamiaDroid per utilitzar l'API de mapes de Google, que, de fet, és el procediment genèric que s'empra en la gran majoria d'aplicacions Android que en fan ús.

En primer lloc cal obtenir una "clau de desenvolupament" proporcionada per Google que et permeti fer servir l'API i afegir aquesta clau al manifest de l'aplicació Android.

Després es pot crear una classe Java que hereti de MapView (que al seu temps hereta d'Activity).

Mitjançant aquest procediment es poden modificar les característiques del mapa a mostrar, i de fet aquesta és la metodologia més estesa quan es treballa amb APIS de Google, heretar de les classes de l'API i estendre-les segons les necessitats concretes del programador. Així si es vol afegir una capa d'informació sobre aquest mapa s'hereta de la classe Overlay, per exemple, i aquest Overlay propi s'afegeix al MapView, etc.

Un cop fet això, normalment només cal tractar de manera adient aquesta classe i integrar-la amb la resta d'activities de l'aplicació.

Actualment (des de febrer de 2013) ha aparegut una nova versió de l'API que han anomenat v2, que permet (entre d'altres coses) visualitzar mapes tridimensionals, i fa servir un procediment lleugerament diferent al esmentat anteriorment. Però quan es va realitzar l'estudi encara no estava disponible, i Zamia no la utilitza (no es preveu la seva utilització a curt termini) així que no la comentarem a l'estudi.

ZamiaDroid també fa ús d'una biblioteca externa anomenada MyTracks que proporciona un servei per guardar i mostrar les rutes recorregudes pels dispositius mòbils, que fa servir el gps i interactua amb GoogleMaps.

Zamia a més permet exportar les dades recollides d'un projecte en els següents formats:

- KML: Un esquema d'XML que representa un format genèric i estàndard per representar dades sobre mapes del que ja s'ha parlat aquí.
- JSON: Acrònim de JavaScript Object Notation) és un format popular per la transmissió de dades. El projecte final de carrera original ja feta servir JSON com a format de comunicació entre el que era el servidor de mostres i el programa client del dispositiu mòbil, es tracta d'un format lleuger i còmode i al hora versàtil i és un estàndard molt estès.
- Fagus: Es tracta d'un format propi, una variant d'XML preparat per a ser usat en un programa de gestió de plantes creat pel departament de la Facultat de Biología, que també desenvolupa el Zamia. (`http://biodiver.bio.ub.es/veganaweb/main/?section=../fagus/content.jsp`).
- TAB: Un fitxer de text pla amb les dades separades per tabulacions que permet treballar fàcilment amb programes de full de càlcul.

- Zamia: Es un fitxer XML amb un format propi pensat per a ser utilitzat en el mateix dispositiu mòbil com a còpies de seguretat, o com format de comunicació entre instàncies del programa en altres dispositius.

Zamia presenta moltes d'altres funcionalitats més o menys específiques per a les tasques de recol·lecció de dades i/o per la ajuda en aquesta obtenció, com ara la comunicació amb un servidor per obtenir llistes taxonòmiques (Thesarus) que facilitin la introducció de dades, o consultes a bases de dades remotes, però a grans trets aquestes serien les seves funcionalitats.

Com es pot veure, en una aplicació tant encarada al treball de camp, la possibilitat de treballar amb mapes offline resultaria molt pràctica, per a poder recol·lectar dades i/o visualitzar-les sobre el terreny fins i tot quan no es disposa d'una bona connexió de xarxa o no n'hi ha cap de disponible.

A més la filosofia de codi obert del projecte lliga molt bé amb la utilització d'OpenStreetMap, i poder fer-lo servir també beneficiaria al projecte.

En resum, tot i que ZamiaDroid és una aplicació pràctica que s'està fent servir actualment de manera satisfactòria, també una aplicació en constant desenvolupament que es podria beneficiar d'algunes millores no prioritàries pel seu equip de desenvolupadors però si bones de tenir.

3.3.2 OsmAnd

OsmAnd (OSM Automated Navigation Directions) és una aplicació de navegació basada en OpenStreetMap.

Com a característiques més interessants destaco:

- Mostrar mapes vectorials d'OSM (*.osm.pbf, *.obf) i altres tipus de mapes basats en cel·les (Mapnik, Osmarender, Google Maps/Satellite/-Terrain, CloudMade, Cyclemap, OpenPisteMap, MapSurfer.Net, Microsoft Maps/Earth/Hybrid)

- És capaç de mostrar mapes offline.

- Permet afegir POIs (punts d'interès) a OpenStreetMap.

El primer que l'aplicació et comenta al iniciar-la per primer cop és que cal que et descarreguis mapes de les zones que necessites, i que la versió gratuïta te un limit de 10 mapes.

Els mapes de zones tenen una mida considerable, de l'ordre de 30Mb per una zona dels voltants d'una ciutat gran, i fins a 400Mb per un país sencer com Anglaterra o Espanya.

Les funcionalitats de l'aplicació estan pensades per ajudar en la navegació de mapes, sobretot per la conducció amb automòbil, tot i que també te modes per a vianants i ciclistes.

A més de veure i navegar els mapes OsmAnd pot ser usat per millorar la qualitat de les dades de OpenStreetMap. Des de la mateixa pantalla de navegació es dona accés fàcil per poder crear i publicar nous punts d'interès (POI) o corregir errors de les dades d'OSM.

Per aconseguir això cal configurar OsmAnd perquè faci servir la teva informació de registre d'OSM des de les preferències, i un cop realitzat això es poden crear nous POIs des d'un menú contextual sobre el mapa.

De manera semblant es poden proposar correccions d'errors de les dades OSM.

Aquesta funcionalitat resulta interessant perquè facilita l'aport comunitari i la voldria destacar especialment.

Malgrat això l'enfoc d'aquesta aplicació és força diferent a les necessitats del projecte, ja que dona molta prioritat a les capacitats de navegació de mapes mitjançant GPS i la seva capacitat per suportar mapes fora de línia, tot i que és molt marcada, no sembla donar facilitats per a la utilització de mapes gestionats per l'usuari de manera local, sinó més aviat pretén que es facin servir mapes proporcionats pels serveis de l'aplicació.

Tot i així és possible treballar amb mapes offline gestionats pels usuaris. El mecanisme que es proporciona per generar mapes d'OsmAnd és processar dades en format OSM amb una aplicació d'escriptori externa (OsmAndMapCreator) i transferir el fitxer resultant en format *.obf al dispositiu que es desitgi.

Figura 3.2: Pantalla principal d'OsmAnd.

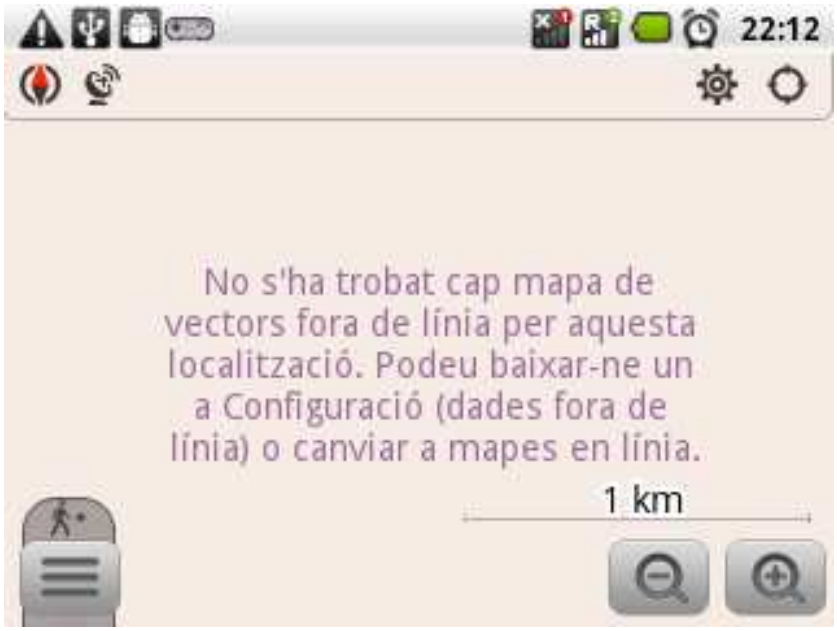

Figura 3.3: Captura d'OsmAnd.

Figura 3.4: Captura del gestor de mapes offline d'OsmAnd.

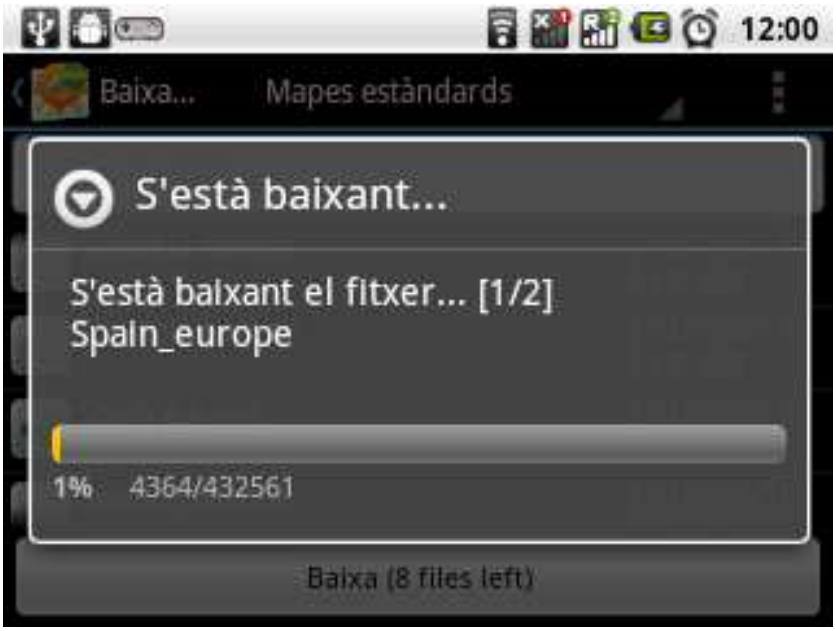

Figura 3.5: Captura de descarrega de mapes offline d'OsmAnd.

3.3.3 OruxMaps

OruxMaps és un cas una mica especial, ja que no és software lliure, tot i que si software "gratuït", però l'afegeixo a l'estudi ja que em va interessar perquè utilitza la biblioteca (que si que és software lliure) mapsforge, que més endavant explicaré amb més deteniment, i em va anar bé analitzar l'aplicació a nivell funcional per comprendre les capacitats d'aquesta i com han estat aprofitades per a desenvolupar aplicacions de GIS.

Oruxmaps permet treballar en dos modes, el mode online, que permet veure diversos tipus de mapes (GoogleMaps, OpenStreetMap, Microsoft Maps, etc.), i el mode offline amb mapes generats especialment per a l'aplicació des d'una eina d'escriptori.

També permet enregistrar camins fent servir el GPS, i exportar aquests camins en format gpx i kml.

A més inclou funcions de navegació per veu, de càlcul d'estadístiques sobre els recorreguts realitzats, alarmes de proximitat, suport pel "geocaching", monitorització de freqüència cardíaca fent servir dispositius bluetooth externs, etc.

El funcionament de l'aplicació és una mica confús, ja que te moltes funcionalitats i la pantalla es veu una mica plena d'opcions, i això fa que a nivell d'interfície costi una mica adaptar-se a l'aplicació.

A part d'això Oruxmaps resulta un programa molt complert i extens, que mostra la capacitat de la biblioteca mapsforge per a mostrar mapes de diverses fonts (entre elles de OpenStreetMap) i les seves capacitats offline.

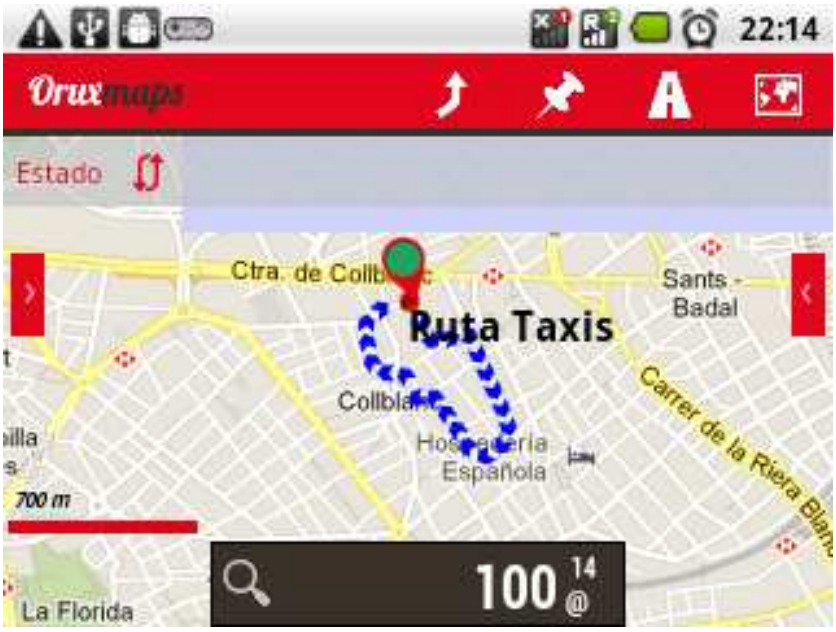

Figura 3.6: Captura de la pantalla principal d'oruxmaps.

Figura 3.7: Captura dels menus d'oruxmaps.

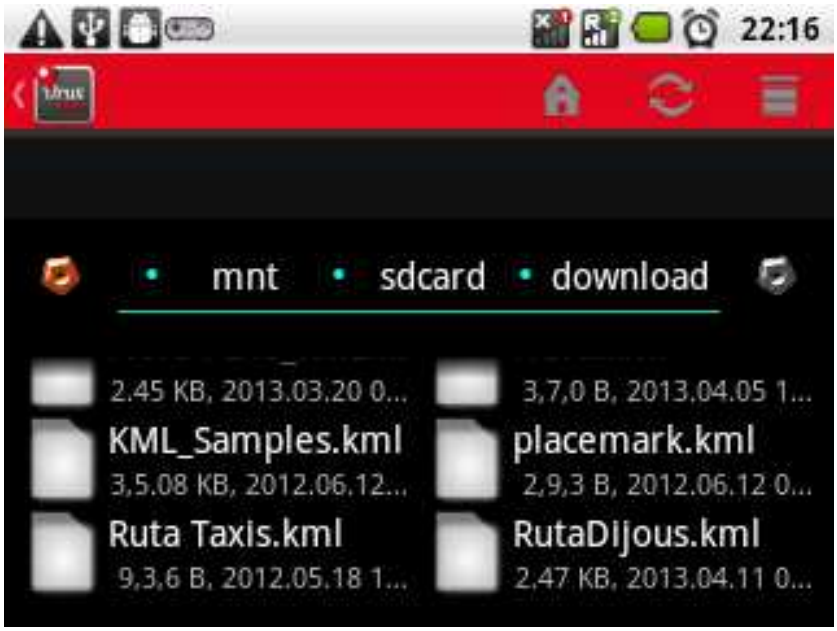

Figura 3.8: Captura de selecció de fitxers d'oruxmaps.

3.3.4 gvSIG Mini

gvSIG és un projecte que va néixer l'any 2004 a València per tal de migrar els sistemes de la Conselleria d'Infraestructures i Transports (de València) a sistemes de software lliure. Això significa que a priori tenien uns objectius i una manera de fer molt lloables, i també recursos per portar a terme aquests objectius. Però al mateix temps aquests objectius pretenien solucionar problemes força concrets i específics, molt enfocats a les necessitats del ministeri.

El projecte gvSIG ha anat desenvolupant diferents productes, per a diferents escenaris i entorns.

La versió de gvSIG pensada per funcionar en dispositius Android s'anomena gvSIG Mini per Android (o gvSIG per abreujar). Això significa que es tracta d'un port de la versió de gvSIG per a dispositius mòbils (gvSIG Mobil) a Android.

gvSIG mini per Android és una aplicació de mapes molt complerta i permet visualitzar mapes de "tiles" (o tessel·les) provinents de fonts molt diverses, per exemple pot descarregar-se les tessel·les generades a partir de OpenStreetMap o el que proporciona GoogleMaps, o Bing (de Microsoft) o els de YahooMaps per anomenar-ne uns quants.

I també dona la possibilitat de guardar aquestes tessel·les descarregades a la memòria del dispositiu i disposar-ne encara que no es tingui connexió a la xarxa (offline). El problema amb això és que es un mètode que descarrega una gran quantitat d'imatges i a vegades (depenent de quants nivells de zoom i de quina àrea es demani descarregar) pot exhaurir l'espai del dispositiu o col·lapsar els servidors (sobretot els de OpenStreetMap i d'alguns altres serveis de mapes, que tenen límits força restrictius per evitar descàrregues de tessel·les en massa).

Permet a més carregar capes addicionals sobre els mapes des de servidors externs que compleixin els estàndards de WMS (Web Map Service), TMS (Tile Map Service) o des de fitxers de capes en format text pla.

Permet cercar adreces i direccions.

També permet utilitzar-lo com a navegador d'adreces (similar al navegador d'un automòbil), tot i que aquesta funcionalitat no funcionava del tot bé en el dispositiu on vaig provar l'aplicació.

Figura 3.9: Menú principal de gvSIG.

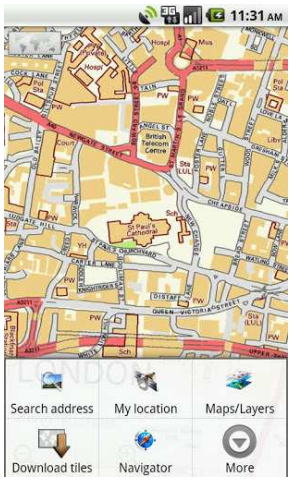

Figura 3.10: Pantalla principal de gvSIG.

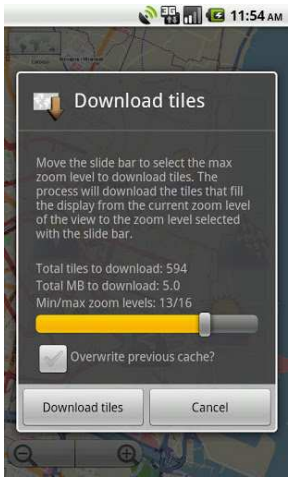

Figura 3.11: Menú de descàrrega de tessel·les de gvSIG.

Figura 3.12: Pantalla de configuració de gvSIG.

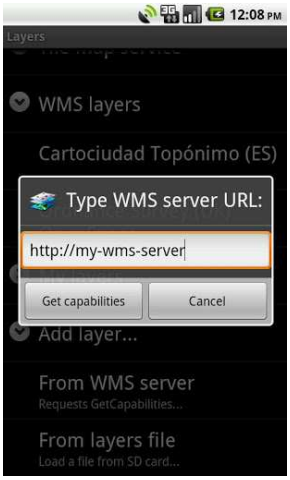

Figura 3.13: Pantalla de configuració de servidors de mapes.

3.3.5 Quantum GIS for Android

Quantum GIS (QGIS) és un Sistema d'informació Geogràfica complet i molt potent, de codi obert (GNU Public License) i multi-plataforma (corre en Linux, Unix, Mac OSX, i fins i tot Windows). Permet treballar amb la majoria de formats vectorials, "raster" i formats de bases de dades.

El QGIS proporciona diferents aplicacions que poden ser usades segons convingui:

- QGIS Desktop Una aplicació d'escriptori que ofereix funcionalitats usuals en sistemes GIS per a visualitzar, editar i analitzar informació geogràfica.
- QGIS Browser Una aplicació centrada a visualitzar simples i ràpidament dades locals o online (WMS).
- QGIS Server Uns servidor que respecta L'estàndard WMS 1.3 que pot ser configurat fàcilment fent servir projectes realitzats amb l'aplicació QGIS Desktop.
- QGIS Client Una "façana" (front-end) basada en OpenLayers i GeoExt, destinada a resoldre les necessitats de proveir mapes basats en la web.

Quantum GIS, però, no estava encarat a dispositius mòbils.

Això va canviar en ocasió de l'esdeveniment Google Summer code de 2011, quan es va crear una adaptació anomenada "QGIS mobile" pensada per funcionar en sistemes Android.

Inicialment estava plantejat com una conversió 1-a-1 de l'aplicació QGIS d'escriptori que tenia com a objectiu les plataformes Android amb pantalles més grans (tauletes d'unes 10.1").

Més endavant s'ha estat transformant en una aplicació també pensada per telèfons, amb una interfície d'usuari més minimalista. Però el desenvolupament d'aquesta versió encara està en execució.

Aquest és un "port" molt interessant pel projecte actual, ja que ha estat clau i molt interessant per comprendre i avaluar la complexitat de portabilitat d'un projecte en c++ i que utilitza les biblioteques gràfiques Qt a un entorn tan peculiar com Android.

En aquest cas concret, fent us de la "traducció" de les biblioteques Qt a Android, anomenades Necessitas, i més concretament de l'aplicació Ministro II, que les descarrega, el QGIS mobile es capaç de mostrar mapes amb

una interfície basada en Qt (no del tot igual a la de l'aplicació d'escriptori naturalment, però si basada en ella).

Ara bé, després d'informar-me de la solució que plantejaven, i de l'excitament inicial, van sorgir complicacions a l'intentar avaluar l'aplicació proporcionada.

El sistema d'instal·lació de totes les biblioteques necessàries per l'execució de QGIS for Android és força complex i llarg. Els desenvolupadors del projecte proporcionen eines per facilitar-lo, però el fet que depengui de tants factors pot ser un dels elements que en compliquen la detecció de problemes. En primer lloc cal instal·lar tot el suport de les biblioteques de Qt a Android, per això l'aplicació Ministro II és de molta utilitat, i es un Procés que vaig aconseguir finalitzar aparentment sense cap error perceptible.

Posteriorment cal instal·lar una de les versions de l'aplicació pròpiament dita, que amb la ajuda d'un programa instal·lador ens permeten seleccionar.

A part de les diferents versions més o menys estables (o com a mínim provades fins a cert punt) i les versions més recents possiblement inestables. També existeixen alguns empaquetaments específics que contenen optimitzacions (armeabi v7a) que només poden ser aprofitades per a alguns dels models més nous de dispositius.

Arribats en aquest punt es van provar extensivament la majoria de les versions disponibles en aquell moment, i es van obtenir resultats similars per a totes elles (errors greus que impedien el carregament de l'aplicació).

Després de comprovar que sorgien problemes similars en executar el programa en tots els dispositiu de proves físics de que disposava en aquell moment (HTC Magic, Google Nexus One i un Samsung Galaxy Tab 7), i veient com també resultava impossible executar-lo degudament en cap de les instàncies del emulador d'Android.

Es va concloure que existien greus problemes d'incompatibilitat relacionats amb els dispositius que teníem a l'abast.

A continuació mostro una taula amb els dispositius que segons la web oficial del projecte QGIS for Android, a dia d'avui permeten la seva utilització sense cap mena de problemes:

Device	Android version	Status
ASUS Transformer	3.2	OK
ASUS Transformer TF300	4.0	OK
Lenovo ThinkPad 6 Tablet	3.1	OK

Com es pot veure es tracta d'un segment de dispositius encara força limitat i de gamma més aviat alta.

Tal i com ho vam interpretar, aquest fet posa en evidència les dificultats inherents del plantejament de "portar" aplicacions d'escriptori complexes a la plataforma Android, i va ser un dels motius per acabar realitzant el projecte tal i com esta plantejat.

Com a apunt final cal dir, però, que QGIS for Android és una opció molt atractiva i encara en constant desenvolupament, a la qual és interessant seguir la pista, ja que en un futur no gaire llunyà podria resultar una eina molt potent en el camp d'estudi que tractem.

3.3.6 pyroute

Pyroute es un programa escrit en Python que semblava molt interessant per les similituds amb el plantejament del projecte present.

Es tracta d'una versió gràfica per a dispositius mòbils d'una interfície de línia de comandes pensada per calcular rutes sobre mapes de osm.

Tal i com s'especifica a la pagina de Pyroute de la wiki de open street maps, esta pensat per córrer sobre Linux, windows i OpenMoko, que es un sistema operatiu lliure per a dispositius mòbils, però no àmpliament suportat.

Utilitza Python, naturalment, però també cairo i Gtk per a mostrar les interfícies i els gràfics (a traves de pycairo i pyGtk).

Per desgràcia les biblioteques de cairo i pyGTk no sembla que estiguin disponibles actualment a la plataforma Android, així que intentar portar aquest projecte a Android va veure's poc viable.

A més existeix la possibilitat que el projecte pugui estar abandonat, ja que a la pagina es menciona que s'està reemplaçant per un altre projecte anomenat "rana" que tampoc sembla molt actiu.

3.3.7 gmapcatcher

Interessant per la seva capacitat offline molt marcada i per haver estat desenvolupat en Python, comprendre els mecanismes que utilitza per a suportar mapes offline sempre és una aportació interessant al projecte, i a més, un possible port a Android hauria estat molt interessant.

El gmapcatcher és una aplicació d'escriptori capaç de descarregar-se de diverses fonts d'internet cel·les de mapes (tiles) i guardar-los per a ser usats offline.

Alguns dels proveïdors de mapes dels quals es pot descarregar la informació són OpenStreetMap, CloudMade, Yahoo, OpenCycleMap o Seznam. Anteriorment es podia descarregar també de Google maps, però degut a una queixa de Google respecte a l'incompliment dels seus termes d'us ara ja no es permet[12].

Naturalment, a banda de descarregar-se aquesta informació de la xarxa, el programa també pot gestionar aquest mapes, carregar-los de fonts locals, i proporciona una interfície per permetre a l'usuari seleccionar quines parts d'aquests mapes es volen guardar.

A continuació segueix una petita avaluació funcional del programa. Per a informació més detallada podeu consultar la seva guia d'utilització[11]:

La interfície principal del programa consta d'un mapa envoltat de barres d'eines (Figure 3.14).

A traves de les barres d'eines es pot controlar el nivell de zoom del mapa, cercar llocs, especificar el mode de treball (online/offline), accedir al menú de configuracions, etc.

La funcionalitat més destacada i que resultava més interessant és la de descarrega de mapes.

Per descarregar una zona del mapa cal clickar amb el ratolí sobre el mapa i seleccionar l'opció "batch download" del menú contextual desplegable que apareix (Figure 3.15). Això es podria traslladar a Android com a una pressió sobre el mapa durant un temps prolongat.

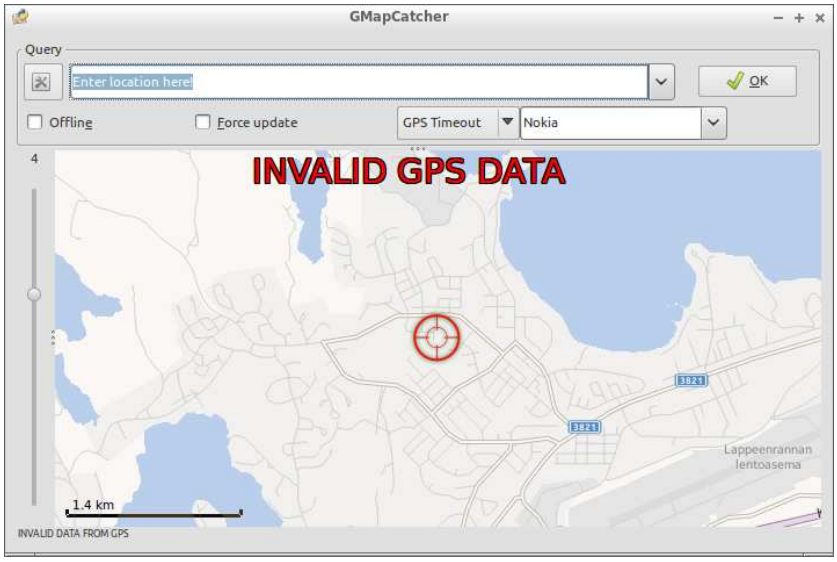

Figura 3.14: Interfície principal de gmapcatcher.

Un cop triada aquesta opció apareix un menú que et permet centrar la posició del mapa i definir-ne l'àrea a descarregar, així com el nivell mínim i màxim de zoom que es guardarà (Figure 3.16):

Així l'us normal del programa seria obrir-lo en mode online, descarregar les parts de mapa que es desitgen als nivell que es vulguin després estarien disponibles permanentment.

La mida dels mapes, però, continua sent molt gran, sobretot si es tenen nivell de zoom molt detallat. Aquest és un "defecte" que tenen tots els sistemes que he vist d'emmagatzemar mapes de tessel·les.

A part d'aquesta funcionalitat, però, l'aplicació no proporcionava gaires més opcions. Així que per desgràcia les capacitats del projecte eren més limitades del que necessitàvem.

Per exemple, mancaven les opcions de carregar mapes en kml i gml, un dels objectius del projecte. Cosa que es podia haver afegit a gmapcatcher.

Però també es va considerar que els mètodes d'interacció amb l'usuari que proporcionava gmapcatcher eren massa complexos i fragosos, molt dependents dels mètodes d'entrada tradicionals (teclat i ratolí) i amb uns dissenys de menús bastant "petits", i per tant molt difícilment portables a dispositius com els telèfons mòbils.

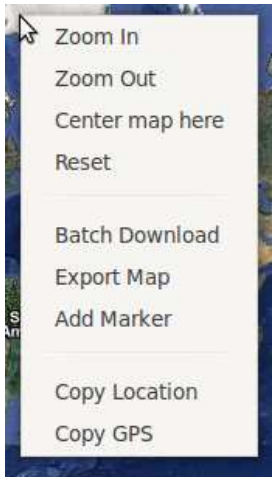

Figura 3.15: Menú desplegable contextual.

Si a més valorem que actualment no es permetia dibuixar línies i polígons als mapes mostrats, i la informació dels marcadors de posició que sí que permetia afegir era molt limitada, i que per tant això era una altre aspecte que també s'hauria de modificar, i que totes aquestes característiques es desviaven de l'objectiu inicial del projecte gmapcatcher.

A més cal afegir la dificultat de portar a Android projectes com aquest, degut a la manca de suport per a les biblioteques Gtk i a la complexitat de portar amb èxit projectes de Python a Android.

Per totes aquestes raons es va decidir simplement estudiar a nivell de funcionalitats i comportament aquest projecte, i no d'implementació.

Tot i així és important mencionar-lo per considerar-ne els mètodes que utilitza de gestió de mapes offline i també per aclarir alguns dubtes que existien respecte les polítiques i els termes d'us d'alguns proveïdors de mapes (concretament Google) en referència a la descarrega en massa de informació.

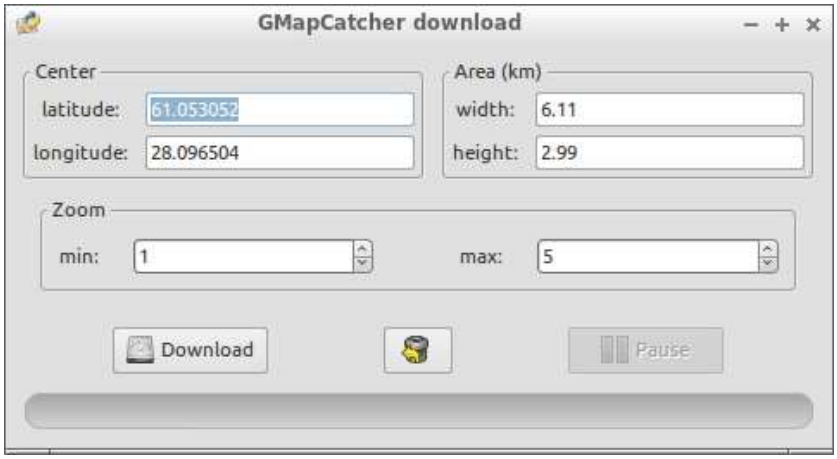

Figura 3.16: Menú de descàrrega.

3.3.8 Taula resum d'aplicacions

A continuació es pot veure una taula/resum de les aplicacions considerades i si tenen o no les capacitats marcades com a objectius del projecte: Android (and), Mapes sense connexió a la xarxa (offline), carregar mapes s'OpenStreetMap (osm), carregar fitxers KML (kml), carregar fitxers GML (gml), dibuix de polígons i altres formes geomètriques (poly), realitzar càlculs sobre mapes (calc) i ser software lliure (open).

Aplicació	and	offline	osm	kml	gml	poly	calc	open
ZamiaDroid	✓	✗	✗	✗	✗	✗	✗	✓
OsmAnd	✓	✓	✓	✗	✗	✗	✗	✓
OruxMaps	✓	✓	✓	✓	✗	✗	✗	✗
gvSIG Mini	✓	✓	✓	✗	✗	✗	✗	✓
QGIS	~	✓	✓	✓	✓	✓	✓	✓
pyroute	✗	✗	✓	✗	✗	✗	✗	✓
gmapcatcher	✗	✓	✓	✗	✗	✗	✗	✓

~ = suport parcial ✓= suportat ✗= no suportat

3.4 Motors de render i biblioteques analitzades

Després d'analitzar aplicacions, es van cercar biblioteques que poguessin ajudar a complir els requisits del PFC. Algunes de les biblioteques són les mateixes que fan servir les aplicacions analitzades.

En aquest cas no tan sols s'ha valorat les biblioteques a nivell funcional, ja que a més de cercar biblioteques que ja complissin els objectius del projecte, també es posava atenció a valorar-les com a possible base per la biblioteca d'aquest PFC, és a dir, també s'ha valorat com podrien ser esteses per afegir-hi les funcionalitats que hi mancaven.

Tal com succeïa a l'estudi d'aplicacions, no s'han valorat exclusivament biblioteques implementades en Android. També s'han valorat biblioteques dissenyades per altres plataformes, i la possibilitat que eventualment poguessin ser "portades".

3.4.1 osmdroid

osmdroid és un projecte de biblioteca per a proporcionar eines i mètodes de visualització de dades de OpenStreetMap per a la plataforma Android.

Entre d'altres coses vol proporcionar una biblioteca de mapes alternativa i lliure a la de Google, però amb una API similar.

osmdroid va ser creat per Nicolas Gramlich, que encara n'és propietari de projecte. El creador de osmdroid va abandonar parcialment el projecte i va dedicar els seus esforços a AndNav, que era un projecte de implementació de navegació gps basat en osmdroid, però que sembla a ser que no ha estat actualitzat des de 2010.

Deixant de banda això osmdroid sembla que encara està mantingut per una comunitat de desenvolupadors.

I segons la seva web[16] ha estat utilitzat en bastants projectes, ja sigui utilitzant la biblioteca que proporcionen íntegrament o bé fent un fork del seu codi font i modificant-la. Alguns d'aquests projectes són OsmAnd, AndNav2, OpenGPX, etc. Aquesta confiança dels desenvolupadors, en certa manera els avala.

Un fet negatiu a destacar és que les biblioteques osmdroid no proporcionen un clar suport per treballar fora de línia. De fet, als fòrums de desenvolupament d'osmdroid, existeixen discussions per afegir-hi aquesta capacitat, però encara no hi ha una solució definitiva.

I el fet d'estar totalment centrat en mostrar mapes osm implicava que, de fer servir aquesta biblioteca pel projecte caldria implementar la part de kml i gml a part.

Malgrat això aquesta era una opció que es va tenir molt en compte a l'hora d'escollir una biblioteca en la que basar-se per implementar el projecte, i de no haver trobat solucions més adients hauria estat la escollida.

3.4.2 osmarender

Osmarender és un motor de renderitzat basat en regles per a generar imatges SVG a partir de dades OSM.

Pren com a dades d'entrada un fitxer de regles i un conjunt de dades d'OpenStreetMap, i genera un fitxer gràfic de dades vectorials en format SVG que concorda amb l'estil definit al fitxer de regles.

Aquest motor de renderitzat semblava força limitat a generar imatges "estàtiques" i mancava de suport per a mètodes d'entrada i per a interaccionar amb els mapes generats i editar-los de manera gràfica (afegir punts d'interès, etc).

Aquesta simplicitat, i la independència d'una plataforma concreta que aporta treballar amb SVG, el feia (a priori) més senzill de portar a Android que d'altres alternatives.

Però aquest mateix plantejament "simplista" del projecte implicava haver de fer moltes modificacions per poder-hi implementar tots els objectius que requeríem.

Malauradament, a més, el projecte no esta essent mantingut des del març de 2012 cosa que feia témer que ens mancaria suport per a qualsevol problema que poguéssim tenir. Per això es va preferir estudiar més a fons altres alternatives.

3.4.3 OpenLayers

OpenLayers és un famós software que permet mostrar un mapa dinàmic a qualsevol pàgina web. És completament gratuït i de codi obert (llicència FreeBSD) i esta programat en JavaScript.

La portabilitat de la tecnologia web va fer plantejar-nos-en la utilització per a mostrar els mapes a Android.

Dintre de la comunitat de software lliure OpenLayers és una de les solucions més utilitzades per afegir mapes interactius a les webs.

D'haver optat per a la utilització de tecnologia web, aquesta opció s'hagués estudiat amb més deteniment. Però els dubtes que van sorgir pel que fa a la integració d'aquesta tecnologia en dispositius Android (amb navegadors web específics), així com la manca de funcionalitats encarades al treball fora de línia, i l'existència de solucions més atractives va fer que s'optés per abandonar aquesta idea.

3.4.4 HTML5 Geolocation

Aquesta és una opció realment interessant i qui sap si podríem estar parlant de la solució més portable i pràctica per a la geo-localització i els sistemes GIS mòbils en general en un futur molt pròxim.

En aquest apartat em refereixo a les capacitats que te HTML5 per a permetre desenvolupar sistemes GIS i no tan sols a la geo-localització en concret.
Estem parlant d'acceleració gràfica per hardware, millor capacitat offline i de cache i accés als dispositius. Combinant totes aquestes possibilitats tenim unes condicions ideals per desenvolupar sistemes GIS que a priori haurien de ser el màxim de portables possibles, ja que qualsevol navegador que suportes HTML5 les podria fer funcionar.

Així tot sembla indicar que només falta un petit pas perque es puguin realitzar aquest tipus de programes. Que es que HTML5 esdevingui un estàndard real i la majoria dels navegadors el suportin.

Per desgràcia sembla a ser que aquest horitzó no resulta del tot pròxim encara, i al dependre en gran mesura del desenvolupament de navegadors web compatibles estem limitats en el que podem fer per arribar-hi.

Per tant ara per ara parlar d'implementar una solució utilitzant aquesta tecnologia es tracta gairebé d'una pura especulació, ja que l'estàndard HTML5 a dia d'avui encara no esta ni completament implementat ni acceptat, en el moment de redacció d'aquesta memòria es troba en estat de "working draft".

Per si no fos poc, aquest tipus de solució planteja alguns dubtes, com ara si els dispositius actuals (o amb uns pocs anys d'antiguitat) serien capaços de fer funcionar navegadors que consumeixin tants recursos com sembla a ser que cada vegada tendeixen a fer, i que serien requerits per poder executar codi HTML5 amb garanties.

De ser així utilitzar HTML5 potser suposaria impossibilitar a un gran nombre de gent la utilització dels sistemes desenvolupats, i això és un factor que s'ha de considerar.

O fins i tot qüestions de seguretat que poden sorgir al donar tanta capacitat als navegadors web.

I al tractar-se d'un sistema tan íntimament relacionat amb Internet, naturalment sorgeix la pregunta de si, malgrat les millores en el sistemes de cache que proclamen els desenvolupadors de HTML5, les capacitats fora de línia d'aquesta mena de solucions serien tant limitades com aparenten, o realment seria viable que algú sense connexió pogués treballar amb normalitat.

Suposo que es necessita una mica més de temps i de proves amb HTML5 per poder respondre a aquestes preguntes i saber si realment pot ser una bona solució per implementar sistemes GIS.

Però malgrat això és interessant observar cap a on sembla que poden anar els trets en un futur pròxim, i jugar amb els exemples de demostració que actualment existeixen[49].

3.4.5 mapnik

Mapnik és un conjunt d'eines per renderitzar mapes. Entre moltes d'altres llocs es usada per generar el mapa de la web principal d'OpenStreetMap.

Permet treballar amb una gran varietat de dades geoespacials i proporciona opcions d'estil molt flexibles per poder generar mapes per àmbits molt diversos. Mapnik esta escrit en C++, però inclou també "bindings" per a treballar amb Python a un més alt nivell. Pot llegir ESRI shapefiles, PostGIS, TIFF, fitxers .osm, i qualsevol dels formats suportats per GDAL o OGR.

Es distribueix per a la majoria de distribucions Linux i també està disponible per Mac OS X i per Windows, però pel que he pogut trobar No existeix una implementació de Mapnik per a Android tot i que es pot fer servir per generar externament mapes que es poden visualitzar al telefon.

Realitzar el procés de renderitzat de mapes en un dispositiu mòbil no és l'objectiu del projecte mapnik, i, de fet, tal i com esta plantejada la biblioteca resultaria força complicat a priori[61].

Però la gran popularitat de mapnik per a realitzar mapes en servidors, va fer que estudiéssim la possibilitat o existència de ports de mapnik a Android, sense èxit, per desgràcia.

Moltes de les aplicacions que s'han comentat anteriorment el fan servir o el suporten. Per exemple:

- OsmAnd
- OruxMaps
- gvSIG Mini

3.4.6 mapsforge

Mapsforge és un atractiu projecte que promet proporcionar biblioteques de codi obert i lliure per a aplicacions basades en OpenStreetMap que permeten mostrar mapes en dispositius Android.

Segons la wiki d'OpenStreetMap mapsforge proporciona una biblioteca gratuïta i lliure que permet renderitzar mapes vectorials fora de línia (i de més tipus) per Android. Amb una API senzilla es pot construir una aplicació de mapes en menys de 30 línies de codi i suporta Overlays, amb punts d'interès (POIs) i formes poligonals.

El projecte va començar l'any 2008 a l'institut de ciències de la computació de la Freie Universität Berlin. I els seus principals objectius són proporcionar un conjunt d'eines lliures que permetin a la comunitat crear fàcilment aplicacions de mapes basades en OpenStreetMap. A tal efecte el projecte proporciona eines per renderitzar mapes, planificar rutes i "navegació", indexar i cercar punts d'interès (POIs), etc.

Algunes de les principals característiques que s'anuncien a la web mateix de mapsforge són:

- Us d'un format compacte per a la renderització ràpida de dades OpenStreetMap als dispositius.
- MapView fàcil de fer servir molt similar a l'API de Google.
- API de capes (overlay) flexible i potent.
- Estils de mapes personalitzables mitjançant fitxers de configuració XML.
- La biblioteca ocupa poc (300 KM aprox.).
- Aplicació de demostració disponible per ajudar als desenvolupadors.
- Proporciona eines per crear fitxers de mapes
- 100% lliure i de codi obert (llicència LGPL3)

En certs aspectes aquest és un projecte similar a osmdroid però semblava tenir un millor suport per a mapes fora de línia i estar més activament desenvolupat, cosa que podia ser de gran ajuda.

Després de descarregar-me la biblioteca i provar l'aplicació de demostració, vaig concloure que, tot i que no aportava totes les funcionalitats que el present projecte necessitava, era un projecte interessant i a tenir molt en compte.

3.5 Android

A part de l'estudi tecnològic de motors de renderitzat de mapes i d'aplicacions similars, el projecte també requeria un estudi de la plataforma Android.

Sempre que es vol treballar sobre una plataforma aliena és necessari un cert estudi previ d'aquesta, per conèixer-ne les característiques i les particularitats a grosso modo, així com l'entorn de treball recomanat, les bones pràctiques a seguir i en resum la filosofia de treball.

Però en aquest cas concret l'estudi ha estat potser més extens del que seria habitual, ja que no tan sols es pretenia inicialment desenvolupar biblioteques i aplicacions en aquesta plataforma aliena, sinó que també es va plantejar la utilització de llenguatges de programació diferents de Java per fer-ho.

L'elecció d'un llenguatge de programació diferent a Java no sol ser habitual per desenvolupar en Android, ja que Java té un estatus de "llenguatge preferent" en aquesta plataforma, i s'espera que els desenvolupadors d'aplicacions el facin servir de manera exclusiva.

Això és degut, en part, al disseny del sistema operatiu Android, que malgrat estar basat en Linux aporta la majoria de les funcionalitats del sistema a través del Dalvik (la maquina virtual de Java especifica per Android).

A continuació es pot veure un diagrama on es veu l'arquitectura d'Android i la importància del Dalvik (Figure 3.17).

Programar en Java no era un impediment del projecte, però tampoc una motivació, així que en un estadi inicial del projecte es va decidir estudiar la possibilitat de prescindir en la mesura del possible de Java i explorar altres alternatives de desenvolupament, en concret es volia mirar de treballar en Python, a ser possible.

Aquest estudi ha portat finalment a l'opció de crear una "façana" de SL4A per tal de poder oferir la capacitat de que scripts escrits en altres llenguatges de programació facin us de les biblioteques creades.

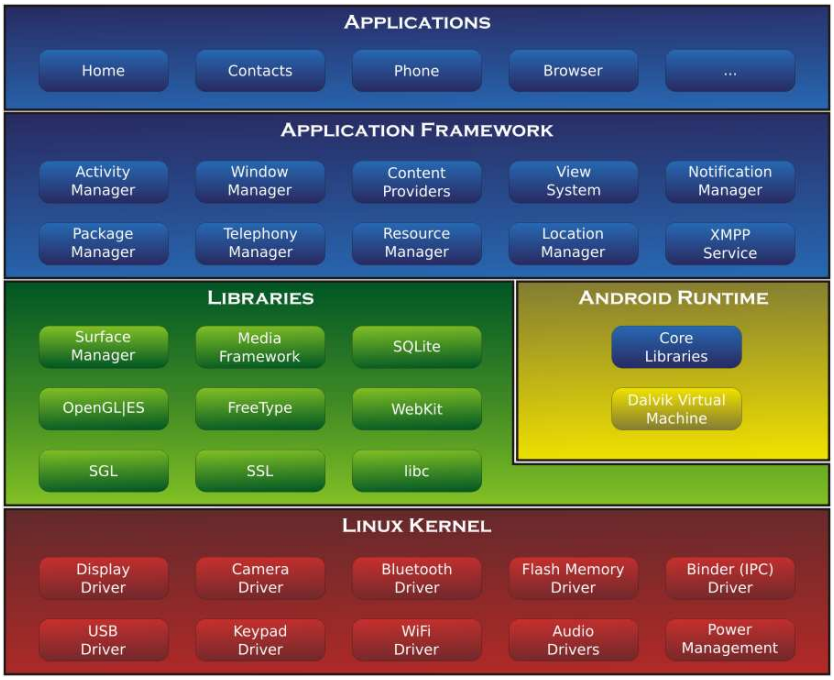

Figura 3.17: Arquitectura del sistema Android.

Per això crec convenient dedicar aquesta secció del capítol d'estudi tecnològic a la recerca que es va realitzar sobre llenguatges alternatius pel desenvolupament en Android.

A continuació esmentaré les opcions estudiades a tal efecte, i també s'explicarà amb una mica més de detall tant la filosofia com la arquitectura de SL4A, que és l'opció que finalment s'ha escollit.

Així també apuntaré aquí els motius pel qual no s'ha desenvolupat el projecte íntegrament en Python, malgrat que aquest era l'objecte d'aquesta part de l'estudi.

3.5.1 Estudi de llenguatges pel desenvolupament en Android

Malgrat que Java és el llenguatge de programació preeminent pel desenvolupament d'Android, existeixen mètodes més o menys complexos per poder desenvolupar aplicacions en altres llenguatges de programació.

Alguns d'aquests mètodes són poc ortodoxos i/o estan encara en desenvolupament (s'ha de tenir en compte la relativa joventut de la plataforma) o

han estat abandonats (encara que sigui relativament jove, hi ha hagut gran interès en ell i s'hi ha treballat amb gran promiscuïtat).

Així que l'estudi es va centrar valorar solucions provades i que requerissin poc nivell de modificació de la plataforma Android.

Aquest criteri es va aplicar ja que no es volia condicionar des d'un bon principi que el resultat final del projecte només pogués ser executat en dispositius amb sistemes alterats.

3.5.2 Native Developement Kit

El Native Developement Kit (NDK) és un conjunt d'eines que permeten implementar aplicacions o "parts" d'aplicacions per a Android fent servir els llenguatges de programació "natius", o sigui C i C++. Són eines oficials d'Android desenvolupades per Google i ben documentades.

Crear programes directament en C/C++ és oportú per a certs tipus d'aplicacions que pretenguin re-utilitzar codi de biblioteques o programes existents que ja estiguin implementades en aquests llenguatges.

O també podria ser útil en casos molt comptats per evitar algunes restriccions que dalvik (la maquina virtual de Java d'Android) imposa al codi que s'hi executa. Tot i que usualment (i tal i com s'adverteix a la documentació d'Android) utilitzar l'NDK no comporta un guany rellevant en el rendiment de les aplicacions.

En general no es recomana fer-lo servir si no és estrictament necessari, i s'adverteix als programadors que només l'emprin si la seva aplicació requereix algun dels beneficis que aporta, i mai, perquè el programador se senti més còmode programant en C/C++ que en Java.

Programar una aplicació "autista" per Android íntegrament en NDK no té moltes pegues, però el problema sembla a ser quan es vol comunicar el codi natiu amb el codi que s'està executant a la maquina virtual de Java. Aparentment aquest procés és més costós del que surt a compte. Així que normalment quan es programa amb NDK es perden les funcionalitats que proporciona l'API d'Android i/o s'afegeix un grau de complexitat elevat a les aplicacions.

Els exemples que es descriuen com a plausibles casos on l'NDK pot ser útil són majoritàriament programes auto-continguts (que no necessitin funcionalitats que proporcioni l'API d'Android), que necessitin realitzar càlculs

intensius amb la CPU i que no necessitin molta memòria, com ara procés de senyals, simulació de física, etc.

3.5.3 Compilació Creuada

La compilació creuada no és altra cosa que generar codi per a una màquina que no te la mateixa arquitectura que la que estàs fent servir per compilar-lo. És a dir compilar des d'un sistema per a un altre.

En el cas que ens ocupa es tractaria de programar des d'un ordinador personal (amb arquitectura amd64 o i686, etc..) i compilar aquest codi fent servir un compilador "especial" i empaquetar-ho de manera que estigui tot preparat per executar-se a les màquines Android (normalment arm).

Aquest procés esdevé més complexe del que pot semblar a primer cop d'ull quan volem fer aplicacions que fan servir moltes biblioteques externes que no estan portades a Android (cal tenir en compte que algunes de les biblioteques que fa servir el sistema Android natiu no són les estàndards de C, sinó biblioteques lleugerament modificades, que dificulten un pèl les coses).

Afortunadament existeixen algunes eines que serveixen per ajudar en aquesta tasca, que és més aviat incòmode.

Aquestes eines han aparegut sobretot en alguns camps concrets, i com es pot veure a continuació, la majoria de les eines que he considerat, han estat pensades per adaptar algun entorn de desenvolupament de videojocs.

Això es degut a que no existeixen eines per facilitat la compilació creuada d'aplicacions de GIS específicament, i al mateix temps les funcionalitats més difícils de portar a una plataforma com Android a priori semblen ser les gràfiques i les de gestió d'entrada (el que seria la interfície amb l'usuari final del sistema).

Per això es va considerar que un "framework" pensat per fer funcionar videojocs podria suplir les necessitats gràfiques d'una aplicació de mapes.

3.5.3.1 Python for Android

Si considerem la compilació creuada i l'objectiu concret de programar en Pyhton per a Android, el primer que es pot pensar és en cercar un mètode per compilar l'entorn de Python (intèrpret d'ordres, més biblioteques bàsiques). De manera que es pugui executar en un dispositiu Android.

A continuació resumeixo molt breument els passos a seguir per tal d'aconseguir-ho, extrets de la guia consultada[27] per compilar Python per Android.

Aquest resum pot donar una idea més concreta del procés que cal seguir per aconseguir el que anteriorment s'ha esmentat (per a més detall consultar l'annex).

Una part important i complicada del procés ha estat facilitada en gran mesura amb un "pedaç" (patch) que molt amablement proporciona l'autor del tutorial, i basat en el que fa servir el projecte Py4A i amb algunes idees de la gent de Pygame For Android (que serà esmentat més endavant), però en cas d'intentar portar un altre versió de Python o d'altres mòduls caldria desenvolupar-ne un altre. Aquest "pedaç" bàsicament prepara l'entorn d'execució Python per les condicions especials d'Android canviant alguns "flags" i la ubicació d'algunes biblioteques (tot i que no he entrat molt en detall a observar exactament quines).

Amb aquest pedaç podem modificar lleugerament el codi font de Python i configurar la seva compilació perquè pugui ser executat en l'entorn Android.

Després d'aplicar el "pedaç" cal modificar certes variables d'entorn i compilar Python fent servir el codi font modificat i un compilador per l'arquitectura ARM.

Naturalment aquest mètode presenta algunes complicacions pràctiques, més enllà de la complexitat del procés en si. En primer lloc, implica haver de sol·licitar que els usuaris facin un procediment semblant i preparin l'entorn per tal d'executar el codi que es crearà, o dissenyar i implementar un mecanisme que ho faci per ells.

En ambdós casos això pot resultar un impediment (o com a mínim una molèstia) per a molts potencials usuaris.

Tal i com s'ha plantejat el mètode a més, només es podrien crear i executar scripts que fessin ús de l'intèrpret de línia de comandes, ja que portar biblioteques d'interfícies gràfiques no és tant "trivial".

A més a més, el mecanisme presenta també el problema endèmic de la compilació creuada (que la resta de mètodes aquí considerats també presenten) que és la manca d'interacció amb les APIs d'Android, cosa que a priori limita molt la utilitat dels programes resultants i es volia evitar a ser possible.

3.5.3.2 Pygame Subset for Android

Aquest cas concret és el port no tan sols de Python sinó també dels mòduls Pygame de Python. Pygame és, al mateix temps, un port de la SDL per a Python, i la SDL és la famosa biblioteca "Simple DirectMedia Layer". Una biblioteca disponible per a múltiples plataformes pel desenvolupament d'aplicacions gràfiques, molt estesa pel desenvolupament de jocs bidimensionals sobretot.

En el fons el nucli de funcionament de Pygame Subset for Android és molt similar a l'explicat en l'apartat anterior, però l'equip de Pygame Subset for Android ha desenvolupat una serie d'"scripts" per ajudar en el procés, que resulten molt pràctics, i permeten fins i tot generar paquets estàndard d'Android APK. Així es pot distribuir l'aplicació programada sense necessitat que els usuaris hagin d'instal·lar prèviament l'entorn d'execució de Python al seu dispositiu, i és molt fàcil de desenvolupar i provar en el dispositiu.

Així el procediment per desenvolupar una aplicació mitjançant Pygame Subset for Android resulta tant senzill com fer el programa en Python sense preocupar-se de res, configurar-lo per al seu desplegament a Android invocant els scripts d'ajuda *(android.py configure mygame)* contestar a una serie de preguntes de configuració i executar altre cop l'script per fer un paquet que podrà ser distribuït en un sistema Android *(android.py build mygame release install)*.

Evidentment, prèviament a això cal instal·lar i configurar l'entorn de desenvolupament d'Android (SDK) i el de Pygame Subset for Android, però com es pot veure el procés de compilació creuada queda molt simplificat amb aquest mètode.

Tot i que, per exemple, he trobat alguns problemes per generar aplicacions que funcionin en tauletes noves (segurament degut a la versió d'arm que tenen configurat els scripts que generen el codi), en general sembla funcionar prou bé i permetre fer coses interessants.

El problema més gran de fer servir aquest sistema és, com en el cas anterior, a l'hora d'aprofitar alguna de les funcionalitats de l'API de Google. com ara accedir al GPS o a la llibreta d'adreces, per exemple.

A més no totes les funcionalitats de Python o de Pygame han estat portades (per això s'especifica "subset" en el nom) i tot i que les funcionalitats presents són suficients per a fer coses prou interessants, no hi havia garanties

que no resultessin en problemes a l'intentar portar aplicacions GIS fetes amb Python, o aprofitar-ne parts.

3.5.3.3 Necessitas

També es va estudiar la possibilitat de crear les biblioteques fent servir les capacitats gràfiques que brindava el port de Qt a sistemes Android (tal i com fa QGIS for Android).

Això hauria permès potser aprofitar alguna de les solucions GIS o de renderitzat de mapes implementades en sistemes d'escriptori que empressin Qt. Per això es va considerar Necessitas, que és el port de Qt per a Android.

Es tracta d'un projecte encara en fase de desenvolupament (al menys ho estava quan es va realitzar l'estudi) que té com a objectiu proporcionar una manera senzilla de desenvolupar aplicacions Qt per a la plataforma Android.

El projecte proporciona un SDK que inclou una modificació del QtCreator (un IDE per editar interfícies gràficament) i permet encarregar-se de tot "màgicament" (com molts IDE's).

En el fons el que fa aquest SDK és utilitzar l'NDK d'Android per compilar l'aplicació Qt generada com a biblioteca dinàmica i crear una aplicació Java "wrapper" que interactui amb l'entorn Android i faci d'interfície.

Teòricament això suposaria un petit "overhead" en l'execució d'aplicacions, però els creadors del projecte asseguren que és negligible.

A la banda del telèfon una aplicació distribuïda pel market mateix (Ministro) s'encarrega de descarregar d'internet totes les biblioteques requerides degudament compilades i preparar així el sistema Android per la seva utilització.

Però tenint en compte les dificultats que la instal·lació i avaluació de QGIS for Android va tenir, l'estadi de desenvolupament del projecte i l'existència d'altres solucions es va decidir no fer ús d'aquesta alternativa.

3.5.3.4 Conclusió

Generalitzant molt, podem dir que la compilació creuada aporta la versatilitat de poder desenvolupar per Android fent servir "frameworks" alternatius, a costa de perdre funcionalitats "estàndards" d'Android, i/o d'afegir un nou

nivell de complexitat tant al procés de desenvolupament com als sistemes que es desenvolupen, cosa que pot comportar problemes de compatibilitat en alguns dispositius.

A més en algunes de les solucions estudiades he notat una petita manca de "rodatge". Sense anar més lluny moltes de les solucions provades estaven en fase de desenvolupament al intentar provar-les, i algunes van resultar en problemes a l'hora de fer proves.

A tot això hi hem d'afegir que la idea que es tenia a l'inici del projecte de la integració amb ZamiaDroid (com s'explicarà més endavant) es complicava si es volia seguir aquest camí.

Per aquests motius, i perquè es va trobar una solució que semblava més encertada, es va decidir no emprar cap d'aquests mètodes, encara que alguns resultaven atractius o interessants, i també val a dir, una mica aterridors.

Potser en un futur no gaire llunyà algun d'aquests entorns esdevindrà una solució estable i més afable que pugui portar (esperem) a una més gran diversitat per a Android sense necessitat de pagar cap "peatge" addicional de desenvolupament.

3.5.4 SL4A

La solució que planteja SL4A va una mica més enllà que la "simple" compilació creuada, tot i que no és una idea radicalment diferent.

Scripting Layer for Android (SL4A) intenta portar els llenguatges de programació de scripts a Android permetent editar i executar scripts i intèrprets interactius directament en un dispositiu Android. Aquests scripts tenen accés a moltes de les APIs disponibles per a les aplicacions Android normals, però amb una interfície més senzilla que facilita fer les coses més ràpid.

Els scripts poden ser executats interactivament en un terminal, o en segon terme. Actualment SL4A suporta Python, Perl, JRuby, Lua, BeanShell, JavaScript, Tcl, i shell. Però en un futur potser n'apareixen més.

Malgrat la bona pinta que feia, SL4A encara esta en estat alfa, i esta dissenyat perquè els que l'utilitzin siguin desenvolupadors de software, així que encara hi ha força "bugs" i s'espera que amb el temps millori bastant.

El problema més gran que suposa la utilització de SL4A és que existeixen limitacions en el que es pot fer a nivell de pantalla (paradoxalment és una mica el problema invers que apareixia al considerar solucions pensades per crear jocs mitjançant compilació creuada).

Una de les API's que no són accessibles als llenguatges de scripting és la de mapes, ni la de gràfics. A més la dificultat de controlar els esdeveniments amb aquesta API pel mig dificulta força la interacció amb els usuaris.

3.5.4.1 Arquitectura

La idea que hi ha darrere l'SL4A és la següent: Tenir un programa servidor de RPC que tingui accés a les API's de Google i que es posi a escoltar, i per altra banda tenir un intèrpret del llenguatge de programació de scripting compilat per a Android (cross-compiled). En el moment que l'interpret detecti que s'està cridant un mètode destinat a les API's de Google es generarà una crida que mitjançant JSON i RPC transmetrà al servidor les dades de la crida. I així podem invocar un mètode remot.

Això significa que, fent un símil entre SL4A i una arquitectura distribuïda en xarxa, el sistema operatiu d'Android seria el servidor i l'script interpretat un client.

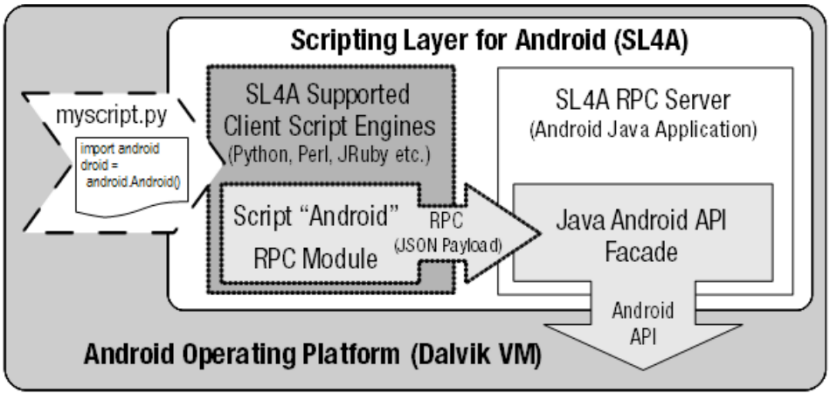

Figura 3.18: Arquitectura de SL4A.

Al nivell més baix, l'aplicació de SL4A que es proporciona és essencialment un host de scripting, cosa que significa que com a aplicació té diferents intèrprets cadascun dels quals per a un llenguatge específic. Aquests intèrprets han estat compilats per a la arquitectura ARM fent servir l'NDK

i es carreguen com a una biblioteca quan l'SL4A s'executa un interpret especific.

A partir d'aquí l'script serà interpretat seqüencialment línia a línia normalment.

Però, el truc és que cada script ha d'importar un fitxer (en el cas de Python és android.py) que defineix una serie de funcions necessàries per comunicarse, mitjançant RPC (remote procedure calls) i JSON (JavaScript Object Notation), amb les "Facanes" de SL4A. Aquestes façanes al seu temps es poden comunicar amb les API d'Android.

Quan l'script s'ha iniciat, també se li ha donat, a part de la seva pròpia instància de l'intèrpret (i una pantalla de text, en cas de no haver estat iniciat en segon pla), també se li proporciona, deia, una connexió TCP amb el servidor re RPC de SL4A.

De fet, el servidor de RPC de SL4A el que fa és obrir un socket, i escoltar peticions entrants, que seran les crides a les apis de Google encapsulades amb JSON.

I pel que fa l'script, en el moment que intenti fer una crida a una de les APIs de Google (per exemple si fem un android.makeToast()), el codi addicional que hem inclòs agafarà (catch) l'excepció que es genera (ja que es tracta d'un mètode desconegut) i en comptes de propagar aquesta excepció passarà el mètode cridat (junt amb els paràmetres corresponents) al servidor RPC com a una petició JSON.

Després aquesta petició serà processada i el resultat es passa de tornada cap a l'intèrpret i a l'script que s'està executant.

Aquesta estructura fa que l'script no necessiti tenir cap coneixement de quina versió de SL4A està essent executada. I a més, aquest nou nivell d'abstracció entre SL4A i el sistema també serveix per prevenir que algun script maliciós pugui arribar a fer mal al sistema.

A continuació podeu veure un diagrama de flux que representa l'execució d'un script Python, i on es veu la interacció dels diferents mòduls i parts de SL4A.

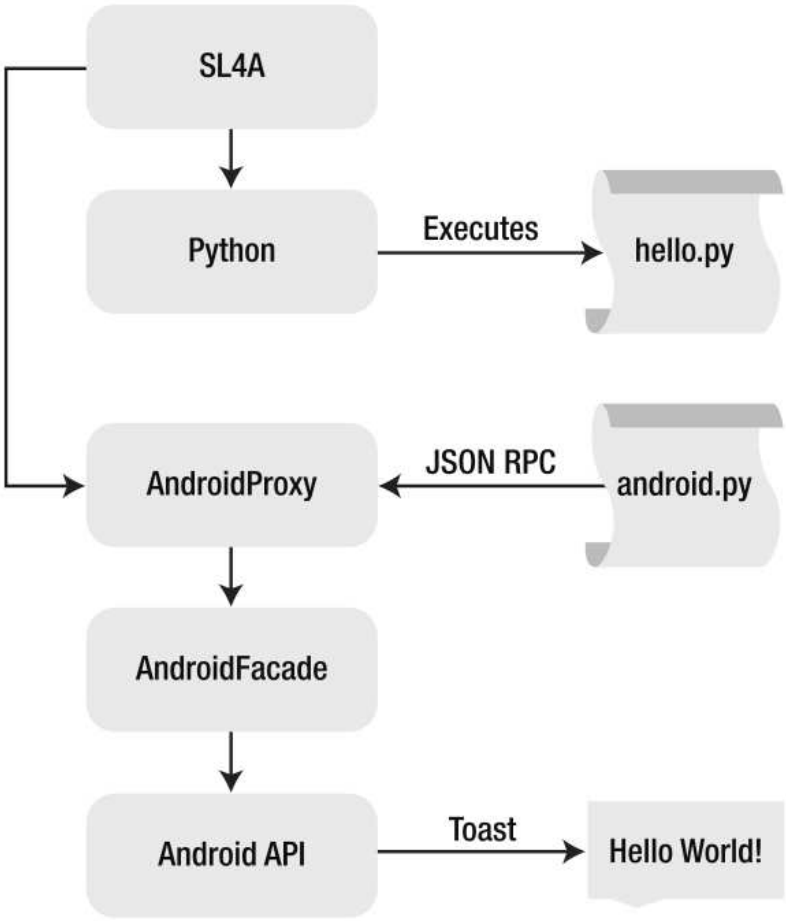

Figura 3.19: Flux d'execució de SL4A.

3.6 Conclusió

Després de fer aquest estudi aquestes són les conclusions a que es pot arribar pel que fa a solucions existents, i a la plataforma en concret que s'ha estudiat.

Pel que he pogut comprovar, el procediment normal que plantegen les aplicacions per a treballar amb mapes en dispositius mòbils sol ser assumir que existeix connexió de xarxa i descarregar sota demanda imatges de mapes processades en servidors externs.

Algunes aplicacions consideren la possibilitat que no es tingui connexió a la xarxa, i les propostes que fan per solucionar aquesta inconveniència són emmagatzemar al dispositiu la informació que serveixen els servidors en

forma de tessel·les (imatges) que consumeixen molt d'espai al dispositiu, o, en el millor dels casos processar aquestes dades en un ordinador personal i copiar-les al telèfon per al seu ús fora de línia més endavant.

Renderitzar els mapes directament al telèfon sembla desitjable quan no es té connexió. Això permetria descarregar només la informació "vectorial" dels mapes al dispositiu (molt menys voluminosa) i "pintar-la" només quan es vulgui mostrar per pantalla.

Però aquesta no sembla ser una opció gaire estesa. Els motius poden ser la alta disponibilitat de xarxes al telèfons. O la relativa limitada capacitat de càlcul que aquests dispositius tenien fins fa poc (sobretot tenint en compte que càlculs intensius poden consumir molta bateria del telèfon i el sistema convida a evitar-los). També pot ser la creixent capacitat de memòria dels telèfons, que fa que guardar els mapes com a imatges en els dispositius no arribi a ser tant dramàtic.

En qualsevol cas, considerant el gran increment en la capacitat de càlcul dels telèfons dels darrers temps, semblava que es podria intentar trobar algun mecanisme per assolir aquesta funcionalitat pel present projecte.

En quant a les conclusions de l'estudi pel que fa a Android pròpiament, després d'haver estudiat tantes alternatives interessants un s'adona de les dimensions que ha abastat la plataforma Android. Tant a nivell de varietat d'aplicacions com de documentació existent. És realment aclaparador, fins i tot en un camp tant aparentment especialitzat com el que s'estava estudiant.

Així queda palesa la grandària de la comunitat de desenvolupadors que han vist possibilitats en Android i treballen amb aquesta plataforma.

Segurament això és fruit de la idoneïtat dels dispositius mòbils actuals per realitzar tasques relacionades amb els sistemes GIS, i també hi ha ajudat, probablement, la relativa "novetat" del sistema i el gran nombre d'usuaris que actualment te. Però malgrat això no esperava trobar tanta cosa.

Com a contrapartida també he pogut comprovar alguns aspectes negatius que comporta la evolució tant ràpida en una plataforma, que fa que els projectes apareguin i siguin abandonats a gran velocitat, o torna obsolets dispositius o solucions molt de pressa.

A més l'estudi també ha servit per comprovar que cap de les solucions existents en el sistema Android satisfeia els objectius del projecte. I també s'ha vist que els intents de portar les alternatives d'altres sistemes (principalment sistemes GNU/LINUX) no havien estat reeixits fins al moment, deixant

entreveure que malgrat tractar-se d'un sistema "parcialment lliure" les seves particularitats dificultaven la tasca de portar codi d'altres plataformes a Android.

Com a apunt final de l'estudi he de reiterar que m'ha sorprès la quantitat de projectes existents en Android, tant en el camp dels sistemes GIS com en el sector de modificacions del sistema o alternatives per a programar-hi en altres llenguatges de programació he anat trobant un gran nombre de treballs, molts dels quals no tenien relació directa amb el present projecte i no han estat inclosos a l'estudi, però han deixat una sensació molt forta d'enormitat.

Estic segur que mentre escric aquestes línies algú està desenvolupant alguna aplicació o biblioteca que estigui relacionada amb aquest PFC, i que hauria pogut ser estudiada i potser, fins i tot, aprofitada per les finalitats del present projecte. Però també cal saber quan abandonar l'etapa de recerca i, treballant amb els coneixements adquirits, aplicar-los per crear una bona solució.

Capítol 4

Anàlisi i Disseny

4.1 Introducció

Després d'haver avaluat tantes alternatives es va arribar a la conclusió que es realitzaria el següent:

- Utilitzar les biblioteques de mapsforge per a cobrir algunes de les funcionalitats bàsiques de les biblioteques del projecte.
- Expandir les biblioteques de mapsforge per a cobrir la resta de les funcionalitats que no estaven garantides per l'ús d'aquestes biblioteques i oferir a la comunitat de mapsforge la possibilitat d'integrar aquestes funcionalitats en el projecte i contribuir-hi així.
- Dissenyar una interfície perquè aquesta biblioteca resultant pugui ser utilitzada pel ZamiaDroid per a mostrar els mapes i aportar noves capacitats.
- Dissenyar una aplicació (erdapfel) fent servir Java i l'SDK d'Android, que permeti mostrar els mapes i provar les funcionalitats de la biblioteca.
- Dissenyar una "façana de SL4A" perquè les principals funcionalitats de la biblioteca siguin accessibles des dels llenguatges de "scripting" de SL4A.

Potser un dels factors claus per l'elecció de mapsforge va ser el fet que a la documentació del projecte existia una proposta per a incloure suport per a KML, que s'havia considerat interessant per la comunitat de desenvolupadors, però desestimat per culpa de restriccions purament temporals (la podeu consultar a l'annex).

A banda d'això, el fet de no trobar una manera clara per desenvolupar les biblioteques en llenguatges alternatius que després en garantís la plena utilitat, també hi va ajudar.

4.2 Especificació de les biblioteques

Les biblioteques resultants d'aquest projecte havien de proporcionar les següents funcionalitats:

4.2.1 Renderitzat de mapes de fonts osm

L'elecció de mapsforge ens garantia el suport de mapes de fons osm de manera "nativa". En aquest apartat, doncs, es contava aprofitar les característiques de la biblioteca.

Mapsforge permet descarregar-se "tessel·les" (tiles) generades amb mapnik a partir de les dades de OpenStreetMap, des dels servidors de tiles del mateix OpenStreetMap o des del servidor d'OpenCycleMap. Es pot estendre per mostrar tessel·les d'altres servidors, si es vol, però per defecte implementa aquests dos.

Ara bé, a més de poder mostrar mapes generats amb dades osm (que és un dels objectius del projecte), també es pretenia permetre llegir directament els formats .osm, que és un xml amb les dades "raw" d'OSM, o .pbf, que és un format binari també amb aquestes dades.

En aquest cas el suport de mapsforge no és tant immediat, però això es comentarà a l'apartat de mapes fora de línia, ja que, encara que siguin "fonts OSM" s'han considerat formats "fora de línia".

4.2.2 Renderitzat de mapes fora de línia

Un factor determinant a l'hora d'escollir Mapsforge com a base de les biblioteques va ser perquè aportava suport per a mapes "offline".

Durant l'estudi tecnològic, però, no es va veure exactament quin era el mecanisme per a proporcionar aquest suport.

Pel que sembla la manera usual d'aportar aquesta funcionalitat en les aplicacions que fan ús de mapsforge es processar les dades OSM del mapa desitjat amb un mòdul específic de l'eina Osmosis creat a tal efecte per la gent de mapsforge (el mapfileWriter).

Fent ús d'aquest mòdul d'osmosis es poden processar dades de diverses fonts OSM (ja sigui en el format XML d'OpenStreetMap, el format binari, etc.) i generar mapes en el format de mapa que després mapsforge llegeix i amb el qual treballa.

Com es pot veure això implica un pre-procés de les dades abans de que mapsforge les mostri, que no es va detectar durant l'estudi tecnològic de l'aplicació, i ha resultat ser una inconveniència imprevista.

Normalment aquest pre-procés es realitza en un ordinador personal i els mapes es transfereixen als dispositius mòbils que requereixin les dades, o alternativament el pre-procés es realitza en un servidor que rep la petició de l'usuari del dispositiu per xarxa i transfereix el mapa en el format propi de mapsforge.

Cap de les dues opcions era plenament satisfactòria per la nostra visió del projecte, així que es va estudiar la possibilitat de portar l'eina Osmosis i el corresponent mòdul de mapsforge a Android perquè aquest a fase de pre-procés dels mapes pogués ser duta a terme en el mateix dispositiu mòbil i així cobrir aquesta funcionalitat sense alterar el funcionament usual de la biblioteca mapsforge.

4.2.2.1 Adaptació d'osmosis a Android

Osmosis és una eina escrita en Java per processar dades d'OSM. És una eina molt versàtil i extensa consistent en un seguit de components connectats que poden encadenar-se i produir operacions complexes. Ha estat dissenyada de tal manera que afegir-hi nous components sigui senzill.

Tal com s'ha dit anteriorment, és l'eina que es fa servir per tractar dades extretes d'OSM i convertir-les en el format de mapes propi de mapsforge.

La necessitat de processar les dades amb aquesta eina fora del dispositiu Android trencava la filosofia de treball que es pretenia aconseguir idealment amb el projecte (i per tant les biblioteques que s'estan especificant).

Davant d'aquest problema es plantejaven dues vies d'actuació:

O bé es modificava la biblioteca de mapsforge per a acceptar aquests formats de manera similar al que es tenia previst fer amb els formats KML i GML, o bé s'intentava mantenir la metodologia de treball plantejada per mapsforge, però adaptant-la a les necessitats particulars d'aquest projecte intentant aportar els mitjans per realitzar el mateix pre-procés que es faria en un servidor o ordinador personal, al telèfon mòbil.

L'opció de modificar les biblioteques de mapsforge es va veure exagerada. A més de comportar certa redundància, semblava molt invasiva i dificultava la possible integració posterior de les modificacions al projecte mapsforge.

Així que es va creure que l'opció més adient, menys problemàtica i amb possibilitats d'èxit era fer un port d'osmosis a la plataforma Android.

Aquest "port" va comportar certes dificultats tècniques que s'explicaran a l'apartat d'implementació, però teòricament era viable.

A més de les dificultats tècniques, afegia algunes contrapartides inherents, com són un previsible increment del temps que el pre-procés trigaria al dispositiu Android, o la reducció de la vida de la bateria del telèfon al realitzar els càlculs. Però es va considerar que aquestes contrapartides eren assumibles, ja que aquest seria un procés que es realitzaria esporàdicament.

4.2.3 Suport per a GML & KML

Pel suport dels formats GML i KML, la cosa era diferent, ja que, per començar la biblioteca mapsforge no proporcionava cap mitjà per a permetre'n la visualització (ni tan sols un d'inconvenient com en el cas anterior).

A més, tal i com podeu veure a l'annex, l'existència d'una proposta d'afegir aquestes funcionalitats a mapsforge, feia que al mateix temps que es cobrien les necessitats del present projecte, es cobrien també els requeriments del projecte mapsforge.

Això, al nostre entendre justificava molt més la modificació de les biblioteques de mapsforge.

Al plantejar l'"expansió" de les capacitats de mapsforge calia considerar molt bé, però, com plantejar-ho.

Resultava evident que un dels passos era afegir-hi la capacitat d'interpretar documents GML i KML (o parsejar-los), i també que calien funcionalitats per a mostrar-los.

Una de les maneres que es podria haver decidit per tal de mostrar els documents KML/GML, hauria pogut ser un procés semblant a l'existent per carregar fitxers de mapes propis de mapsforge (.map). És a dir crear el que dins el projecte hauria estat alguna subclasse de "mapgenerator".

En aquest punt es va decidir, però, no tractar els documents GML i KML com a fitxers de "mapes" pròpiament dits, sinó com a fitxer de "capes".

M'explicaré: malgrat que un mapa no és més que un conjunt de capes, en el projecte mapsforge es fa una distinció entre un fitxer que conté un mapa sencer i les capes que posteriorment s'hi poden anar afegint a sobre. Els fitxers KML es fan servir sobretot per a exportar capes i no tant "mapes" sencers.

El cas d'ús que la comunitat de mapsforge esperava cobrir aportant aquesta nova funcionalitat era que un usuari exportes d'algun o altre programa extern una capa en KML, i la pogués carregar sobre un mapa de mapsforge.

Així per a garantir el suport a KML i GML, es va decidir implementar "parsers" de GML i de KML i estendre els Overlays (capes) de mapsforge, per intentar fer un "mimetisme" amb els diferents elements ja existents a la biblioteca mapsforge, i que la seva utilització fos pràctica i similar a la que ja s'estava fent servir, i no fos traumàtica per a desenvolupadors avesats a l'API d'Overlays de mapsforge.

Això va comportar greus problemes quan mapsforge va remodelar radicalment aquesta API en una actualització de la biblioteca, tal i com s'explicarà amb meś detall més endavant.

Per altra banda, la decisió d'implementar un "parser" de GML/KML va sorgir després de cercar extensivament implementacions ja existents sense èxit.

Sembla a ser que tot i que KML i GML són un estàndards força estesos basats en XML, no existia cap solució implementada que poguéssim trobar i que fos adient per a ambdós formats.

4.3 Integració amb ZamiaDroid

Per a la integració amb ZamiaDroid, es van tenir en compte diverses possibilitats.

El primer que es va considerar va ser, tenint en compte que la API de mapes de Google que utilitza ZamiaDroid és molt similar a la API de mapsforge, fer les modificacions pertinents perquè ZamiaDroid fes servir la nostra expansió de mapsforge en comptes de la biblioteca de Google.

Aquesta solució no era prou bona perquè implicava alguns problemes:

- Mapsforge no garantia la total equivalència entre la seva API i la de Google, cosa que vol dir que canviar una biblioteca per l'altre no era immediat, sinó que podia significar una gran modificació del codi de Zamia.

- Necessitat de fer una branca alternativa en el desenvolupament de ZamiaDroid per a realitzar aquesta integració, i possible dificult de la posterior "mescla" de branques degut a la evolució continua de Zamia i a l'alta volatilitat de les parts que treballen amb la API de mapes (que són moltes i molt poc estables).

- Mantenir el suport per a la utilització dels mapes de Google era un punt a favor per l'equip de Zamia, i aquesta solució implicava prescindir-ne.

Així que es va descartar en favor d'una opció que permetés la coexistència d'ambdós biblioteques de mapes.

Per garantir aquesta coexistència es va considerar la possibilitat d'incloure la biblioteca estesa de mapsforge a dins del ZamiaDroid. Però el sistema d'herència de Java dificultava molt el procés.

Tant l'API de mapes de google com la de mapsforge segueixen la filosofia d'Android d'integrar funcionalitats als programes mitjançant herència. És a dir, que per fer servir un mapa s'espera que s'hereti de la classe de mapes que proporciona la API (qualsevol de les dues) i es modifiqui al gust i/o necessitats de l'aplicació que s'està fent.

Això no només succeeix amb la classe MapView, sinó amb la pràctica totalitat de les classes que Zamia fa servir per treballar amb mapes (GeoPoint, Overlay, etc.).

I Java no permet herència múltiple, així que la integració amb Zamia per aquesta via suposava una duplicació de codi gairebé idèntic exageradament gran que no agradava ni als desenvolupadors de Zamia ni als del present projecte.

Es va mirar de dissenyar una jerarquia de classes que permetés aquesta integració de manera neta (sense haver de reescriure gairebé el mateix codi per a les dos versions de la biblioteca), però aquesta estructura va resultar d'una complexitat elevada degut al gran nombre de classes que intervenien, i amb pocs beneficis. Així que es va considerar que no aportava netedat al projecte ans al contrari.

Finalment es va trobar una alternativa que comportava una modificació molt menys intrusiva de Zamia i aportava totes les funcionalitats que es requerien.

Aquesta alternativa va ser emprar les capacitats ja existents de ZamiaDroid per a exportar mapes en format KML junt amb el disseny d'una interfície entre el ZamiaDroid i l'aplicació de visualització de mapes erdapfel.

Fent servir la comunicació entre intents estàndard de la plataforma Android i KML com a suport de dades ZamiaDroid pot cridar l'aplicació de visualització erdapfel quan es vulgui visualitzar qualsevol dada de Zamia (ja sigui una citació concreta, com una selecció d'aquestes o un recorregut, etc.).

Erdapfel fa servir la biblioteca estesa de mapsforge, i, per tant, permet tant la càrrega de GML/KML com la possibilitat de treballar fora de línia. I mitjançant els mateixos intents es pot comunicar a erdapfel quin comportament ha de tenir quan l'usuari interactua amb el mapa (què cal fer quan es pressiona una citació, etc.) que en el nostre cas concret serà enviar dades de la interacció i retornar el control a ZamiaDroid.

Aquesta interacció s'ha concebut de manera que sigui prou genèrica com perquè qualsevol aplicació que ho desitgi en pugui fer ús (no necessàriament ZamiaDroid).

A continuació es pot veure un esquema conceptual d'aquesta interacció:

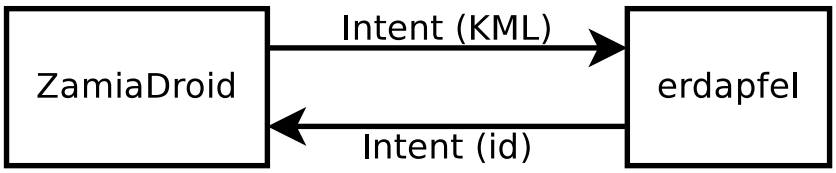

Figura 4.1: Interacció Zamia erdapfel.

Com a contrapartida, aquest mètode podria limitar la interacció que pot tenir l'usuari amb el Zamia, ja que afegeix un estrat entre la comunicació del Zamia i l'usuari.

4.4 Funcionalitats de l'aplicació erdapfel

A part de demostrar les funcionalitats abans mencionades de la biblioteca estesa de mapsforge, erdapfel havia de permetre la integració amb ZamiaDroid per ajudar en la visualització. Així que erdapfel és una mena de visor avançat de mapes pensat perquè pugui ser cridat des d'una aplicació externa, o pel seu funcionament autònom. Aquestes són les seves funcionalitats principals:

- Visualitzar mapes de fonts osm en línia
- Obrir mapes fora de línia
- Obrir fitxers KML i GML
- Generar elements geomètrics (punts i polígons)
- Realitzar mesures aproximades de distàncies i àrees sobre el mapa.
- Guardar els elements geomètrics generats.
- Permetre a l'usuari gestionar els fitxers generats i els fitxers a visualitzar de manera gràfica.

També havia de permetre la invocació com a "Intent" a més de la invocació autònoma de l'aplicació, i acceptar paràmetres per tal de configurar-ne l'execució al fer-ho.

Les "Activities" que es van pensar per l'aplicació doncs van ser:

- Mapa/Activity principal
- Pantalla de configuració
- Pantalla de selecció de fitxers

Tant la creació d'elements geomètrics com les mètriques es prendrien sobre el mapa de la pantalla principal directament.

Per gestionar els mapes es va pensar en una interfície similar a la d'un navegador de fitxers.

I per facilitar la utilització de l'aplicació i no sobrecarregar-ne la pantalla principal del mapa també es necessitava un menú de configuració, per definir paràmetres per defecte, temes d'estil, etc.

4.5 Disseny de l'aplicació erdapfel

El disseny d'aplicacions amb l'SDK d'Android està força condicionat per l'arquitectura d'aquest sistema, així que no es permet gaire llibertat en aquest aspecte (com podeu comprovar llegint aquest article[66]). Però per altra banda si que es dona molta llibertat i facilitat en el disseny d'interfícies d'usuari.

Aquest apartat és per mostrar aquest disseny d'interfícies d'usuari (no el disseny de l'arquitectura de l'aplicació en si, ja que aquest és el genèric emprat en totes les aplicacions Android).

A continuació es mostraran els prototips de disseny per a les pantalles de l'aplicació erdapfel.

4.5.1 Pantalla principal i de mapes

Figura 4.2: Prototip pantalla principal d'erdapfel.

4.5.2 Pantalla de selecció de fitxers

4.5.3 Pantalla de configuració

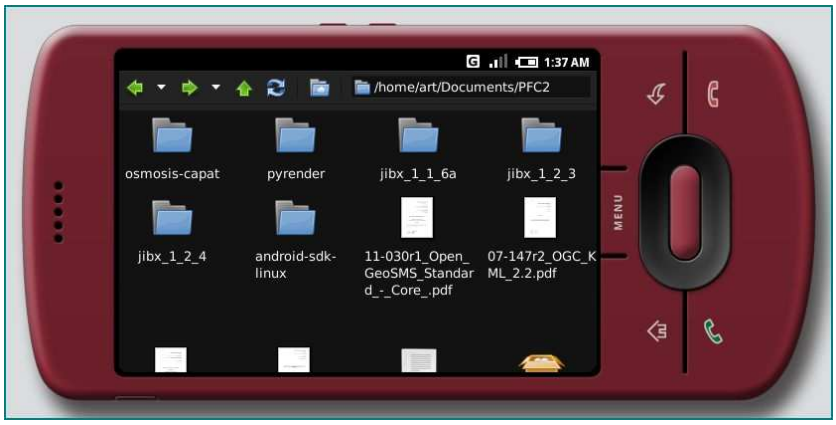

Figura 4.3: Prototip pantalla de selecció de fitxers d'erdapfel.

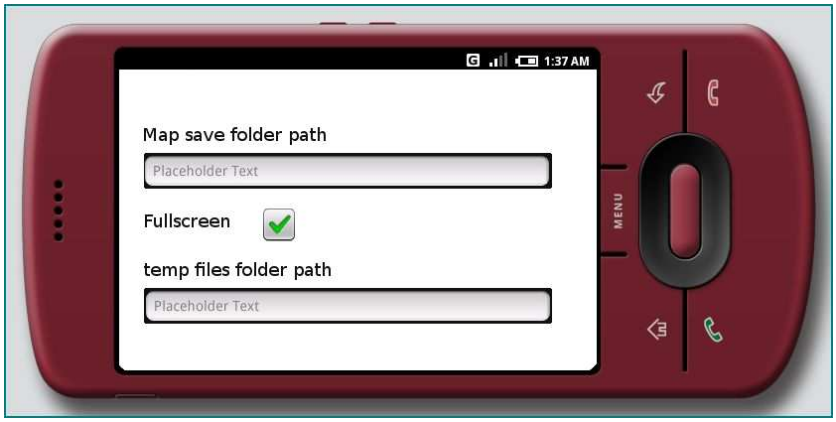

Figura 4.4: Prototip pantalla de configuració d'erdapfel.

4.6 Funcionalitats de la façana de mapes

La façana de mapes de SL4A havia de ser una abstracció a molt alt nivell del que podia fer la biblioteca estesa de mapsforge.

La idea d'aquesta façana seria, per exemple, processar un fitxer KML des d'un script Python, i invocar tot seguit una funció com "showMap(kml)" que el mostri per pantalla.

Aquestes són les funcionalitats que es van definir en un començament:

- Visualitzar un mapa de fonts osm i navegar sobre seu.
- Obrir un fitxer de mapa fora de línia .map, KML o GML

- Permetre Generar elements geomètrics (punts i polígons) sobre un mapa obert.
- Realitzar mesures aproximades de distàncies i àrees sobre el mapa obert.
- Guardar els elements geomètrics generats en KML o GML.

Ara bé, a diferència de les invocacions a la API de google, que no comporten cap problema, ja que aquesta està distribuïda "de fàbrica" en tots els dispositius Android, la nostra biblioteca no està distribuïda amb el sistema Android per defecte.

Això implicava que no tan sols calia crear una façana en el servidor de SL4A sinó que aquesta façana havia de tenir com a dependència la biblioteca estesa de mapsforge.

Així que es va decidir aprofitar que les capacitats requerides de la façana eren força compatibles amb les d'erdapfel per fer servir el programa com a punt d'entrada de la façana.

Això, tot i que no és una solució òptima, facilitava molt la distribució i instal·lació de la biblioteca i no suposava una dramàtica inconveniència per als usuaris, ja que les dimensions de l'aplicació erdapfel no són gaire més grans que la de la biblioteca en sí (es tracta d'una mida aproximada de 500Kb).

I de moment es van considerar aquests avantatges més importants que els inconvenients.

4.7 Arquitectura del sistema

Partint de les explicacions anteriors, aquest és l'esquema conceptual dels diferents elements del sistema i com interactuen entre ells (Figure 4.5).

Noteu que el port d'Osmosis es un element del sistema separat de la biblioteca estesa de mapsforge pròpiament dita (el port és el que en el diagrama es veu com a mapfileWriter4And).

Com es pot comprovar, erdapfel, tot i que havia de ser inicialment una senzilla aplicació de prova, ha esdevingut el punt d'entrada a les biblioteques generades tant per a Zamia com per a la façana de SL4A.

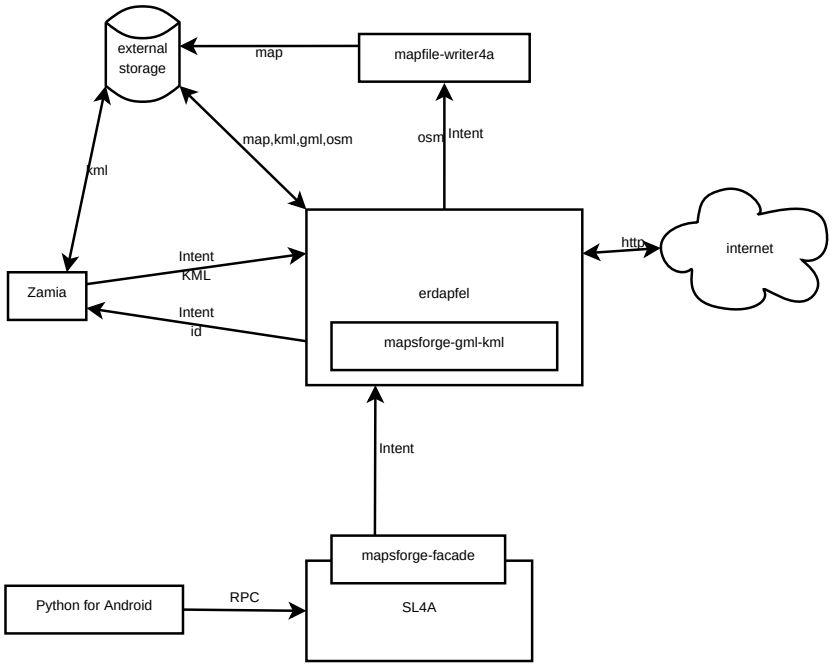

Figura 4.5: Arquitectura del sistema.

4.8 Treball comunitari: Integració del GML i KML amb mapsforge

En aquest apartat s'explicarà una mica més a fons els motius concrets pels quals es va decidir provar d'integrar les modificacions GML i KML de la biblioteca mapsforge a aquest projecte, les conseqüències que això va aportar al disseny i a la implementació d'aquesta part del projecte. La valoració final d'aquesta experiència es podrà trobar més endavant a les conclusions.

Abans de començar a implementar el "parser" de GML/KML es va considerar la següent possibilitat:

Sabent que mapsforge intentava mostrar mapes de diferents orígens i tenia suport per a mapes offline, i ja que tal i com es pot observar hi havien propostes de millora de mapsforge referents a un suport per KML acceptades però amb molt poca prioritat degut a restriccions de temps i de personal.

I sabent també que aquesta part havia de ser implementada per tal d'assolir els objectius del present projecte, semblava una bona idea intentar treballar en equip amb ells.

Així que ens vam posar en contacte amb la comunitat de desenvolupament de mapsforge i se'ls va proposar de desenvolupar aquesta funcionalitat dintre de la seva mateixa biblioteca, i col·laborar així amb el projecte.

La comunitat de mapsforge va dir que sí, però amb reserves. La veritat és que no va ser una resposta gaire entusiasta (com es pot veure a l'annex). La seva proposta va ser que fes un "branch" independent de mapsforge en un repositori de versions extern, i implementés la meva proposta allà, i que si resultava satisfactòria pels seus estàndards es faria una fusió d'aquesta "branch" amb la versió principal de mapsforge.

Així que, en un intent de contribuir a la comunitat de codi lliure, aquesta part del projecte s'ha intentat realitzar en màxima sintonia amb aquesta comunitat, utilitzant les eines i seguint les convencions que ells proposen (que d'altra banda són força raonables).

A continuació les esmento:

4.8.1 Mapsforge conventions

Les convencions del projecte mapsforge pel que fa al codi, en realitat estan basades gairebé completament en les convencions oficials del llenguatge de programació Java.

A més d'aquestes convencions hi ha una serie de normes definides per als membres del projecte.

Respecte a l'idioma i a la codificació dels caràcters.

El projecte fa servir tan sols anglès. Per tant tots els noms de paquets, arxius, mètodes, variables, així com també els comentaris i els missatges de les revisions han d'estar en anglès.

Per tal de no incorporar errors ortogràfics al projecte cal passar un corrector automàtic al codi (ells proposen el que està integrat amb l'Eclipse).

També demanen fer servir codificació UTF-8 i salts de línia de Unix ('`\n`') per a tots els fitxers de font, comentaris i altra documentació escrita.

Per pujar versions al subversion (repository) cada "commit" ha de tenir un comentari adient esmentant totes les modificacions. I sempre que sia possible, tots els canvis que estiguin conceptualment relacionats entre ells hauries de ser pujats conjuntament.

Com a convencions específiques de Java sol·liciten que es treballi amb l'IDE (Integrated Development Environment) Eclipse i un formatejador de codi que està contingut en ell. A tal efecte proveeixen un perfil creat pel formatejador de codi (que demanen que no sigui modificat per als usuaris sense la aprovació dels membres del projecte), i demanen que aquest perfil s'apliqui al codi abans de fer qualsevol registre al supositori de svn.

A part demanen fer servir les bones pràctiques de programació de mantenir la visibilitat de les classes i mètodes al mínim. i fer servir el modificador "public" només quan sigui estrictament necessari.

També consideren que documentar el projecte utilitzant Javadoc és primordial. Per això demanen que totes les classes, mètodes i camps tinguin el seu comentari de Javadoc apropiat. Concretament aquests comentaris han de documentar (en el cas dels mètodes) tots els paràmetres i el valor de retorn. Naturalment també demanen que quan es realitzi qualsevol modificació a un mètode ja documentat s'actualitzi el comentari degudament.

Respecte als senyals d'avís del compilador (Warnings), i tenint en consideració que molts dels errors més usuals de programació poden ser evitats utilitzant els esmentats avisos, es demana que habilitem tots els avisos que garanteixin un elevat nivell de qualitat de software.

També demanen que els warnings siguin resolts preferiblement pels desenvolupadors que els han generat, i que en cap cas s'utilitzin "trucs" per evitar els warnings (fer servir les anotacions de Java @SupressWarnings) o es modifiqui la configuració del compilador a tal efecte.

Naturalment també demanen que mai es pugi una versió amb errors de compilació al subversion.

4.8.1.1 CheckStyle

Checkstyle és una eina de desenvolupament per ajudar als programadors a escriure codi Java que compleixi els requisits d'un estàndard de codi. Automatitza el procés de comprovar el codi de Java per evitar que els desenvolupadors hagin de realitzar una tasca tant empipadors (però important). Això fa que sigui una eina ideal per a projectes que vulguin forçar l'aplicació d'unes convencions pel redactat del codi (és a dir un estàndard de codi).

Checkstyle és molt configurable i pot ser emprat per forçar l'aplicació de gairebé qualsevol estàndard de codi imaginable. L'estàndard que empra el projecte mapsforge és una petita variació de les convencions de Sun per a Java.

Això, a nivell del programador implica que després d'escriure una part de codi, cal executar el CheckStyle i comprovar els avisos que genera. Una de les maneres més pràctiques de realitzar aquest procés és instal·lar un "plug-in" per eclipse que existeix i que integra CheckStyle a l'entorn de desenvolupament. Però també es podria fer des de línia de comandes, o fent servir eines com l'ant, etc. Els "errors" que mostra checkstyle poden ser tant de disposició del codi com de problemes de disseny, duplicitat de codi, o mala aplicació de patrons.

Aquí podeu trobar més informació[5].

4.8.1.2 FindBugs

FindBugs™ és un programa per trobar errors (bugs) en programes Java. Es dedica a cercar instàncies de "patrons d'error", és a dir, patrons en el codi que probablement indueixin a la aparició de bugs.

De forma similar a CheckStyle existeix un plug-in per a eclipse que en facilita l'execució.

Aquí podeu trobar més informació[6].

4.8.1.3 PMD

PMD busca potencials problemes o pràctiques de programació que considera inadequades en el codi font Java, com ara:

1. Possibles bugs, sentències try/catch/finally/switch buides.
2. Codi mort, variables locals o paràmetres no usats, o mètodes inservibles.
3. Codi no optim, Cadenes de caràcters "malgastades".
4. Expressions massa complicades, sentències "if" innecessàries, sentències for que podrien ser whiles

5. Codi duplicat, fragments de codi que podrien ser "factoritzats"

Com es pot veure la funció d'aquest programa és força similar a la de l'anterior. Suposo que la redundància deu tenir alguna raó de ser i ambdós programes són complementaris, la veritat és que els errors que PMD detecta solen ser més "paraniocs" que el de FindBugs, però com que era un requeriment "ineludible" de mapsforge tampoc s'ha investigat molt en aquest sentit.

Les sigles PMD no tenen cap significat concret, segons els seus creadors:

4.8.1.4 JUnit Tests

JUnit és una biblioteca per realitzar "tests" automatitzats sobre classes Java.

Un "test unitari" és un mètode per provar fragments de codi font de manera individual per determinar si responen com és previst.

En el cas concret de les classes que hereten d'Android, però, els testos mitjançant JUnit es compliquen, ja que no es poden automatitzar sense fer servir el "framework" de JUnit específic d'Android (Android Test framework), que té components específics per provar classes relacionades amb el framework d'Android (que d'altra manera, i fent servir els mòduls de JUnit convencionals, fallarien).

El cas és, però, que el projecte mapsforge no proporcionava "tests" o proves per a cap classe que heretes d'Android, i només realitzava proves sobre les classes Java Bàsiques fent servir la implementació de JUnit "estàndard".

Així que vaig seguir la mateixa política, i no es van dissenyar "tests" per a cap classe que heretes d'Android.

Per a més informació consulteu[8].

4.8.2 Deliveracions sobre la implementació dels Parsers

Una de les primeres qüestions que vaig plantejar a la comunitat (mitjançant un debat del grup de desenvolupadors), va ser que aclaríssim quin mètode faríem servir per parsejar els fitxers.

Hi ha moltes més maneres de parsejar un XML de les que semblen a primera vista, i en el cas que ens ocupa vaig pensar que calia tenir-ho molt present, ja que al desenvolupar per a un dispositiu mòbil, les limitacions, sobretot de memòria semblava que podien resultar d'una incidència important, però a més també calia valorar de no sobrecarregar el projecte amb moltes biblioteques externes, ja que el sistema que triés quedaria inclòs a mapsforge com a una nova dependència. Per això volia saber l'opinió de la comunitat.

Les dos principals maneres de parsejar i crear XML's proposades per W3C (que poden ser considerades estàndards), que són DOM i SAX.

4.8.2.1 DOM

Es tracta de carregar tot el document en memòria i analitzar-lo per crear un objecte amb estructura d'arbre que pot ser accedit pel programa. En entorns amb molta memòria disponible aquest pot ser un mètode pràctic, però tenint en compte les limitacions de memòria dels dispositius Android, i la mida d'alguns fitxers GML o KML, es va pensar que no seria el sistema més adient.

4.8.2.2 SAX

En aquest cas, es tracta d'un parser "event-driven". Que no carrega el fitxer en memòria ni proporciona accés a un objecte estructurat amb tota la informació, sinó que cerca informació concreta i crida a determinades funcions al trobar-la. El consum de memòria d'aquest parser doncs és molt menor que la d'un DOM, però com a contrapartida cal estructurar les dades "manualment".

A partir d'aquí altres alternatives que s'han valorat han estat majoritàriament implementacions del que es coneix com a XML bindings (concretament Java-to-XML bindings).

Essencialment són procediments per representar els documents XML com a Objectes, en aquest cas objectes de Java directament.

Segons proposa Donny Hallman utilitzant aquests procediments es pot millorar el processament de GML (i també KML).

Existeixen moltes implementacións d'aquest mètodes, de les quals s'ha valorat la anomenada JiBX.

4.8.2.3 JiBX[41]

Tal i com es menciona a aquest blog[40], JiBX planteja una proposta aparentment excel·lent per a vincular dades XML a objectes Java sobre la plataforma Android.

Ja que és ràpida, lleugera i amb un impacte reduït en el projecte actual (només cal afegir uns 60KB addicionals a l'aplicació), fa servir l'XMLPull parser d'Android per defecte i és teòricament fàcil i còmode d'utilitzar.

Així que amb les atractives expectatives es va decidir fer una prova de la utilització d'aquesta biblioteca, i trobant la prova satisfactòria, acabar implementant la solució amb JiBX.

Es per això que més endavant es detallarà més aquesta biblioteca i la seva filosofia.

4.9 Conclusió

Com es pot veure el disseny global d'aquest projecte és complex. Això és fruit, suposo, de que és un projecte força heterogeni, i cadascuna de les parts que el conformen ha condicionat d'alguna forma el resultat final. Així doncs el disseny ha estat molt condicionat per elements externs i l'elecció de les tecnologies a fer servir. I les decisions de disseny que s'han pres han estat sempre per intentar mantenir el més simple possible la interacció entre les parts del sistema.

En termes generals s'ha intentat respectar la filosofia de cada un dels elements a integrar, així, per posar un exemple, per la comunicació dintre d'Android s'ha preferit la utilització d'Intents abans que d'altres alternatives ja que aquesta és "la manera de fer d'Android".

Per aconseguir això ha calgut documentar-se i entendre bé cadascuna de les solucions triades, i en algunes ocasions això ha resultat complicat degut a la manca de documentació a molt alt nivell d'alguns d'aquests projectes.

Així podríem concloure que la part fonamental d'aquest projecte ha estat la integració de sistemes.

Capítol 5

Implementació

5.1 Introducció

Podem separar la implementació d'aquest projecte en dos grans blocs conceptuals:

1. Eixamplar la biblioteca de mapes

   (a) En primer lloc la implementació i integració dels mòduls de GML i KML en la biblioteca mapsforge.

   (b) També portar la habilitat de escriure mapes directament al dispositiu, cosa que requeria portar Osmosis a Android (mapfile-writer4and).

   (c) I finalment la implementació de la façana de SL4A.

2. En segon lloc la integració de les biblioteques modificades en el projecte ZamiaDroid a nivell de demostració funcional.

   (a) Crear l'aplicació autònoma de visualització de mapes (erdapfel) i implementar-hi els mètodes de comunicació dissenyats.

   (b) Modificar el Zamia per assolir la comunicació bidireccional amb l'aplicació creada.

Cadascun d'aquests blocs ha tingut les seves particularitats i entorns de desenvolupament diferents, tal i com s'ha esmentat anteriorment, i a continuació es poden veure els detalls d'implementació més rellevants de cadascun.

5.2 Biblioteca de mapes

5.2.1 Problemàtica amb JiBX

Abans d'entrar en detalls sobre la implementació de les extensions de la biblioteca mapsforge per a KML i GML, deixeu-me comentar alguns problemes sorgits de la utilització de la implementació/utilització de JiBX.

El projecte JiBX proporciona un mètode automàtic per a generar classes Java a partir d'un esquema XML (codegen). Després d'això cal decorar aquestes classes ja compilades (en un procés anomenat bytecode enhancement) mitjançant un component anomenat "binding compiler", que llegeix un fitxer (definició del binding) on està estipulada com s'ha de fer la conversió de document XML a classe de Java. Aquest pas es necessari per tal que en temps d'execució JiBX sàpiga on posar cada element del fitxer a parsejar. Un cop seguits aquests passos ja es pot fer servir la biblioteca run-time de JiBX per convertir dinàmicament un fitxer que compleixi l'esquema en instàncies d'aquest conjunt de classes, i treballar amb elles.

La primera fase del procés per poder gaudir dels avantatges que brinda parsejar documents XML amb JiBX, doncs, és fer servir aquest mètode automàtic (codegen) que genera tant les classes d'objectes Java com el fitxer de definició de binding per continuar el procés.

Desgraciadament, el mètode automàtic de generació de codi que s'havia provat satisfactòriament durant l'anàlisi de JiBX (amb la versió 2.1 de l'esquema de kml) no funcionava per als esquemes de la versió "oficial" de kml, la versió 2.2 de opengis.

Després d'intentar arreglar els problemes i contactar amb la comunitat de JiBX sense èxit (podeu trobar més detalls a l'annex corresponent) es va optar per crear manualment un binding i utilitzar una jerarquia de classes de la versió 2.1.

El fitxer de definició de binding a més a més, és molt complex i te una gran varietat d'opcions i possibilitats, i la versió que es va fer manualment segurament no aprofita totes les capacitats de JiBX.

A més, degut a la complexitat d'elaboració d'aquests fitxers no es va aconseguir fer que diferents esquemes de KML convergissin en les mateixes classes

generades, obligant a crear una classe abstracta en les biblioteques esteses de mapsforge, que finalment s'especifiqui en cadascuna de les versions dels esquemes KML que es vulguin implementar/suportar (com es pot veure aquí Figure 5.1).

Tot això pot suposar que el parser sigui inestable per algunes determinades instàncies de fitxers KML, ja que no es garanteix la completa compatibilitat entre versions dels esquemes, ni que el procés de binding sigui prou robust com per tractar qualsevol eventualitat que es pugui torbar en un fitxer kml.

S'ha comprovat/garantit que tant els fitxers kml descarregats de google maps, com els fitxers generats per ZamiaDroid sigui acceptats sense problema, però no es descarta que en altres casos s'interpreti com a mal format un fitxer que teòricament compleix l'esquema de KML.

Un inconvenient semblant va succeir amb els esquemes de GML.

Aquest inconvenient es va detectar molt tard en el procés, i l'única alternativa viable en aquell moment va ser aquesta.

De tenir més temps, o un coneixement/suport més extens de la biblioteca JiBX, segurament es podria arreglar aquest problema i evitar errors d'execució.

També s'hauria pogut abandonar el suport de JiBX i implementar un parser SAX senzill, adaptat només per a les necessitats concretes d'aquest projecte. Però es va preferir mantenir JiBX i deixar com a treball futur la optimització del binding de JiBX.

5.2.2 Mapsforge-GML-KML

La arquitectura de mapsforge ja estava predefinida, i per tant no es podia modificar completament.

En aquesta secció parlaré exclusivament de la implementació dels mòduls de GML/KML propis i no de la implementació de la biblioteca mapsforge, informació sobre la implementació i altre documentació tècnica d'aquesta biblioteca es pot trobar aquí[24].

La manera menys invasiva que es va trobar per a afegir suport de KML i GML va ser crear una nova classe Overlay que permetes llegir fitxers d'aquest tipus, i generes les característiques contingudes en ells amb els Overlays ja existents per defecte a la biblioteca. És a dir una classe que contingues totes les capes representables del fitxer GML/KML i també els mètodes per tractar-les.

Una alternativa hagués estat modificar la classe MapView i permetre que de la mateixa manera que es creen mapes a partir de fitxers .map en el mètode setMapFile "entengués" també els fitxers KML i GML.

Això tenia però alguns inconvenients.

Per començar significaria canviar la manera com actualment es carreguen els fitxers de mapes, ja que ara mateix la carrega d'un fitxer implica la substitució completa d'un mapa per un altre, i no existeix la possibilitat de sobreposar les capes del mapa carregat actualment amb les del nou fitxer a carregar.

Aquest és un canvi que podria suposar poc important. Però si tenim en compte que la posició de la comunitat de mapsforge pel que s'entén de la proposta inicial en referència a la inclusió de KML, aquesta funcionalitat no es veu tan pràctica per a carregar mapes complets en aquests formats (cosa que també seria possible) sinó com a una manera de incloure dades addicionals als mapes ja existents.

És a dir, que l'escenari pel qual la comunitat de mapsforge veu més útil aquesta nova característica és el d'afegir informació, per exemple rutes o punts d'interès exportats des de google maps, o google earth, o qualsevol altre programa que permeti exportar a KML o GML.

En altres paraules, que la utilitat més gran que es veu a la addició del suport per a fitxers KML i GML és poder estandarditzar la manera com es gestionen aquests fitxers, que d'altra banda s'hauria de fer el desenvolupador final, amb el risc i les inconveniències que això comporta, i la poca eficiència d'haver de reescriure codis tant similars tantes vegades.

És per això que semblava més bona idea no haver de modificar la manera com ara mateix estan definides les interfícies per a la càrrega de fitxers de

mapes que ara existeixen, afegint la possibilitat de un "append" per entendre'ns.

I en darrer lloc aquesta estratègia hagués implicat una modificació important de la classe MapView, que és una classe molt important i que encara esta en constant desenvolupament per la comunitat de mapsforge, cosa que hagués dificultat molt la coordinació entre el desenvolupament dels mòduls GML/KML de la manera com s'havia plantejat.

Així que es va creure que la millor manera era crear Overlays capaços d'acceptar com a entrada fitxers KML o GML i generar-ne les característiques internament.

5.2.2.1 KMLBinding.java

Aquesta és una classe abstracta que representa una vinculació concreta de JiBX. Ha de ser implementada en alguna classe que accepti un xml schema de KML concret, i proporcionarà mètodes per parsejar fitxers KML vàlids i obtenir l'equivalent en Overlays de les seves "features".

Les classes que heretin de KMLBinding i implementin un esquema concret importaran les classes de Java generades gràcies al Procés previ de vinculació i generació de codi amb JiBX.

Així en el moment requerit faran ús dels mètodes d'unmarshall de la biblioteca jibx-run (la petita biblioteca que en temps d'execució parseja els fitxers a objectes Java) ja podem treballar amb la representació del fitxer en forma d'instàncies de classes Java.

El mètode més rellevant de la classe és parseKML, que s'encarrega de fer aquesta crida i a continuació processar els objectes Java obtinguts.

És en aquest moment on es fa la "transformació" de les característiques ("features") de KML a els Overlays més apropiats de que disposem.

Així aquesta "transformació" agafa les característiques especificades al fitxer KML i, depenent del seu tipus genera Overlays amb característiques equivalents.

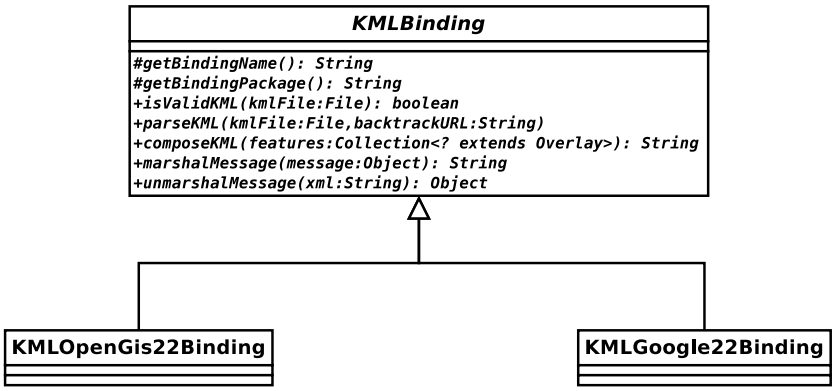

Figura 5.1: Diagrama de classes de KMLBinding i els seus hereus.

Actualment es tracten degudament els elements geomètrics poligonals, els camins i els punts, que són "transformats" als Overlays de mapsforge de OverlayWay amb emplenat intern, OverlayWay sense emplenat i KMLOverlayItem respectivament.

Els dos esquemes de KML que s'han vinculat amb JiBX, i que per tant la biblioteca pot interpretar són el de Google i el d'opengis.

5.2.2.2 KMLLayer.java

La classe KMLLayer mimetitza l'estructura d'OverlayList, ja que en el fons també és una llista heterogènia d'Overlays que implementa les interfícies Java de RandomAccess i de List.

En essència consisteix en una llista d'objectes de la classe abstracta Overlay amb tots el mètodes necessaris per garantir la implementació de les dos interfícies Java anteriorment mencionades i també una instància de la classe KMLBinding, que és una instància que representa una vinculació concreta de JiBX, i que conté una sèrie de mètodes per parsejar fitxers KML (en aquest cas concret).

Fent servir aquest mètodes del KMLBinding es poden obtenir les Overlays que conté el fitxer KML, i afegir-los a la llista que aquesta classe manté.

Aquesta classe no és "thread-safe" i se suposa que serà inclosa a la llista d'Overlays del mapView o en algun altre contenidor que n'asseguri el tractament com es degut durant el cicle de vida de l'aplicació.

5.2.2.3 GMLOverlay.java

La idea de GMLOverlay era inicialment la mateixa que per a KMLOverlay.

5.2.3 mapfile-writer4and

Mapfile-writer4and és la combinació del port de l'eina osmosis a Android i del mòdul d'osmosis mapfile-writer, que forma part de mapsforge.

La totalitat del codi que es volia "portar" era codi Java amb interfície de línia de comandes (sense GUI), així que semblava factible adaptar-ho a l'entorn Android, compilar-ho i distribuir-ho com a un paquet normal.

Llavors la comunicació entre les aplicacions que en volguessin fer ús i el mapfile-writer4and hauria de ser efectuada mitjançant intents com de costum en Android.

El procés de desenvolupament de mapfile-writer4and constava, doncs, de les següents parts en teoria:

- La adaptació d'Osmosis per a la plataforma Android
- La adaptació del mòdul de mapsforge
- La creació d'una aplicació d'Android

Abans de començar, però, vaig enviar un missatge a la llista de correu dels desenvolupadors d'osmosis (osmosis-dev) per saber si s'havia intentat abans realitzar aquest port, o si existia algun impediment de base que fes que es tractés d'una tasca inviable, i al veure que no era així, vaig començar.

Aquests són algunes de les dificultats tècniques van sorgir durant el procés. A continuació esmento les més importants:

5.2.3.1 Conversion to Dalvik format failed: Unable to execute dex: Java heap space

Per tal d'aconseguir el mapfile-writer4and, el primer que es va intentar va ser utilitzar l'eina osmosis sense cap modificació ni adaptació.

Per tant el primer que es va fer va ser crear una aplicació d'Android que inclogués l'eina osmosis i tots els seus mòduls descarregats directament de la pàgina del projecte.

Aquesta aplicació d'Android havia de permetre l'execució mitjançant intents, i també es va pensar de proporcionar-li una petita interfície gràfica pel cas de que es volgués executar com a "stand-alone".

Per aconseguir-ho el que calia fer en la activitat principal (i única) de l'aplicació era comprovar si s'havia invocat amb paràmetres des d'un intent (en cas de que fos així s'esperava tenir un paràmetre a "extras" que fos l'argument de línia de comandes per a l'eina osmosis) i en cas de que no fos així mostrar una simple interfície on es podia introduir aquests arguments i executar-los.

En ambdós casos, el que es feia era cridar a la classe principal d'osmosis (inclosa al projecte en forma de jar) amb un "execute", amb aquests arguments.

Però naturalment aquesta aproximació tant directa no va funcionar, posant de manifest que seria necessària una adaptació d'osmosis per a la plataforma Android.

De fet el primer problema que vaig trobar va resultar ser un problema de l'entorn de desenvolupament i no pas un problema conceptual o de procediment.

L'error "Conversion to Dalvik format failed: Unable to execute dex: Java heap space" era degut (en aquest cas) a una configuració errònia de l'eclipse (i al meu entendre a una implementació no gaire eficient del compilador dex per Android).

El problema era que la memòria de la màquina virtual de Java assignada a l'eclipse mateix s'esgotava quan es volien processar una gran quantitat de paquets jar per incloure al projecte.

D'alguna manera el consum de memòria es disparava de manera gairebé exponencial per cada nou jar que s'afegia al projecte, i el fet que inicialment s'intentessin incloure tots els mòduls d'osmosis al projecte resultava ser massa (encara que el conjunt de biblioteques "només" pesés uns 8Mb).

Per tal de solucionar aquest error es va incrementar la memòria de que disposava l'eclipse mitjançant la modificació d'un fitxer de configuració, i també es van eliminar mòduls innecessaris d'osmosis del projecte.

5.2.3.2 "No validating SAXParser implementation available"

Després d'això, un dels problemes més importants que van sorgir va ser un error d'execució de l'Osmosis en la plataforma Android, que es queixava de la manca de un parser SAX a l'entorn d'execució *"No validating SAXParser implementation available"* quan osmosis intentava llegir el fitxer de configuració plugin.xml.

Això molt probablement fos degut a que l'entorn d'execució d'Android, tot i ser una màquina virtual de Java, té algunes particularitats, i li manquen alguns elements estàndard.

Per tal de solucionar aquest problema la ajuda de la comunitat osmosis-dev va ser molt valuosa.

Un dels membres d'aquesta comunitat em va comentar que aquest error succeïa en un estadi on l'aplicació intentava cercar els plugins presents en el sistema per configurar-se. I també em va fer saber que hi havia diversos mecanismes per a poder dir a osmosis quins plugins existien en el sistema, i el mètode que provocava aquest error era l'ús de JPF (Java Plugin Framework). També em va comentar que el JPF no era ni tan sols el mètode més emprat.

El fet que osmosis permetés altres maneres de carregar plugins que no fos amb aquest fitxer xml obria la porta a l'opció de modificar osmosis perquè prescindís completament de JPF i d'aquest mecanisme.

Així es va descarregar el codi font d'osmosis i modificar perquè no utilitzés el sistema de gestió de plugins JPF.

Aquesta versió "capada" d'osmosis requeria la configuració "manual" dels plugins.

Això és va aprofitar per fer una tria dels plugins que eren estrictament necessaris per proporcionar les funcionalitats que requeria la nostre biblioteca, i prescindir de la resta.

Els mòduls que es van conservar van ser els que permeten la lectura de fitxers binaris d'OpenStreetMap (.pbf), els que permeten la lectura de fitxers XML d'OpenStreetMap (.osm), així com alguns plugins auxiliars d'aquests

dos mòduls, i evidentment el mòdul de mapsforge-map-writer (que permet escriure el fitxer .map).

Aquest procés de purga de mòduls va ajudar també a solucionar l'error anterior de manera definitiva.

5.2.3.3 OutOfMemoryError

Un cop adaptat osmosis per a Android i configurats els plugins correctament l'aplicació mapfile-writer4and ja estava preparada, i va ser provada satisfactòriament a l'emulador d'Android, i encara que evidenciava que era un procés lent, funcionava correctament.

Però per desgràcia, al provar aquesta funcionalitat en el meu telèfon de proves més endavant (en aquell moment un HTC magic) va sorgir un lleig error en temps d'execució que es queixava de que la memòria per a la pila de Java s'havia esgotat i no era possible generar el fitxer.

En el cas de dispositius més potents on es va provar aquesta aplicació (com el Nexus 7), el procés s'executava satisfactòriament.

El problema en els dispositius més antics i/o limitats, era que la memòria que el Dalvik proporcionava per a la pila no era suficient per executar el mòdul mapsforge-map-writer.

El procés per generar un mapa de mida petita amb osmosis+mapsforge-map-writer consumeix aproximadament 90Mb.

La mida de la pila de Java d'Android varia molt segons el dispositiu, però, per exemple en el cas d'un telèfon de gamma mitjana/baixa sol ser de l'ordre de 64Mb mentre que en el d'una tauleta de gamma alta pot ser de 128Mb o 256Mb. En un dispositiu com l'HTC magic, que ja es força antic, aquesta memòria és de tan sols 24Mb.

Està clar, doncs, que aquest error era molt preocupant i desagradable, ja que limitava molt la utilització d'aquesta funcionalitat de la biblioteca. Per això es va dedicar un temps considerable a cercar possibles solucions al respecte[46][47]. Tant per intentar optimitzar osmosis o mapsforge-map-writer, com per intentar expandir la memòria de la pila de Java als dispositiu.

Els resultats d'aquesta recerca van ser decebedors.

En primer lloc es va comprovar que malgrat que el dispositiu pogués tenir suficient memòria física per executar la tasca, degut al restringit entorn de la màquina virtual de Java que és el Dalvik resultava impossible incrementar la memòria de que disposava una aplicació Android.

També es va veure que osmosis i el mòdul mapsforge-map-writer (que és el pas que, de fet, esgotava la memòria) no podia ser optimitzat per consumir menys recursos, ja que això anava en contra dels principis de disseny del mòdul mateix, que prioritzava un alt consum de recursos per proporcionar un baix temps d'execució (cal recordar que aquest mòdul va ser dissenyat tenint en ment la seva execució en un servidor). A l'annex podeu trobar més informació.

Sembla a ser que si es prescindeix del Dalvik (per exemple fent servir l'NDK) és possible aprofitar més la memòria del dispositiu sense restriccions, però portar osmosis i el mòdul mapsforge-map-writer (escrits en Java) al llenguatge natiu de l'NDK (C o C++) es va descartar per ser inviable. També es va descartar reescriure el mòdul d'escriptura de fitxers .map per fer-lo més "eficient" però lent, ja que es va creure que es desviava molt del marc del projecte i no interessava al projecte mapsforge.

Així que malgrat els esforços per portar osmosis a Android, i malgrat també el relatiu èxit en aquesta empresa (osmosis i la majoria dels seus mòduls podien ser executats sense inconvenient), al final va resultar poc útil, ja que només dispositius de segment elevat són capaços d'executar aquesta tasca al dispositiu mateix.

La resta d'usuaris han de processar els mapes en un ordinador personal o descarregar-los d'algun servidor. Cosa que no resulta ideal (però recordem que és el "modus operandi" d'altres opcions, com ara OsmAnd).

5.2.4 SL4A Facade

L'arquitectura de SL4A ja ha estat explicada anteriorment (Figure 3.18). Segons aquesta arquitectura, per poder afegir suport per a una "nova" biblioteca de Java calia modificar el codi de SL4A (ja que no es preveia cap mecanisme dinàmic per afegir noves façanes). Concretament el que cal fer és un nou projecte façana amb una classe que hereti de "RpcReceiver", i allà

posar-hi els mètodes que es volen "publicar". I modificar la classe FacadeConfiguration del nucli de SL4A perquè sàpiga de l'existència d'aquesta nova façana.

Fent això, en el moment que un dels llenguatges d'scripting cridi a la "nostra" funció, la crida s'interceptarà i s'executarà el codi que hàgim implementat a la façana seguint el mecanisme convencional de SL4A.

Així les classes que s'han implementat o modificat són aquestes:

1. *com.googlecode.android_scripting.facade.MapsforgeFacade.java* dins del projecte MapsforgeFacade, que serà la nova façana encarregada de comunicar-se amb la biblioteca de mapes.

2. *com.googlecode.android_scripting.facade.FacadeConfiguration.java* dins del projecte ScriptingLayer, que és el projecte "principal" del codi font de SL4A, i cal ser modificada per afegir la façana creada i que el sistema la reconegui.

La creació del nou projecte de Java que conté la façana no te cap particularitat (només calia tenir en compte el nom del paquet fos *com.googlecode.android_scripting.facade*. La implementació d'aquesta classe senzillament proporciona un mètode showMapsforgeMap, que espera tres cadenes de caràcters com a paràmetres, i fa una crida a erdapfel amb els mateixos arguments que acceptaria la invocació de l'intent, i que estan especificats aquí, i que són opcionals.

Com s'ha comentat abans, la utilització d'erdapfel com a punt d'entrada de la façana SL4A es va decidir per tal d'evitar afegir les biblioteques esteses de mapsforge coma dependència de SL4A, cosa que no tenia massa sentit.

Això, però, força a que per fer servir aquesta funcionalitat és necessari instal·lar erdapfel (a meś de la versió "modificada" de SL4A).

Tot plegat fa que ara mateix aquesta sigui més aviat una implementació "experimental", encara que funcional.

5.3 Integració amb ZamiaDroid

Per a la integració amb ZamiaDroid calia fer unes quantes modificacions a Zamia i també crear l'aplicació de visualització de mapes erdapfel.

En primer lloc es comentaran les modificacions a Zamia, i posteriorment es detallaran els apartats de desenvolupament d'erdapfel més interessants.

5.3.1 Modificacions a ZamiaDroid

Tal i com s'ha explicat a l'apartat d'anàlisi i disseny, s'ha procurat haver de modificar el mínim possible el ZamiaDroid.

Malgrat això, algunes modificacions han estat indispensables per poder permetre l'ús de les biblioteques esteses de mapsforge (a través d'erdapfel).

Per realitzar aquestes modificacions s'ha comptat amb la inestimable ajuda de David Martí Pino, el principal desenvolupador i creador de ZamiaDroid.

A part d'afegir la invocació d'erdapfel mitjançant intents des de Zamia (la modificació més senzilla d'entendre), ha calgut fer un parell d'altres coses, una mica menys evidents.

Ara les esmentaré, i més endavant, a l'apartat d'erdapfel s'acabarà de veure clar la seva raó de ser.

5.3.1.1 Afegir referències als fitxers KML generats

Per tal de poder referenciar els objectes del KML i que el Zamia entengui quin dels objectes ha "premut" l'usuari des d'erdapfel, s'ha fet servir el mateix camp "id" que totes les "features" d'un KML tenen.

Aquest és un camp que no es fa servir per la representació visual, i que Zamia tampoc necessitava, així que es va acordar que seria l'identificador de tornada.

Aquest identificador no te perquè ser el mateix de la base de dades de Zamia, només ha de ser un paràmetre que el programa hoste (en aquest cas Zamia) sàpiga interpretar inequívocament.

5.3.1.2 Afegir un punt d'entrada per a la "tornada" des de l'aplicació erdapfel

També calia preparar ZamiaDroid per a ser "iniciat" des d'erdapfel.

En aquest cas això va consistir en afegir un "punt d'entrada" directament a l'editor de citacions mitjançant un intent.

Bàsicament les modificacions a fer al Zamia van ser modificar lleugerament l'Activity de l'editor de citacions perquè pogués ser cridada directament.

5.3.2 Erdapfel

L'aplicació erdapfel consta d'una pantalla (Activity) principal i dues d'auxiliars. La implementació de la Activity de configuració no te massa rellevància, així que no es comentarà, però em sembla interessant comentar alguns detalls de la pantalla selecció de fitxers:

5.3.2.1 FilePicker

Android no disposa d'un navegador de fitxers per defecte, que permeti recórrer la estructura de directoris del sistema. És possible instal·lar aplicacions que proporcionin aquesta funcionalitat, però no sembla haver-hi cap estàndard, ni tan sols estàndard "de facto". Així doncs, si una aplicació vol donar la possibilitat de seleccionar fitxers de manera visual, normalment cal que implementi el selector "manualment".

És clar doncs que existeixen moltes implementacions doncs per a aquesta funcionalitat, i moltes d'elles de codi obert. En el nostre cas concret, s'ha aprofitat que un dels exemples de mapsforge implementava una interfície similar per adaptar-la a les nostres necessitats i fer-ne ús.

Es tracta de la classe filePicker i de les classes fileFilter.

filePicker és una classe que hereta d'Activity i implementa la interfície AdapterView.OnCLickItemListener.

A grans trets el que fa és mostrar els continguts d'un directori (ja siguin fitxers o carpetes) i permet la navegació per les carpetes i la selecció dels fitxers.

S'ha modificat, però, perquè només es mostrin els fitxers vàlids que es permet carregar en cada moment gràcies a la utilització de les classes fileFilter.

Així un cop seleccionat un fitxer vàlid, la activitat acaba i publica el resultat fent ús del procediment usual en Android (setResult).

Com que erdapfel haurà cridat aquesta Activity emprant el mètode "startActivityForResult" aquestes dades estaran disponibles per la Activity prin-

cipal del programa. (aquest és el procediment de comunicació més usual entre Activities en al plataforma Android)

També cal dir que els "recursos gràfics" per defecte d'Android no disposen de cap representació d'un fitxer o d'una carpeta (almenys a les versions per les quals es pretén donar suport amb erdapfel). Per tant es van haver de manllevar icones adients i empaquetar-les amb l'aplicació.

Les icones de directoris es van manllevar del Tango desktop project[65], ja que es tracta d'un projecte completament Lliure i gratuït, i fa servir estàndards visuals molt estesos i en certa manera intuïtius per als usuaris.

Les icones per als documents que suporta erdapfel són les mateixes que la icona principal del projecte, i va ser dissenyada especialment per a l'ocasió.

Figura 5.2: Captura d'erdapfel filePicker.

La resta de parts interessants de la implementació formen part de la pantalla principal a on es veu el mapa.

5.3.2.2 MainActivity

La activitat principal d'erdapfel és la que mostra el mapa, i es carrega directament al començar l'aplicació.

El primer que comprova aquesta Activity (en el mètode OnCreate) és si ha estat invocada amb paràmetres per carregar directament un fitxer de mapes (KML, GML o map).

Això ho determina fent una cerca a la estructura d'extres del mateix intent. Aquesta estructura és el sistema usual de passar paràmetres quan s'invoca un intent d'Android, i bàsicament es tracta d'un diccionari clau/-valor on es guarden paràmetres que poden ser de tipus simples de Java, o qualsevol classe que implementi la interfície serializable. Les claus que es cerquen en aquesta estructura són les següents:

- "MapFile"
- "KMLFile"
- "KMLCallBackURL"

La primera clau espera tenir per valor un String que contingui el camí (path) a un fitxer de tipus map, i si al carregar-se l'aplicació es detecta aquesta clau no s'intenta carregar un mapa des d'internet sinó que es carrega directament el mapa indicat.

En cas de que el mapa no es pogués carregar, o senzillament aquesta clau no fos present a la estructura d'extres, es procedeix a intentar descarregar el mapa d'OpenStreetMap generat amb mapnik.

La segona clau està emparellada amb un String que representa també una ubicació de fitxer, en aquest cas s'espera un fitxer KML.

La distinció entre KML i Map en aquest nivell és conseqüència del tractament dels fitxers KML com a Overlay i no com a format de "mapa" tal i com mapsforge ho entén (com es va comentar a l'apartat d'anàlisi i disseny).

Això fa que el tractament per a aquest fitxers sigui molt diferent del dels de mapes. A més aquesta separació té sentit si en un futur es volgués, per exemple, carregar un mapa i al mateix temps afegir-hi una capa des d'un KML. O donar la possibilitat de carregar més d'un KML (en diferents capes) a l'inici de l'aplicació, cosa que no te sentit amb els mapes perquè només es mostraria un sol mapa a cada instant.

La tercera clau és completament opcional, però només te sentit la seva presència quan es carrega un fitxer KML.

Es tracta d'una dada necessària per establir la comunicació entre aplicacions des d'erdapfel cap a la possible aplicació "mare" que el vol fer servir com a visor.

El valor d'aquesta clau ha de ser una URI (de fet un String que conté una URI) que especifiqui el "punt de tornada" que l'aplicació "mare" vol que erdapfel empri.

Això o algun mecanisme similar és necessari per poder establir una comunicació bidireccional entre erdapfel i les aplicacions que el volen usar, tal i com s'ha explicat a l'apartat de disseny (en aquest cas concret es demostra la comunicació amb Zamia, però el mecanisme pot ser estès per a qualsevol possible "client").

El punt d'entrada a especificar no és altre que el "nom" estàndard d'Android per l'intent que es vol executar quan l'usuari interaccioni amb els objectes carregats del KML. En el cas concret de Zamia, per exemple, el punt d'entrada és l'editor de citacions. Així, quan es realitza un "onTap" sobre una citació a l'erdapfel, aquest, invoca un intent amb la URI definida i amb els paràmetres de l'objecte del KML que l'usuari ha pitjat. Zamia, al ser cridat mitjançant aquest intent, agafa els paràmetres que li proporciona erdapfel i els interpreta (en aquest cas concret el paràmetre més rellevant és l'identificador de citació), fa una consulta sobre la seva base de dades i mostra l'editor de citacions, de la mateixa manera que ho hauria fet si s'hagués "pitjat" una citació des del mapa de google integrat dins de ZamiaDroid.

El fet que Android permeti invocar intents a partir d'una URI d'aplicació facilita que aquest "punt d'entrada" pugui ser especificat de manera unívoca, clara i d'una manera acceptada pel sistema Android.

Actualment la càrrega de fitxers GML a l'inici de l'aplicació no està contemplada.

Això es degut a que ZamiaDroid (que és la principal aplicació que fa ús d'aquest "punt d'entrada" a erdapfel ara mateix) no requeria aquesta característica, i també perquè en el disseny inicial d'erdapfel no s'especificava aquesta funcionalitat.

Si es volgués implementar tan sols caldria afegir una nova clau que fes possible, o estendre l'ús de la clau "KMLFile", encara que això podria portar lloc a confusió.

5.3.2.3 Detalls de la interfície principal

Respecte la pantalla principal comentaré breument com s'han implementat els modes de dibuix de polígons i de mètriques, i com s'interacciona amb

les biblioteques esteses de mapsforge, tant per mostrar documents kml o les figures geomètriques, com el mecanisme per generar mapes offline que proporciona mapfile-writer4and.

Abans d'això, però comentaré breument algun detall de la interfície principal que es mostra.

Tal i com s'ha dit al capítol d'anàlisi i disseny, es volia que la interfície d'usuari per a la visualització de mapes fos neta i clara, i que els controls no obstruïssin la visió dels mapes.

Per assolir això es va optar per fer un ús intensiu del menú desplegable d'Android (OptionsMenu) per a les opcions de carregar capes KML/GML, d'obrir fitxers de mapes locals, guardar mapes offline, exportar les capes dibuixades a KML o accedir a la configuració de l'aplicació.

Això, però, va fer sorgir problemes en les versions més noves d'Android (a partir de la 4) on es "desencoratja" l'ús dels botons físics i d'aquest menú en favor de barres d'eines desplegables (action bar). Google recomana als desenvolupadors canviar les seves interfícies considerant aquesta eventualitat, ja que en un futur no garanteix que aplicacions dependents del botó físic de menú siguin acceptades.

Per mantenir el nivell de compatibilitat amb sistemes antics, però es va decidir obviar aquestes recomanacions i solucionar els errors que poguessin aparèixer en versions noves en el moment que aquests errors apareguin.

Fent servir aquest "menú" d'opcions desplegable ens asseguràvem que la pantalla estava "neta" mentre l'usuari consultava el mapa.

Però per a permetre a l'usuari pintar polígons o mesurar distàncies es va voler implementar una solució més accessible i "dinàmica". Per això es va intentar imitar els controls de "zoom" que mostra el mapa de mapsforge (i el de google), que només apareixen durant un temps limitat després que l'usuari hagi realitzat una acció.

Dissortadament degut al fet que per aquesta aplicació s'ha interceptat i "sobreescit" el control d'esdeveniments (events), tal i com s'explicarà més endavant amb detall, aquesta "imitació" no és perfecte, i a vegades es descoordina respecte als controls de zoom.

Figura 5.3: Captura d'erdapfel mainActivity.

Figura 5.4: Captura d'erdapfel mainActivity amb la ui amagada.

5.3.2.4 Guardar mapa offline

Tal i com s'ha explicat a l'apartat de mapfile-writer4and per convertir dades d'osm en format xml (.osm) o binari (.pbf) fent servir aquesta eina cal invocar-la com un intent i proporcionar un String (mitjançant els extras) amb els arguments per a osmosis, és a dir, quins plugins ha de carregar, i quins paràmetres es volen executar (en el nostre cas un fitxer d'entrada, un de sortida i algunes opcions addicionals).

Però a més d'invocar el mapfile-writer4and, l'aplicació erdapfel ha de fer més coses. A continuació es detalla el procés:

- Delimitar la zona que es vol guardar.
- Construir una petició per l'API v0.6 de OpenStreetMap amb aquestes dades[45].
- Obtenir les dades des d'internet i guardar-les localment.
- Invocar mapfile-writer4and per crear un fitxer .map que es pugui obrir amb erdapfel.
- Netejar dades temporals.

Aquests poden ser processos llargs en alguns dispositius, ja sigui per la gran capacitat de càlcul que fa servir mapfile-writer4and, com pel fet d'esperar la resposta de servidors remots per a obtenir les dades d'OpenStreetMap.

És per això que es va decidir implementar aquestes tasques de manera asíncrona en segon pla, fent servir els AsyncTasks i Threads d'Android.

La classe AsyncTask empaqueta la creació de Threads (fils d'execució) i Handlers per gestionar-los.

Quan la tasca ha finalitzat es comunica el resultat a l'usuari i es retorna a erdapfel.

Si la operació falla (perquè mapfile-writer4and no s'ha trobat al sistema o no ha tingut prou memòria, o perquè no s'han pogut obtenir les dades d'internet) també es comunica i no es fa cap més acció.

Amb més temps s'hauria pogut buscar un mecanisme per ajudar als usuaris a acabar aquest procés de generació de mapa en els seus ordinadors personals (conservant les dades de la zona que es volia processar, per exemple). Però finalment això no es va poder fer.

5.3.2.5 PolygonMode

Activar el mode d'edició de polígons implica modificar el comportament de la pantalla principal per permetre dibuixar sobre el mapa formes poligonals.

A tal efecte, s'ha implementat una interfície simple i força intuïtiva, tot i que no perfecte.

Quan el mode d'edició està activat els "controls" que fins ara servien per navegar pel mapa (desplaçant-lo) o fer "Zoom" (amb un doble "tap", per exemple) es veuen alterats i passen a ser usats per a inserir nous vèrtex de polígons o modificar els vèrtexs existents (podeu veure més detalla a la guia d'utilització del programa).

La implementació que s'ha fet servir per aconseguir això, consisteix en capturar els events generats per l'usuari i tractar-los manualment en cas d'estar en aquest mode d'edició (sobreescribint la funció dispatchTouchEvent i evitant que l'esdeveniment es propagui a classes superiors de la jerarquia d'herència si s'està en el mode d'edició).

Això fa que el mode d'edició tingui control absolut sobre els gestos que l'usuari fa per pantalla, i inhabilita efectivament els controls "normals" del mapa.

Aquesta va ser la millor manera que es va trobar per permetre una edició pràctica i "intuïtiva" de manera senzilla.

Com a contrapartida, aquest mètode implica haver d'implementar "manualment" la detecció del tipus d'esdeveniment (double tap, long press, etc.), o de multi-touch, i això suposa una certa redundància, i pot arribar a comportar lleugeres diferències en el comportament i la resposta que l'usuari percep respecte la gestió d'events que s'efectua normalment al dispositiu (cosa que pot portar a certa confusió).

El mètode de mantenir estàtic el mapa mentre es dibuixa i no poder desplaçar-lo ni ampliar-lo durant la edició també pot ser problemàtic o frustrant per alguns usuaris (sobretot amb dispositius amb pantalles petites). Però a banda d'això, es poden aconseguir pintar polígons complexos de manera força intuïtiva i simple, suportant telèfons que no disposen de multi-touch, i es va considerar una implementació prou satisfactòria.

5.3.2.6 MetricsMode

La implementació dels controls del mètode de mètriques són gairebé idèntics als del d'edició de polígons, així que no els comentaré. Podeu veure amb més detall la utilització d'aquest mode de treball aquí.

En canvi la implementació d'aquest "mode" fa servir unes capacitats de la biblioteca de mapes que encara no s'han comentat, i que són interessants.

El mètode de mètriques treballa (com en el cas anterior, de fet) principalment amb coordenades de pantalla (no amb punts geogràfics).

Però per calcular distàncies aproximades és necessària la conversió de coordenades de pantalla a punts geogràfics (naturalment).
Això es realitza fent servir els mètodes "fromPixels" de la classe Projection (a la que es té accés, ja que cada MapView en conté una instància concreta) i el mètode "distanceBetween" de la classe Location (que accepta dos parells de coordenades lat/lon).

Tant els mètodes necessaris per fer aquesta transformació com els mètodes per obtenir distàncies aproximades entre dos punts geogràfics els proporciona la biblioteca de mapsforge.

Però per calcular àrees he hagut d'implementar una versió d'un algorisme de calcul d'àrees de polígons irregulars.

L'he fet basant-me en l'algorisme descrit aquí[50]. Aquest mètode te certes limitacions. Per començar assumeix una terra "plana" i sense accident geogràfics (no te en compte altituds ni la curvatura de la terra) així, per a àrees molt grans l'error acumulat pot ser notable i els resultats mostrats s'han de prendre tan sols com a aproximacions. A més tampoc mesura bé l'àrea d'un polígon intersecant.

Malgrat aquestes limitacions s'han fet algunes proves amb el càlcul d'àrees de zones geogràfiques conegudes, i els resultats són prou acceptables.

Com a petit apunt final, també s'ha implementat un senzill mecanisme de "precisió variable" per a mostrar les magnituds més significatives en cada moment, tant per les distàncies (que es pinten a sobre les línies) com per les àrees.

5.4 Conclusió

Finalment, per tancar aquest apartat, enunciaré breument les conclusions més rellevants que n'he extret.

5.4.1 JiBX

Sobre l'ús de l'eina JiBX com a "parser" de XML, he dir que no estic content d'haver-hi treballat.

Malgrat els beneficis que aquesta metodologia aportava en aquest cas concret, la manca de desenvolupament de les eines automatitzades de generació de codi i bindings va suposar un gran entrebanc pel desenvolupament d'una part clau del projecte.

A més, tot i tenir una documentació molt àmplia, complerta i ben estructurada, la gran complexitat d'aquesta eina, sumat a les dimensions dels esquemes que es volien tractar (GML en particular és d'una complexitat notable) va dificultar molt la generació "manual" d'aquests elements, i en definitiva va acabar causant que el resultat final fos, com a mínim, poc òptim i va convertir el parser de les biblioteques en un dels elements més inestables i propens a errors de tot el sistema.

També vaig trobar a faltar el suport de la comunitat en els meus dubtes i problemes, ja que els desenvolupadors a que vaig consultar mitjançant la llista de correu no em van ajudar a resoldre cap dels problemes que vaig patir.

5.4.2 mapsforge

L'elecció del projecte mapsforge també va tenir els seus inconvenients.

En primer lloc, la documentació de mapsforge no era tant detallada com jo hauria desitjat, i en alguns casos era obsoleta.

La biblioteca contenia alguns "bugs" que es van haver d'anar arreglant amb pedaços mentre s'hi treballava (malgrat que la expansió de GML/KML s'estava realitzant sobre una versió "estable" de mapsforge (la 0.3.0)). Recordo especialment un problema amb la gestió de les "tessel·les" que apareixia quan s'executava el programa en la versió 4 d'Android[70].

A més l'API d'Overlays amb la que vaig treballar tenia seriosos defectes de disseny i vaig haver de fer algunes modificacions en classes d'aquesta API per tal de poder-hi treballar (cosa que a priori no hauria d'haver passat mai). Per exemple, les classes que heretaven dels ItemizedOverlays no tenien cap mena d'accés sobre les dades que aquesta classe contenia, i pràcticament només es podia mostrar, però si un cop mostrat un punt (per exemple) es volia consultar a quina posició geogràfica es trobava, això no es podia fer. És a dir, que havien implementat els mètodes "set" però no els "get" quan això no tenia cap raó de ser que jo pogués veure.

El pitjor de tot, però va ser que un cop ja s'havia implementat bona part de l'"expansió" de KML i GML sobre aquesta API d'overlays se'm va comunicar que estava previst modificar l'API d'Overlays radicalment per a la nova versió 0.3.1 (segurament per solucionar alguns dels defecte que vaig detectar mentre hi treballava).

Això va causar certa sensació de "desemparament", i fa molt difícil la futura integració d'aquesta "extensió" de mapsforge al projecte principal, ja que implica reescriure gran part (per no dir tot) el suport de KML i GML i adaptar-lo a la nova API d'Overlays.

Quan aquest fet va tenir lloc es va decidir de continuar desenvolupant amb la versió antiga de la API més que res perquè ja estava molt avançada la implementació, però també per temes d'estabilitat.

Durant el desenvolupament d'aquest PFC també es va desplaçar el repositori de mapsforge des d'un control de versions de subversion a git, i durant el procés es va perdre totes les revisions anteriors al canvi.

Tot plegat, junt amb el poc interès que ha sostingut la proposta d'ampliació, fa que sigui poc probable que al final el treball realitzat s'integri al projecte mapsforge.

5.4.3 SL4A, osmosis i treball sobre codi de tercers

El fet d'haver hagut de modificar el codi font d'alguns d'aquest projectes (com el d'osmosis, el de SL4A o el de mapsforge) m'ha permès conèixer en primera persona els avantatges d'un bon disseny de software.

En aquest aspecte valoro molt positivament tant el projecte osmosis com el de SL4A, ja que tenen un codi molt net i clar, modular i fàcilment adaptable a les necessitats que han sorgit, i amb una documentació prou entenedors (tot i que no tant extensa com hauria requerit una assignatura d'enginyeria del software de la carrera).

Així que "submergir-se" en aquests codis font no has estat tant traumàtic com esperava.

Capítol 6

Planificació i estudi econòmic

6.1 Planificació

Les converses per començar aquest projecte van començar pels volts de febrer de 2012.

Com es pot veure doncs, aquest projecte s'ha allargat molt en el temps, majoritàriament per raons externes al projecte.

Això ha fet que la planificació inicial del projecte hagi quedat obsoleta. Malgrat aquests imprevistos adjunto tot seguit els diagrames originals per a la planificació del projecte (Figure 6.1).

Aquest és un diagrama més acurat i realista que es va fer després de trobar aquests inconvenients i veure que les previsions inicials no eren assumibles (Figure 6.2). Bàsicament es tracta de les mateixes tasques però on s'han afegit algunes tasques en funció de les conclusions tretes de solucionar els esmentats inconvenients.

Durant aquest període també es van aprofitar per replantejar algunes de les tasques a realitzar, sobretot pel que fa a l'aplicació de prova. I es va afegir la integració al ZamiaDroid com a possibilitat pràctica (anteriorment només s'havia estudiat).

Per això hi ha plantejades unes noves fases de disseny i especificació.

Aquesta és una versió del mateix gantt amb les tasques de la memòria desplegades (Figure 6.3).

Aquí podem observar amb més detall la part d'Implementació del "redisseny" (Figure 6.4).

Aquests gantts mostren el repartiment de les tasques al llarg del temps que s'ha estat treballant en el PFC. Algunes de les tasques s'han perllongat molt en el temps, ja que es va contar a dedicar-hi un temps més limitat del que s'havia plantejat inicialment en la primera fase.

Malgrat això hi ha certes fases que no s'adiuen tampoc a aquesta segona planificació. Per exemple el re-disseny/re-plantejament no va ser tan prolongat com es preveia ja que es van aconseguir mantenir moltes de les decisions de disseny plantejades anteriorment. En canvi la implementació va comportar alguns problemes tècnics d'integració de sistemes que ja s'han pogut veure a la memòria, i es van haver de realitzar d'altres tasques que no van ser previstes.

La poca fiabilitat de les planificacions ha estat deguda principalment a motius externs al desenvolupament del projecte, però si que és cert que he notat una manca d'experiència en avaluar el temps necessari per a realitzar tasques d'investigació i integració de sistemes.

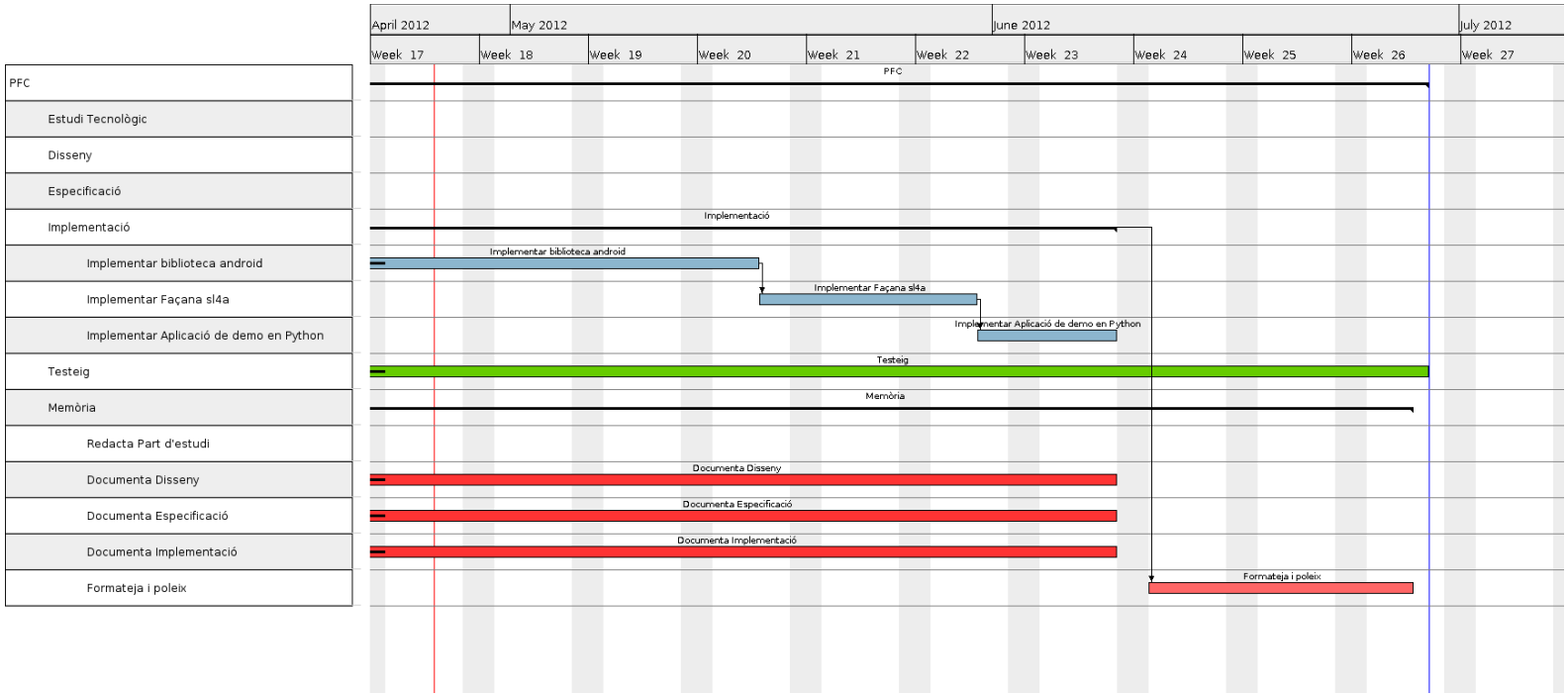

Figura 6.1: Diagrama de gantt inicial presentat a l'informe.

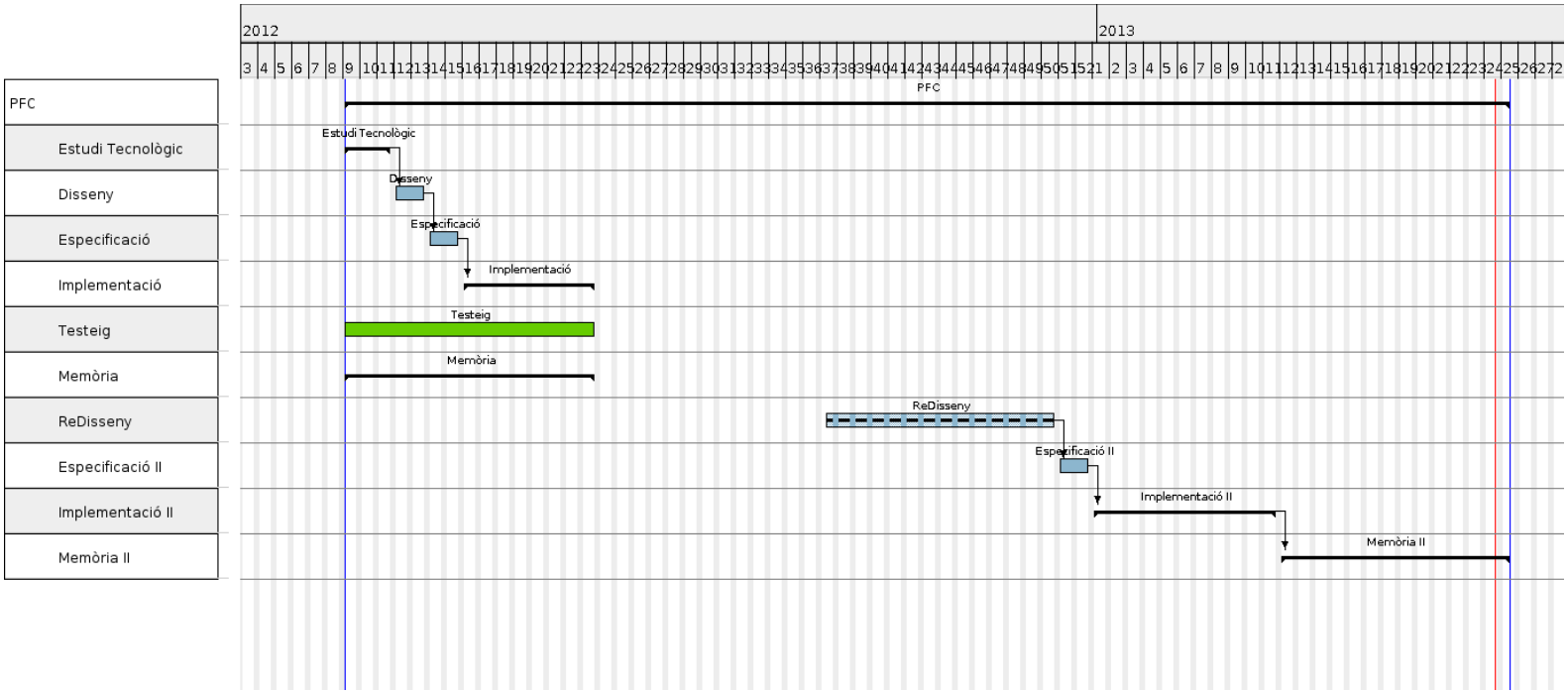

Figura 6.2: Diagrama de gantt revisat.

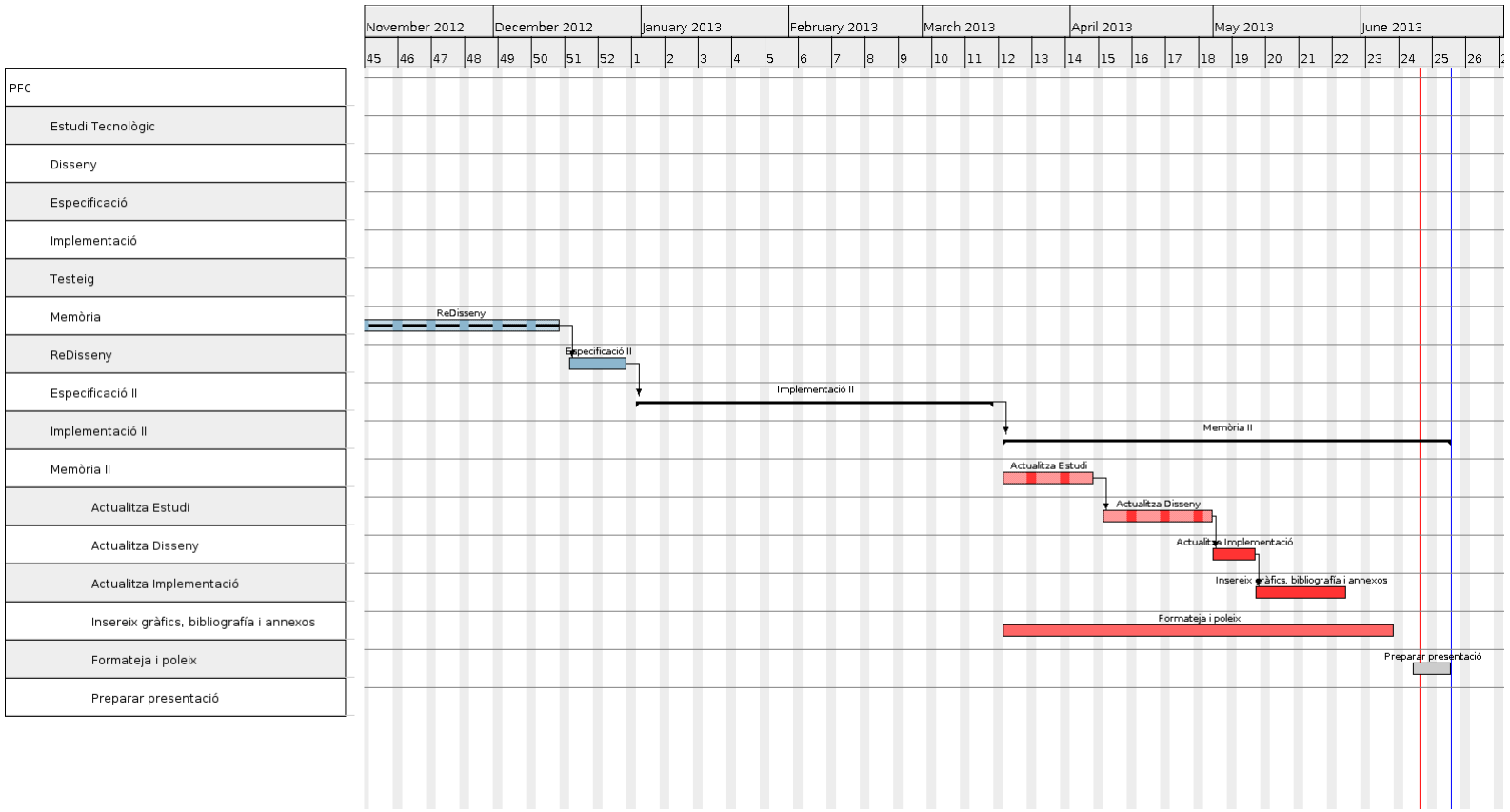

Figura 6.3: Diagrama de gantt revisat, amb detall de la memòria.

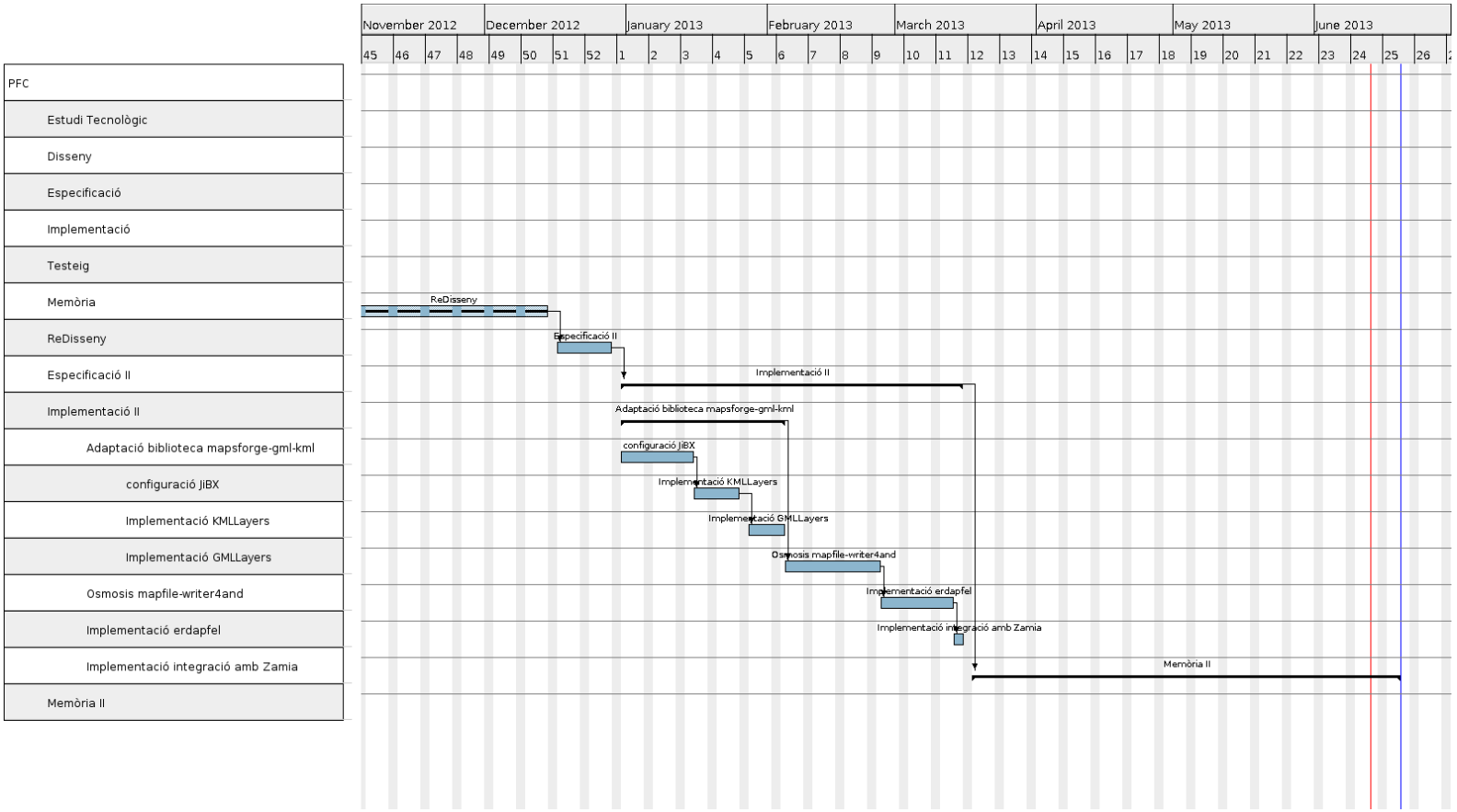

Figura 6.4: Diagrama de gantt revisat, amb detalls de desenvolupament.

6.2 Estudi econòmic

A continuació presento una estimació dels costos econòmics del projecte.

6.2.1 Personal

Els costos més importants d'aquesta estimació corresponen als costos de personal (recursos humans).

Això es degut a la llarga duració del projecte en el temps, però també (com veurem a continuació) a l'elecció de software lliure per a totes les eines i plataformes de desenvolupament, que fa que la resta de parts de la estimació siguin gairebé negligibles.

Per fer l'aproximació de costos he assumit un sou brut de 35€ l'hora pel rol de programador i 55€ l'hora pel d'analista/dissenyador.

Cal tenir present que (sobretot durant la segona part del projecte) no es va realitzar una jornada complerta de 8h al dia. Tot i això els costos de personal ascendeixen bastant.

Rol	Jornades (8h)	Cost
Analista/Dissenyador	82,5	36300€
Programador	76,667	21466,667€
	Total:	57766,667€

Figura 6.5: Costos de personal aproximats.

6.2.2 Hardware

Els costos de material (hardware) són molt menors que els de personal. Malgrat això, els he volgut especificar, ja que no els considero negligibles, i em sembla interessant comprar-los amb els de personal.

En primer lloc cal dir que al començar aquest projecte no disposava d'un dispositiu Android i confiava en fer un ús intensiu de l'emulador proporcionat amb l'SDK pel desenvolupament i prova d'aplicacions, i realitzar només proves eventuals en dispositius amb l'ajuda esporàdica de col·legues que me'ls poguessin deixar.

Això, però, va resultar ser molt més molest del que esperava ja que l'emulador té força limitacions, difícils o empipadores de suplir (emulació de GPS, lentitud, limitacions en la simulació d'interacció amb "touchscreen", etc.).

Durant el període inicial, un company em va deixar un HTC Magic que no utilitzava per ajudar-me en el desenvolupament. Més endavant, a mitjans de desenvolupament del projecte, es va adquirir un dispositiu Android de gamma baixa (amb GPS).

Gràcies als baixos requeriments de les biblioteques i aplicacions que es desenvolupaven (excloent el mapfile-writer4and) aquests dispositius va ser suficients per la major part del desenvolupament del projecte, i són els únics que considero en aquest estudi de costos.

En realitat, les proves de mapfile-writer4and s'han realitzat amb un altre dispositiu "deixat" de cost molt més considerable, però no està inclòs al còmput, ja que les proves d'aquesta part del desenvolupament es podrien haver realitzat amb l'emulador sense que els seus inconvenients fossin prohibitius.

La resta de hardware requerit com a cost del projecte ha estat computat segons el cost d'un sistema informàtic de gamma baixa, ja que no es necessitava res específic ni cap equip de molt altes prestacions. Si bé es cert que en el nostre cas concret s'hauria pogut desenvolupar el projecte en ordinadors de la facultat sense cap cost addicional, en un projecte "real" probablement no s'hagués tingut aquesta sort, i la inversió en equipament hauria estat imperativa.

En el cas dels serveis de control de versions i correu, s'ha assumit que s'haguessin pogut fer servir opcions gratuïtes (com a alternativa als serveis que proporciona la universitat). Així aquests costos no s'han considerat.

Component	Model	Cost
Telèfon Android	mtk6516 F605 (Android 2.3)	66,9€
PC gamma baixa	HP ProLiant G7	205€
Monitor	Samsung LS19B150NS 18.5"	84,95€
Perifèrics bàsics (teclat+ratolí)	Gigabyte KM6150	13,25€
	Total:	370,1€

Figura 6.6: Costos de hardware aproximats.

6.2.3 Llicències

Per acabar el capítol d'estudi econòmic cal remarcar què ha implicat a nivell econòmic l'elecció de llicències de software lliure.

Tot el software emprat per a desenvolupar aquest projecte ha estat software lliure i gratuït. Des del sistema operatiu de l'ordinador de desenvolupament fins al que s'ha acabat generant, passant per els programes analitzats i l'entorn de desenvolupament (eclipse, Android SDK, emacs, LaTeX).

A continuació es pot veure un breu resum de les llicències de software que empren els principals programes utilitzats.

Com es pot veure, a part de ser llicències amb més o menys "llibertat", totes permeten la lliure distribució del software i no han requerit cap cost econòmic per a la seva utilització.

Però a part de calcular els costos d'aquestes llicències, per aquest estudi també s'han valorat les altres implicacions que podia tenir fer servir llicències

Programa	Llicència	Cost
debian 6.0.7	Diverses llicències lliures	-
eclipse indigo	(EPL[9]) Eclipse Public License	-
Android	(ASL[10]) Apache Software License (principalment)	-
emacs	(GPL[1]) General Public License	-
LaTeX	(LPPL[4]) Latex Project Public License	-
ganttproject	(GPL[1]) General Public License	-

Figura 6.7: Costos de llicències de software.

lliures, sobretot en quant a biblioteques desenvolupades/utilitzades, i com es combinen.

Així es van detectar alguns "conflictes" de llicències al valorar l'ús de biblioteques que fessin servir la llicència LGPL (Lesser GPL), tenint en compte que s'havia previst que el codi generat estigues sota la llicència GPL. En concret aquesta problemàtica es va manifestar al valorar l'ús de mapsforge.

La llicència LGPL es diferencia en molts aspectes de la GPL, i això ha fet que ens plantegéssim el fet de no fer servir biblioteques que la fessin servir.

La mateixa GNU adverteix d'algunes raons per les que usualment un bon defensor del software lliure no hauria de fer servir la LGPL. Almenys no a la lleugera. (per a la versió completa vegeu aquí[3]).

Resumint molt, la LGPL és més "permissiva" i permet "tancar" el codi generat i fer-lo servir en projectes de software privatiu, mentre que la GPL no ho permet, i garanteix que tot el codi publicat sota aquesta llicència sempre es mantindrà lliure.

Així, per defensar els interessos del software lliure, quan es creen biblioteques innovadores o característiques que poden ajudar a decantar al balança dels usuaris entre escollir aplicacions de software lliure o software privatiu, és important emprar la GPL per ajudar al desenvolupament del software lliure.

Però en el cas del desenvolupament de funcionalitats que ja existeixen en alternatives de software privatiu (com és el cas, per exemple de les biblioteques de mapes) no és tant rellevant forçar que mai es pugui "tancar" el codi.

Per tant, tot i que aquest tipus de llicència no hauria de ser la primera opció desitjable, en el cas que ens ocupa, i posat que existeixen moltes alternatives comercials a la biblioteca de mapes, sembla bastant lícit que els desenvolupadors de mapsforge hagin optat per aquesta llicència.

I això també ens valida a nosaltres per a utilitzar-la en aquest cas concret.

Així doncs aclarits aquests petits conflictes ideològics entre llicències es va concloure que es podia fer ús de les biblioteques o del software que estiguessin sota la LGPL o llicències similars i contribuir-hi sense impediments mentre les funcionalitats que s'aportessin no "marquessin la diferència" d'acord amb la filosofia anteriorment citada.

Això ha permès al present projecte beneficiar-se de molt software sense costos addicionals, i al mateix temps contribuir a la comunitat de software lliure aportant-hi les hores de desenvolupament, fent que l'esforç i els costos d'aquest projecte tinguessin molt més sentit.

Capítol 7

Conclusions i Treball Futur

7.1 Conclusions

Arribats a aquest punt cal valorar si els objectius del projecte s'han complert o no.

A més de la valoració dels objectius també escriuré algunes consideracions finals sobre algunes de les eines/parts d'aquest projecte que m'agradaria detallar.

I per acabar una valoració del que ha representat aquest projecte final de carrera a nivell personal.

7.1.1 Valoració del compliment dels objectius

Suport GML/KML/OSM: Gràcies a la utilització de mapsforge s'ha garantit el suport de mapes d'OpenStreetMap.

Com s'ha vist també s'ha aconseguit modificar (estendre) aquesta biblioteca per acceptar els formats KML i GML.

Suport fora de línia: La funcionalitat completa de generació/carrega de mapes fora de línia és viable en determinats entorns (dispositius amb suficients prestacions), i per la resta de casos es garanteix la visualització de mapes sense connexió (si bé prèviament generats des d'un sistema extern).

De fet, en el pitjor dels casos (per als dispositius menys potents) s'obté un funcionament igual a la resta d'aplicacions de les mateixes característiques que s'han trobat. És a dir, les aplicacions que proporcionen suport de mapes fora de línia, però que no sigui l'emmagatzemament de "tessel·les", i per tant sense consumir la memòria que aquest mètode implica.

I en el millor dels casos es millora notablement aquest funcionament i es facilitat el procés, ja que no cal processar les dades en un sistema extern.

Dibuixos geomètrics sobre el mapa: L'aplicació erdapfel permet dibuixar elements geomètrics sobre un mapa, i després guardar-los/exportar-los a format KML.

Càlculs d'elements geomètrics: L'aplicació erdapfel també permet calcular distàncies i àrees de manera aproximada.

Encara que actualment els càlculs que es poden realitzar són limitats, s'ha de tenir en compte que en aquest aspecte erdapfel no deixa de ser una aplicació de demostració. A mesura que es vagin necessitin funcionalitats més concretes en aplicacions de treball de camp, no ha de ser difícil afegir-les.

Facilitar la creació d'interfícies gràfiques: Aquest objectiu s'ha assolit al crear l'aplicació de prova, ja que les classes de MetricsOverlay i PolygonOverlay poden ser usades a tal efecte.

Façana de SL4A: Finalment s'ha aconseguit desenvolupar una façana pròpia i integrar-la a SL4A, de manera que les funcionalitats de la biblioteca quedin exposades als llenguatges de scripting.

Aplicació de prova: erdapfel és aquesta aplicació de "prova". Crec que aquesta aplicació només deixa entreveure tot el que es pot arribar a fer amb les biblioteques que s'han creat, però, de fet aquest era el seu principal objectiu.

A més d'això, aquesta aplicació ha servit per expandir el projecte ZamiaDroid.

Publicar el codi sota la GPL: Tècnicament, si ens prenem el redactat inicial d'aquest objectiu al peu de la lletra, s'ha d'admetre que no s'ha pogut mantenir, ja que la utilització de mapsforge implicava l'ús de LGPL en lloc de GPL pel codi que n'havia de formar part.

Malgrat això, es pot interpretar que el significat "real" d'aquest objectiu era que el codi i els esforços esdevinguts d'aquest projecte ajudessin, encara que fos una mica, a la comunitat de software lliure a seguir evolucionant i millorant.

Només el temps dirà en veritat si aquest objectiu s'ha complert o no, si algun dels esforços d'aquest projecte serà útil més endavant a algú.

Però s'ha fet tot el possible perquè així sigui.

7.1.2 Valoració del treball en comunitat

Com ja s'ha explicat a l'apartat de conclusions sobre la implementació de mapsforge, la possibilitat real d'integració de les expansions de mapsforge que aquest PFC proporciona s'ha vist dificultada per l'actualització de l'API d'overlays de mapsforge.

Això ha estat una veritable llàstima, ja que era una part que resultava molt interessant del plantejament del projecte.

Malgrat tot, encara existeix la possibilitat que en un futur es reprengui la activitat d'aquesta branca, i, aprofitant tot el treball realitzat, s'acabi integrant aquest suport de KML i GML a mapsforge.

A nivell més personal, atribueixo aquests petits inconvenients a la gran inexperiència que tinc en l'àmbit del treball en comunitat. Així, també espero que això em serveixi per saber com enfocar un futur treball d'aquestes característiques.

7.1.3 Valoració de la integració amb ZamiaDroid

Tot i la vinculació que aquest projecte tenia amb Zamia, la integració d'aquest projecte al Zamia no era un objectiu, i, de fet, es volia deixar com a futur treball. Però el fet que s'hagi trobat una manera d'aconseguir-la ha estat un "bonus" per aquest projecte que ha resultat molt satisfactori.

Encara que actualment les capacitats estiguin una mica limitades per l'aplicació erdapfel, espero que en un futur no molt llunyà es puguin aprofitar totes les capacitats de la biblioteca des del Zamia, i això contribueixi a la seva usabilitat.

7.1.4 Valoració del treball amb SL4A

En aquest apartat considero que he descobert un món fascinant i amb moltes possibilitats que no s'ha explotat tant com es voldria en el present projecte.

Crec que SL4A és una idea molt interessant i una bona mostra del que es pot arribar a fer amb la motivació i els coneixements adequats.

Així que estic content d'haver explorat aquests mecanismes i acabar-ne trobant un de tant satisfactori. Crec que el faré servir per a desenvolupar algun projecte que tinc en ment.

7.1.5 Consideracions finals

Aquest projecte m'ha resultat força complicat. Realment he sentit que es tractava de la culminació d'un procés que ha durat molts anys, i, en certa manera, ja tenia ganes de que s'acabés.

Una de les coses que m'ha resultat més complicada ha estat la gestió de tants projectes diferents. Cal tenir present que pràcticament cada projecte tenia les seves eines o mecanismes de treball particulars, diferents de la resta. Així, he fet servir, a banda d'eclipse, tant make, ant, mvn per a la compilació d'algunes dels projectes, i gairebé totes les eines de control de versions per a obtenir el codi dels projectes a estudiar: hg (mercurial), svn, cvs i git.

Per posar només un exemple, el procediment normal per compilar erdapfel consistia en compilar i empaquetar les classes de JiBX generades manualment en un jar, incloure aquest jar al projecte d'eclipse de la biblioteca estesa de mapsforge (mapsforge-gml-kml), compilar aquesta biblioteca de manera externa amb l'eina mvn i publicar-ne un "snapshot" al repositòri local de maven, tornar a l'eclipse, en aquest cas al projecte d'erdapfel i assegurar-me que s'havia actualitzat la referència de la biblioteca de mapes. Abans de compilar erdapfel calia també preparar a mà un paquet amb els mòduls "capats" i correctament configurats d'osmosis per tal que la part del mapfile-writer4and es

pogués generar, i finalment podia fer servir eclipse per a compilar l'aplicació d'Android.

Gestionar i/o automatizar tots aquests processos m'ha resultat molt difícil degut a la heterogeneïtat de les eines de desenvolupament que fa servir cada projecte diferent.

També he trobat a faltar alguns cops disposar de documentació conceptual a nivells prou "elevats" com per no haver-se d'introduir molt en els ets i uts de cada projecte. Una documentació semblant a la que he vist durant la carrera en assignatures d'enginyeria de software.

Però en general en aquest aspecte estic satisfet, ja que he estat capaç d'entendre la majoria de projectes que he hagut de modificar a nivell de codi font, i aconseguir resultats.

Com que es tractava d'un projecte pedagògic a més de pràctic, s'ha donat també molta importància a la part d'investigació i això també m'ha aportat molt.

Potser en algun moment s'hagi abusat d'aquesta visió i s'hagin deixat una mica al marge els usos pràctics del projecte a favor d'una visió més "expedicionària".

Personalment, però, estic satisfet d'haver fet aquestes proves per tenir una mica més clar què es pot fer i què no es pot fer amb aquests aparells que ara tots tenim a la butxaca.

Certament no s'han investigat a fons totes les vies, i potser algun dels camins que no s'han agafat hauria portat a conclusions diferents.

Però en acabar aquest projecte tinc la sensació que he après moltes coses del sistema amb el que he treballat, i això és bo.

Si bé és cert que moltes vegades he tingut la sensació que el sistema operatiu Android restringeix en certa manera les capacitats dels dispositius amb que he treballat, també és veritat que Android és un sistema molt "obert" que permet molt de joc i realitza bé les seves funcions, i que el fet que un sistema tan pròxim a la filosofia del codi obert sigui tan popular aporta més beneficis que no inconvenients.

Potser en aquest cas queixar-se de les restriccions d'Android podria ser semblant a la cèlebre metàfora d'Immanuel Kant, de l'ocell que al notar la resistència de l'aire al volar creu que volaria més bé en el buit, quan és la fricció d'aquest aire que sustenta en el seu vol.

> "Die leichte Taube, indem sie im freien Fluge die Luft teilt, deren Widerstand sie fühlt, könnte die Vorstellung fassen, dass es ihr im luftleeren Raum noch viel besser gelingen werde.- Kritik der reinen Vernunft (1787), Einleitung, III.

7.2 Millores i possibles ampliacions

7.2.1 Introducció

Aquest era un projecte ambiciós en el seu plantejament, que dona lloc a un gran ventall de possibilitats d'expansió futura. En aquest apartat es comenten algunes d'aquestes possibles millores futures, les que resulten més importants i també les que són més atractives i llamineres.

Per assolir resultats en un temps raonable no s'ha realitzat tot el que vam idear, així, en l'acord que representen els objectius del projecte, es van deixar de banda algunes funcionalitats.

7.2.2 Expansió de les funcionalitats de la façana

Degut a restriccions de temps la façana permet una interacció limitada amb els usuaris, i això seria un bon punt per on millorar el projecte.

Per assolir això hi ha un gran ventall de possibilitats, però jo proposaria integrar la biblioteca estesa de mapsforge a la façana, i redefinir funcions que puguin resultar útils als usuaris de SL4A.

En aquest cas, un petit estudi al respecte seria necessari.

7.2.3 Integració "oficial" a el projecte mapsforge

Tal i com he explicat en l'apartat de treball en comunitat els mòduls de tractament de kml i gml han estat desenvolupats sobre les biblioteques mapsforge, però aquesta modificació encara ha de ser aprovada per la comunitat i integrada al projecte mapsforge, per això es possible que encara s'hagi de

desenvolupar i adaptar a alguns requisits que puguin anar sorgint segons la comunitat revisi el codi.

Actualment és una modificació funcional de mapsforge i ja es accessible a la comunitat de software lliure per a qui la pugui voler emprar, però l'objectiu sempre havia estat que no es tractés d'un "fork" del projecte sinó d'integrar les modificacions a ell.

Però per això és necessari un procés relativament lent de revisió del codi que he fet per adaptar-lo a la versió actual de l'API d'Overlays (ja que els membres de la comunitat ho van plantejar així).

Seria desitjable que aquesta tasca, o almenys les parts més importants, la realitzes jo mateix. Per tant, tinc pensat de continuar-hi treballant en la mesura que sigui em possible.

7.2.4 Millora de les interfícies i les funcionalitats aprofitant Python

Al proporcionar una façana per a SL4A algunes de les funcionalitats de les biblioteques de mapes han estat reduïdes, ja que s'ha hagut de crear una abstracció a molt alt nivell de les funcionalitats.

Això no vol dir, però que no es pugui aprofitar el potencial de Python per incrementar les funcionalitats de la biblioteca i també per millorar-ne la interfície.

Una interessant proposta podria ser modificar i tractar els mapes offline amb eines de Python a nivell de fitxer kml/gml o osm i mostrar-los posteriorment.

7.2.5 Expansió de les capacitats del modul de KML GML per visualitzacions avançades

Actualment, i degut a limitacions, tant de temps per part meva com degut al propi disseny de les biblioteques de mapsforge per renderitzar mapes, no s'estan aprofitant al 100% les capacitats dels formats kml i gml.

Per posar un exemple, kml permet representacions en 3D de estructures (el programa Google earth les permet visualitzar)

A continuació llisto les capacitats més interessants que se'm acudeixen per poder suportar en un futur:

- Mostrar estructures tridimensionals.
- Permetre personalitzacions d'estil configurades en els fitxers (actualment les personalitzacions s'apliquen a nivell de programa i les que es puguin definir al fitxer són ignorades).

7.2.6 Contribució a OpenStreetMap

Una funcionalitat molt interessant que es va observar del projecte OsmAnd, i que no estava plantejada als objectius del projecte, era la d'aportar noves dades a la comunitat d'OpenStreetMap fent servir l'aplicació generada, o proporcionar mètodes encarats a tal objectiu a la mateixa biblioteca de mapes.

Actualment mapsforge no proporciona aquesta funcionalitat, i tot i que es tracta d'una característica complexa i que cal estudiar amb deteniment, no veig cap motiu pel qual no es podria implementar.

Els beneficis de tal funcionalitat són evidents, ja que permetria la actualització/correcció o simplement generació de noves dades sobre els mapes al mateix lloc on es troben, i no tan sols de punts d'interès sinó també de polígons, etc. I tot això independentment de si es disposa de connexió de xarxa al moment.

Certament aquesta seria una molt bona millora a proporcionar, tant si es planteja com una millora a nivell d'aplicació, com a nivell de biblioteca (ambdós podrien tenir sentit si es plantegen adequadament).

7.2.7 Optimització de recursos

Mapsforge es una biblioteca pensada per a ser executada en la plataforma Android, cosa que comporta certes limitacions de memòria disponible i capacitat de càlcul. Això fa que la eficiència sigui bastant rellevant per a la comunitat de mapsforge a l'hora de fer les coses.

Tot i així la gestió dels Overlays que fa no és perfecte i te certes limitacions a la escalabilitat.

Prova d'això és la modificació de la API d'Overlays que els mateixos desenvolupadors van fer per intentar solucionar-ho.

Si a això hi afegim la carrega addicional a parsejar i gestionar els fitxers KML (que tot i ser molt menor no és negligible quan parlem en plataformes limitades com Android), i el fet que ara per ara no hi hagi cap control o limitació sobre els possibles fitxers que els usuaris carregaran, provoca que es pugui donar la situació que al carregar determinats fitxers l'aplicació que faci servir les biblioteques sigui pràcticament inusable.

Tota aquesta casuística pot ser controlada i gestionada per les aplicacions que utilitzin la biblioteca, però malgrat això, hi ha certs aspectes que seria bo de millorar a la banda de la biblioteca per facilitar als usuaris aquesta gestió dels recursos.

Apèndix A

Annexos

A.1 Compilació creuada de python

Aquests són els passos detallats per a compilar l'entorn Python per a un dispositiu Andoird[27].

En primer lloc cal obtenir una versió "hoste" de l'entorn d'execució de Python. Això no hauria de tenir més complicació que la l'obtenir el codi font de Python i executar un simple ./configure; make; make install.

Després d'això cal aplicar el "pedaç" al codi font de Python (prèviament copiat en algun altre lloc per no modificar l'hoste). Aplicar el pedaç es fa naturalment amb l'eina "patch" (patch -p0 python-2.7.2.android.diff), i a continuació cal preparar l'Android NDK, i modificar algunes variables d'entorn:

```
export ANDROID\_NDK=[PATH WHERE THE ANDROID NDK IS]
export PATH=``\$ANDROID\_NDK/toolchains/arm-linux-
    androideabi-4.4.3/prebuilt/linux-x86/bin/:\$ANDROID\
    _NDK:\$ANDROID\_NDK/tools:/usr/local/bin:/usr/bin:/
    bin''
export ARCH=``armeabi''
export CFLAGS=``-DANDROID -mandroid -fomit-frame-
    pointer --sysroot \$ANDROID\_NDK/platforms/android
    -5/arch-arm''
export CXXFLAGS = ``\$CFLAGS''
export CC=``arm-linux-androideabi-gcc \$CFLAGS''
export CXX=``arm-linux-androideabi-g++ \$CXXFLAGS''
export AR=``arm-linux-androideabi-ar''
```

```
export RANLIB=``arm-linux-androideabi-ranlib''
export STRIP=``arm-linux-androideabi-strip --strip-
    unneeded''
export MAKE=``make -j4 HOSTPYTHON=[PATH TO HOST PYTHON]
     HOSTPGEN=[PATH TO HOST PGEN] CROSS\_COMPILE=arm-
    eabi- CROSS\_COMPILE\_TARGET=yes''
```

Naturalment posant a on cal les direccions cap a l'executable de python i el de pgen del host.

Després cal configurar la compilació creuada de la següent manera per fer servir el compilador per arm del NDK:

```
./configure LDFLAGS=``-Wl,--allow-shlib-undefined''
    CFLAGS=``-mandroid -fomit-frame-pointer --sysroot   \
    $ANDROID\_NDK/platforms/android-5/arch-arm''
    HOSTPYTHON=[HOST PYTHON PATH] HOSTPGEN=[HOST PGEN
    PATH] --host=arm-eabi --build=i686-pc-linux-gnu --
    enable-shared --prefix=``[WHERE YOU WANT TO PUT THE
    GENERATED PYTHON STUFF]''
```

Però abans de fer servir el makefile generat cal fer-hi una petita modificació:

```
sed -i "s|^INSTSONAME=\(.*.so\).*|INSTSONAME=\\1|g"
    Makefile
```

I ara si que s'hauria de poder compilar sense problema.

A.2 Mapsforge Binary MapFile

Specification: Mapsforge Binary Map File Format

- Specification: Mapsforge Binary Map File Format
  - Conceptual design
  - General remarks
  - File structure
  - File header
    - Tile index header
    - Tile index entry
    - Tile header
    - POI data
    - Way properties
    - Way data
  - Version history

Conceptual design

The *mapsforge binary map file format* is designed for map rendering on devices with limited resources like mobile phones. It allows for efficient storage of geographical information (e.g. OpenStreetMap data), fast tile-based access, and filtering of map objects by zoom level.

The map file consists of several sub-files, each storing the map objects for a different zoom interval. Zoom intervals are non-overlapping groups of consecutive zoom levels. Each zoom interval is represented by a single member of the group, the so-called *base zoom level* .

General remarks

- All latitude and longitude coordinates are stored in microdegrees (degrees $\times$ 10 6).

- Numeric fields with a fixed byte size are stored with *Big Endian* byte order.
- Unsigned numeric fields with a variable byte encoding are marked with `VBE-U` *INT* and stored as follows:
  - the first bit of each byte is used for continuation info, the other seven bits for data.
  - the value of the first bit is 1 if the following byte belongs to the field, 0 otherwise.
  - each byte holds seven bits of the numeric value, starting with the least significant ones.
- Signed numeric fields with a variable byte encoding are marked with `VBE-S` *INT* and stored as follows:
  - the first bit of each byte is used for continuation info, the other six (first byte) or seven (all other bytes) bits for data.
  - the value of the first bit is 1 if the following byte belongs to the field, 0 otherwise.
  - each byte holds six (first byte) or seven (all other bytes) bits of the numeric value, starting with the least significant ones.
  - the second bit in the first byte indicates the sign of the number. A value of 0 means positive, 1 negative.
- All strings are stored in UTF-8 as follows:
  - the length L of the UTF-8 encoded string in bytes as `VBE-U` *INT* .
  - L bytes for the UTF-8 encoding of the string.

File structure

For each zoom interval a so called ***sub-file*** is created. A sub-file consists of a ***tile index segment*** that stores a fixed-size pointer for each tile created in the ***tile data segment*** . The order of storing tiles to the tile data segment and their corresponding pointers to the tile index segment is row-wise and within a row column-wise. Rows and columns are inherently given by the grid layout of the tiles that is defined by the rectangular bounding

box. For each tile in the grid, meta information is available in the ***tile header*** accompanied by its payload data (POIs and ways).

To read the data of a specific tile in the sub-file, the position of the fixed-size pointer in the index can be computed from the tile coordinates. The index entry points to the offset in the sub-file where the data is stored. Tile coordinates are implicitly given due to the structure of the tile index, thus no tile coordinates need to be stored along each tile.

```
# meta data
file header

# sub-files
for each sub-file
    # tile index segment
    tile index header
    tile index entries

    # tile data segment
    for each tile
        tile header
        for each POI
            POI data
        for each way
            way properties
     way data
```

File header

bytes	optional	name	description
20		magic byte	`mapsforge binary OSM`
4		header size	size of the file header in bytes (without magic byte) as 4-byte *INT*
4		file version	version number of the currently used binary file format as 4-byte *INT*
8		file size	The total size of the map file in bytes
8		date of creation	date in milliseconds since 01.01.1970 as 8-byte *LONG*
16		bounding box	geo coordinates of the bounding box in microdegrees as 4*4-byte *INT* , in the order minLat, minLon, maxLat, maxLon
2		tile size	the tile size in pixels (e.g. 256)
variable		projection	defines the projection used to create this file as a string
1		flags	• 1. bit: flag for existence of debug information • 2. bit: flag for existence of the *map start position* field • 3. bit: flag for existence of the *start zoom level* field • 4. bit: flag for existence of the *language preference* field • 5. bit: flag for existence of the *comment* field • 6. bit: flag for existence of the *created by* field • 7.-8. bit: reserved for future use
8	yes	map start position	geo coordinate in microdegrees as 2*4-byte *INT* , in the order lat, lon
1	yes	start zoom level	zoom level of the map at first load
variable	yes	language preference	The preferred language for names as defined in ISO 3166-1 as string
variable	yes	comment	comment as a string
variable	yes	created by	The name of the application which created the file as a string
variable		POI tags	• amount of tags as 2-byte *SHORT* • tag names as a strings • tag IDs are implicitly derived from the order of tag names, starting with 0
variable		way tags	• amount of tags as 2-byte *SHORT* • tag names as a strings • tag IDs are implicitly derived from the order of tag names, starting with 0

1		amount of zoom intervals	defines the amount of zoom intervals used in this file

variable		zoom interval configuration	• for each zoom interval:
			  ◦ base zoom level as *BYTE*
			  ◦ minimal zoom level as *BYTE*
			  ◦ maximal zoom level as *BYTE*
			  ◦ absolute start position of the sub file as 8-byte *LONG*
			  ◦ size of the sub-file as 8-byte *LONG*

Tile index header

bytes	optional	name	description
16	yes	index signature	If the debug bit in the file header is set: `+++IndexStart+++`

Tile index entry

bytes	optional	name	description
5		index entry	• 1. bit: flag to indicate whether the tile is completely covered by water (e.g. a tile amidst the ocean) • 2.-40. bit: offset of the tile in the sub file as 5-byte *LONG* (optional debug information and index size is also counted) If the tile is empty offset(tile $_i$) = offset(tile $_{i+1}$)

Tile header

bytes	optional	name	description
32	yes	tile signature	If the debug bit in the file header is set: `###TileStartX,Y###` where X and Y indicate the tile coordinates of the current tile; the text is always padded to 32 bytes by adding whitespaces
variable		zoom table	A table indicating the number of POIs and ways in this tile for the different zoom levels covered by the enclosing sub-file. Let Z be the number of zoom levels supported by the enclosing sub-file (e.g. 6 for a sub-file that covers levels 12-17). Then the table has Z rows and 2 columns (first column: POIs, second column: ways). Each cell in the table represents the number of POIs or ways on the specific zoom level. The table is written row-wise and values are encoded as `VBE-U` *INT*.
variable		first way offset	offset in bytes to the first way in this tile as `VBE-U` *INT*. The counting starts at the following byte (i.e. first way offset itself is not counted).

POI data

bytes	optional	name	description
32	yes	POI signature	If the debug bit in the file header is set: `***POIStartX***` where X defines the OSM-ID of the POI; the text is always padded to 32 bytes by adding whitespaces
variable		position	geo coordinate difference to the top-left corner of the current tile as *VBE-S INT*, in the order lat-diff, lon-diff
1		special byte	• 1.-4. bit: layer (OSM-Tag: layer=...) + 5 (to avoid negative values) • 5.-8. bit: amount of tags for the POI
variable		tag id	for each tag of the POI: • tag id as *VBE-U INT*
1		flags	• 1. bit: flag for existence of a POI name • 2. bit: flag for existence of a house number • 3. bit: flag for existence of an elevation • 4.-8. bit: reserved for future use
variable	yes	name	name of the POI as a string
variable	yes	house number	house number of the POI as a string
variable	yes	elevation	elevation of the POI in meters as *VBE-S INT*

Way properties

bytes	optional	name	description
32	yes	way signature	If the debug bit in the file header is set: `---WayStartX---` where X defines the OSM-ID of the way; the text is always padded to 32 bytes by adding whitespaces
variable		way data size	number of bytes that are needed to encode the current way as *vbe-u INT*, starting from the sub tile bitmap (i.e. way signature and way size are not counted)
2		sub tile bitmap	A tile on zoom level z is made up of exactly 16 sub tiles on zoom level z+2 for each sub tile (row-wise, left to right): • 1 bit that represents a flag whether the way is relevant for the sub tile Special case: coastline ways must always have all 16 bits set.
1		special byte	• 1.-4. bit: layer (OSM-Tag layer=...) + 5 (to avoid negative values) • 5.-8. bit: amount of tags for the way
variable		tag id	for each tag of the way: • tag id as *vbe-u INT*

1		flags	• 1. bit: flag for existence of a way name
			• 2. bit: flag for existence of a house number
			• 3. bit: flag for existence of a reference
			• 4. bit: flag for existence of a label position
			• 5. bit: flag for existence of *number of way data blocks* field
			  ◦ case 0: field does not exist, number of blocks is one
			  ◦ case 1: field exists, more than one block
			• 6. bit: flag indicating encoding of way coordinate blocks
			  ◦ case 0: single delta encoding
			  ◦ case 1: double delta encoding
			• 7.-8. bit: reserved for future use

variable	yes	name	name of the way as a string
variable	yes	house number	house number of the way as a string
variable	yes	reference	reference of the way as a string
variable	yes	label	geo coordinate difference to the first way node in microdegrees as 2 Ã—

variable	yes	position	`VBE-S` *INT* , in the order lat-diff, lon-diff
variable	yes	number of way data blocks	The amount of following way data blocks as `VBE-U` *INT* .

Way data

bytes	optional	name	description
variable		number of way coordinate blocks	The amount of following way coordinate blocks as *VBE-U INT*. An amount larger than 1 indicates a multipolygon with the first block representing the outer way coordinates and the following blocks the inner way coordinates.
variable		way coordinate block	for each way coordinate block: • amount of way nodes of this way as *VBE-U INT* • geo coordinate difference to the top-left corner of the current tile as *VBE-S INT*, in the order lat-diff, lon-diff • geo coordinates of the remaining way nodes stored as differences to the previous way node in microdegrees as 2 Ã— *VBE-S INT* in the order lat-diff, lon-diff. Let x1 be the lat of the previous way node and x2 be the lat of the current way node. Then the difference is defined as x2 - x1.

Font: `http://code.google.com/p/mapsforge/wiki/SpecificationBinaryMapFile`

A.3 OpenStreetMap tile usage policy

We are in principle happy for our map tiles to be used by external users for creative and unexpected uses - in contrast to most web mapping providers, which insist that you use only their supplied API.

If you see this image instead of the requested tile your application may be making unreasonable demands on OSM infrastructure. However, OpenStreetMap's own servers are run entirely on donated resources. They have strictly limited capacity. Heavy use of OSM tiles adversely affects people's ability to edit the map, and is an abuse of the individual donations and sponsorship which provide hardware and bandwidth. As a result, we require that users of the tiles abide by this tile usage policy.

OpenStreetMap data is free for everyone to use. Our tile servers are not.

Below are the minimum requirements that users of tile.openstreetmap.org must adhere to. These may change in future, depending on available resources. Should any users or patterns of usage nevertheless cause problems to the service, access may still be blocked without prior notice. We will try to contact relevant parties if possible, but cannot guarantee this.

But because OpenStreetMap data is free, many other organisations provide map tiles made from OSM data. If your project doesn't meet these requirements, you can get OSM-derived map tiles elsewhere.

Requirements

- Heavy use (e.g. distributing an app that uses tiles from openstreetmap.org) is forbidden without prior permission from the System Administrators. See below for alternatives.
- Clearly display license attribution.
- Do not actively or passively encourage copyright infringement.
- Calls to /cgi-bin/export may only be triggered by direct end-user action. (For example: "click here to export".) The export call is an expensive (CPU+RAM) function to run and will frequently reject when server is under high load.

Technical Usage Requirements

- Valid User-Agent identifying application. Faking another app's User-Agent WILL get you blocked.
- If known, a valid HTTP Referer.

- DO NOT send no-cache headers. ("Cache-Control: no-cache", "Pragma: no-cache" etc)
- Cache Tile downloads locally according to HTTP Expiry Header, alternatively a minimum of 7 days.
- Maximum of 2 download threads. (Unmodified web browsers' download thread limits are acceptable.)

Note: modern web browsers in standard configuration generally pass all the above technical requirements.

Bulk Downloading

Bulk downloading is strongly discouraged. Do not download tiles unnecessarily. In particular, downloading significant areas of tiles at zoom levels 17 and higher for offline or later usage is forbidden without prior consultation with a System Administrator. These tiles are generally not available (cached) on the server in advance, and have to be rendered specifically for those requests, putting an unjustified burden on the available resources. To avoid having your access blocked, please discuss your requirement with system administrators either via their wiki pages or on the IRC channel prior to starting.

Changes to this policy

This policy may change at any time subject to the needs and constraints of the project. Commercial services, or those that seek donations, should be especially aware that access may be withdrawn at any point: you may no longer be able to serve your paying customers if access is withdrawn.

Alternative OpenStreetMap Tile Providers

Free tile servers based on OSM data:

- MapQuest Open
- TMS - listing more

Paid-for tile servers:

- See Commercial OSM Software and Services

Setting up your own tile server: the switch2osm website, with up to date information on running your own OSM based services

- Tiledrawer
- MapBox

This policy relates to the the default "Mapnik" tiles rendered and served by a,b,c.tile.openstreetmap.org as part of the OpenStreetMap project. It does not relate to other tiles that are viewable on the OpenStreetMap.org homepage, which may have their own usage policies. You should contact the individual projects, such as OpenCycleMap, if you wish to use their tiles.

A.4 Extracte dels "issues" de mapsforge

Aquest és un extracte dels "issues" de mapsforge, on es plantejava l'implementació del suport a KML, i es descartava per manca de recursos:

Project Member Reported by thilo.mu...@gmail.com, Jun 5, 2011

> At the moment all overlay classes accept only "raw" coordinates, represented by the GeoPoint class. But in many cases the actual input data comes from a KML file which is stored somewhere on the device. Although parsing KML files is not a big deal in theory, it means that developers need to write the same code over and over again.
>
> We should therefore extend the overlay interface so that developers can pass KML files directly to WayOverlays for example. Having one stable and correctly designed implementation will also reduce the risk of bugs in the client application.

Sep 2, 2012 Project Member #1 thilo.mu...@gmail.com

> Due to our very limited resources we cannot implement this feature in the mapsforge-map library as there are also many more important issues that need to be fixed first.

Font: `http://code.google.com/p/mapsforge/issues/detail?id=121`

A.5 Extractes de la llista de correus de mapsforge-dev

A.5.1 Converses inicials

Google Grups rendering GML and KML data?

jo <achtungwolf@gmail.com>
11/04/2012 10:24
Publicat al grup: mapsforge-dev

> Hi all!
> This is kind of a weird request...
> I'm currently working on a university project (I think in english the correct term is Master Thesis), and I was wondering, what's the best way to display a map in GML / KML on an Android Device using mapsforge.
> I found that
> issue http://code.google.com/p/mapsforge/issues/detail?id=121
> but I assume it's not implemented yet (or I could'n find it)
> I'v been looking around,and I don't know exactly what should I do or how to proceed... should I Parse The files and try to create overlays with the suitable overlay classes provided?
> Is there any limitation to this method which I should be aware of? (some information in the gml that i can't represent out of the box?)
> Should I try to extend some other class, i don't know, like creating some sort of mapgenerator or something?
> Is there any of these alternatives, or some other that you may think interesting for the sake of the mapsforge project? (I mean.. . I have to do the work anyway, so if what I do is useful to you all, it will be nice...).
> As you May see, I'm a little bit confused (as always), and maybe what I say has been already done, or it's trivial using some feature which i didn't know of.... so a little of experienced input from you wolud be much appreciated.
> Also any comment or thought about the issue will be helpful.
> Thanks all.

Re: [mapsforge-dev] rendering GML and KML data?

Thilo
11/04/2012 13:17
Publicat al grup: mapsforge-dev
 The status of the issue is up-to-date. There is no GML or KML parser in
  the mapsforge library yet. If you need to display such files on an
  MapView overlay, you will have to handle that process on your own.
  You should also not expect a resolution of this issue in the near
  future, as we have many other more important issues to fix first. This
  is really just a "nice to have" idea and has a very low priority for us.
  Greetings,
  Thilo

Re: [mapsforge-dev] rendering GML and KML data?

jo <achtungwolf@gmail.com>
11/04/2012 14:26
Publicat al grup: mapsforge-dev
   On 11 Abr, 13:17, Thilo Mühlberg <thilo.muehlb...@gmail.com> wrote:
   > The status of the issue is up-to-date. There is no GML or KML parser in
   > the mapsforge library yet. If you need to display such files on an
   > MapView overlay, you will have to handle that process on your own.
   >
   > You should also not expect a resolution of this issue in the near
   > future, as we have many other more important issues to fix first. This
   > is really just a "nice to have" idea and has a very low priority for us.
   >
   Oh, yes, I understand that.
   The thing is... since I have to do that work anyway, for requirements
   of my project...
   I was wondering, if you'r interested in a collaboration while I do.
   You know.. that the parser I made can be... integrated in the
   project...
   And since I'm not used to work in an open source project I was asking
   which approach seems better to make it the right way. So it can be
   useful in the future and it respects the philosophy of the project...
   I don't know if I've explain it well enough... sorry, my english is
   not as good as it should be.
   Greetings,
   Adrià

Re: [mapsforge-dev] rendering GML and KML data?

Thilo
15/04/2012 13:43
Publicat al grup: mapsforge-dev

> As you have just started your project and as this is - as i mentioned before - only a low-priority issue for us, i suggest that you begin implementing your GML/KML parser after carefully looking at the existing Java libraries and XML tools. Write well-structured code with unit tests and Javadoc comments that follows the code conventions [1] and code quality guidelines [2] from the mapsforge project.
> Use your own public SVN or GIT repository while developing so that we can easily track the current state. Maybe you should just create a new Google project with a built-in issue tracker for bug reporting. When you have finished your work, we can then decide whether we like the result and - if so - merge your code into the mapsforge project.
> In case you have any questions, don't hesitate to use this mailing list.
> Greetings,
> Thilo
> [1] https://code.google.com/p/mapsforge/wiki/ProjectConventions
> [2]
> https://code.google.com/p/mapsforge/wiki/GettingStartedDevelopers

Font: https://groups.google.com/forum/?fromgroups=#!topic/mapsforge-dev/BL2AqAaC5nY/discussion

A.5.2 Converses per a la optimització de map-file writer

Google Grups mapsforge-map-writer memory usage.

jo (Adrià Ribatallada)
05/12/12

Hi.

I'm trying to run osmosis in an Android device to convert downloaded osm data to the mapsforge .map format, and store that to the device for latter use.

And after a lot of trouble I almost got it working.

I say almost, because in the emulator it does work, but it fails in all the "real" devices I'v tested with an Out Of Memory Exception (so it's technically not working).

I'm only being able to perform the actions correctly in the android emulator because there I'v set a "Max VM application heap" to 132Mb.

But all my devices have less than that, about 64Mb , and eventually got the OOM exception.

And it seems that where all fail is in the last stage of my osmosis pipeline, which is the stage where I perform the --mapfile-write.

I'v tried to force the option type=hd, with some problems, but at the end, even using that option the problem persists. And all seems to indicate that the largest memory consumption is when executing the map-writer plugin, so I don't think that's something that the osmosis-dev team could help me solve.

I don't know exactly if all that memory is strictly required, but it shocks me a little, because I'm working with very small osm data files, so I'm wondering...

Is there a chance to do some improvement in that aspect in the map-writer plugin, or that a memory leak exists there and we can easily reduce that amount of memory consumed?

Here is a verbose output log of a crashed execution in the android device (I painted the more relevant osmosis output sections), in case you need more information don't hesitate to request it.

Thanks in advance for any help.

[...]

Atentament,

Adrià Ribatallada i Torelló.

Re:

Jürgen Broß
05/12/12

Hi Adrià,

the map-file writer is intended to be executed on a rather large and powerful machine and is simply not suited to be run on a mobile device. It does some heavy computations and to speed this up, it generally trades memory requirements for gains in execution time.

The available heap size for a single app on Android is generally artificially restricted to a certain amount. Unless you have a "rooted" device, you cannot adjust this artificial restriction.

I'm curious about your use case. Why do you want to convert OSM data on a mobile device?

Best regards,
Jürgen

Jürgen Broß
Institute of Computer Science
Databases and Information Systems
Freie Universität Berlin
Takustr. 9
D-14195 Berlin, Germany
phone: +49 30 838-75108
email: juerge...@fu-berlin.de

Re:

Hi Jürgen,

Thanks for the quick response.

2012/12/5 Jürgen Broß <juergen.bross@fu-berlin.de>
Hi Adrià,

the map-file writer is intended to be executed on a rather large and powerful machine and is simply not suited to be run on a mobile device. It does some heavy computations and to speed this up, it generally trades memory requirements for gains in execution time.

I see.

The available heap size for a single app on Android is generally artificially restricted to a certain amount. Unless you have a "rooted" device, you cannot adjust this artificial restriction.

I'm curious about your use case. Why do you want to convert OSM data on a mobile device?

The issue is, we have an application to collect some geo-localized data, and we wanted to easily download some parts of the map when available networks were present, and then when we were in the "field" collecting our data, we can use that offline maps, and synchronize the data later.

It was an... "ease of use" improvement considering having to process that data on a PC and then transferring that maps to the phone. And since the way that mapsforge handles the offline maps support is using the .map format and the way to generate that kind of files is using osmosis, and I didn't wanted to setup a server just for generating that maps... I figured that the processing might be performed on the mobile device itself, I thought... why not?

Well, as you can see, the use case is rather simple, but maybe mine might not have been the best approach.

Apart from setting up a server that processes all the info, is there a way to achieve our goal?

Thanks.

Kind regards,
Adrià Ribatallada i Torelló.

Font: https://groups.google.com/forum/?fromgroups#!msg/mapsforge-dev/ioBORO3zBNo/y7NlsRiQUZwJ

A.6 Extractes de la llista de correus d'osmosis-dev

[osmosis-dev] Android?
http://lists.openstreetmap.org/pipermail/osmosis-dev/2012-November/001410.html
Adrià Ribatallada Torelló achtungwolf at gmail.com Tue Nov 6 16:41:55 GMT 2012

Hi there!

I was trying to use the osmosis tool in an android device, and after including all the required libraries to an android project and make it compile and install in the device I'v encountered some "runtime" obstacles that made me consider that the best approach might be to adapt the osmosis source code, adapting it to use the android libraries.

(the most severe "obstacle" is a "No validating SAXParser implementation available" while trying to read the file plugin.xml. Is the same error explained here with more detail by this other guy: http://stackoverflow.com/questions/10222230/trying-to-use-osmosis-in-android-environment-duplicate-file-error-with-jar-file/13251374#13251374).

So right now I'm trying to "port" osmosis or at least part of it to use it on an Android device.

But before I start what I consider a huge amount of work, I must ask:

Has somebody tried to do that or something similar in the past? (I couldn't found anyone, but maybe I'm not searching in the right places)

Is there (a priori) any reason why this endeavor should fail miserably?

Thanks in advance for any help you might provide.

Sincerely,

Adrià Ribatallada i Torelló.

Raphael Volz rv at nogago.com Tue Nov 6 16:50:23 GMT 2012

Hi Adria,
I don't know why you would want to have osmosis on an Android device, but nevertheless, if you want to achieve this, you need to implement those libraries used from Java SDK that are not present in the Android SDK. For many, you will find a open source implementations in openjdk http://hg.openjdk.java.net/.
You will have to rename the packages (e.g. appending a common prefix to avoid conflicts, and do so throughout the osmosis code base).

Good luck +

Raphael

Adrià Ribatallada Torelló achtungwolf at gmail.com
Tue Nov 6 17:19:06 GMT 2012

> Hi Adria,
> I don't know why you would want to have osmosis on an Android device,
>

I wanted to do so because I'm creating an android app that uses the mapsforge library, and I was trying to convert downloaded osm data to the "mapsforge format" to use it later in offline mode, and the way mapsforge converts from osm to its internal map format is by using an osmosis plugin, so I thought it may be possible to perform that task in the device itself.

but nevertheless, if you want to achieve this, you need to implement those
> libraries used from Java SDK that are not present in the Android SDK. For
> many, you will find a open source implementations in openjdk
> http://hg.openjdk.java.net/.
> You will have to rename the packages (e.g. appending a common prefix to
> avoid conflicts, and do so throughout the osmosis code base).
>

Ok, many thanks. I'll start to fiddle with it right away.

>
> Good luck +
>
>
Raphael
>
> Thanks again :D

Brett Henderson brett at bretth.com Wed Nov 7 10:29:35 GMT 2012

Hi Adrià,

On 7 November 2012 03:41, Adrià Ribatallada Torelló
<achtungwolf at gmail.com>wrote:

> Hi there!
>
> I was trying to use the osmosis tool in an android device, and
> after including all the required libraries to an android project and make
> it compile and install in the device I'v encountered some "runtime"
> obstacles that made me consider that the best approach might be to adapt
> the osmosis source code, adapting it to use the android libraries.
>
> (the most severe "obstacle" is a "No validating SAXParser implementation
> available" while trying to read the file plugin.xml. Is the same error
> explained here with more detail by this other guy:
> http://stackoverflow.com/questions/10222230/trying-to-use-osmosis-
in-android-environment-duplicate-file-error-with-jar-file/13251374#13251374
>).
>

I'm not sure if this helps, but the plugin.xml loading code is used by JPF (Java Plugin Framework) support. JPF provides one way of packaging Osmosis plugins, but is rarely used. The majority of plugins use a simpler mechanism involving a properties file called osmosis-plugin.conf bundled in each plugin jar file. If you don't need JPF you could remove the JPF code from Osmosis completely and eliminate the error you're receiving.

Brett

Adrià Ribatallada Torelló achtungwolf at gmail.com Thu Nov 8 20:37:36 GMT 2012

Thank you Brett, I'v followed your advice and tried to remove the JPF from a local clone of the osmosis source, and adapted the plugin I need to use the osmosis-plugins.conf.

And all compiles good and fine, and if I execute the command I need using the bin/osmosis script in my machine all goes fine.

But in the Android device I need to do it from the code (or so I think), calling Osmosis.run(args) or Osmosis.main() I presume, and there is where it fails now...

That's the code I'm trying to run:

```
        File pbfFile = new
File(Environment.getExternalStorageDirectory()+"/download/"+"in.osm.pbf");
File outFile = new
File(Environment.getExternalStorageDirectory()+"/download/"+"out.map");

Osmosis.main(new String[] {
"--read-pbf file="+pbfFile,
"--mapfile-writer file="+outFile
});
```

That's my pretty error:

[..]

I assume that is thrown because the envirenoment it's not well configured, but I don't know how to "do the same" that does the script from my program code :P

I'v tried to run the classworlds.Launcher, but I didn't suceed there...

Can some of you help me understand what needs to be done "programatically" to setup the environment?

Thanks again.

Brett Henderson brett at bretth.com Fri Nov 9 04:55:00 GMT 2012

Hi Adrià,

Simply calling main should be okay, but you'll need to configure the classpath beforehand. That's what the classworlds launcher does for you. During startup, Osmosis scans the classpath for all instances of osmosis-plugins.conf and uses them to dynamically register plugins. If it doesn't find the pbf plugin (ie. the pbf jar file isn't on the application classpath) then you'll see the missing task error.

Brett

Adrià Ribatallada Torelló achtungwolf at gmail.com Fri Nov 23 12:09:28 GMT 2012

Hi folks!

I forgot to thank you all for your help!

At the end I got it working in the phone.

The last problem i commented was my mistake (a shamefully idiotic one) when passing the parameters to the run method I wasn't separating the strings correctly and the commandline parser complained to that (as natural).

So in resume, if the JPF (as Brett suggested) is "deactivated" the osmosis tool is able to run on an android device (at least some of the tasks/plugins).

Provided that some minor modifications are done to the lib jar's to avoid repeated files in the final apk. That's basically removing the

repeated osmosis-plugin.conf file present in the plugins jars, and then, of course, force the loading of that plugins via the run args (using -plugin option).

That's probably not a very good solution, because it demands to modify the source of osmosis, and at the end the idea of running osmosis in the phone is not very useful to anyone, since processing the data in the phone is really slow and has plenty of limitations (due to android limitations I presume), and most of the time prforming the task on a server or on a local computer is the smartest way to do it.

But just to let you know, it works.

Thanks!

Sincerely,

Adrià Ribatallada i Torelló.

Brett Henderson brett at bretth.com Fri Nov 23 12:24:08 GMT 2012

Hi Adrià,

On 23 November 2012 23:09, Adrià Ribatallada Torelló
<achtungwolf at gmail.com>wrote:
<snip>

>
> But just to let you know, it works.
>

Thanks for giving us the update. No doubt the question will come up again so it's good to hear that it's possible.

Cheers,
Brett

A.7 Extractes de la llista de correus de jibx-dev

adria . ribatallada Thu, 14 Jun 2012 10:12:49 -0700

Hi All,

I was trying to generate bindings for kml2.2 using the standard schemas provided by ogc (`http://schemas.opengis.net/kml/2.2.0/`), but I'm stuck at the first step since the codeGen tool throws this "IllegalArgumentException" exception.

Using google's previous schema codeGen was working good, and all seemed to work well (`https://developers.google.com/kml/schema/kml21.xsd`) but it's vital for me to use this new version of the kml schema, and I don't know what to do.

As you can imagine I don't have control over the schema since it's given as is, and I'm completely at a loss.

Anyone have a clue why there is a problem? I'm doing something wrong? is there something I can do to solve this?

I'v searched and found some cases of this exception in the mailing list, but it were related to empty enumeration values, and I don't think this is the concrete case here. Also most of that old messages apparently were solved in newer versions of jibx, so.

What should I do?

Also, there are a LOT of Warning: No type defined for element 'something' at (line 1615, col 59, in ogckml22.xsd) but I don't think it's related to the IllegalArgumentException.

I write here because I don't know who else can help me solve this problem, any kind of advice will de helpful.

Thanks all in advance.

Greetings,

Adrià Ribatallada Torelló.

P.D. this is the exact output if it's of any help: [..]

Yosvany Wed, 03 Oct 2012 02:12:54 -0700

<adria.ribatallada <at> est.fib.upc.edu> writes:

> > Hi All, > > I was trying to generate bindings for kml2.2 using the standard > schemas provided by ogc (`http://schemas.opengis.net/kml/2.2.0/`), but > I'm stuck at the first step since the codeGen tool throws this > "IllegalArgumentException" exception. >

Hi all

I'm getting this error working while trying to generate the code using the travelport uAPI WSDLschema: `http://support.travelport.com/webhelp/uapi/Content/Resources/uAPI_WSDLschema_Release-3.1.0.88.zipandjibx_1.2.4` (will try a previous version)

I'm using the codegen's build.xml from the examples directory just pointing to the other schema.

codegen: [echo] Running code generation from schema [java] Loaded and validated 1 specified schema(s) and 2 referenced schema(s) [java] Exception in thread "main" java.lang.IllegalArgumentException [java] at org.eclipse.jdt.core.dom.SimpleName.setIdentifier(SimpleName.java:191) [java] at org.eclipse.jdt.core.dom.AST.newSimpleName(AST.java:1303) ..... blabla (the same trace from the original post)

Any help with this guys?, i'm really new with all this stuff Thank you

adria . ribatallada Fri, 08 Feb 2013 15:25:11 -0800

Hey, sorry to revive this old and long forgotten issue.

But I forgot to mention that recently, and using a rather rudimentary method, I had been able to isolate the schema part that was causing that IllegalArgumentException.

In the 2.1 version (which WASN'T causing any error) the problematic complexType looked like that:

```
<complexType name="ItemIconType"
    final="\#all">
    <complexContent>
      <extension base="kml:ObjectType">
        <sequence>
          <element name="state"
              type="kml:itemIconStateType"
              minOccurs="0"
              maxOccurs="unbounded"/>
          <element name="href" type="anyURI"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

But in the 2.2 version of the schema this was changed so it looked like this:

```
<complexType name="ItemIconType"
    final="\#all">
  <complexContent>
    <extension base="kml:AbstractObjectType">
      <sequence>
        <element ref="kml:state"
            minOccurs="0"/>
        <element ref="kml:href" minOccurs="0"/>
        <element
            ref="kml:ItemIconSimpleExtensionGroup"
            minOccurs="0"
          maxOccurs="unbounded"/>
        <element
            ref="kml:ItemIconObjectExtensionGroup"
            minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

And then previously, the conflicting element was defined:

```
<element name="state"
    type="kml:itemIconStateType"/>
```

With a bunch of other elements.

Well then, this state element was causing the trouble. I don't know why (I guess it has to do with the minOccurs/maxOccurs attribute?), or if there was a better way to avoid that exception (like a configuration parameter, or something) but after manually editing the schema and reverting this concrete object, the code (and the bindings) are created.

Maybe someone reading this can correct the "bug" in the codeGen, or treat that exception nicely so the user can understand what's going on.

I will welcome any further explanation on the matter, too.

Kind Regards.

Adrià Ribatallada i Torelló.

Quoting adria.ribatall...@est.fib.upc.edu:

> Hi All, [...] > at org.jibx.schema.codegen.CodeGen.main(CodeGen.java:2206)

Font: http://www.mail-archive.com/jibx-users@lists.sourceforge.net/msg04861.html

A.8 Manual d'utilització d'erdapfel

ERDAPFEL

Enhanced Android map viewer User's Guide

By: Adrià Ribatallada Torelló

19 de juny de 2013

A.8.0.1 Overview

The erdapfel Android application is a test program to display, manage and edit maps.
It's intended to demonstrate the capabilities of the mapsforge-gml-kml library, which is an enhancement of the regular mapsforge library with gml and kml file support.

A.8.0.2 Installing

The package erdapfel.apk requires a minimum Android version 2.1 Eclair (API level 7),
To allow on device .map file generation you have to install the additional package mapfile-writer4and.apk, which requires minimum Android version 2.3.3 Gingerbread (API level 10), and a minimum of 128 Mb of virtual machine heap memory (high-end device).
Once installed, you can configure the settings of the application to fit your device needs.

A.8.0.3 settings menú

The settings menú contains the following configuration options:

- – Fullscreen: If enabled the main screen should cover the whole device screen space. If disabled, the main Android task bar will remain visible. This options requires a program restart.
- – Layers Save Folder: Specifies the location where the layers saved using the "Save Layers" option should be stored. We recomend somewhere in the external storage (default value is `/mnt/sdcard/download/out`)

Figura A.1: settings screenshot

A.8.0.4 main screen

The main screen displays a map and allows some operations over it.

You can interact with that map with the touchscreen. Dragging the finger through the screen will allow you to move the map area displayed, and will also show for a short time some buttons (which will disappear after certain amount of time showing the whole map on the screen again).
The buttons displayed over the map are the following:

- – PolygonMode: enables polygon edition mode.
- – MetricsMode: enables distance measurement and area calculation mode.
- – Clear: Clears all the user generated polygons (but leaves the map).
- – ZoomControls: increase or decrease the zoom level.

Additional options may be accessed when the Android Menú button is pressed. These options are:

- – Save Offline Map: Save the desired zone as a local mapfile (requires mapfile-writer4and application).

Figura A.2: map screenshot

Figura A.3: map with visible buttons

- – Load Offline Map: Load a mapfile (.map).
- – Load/Add Layer: Load a GML or KML file as a layer (and add it

to the map).

- – Save Layers: Save the user generated polygons as a kml file.
- – Settings: Access the settings screen.

Figura A.4: map with visible menú

By default, when the application starts, it tries to load a tile map from the internet, if no internet connection is available, you can still load an offline map using the appropriate menú option.

A.8.0.5 offline map loading

When the "Load Offline Map" option is selected from the menú, a file browser appears and lets you navigate through the android file system and choose a valid .map file to load.
Once the file is selected, the file browser disappears and the map is loaded in the main map view, overwriting whatever map was displayed previously and centring the view as the new .map file specifies.

Figura A.5: file picker

A.8.0.6 offline map generation (with mapfile-writer4and app)

This option only works when online, and it's intended to work with the mapfile-writer4and app to generate map files using the Android device.
When the option is selected from the menu, the map scrolling is dasabled, and instead the user input allows the user to select a rectangular area, which will be the stored map.
Note that sue to limitations with the OpenStreetMap servers, if the area selected is too big, this operation will fail.
Once the area is selected you can double-tap on the screen, and this will trigger the actual downloading of osm data and generation of the .map file.
This .map file generation requires that the mapfile-writer4and application is present in the system, if it is, then this process calls it and generates a new map file with the specified location.
Beware! this feature requires a high end device with a great amount of Dalvik Java heap memory (more than 128Mb).

A.8.0.7 polygon drawing

When the polygon drawing button is pressed, the application changes to polygon draw mode and the user input behaviour is changed.
You will always know that you are in polygon draw mode, for the green text

on top of the map.
In this operation mode the "slippy" map and zoom controls are locked, so make sure that the whole area where you want to draw is on screen before starting this mode.
In polygon draw mode whenever the screen is pressed you add a new polygon vertex. You can also edit an existing vertex position by dragging it.

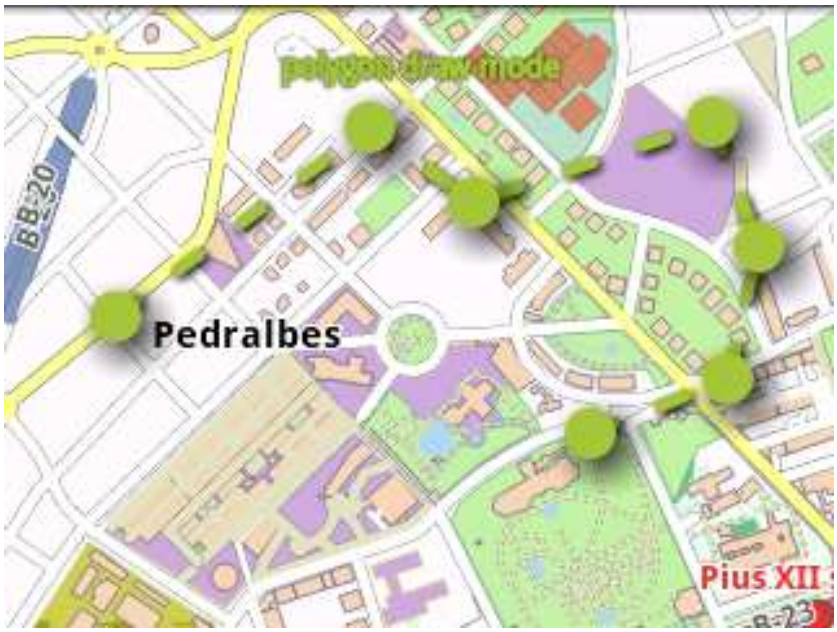

Figura A.6: polygon draw screenshot

A polygon is "completed" by joining the last vertex inserted with the first, thus closing the polygon, converting it to an actual layer over the map, and exiting the drawing mode.

You can also erase the last inserted vertex if you make a mistake by pressing the "back" Android button.
If all the vertexes are removed the application exits polygon draw mode and returns to the main map view mode.

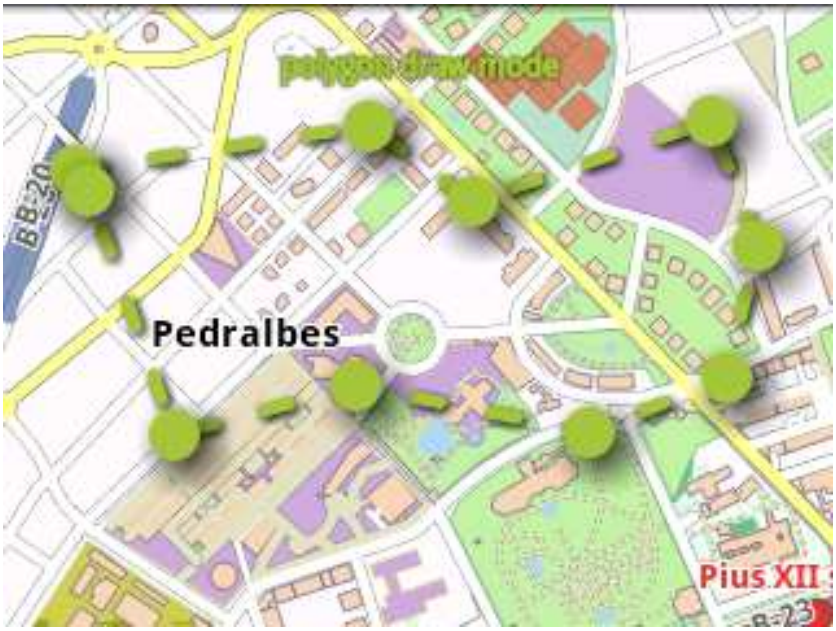

Figura A.7: polygon closing

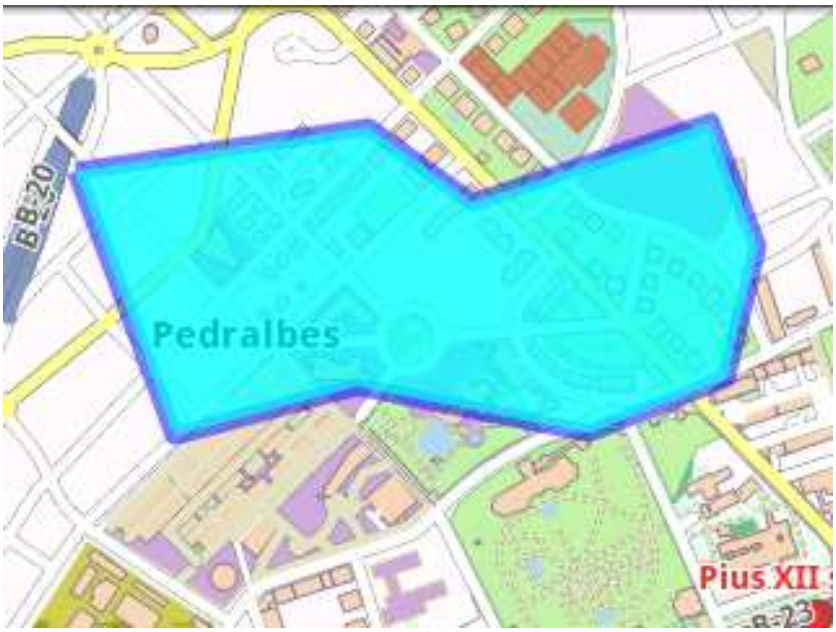

Figura A.8: polygon closed

A.8.0.8 metrics mode

The metrics mode is similar in operating to the polygon drawing mode.
In this case, the displayed text that lets you know you are in metrics mode, is light blue.
Whenever two vertexes are inserted in this mode, the approximate distance between points is painted over the line between the vertexes.

Figura A.9: distance between points

Please, note that this distance calculation is only an approximation and doesn't account for height or terrain irregularities.
If more than 2 vertexes are inserted, an approximate area calculation for the polygon represented is displayed on top of the screen.

This area approximation uses a simple area calculation algorithm and does not consider earth curvature nor geographical accidents. It also fails for cross intersecting polygons, so beware.

Figura A.10: area calculation

A.8.0.9 layer loading

The layer adding/loading option opens a file browser (like the one for map loading) that let's you choose a GML/KML file.
Once selected, if the file can be understood by erdapfel its contents will be added as a layer on top of the currently displayed map.

A.8.0.10 layer saving

This option allows to save all the generated polygons and loaded layers to a single kml file.
This file is placed under the folder specified in the settings screen.

A.8.0.11 invocation through Intents

It is possible to call erdapfel from another application via intents.
When doing so the application can be launched with parameters telling what Map file and/or Layer files to display.
This is done by putting extra parameters in the Intent, here's an example:

```
PackageManager pm = getPackageManager();
Intent intent = pm.getLaunchIntentForPackage("edu.fib.upc.erdapfel");
if (intent == null) throw new PackageManager.NameNotFoundException();
```

```
intent.putExtra("KMLFile","MyFile");
startActivity(intent);
```

For more information on the supported parameters and its syntax please read the technical documentation of erdapfel.

A.8.0.12 License

This program is under GPL license.

A.8.0.13 Contact

Adrià Ribatallada Torelló adria.ribatallada@est.fib.upc.edu

A.9 Exemple de fitxer GML

```
    <gml:boundedBy>
      <gml:Box srsName="EPSG:4326">
            <gml:coordinates>-0.768746,47.003018
              3.002191,47.925567</gml:coordinates>
      </gml:Box>
    </gml:boundedBy>
  <gml:featureMember>
    <ms:polygon fid="1">
      <gml:boundedBy>
            <gml:Box srsName="EPSG:4326">
                  <gml:coordinates>-0.768746,47.003018
                    0.532597,47.925567</gml:coordinates>
            </gml:Box>
      </gml:boundedBy>
      <ms:msGeometry>
      <gml:MultiPolygon srsName="EPSG:4326">
      <gml:polygonMember>
        <gml:Polygon>
          <gml:outerBoundaryIs>
            <gml:LinearRing>
              <gml:coordinates>-0.318987,47.003018
                -0.768746,47.358268 -0.574463,47.684285
                -0.347374,47.854602 -0.006740,47.925567
                0.135191,47.726864 0.149384,47.599127
                0.419052,47.670092 0.532597,47.428810
                0.305508,47.443003 0.475824,47.144948
                0.064225,47.201721 -0.318987,47.003018
                </gml:coordinates>
            </gml:LinearRing>
          </gml:outerBoundaryIs>
          <gml:innerBoundaryIs>
            <gml:LinearRing>
              <gml:coordinates>-0.035126,47.485582
                -0.035126,47.485582 -0.049319,47.641706
                -0.233829,47.655899 -0.375760,47.457196
                -0.276408,47.286879 -0.035126,47.485582
                </gml:coordinates>
```

```
            </gml:LinearRing>
          </gml:innerBoundaryIs>
        </gml:Polygon>
      </gml:polygonMember>
      </gml:MultiPolygon>
      </ms:msGeometry>
      <ms:ogc_fid>1</ms:ogc_fid>
      <ms:name>My Polygon with hole</ms:name>
      <ms:id>0</ms:id>
    </ms:polygon>
  </gml:featureMember>
  <gml:featureMember>
    <ms:polygon fid="2">
      <gml:boundedBy>
            <gml:Box srsName="EPSG:4326">
                  <gml:coordinates>1.511919,47.088176
                    3.002191,47.882988</gml:coordinates>
            </gml:Box>
      </gml:boundedBy>
      <ms:msGeometry>
      <gml:Polygon srsName="EPSG:4326">
        <gml:outerBoundaryIs>
          <gml:LinearRing>
            <gml:coordinates>1.625463,47.357844
              1.511919,47.741057 1.880938,47.882988
              2.420275,47.797830 2.789295,47.485582
              3.002191,47.457196 2.874453,47.088176
              2.178993,47.343651 1.625463,47.357844
              </gml:coordinates>
          </gml:LinearRing>
        </gml:outerBoundaryIs>
      </gml:Polygon>
      </ms:msGeometry>
      <ms:ogc_fid>2</ms:ogc_fid>
      <ms:name>My simple Polygon</ms:name>
      <ms:id>0</ms:id>
    </ms:polygon>
  </gml:featureMember>
  <gml:featureMember>
    <ms:polygon fid="3">
      <gml:boundedBy>
```

```
            <gml:Box srsName="EPSG:4326">
                <gml:coordinates>0.000000,45.000000
                    2.000000,47.000000</gml:coordinates>
            </gml:Box>
        </gml:boundedBy>
        <ms:msGeometry>
        <gml:MultiPolygon srsName="EPSG:4326">
        <gml:polygonMember>
          <gml:Polygon>
            <gml:outerBoundaryIs>
              <gml:LinearRing>
                <gml:coordinates>0.000000,45.000000
                    2.000000,45.000000 2.000000,47.000000
                    0.000000,47.000000 0.000000,45.000000
                    </gml:coordinates>
              </gml:LinearRing>
            </gml:outerBoundaryIs>
            <gml:innerBoundaryIs>
              <gml:LinearRing>
                <gml:coordinates>0.500000,45.500000
                    1.500000,45.500000 1.500000,46.500000
                    0.500000,46.500000 0.500000,45.500000
                    </gml:coordinates>
              </gml:LinearRing>
            </gml:innerBoundaryIs>
          </gml:Polygon>
        </gml:polygonMember>
        </gml:MultiPolygon>
        </ms:msGeometry>
        <ms:ogc_fid>3</ms:ogc_fid>
        <ms:name>my polygon with hole</ms:name>
        <ms:id>3</ms:id>
      </ms:polygon>
    </gml:featureMember>
```

A.10 Exemple de fitxer KML

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns=”http://earth.google.com/kml/2.2”>
<Document>
  <name>Ruta</name>
  <description><![CDATA[Ruta tuta tuta]]></description>
  <Style id=”style1”>
    <LineStyle>
      <color>73FF0000</color>
      <width>5</width>
    </LineStyle>
  </Style>
  <Placemark>
    <name>Linia 1</name>
    <description><![CDATA[]]></description>
    <styleUrl>#style1</styleUrl>
    <LineString>
      <tessellate>1</tessellate>
      <coordinates>
        2.109350,41.391361,0.000000
        2.110620,41.392216,0.000000
        2.111778,41.392570,0.000000
        2.113688,41.391186,0.000000
        2.114611,41.390526,0.000000
        2.114439,41.390221,0.000000
        2.114718,41.389851,0.000000
        2.115276,41.389915,0.000000
        2.115576,41.390266,0.000000
        2.118537,41.390461,0.000000
        2.122164,41.387661,0.000000
        2.122743,41.387691,0.000000
        2.123322,41.387451,0.000000
        2.123451,41.387257,0.000000
        2.135146,41.390285,0.000000
        2.143965,41.392536,0.000000
        2.146389,41.393085,0.000000
        2.150080,41.390331,0.000000
        2.155101,41.386517,0.000000
```

```
        2.159130,41.389561,0.000000
        2.166667,41.395180,0.000000
        2.170980,41.398495,0.000000
        2.169110,41.399891,0.000000
        2.163084,41.404545,0.000000
        2.163491,41.404900,0.000000
        2.162998,41.405495,0.000000
        2.163490,41.407700,0.000000
        2.164671,41.411724,0.000000
        2.166796,41.412094,0.000000
        2.169242,41.413010,0.000000
        2.171731,41.414364,0.000000
        2.175872,41.417934,0.000000
        2.176559,41.419285,0.000000
        2.177589,41.423759,0.000000
        2.176216,41.425419,0.000000
        2.173400,41.426220,0.000000
        2.169886,41.427219,0.000000
        2.164414,41.427845,0.000000
        2.163641,41.428085,0.000000
        2.162461,41.429165,0.000000
        2.161989,41.429665,0.000000
        2.161453,41.429535,0.000000
        2.160000,41.429668,0.000000
        2.159135,41.429630,0.000000
        2.158191,41.429470,0.000000
        2.157869,41.429340,0.000000
        2.156947,41.429810,0.000000
        2.156410,41.430435,0.000000
        2.155809,41.430470,0.000000
        2.155638,41.430080,0.000000
        2.156239,41.429455,0.000000
        2.156818,41.429054,0.000000
        2.157269,41.428764,0.000000
        2.157869,41.428440,0.000000
        2.158980,41.428669,0.000000
      </coordinates>
    </LineString>
  </Placemark>
</Document>
</kml>
```

Bibliografia

[1] *Llicència GPL v3*
http://www.gnu.org/licenses/gpl-3.0.ca.html

[consultat per últim cop el 6 de juny de 2013].

[2] *Llicència LGPL v3*
http://www.gnu.org/licenses/lgpl-3.0.ca.html

[consultat per últim cop el 6 de juny de 2013].

[3] *Perquè no LGPL*
http://www.gnu.org/philosophy/why-not-lgpl.html

[consultat per últim cop el 6 de juny de 2013].

[4] *Latex Project Public License*
http://www.latex-project.org/lppl.txt

[consultat per últim cop el 10 de juny de 2013].

[5] *Checkstyle home*
http://eclipse-cs.sourceforge.net/

[consultat per últim cop el 6 de juny de 2013].

[6] *FindBugs home*
http://findbugs.sourceforge.net/downloads.html

[consultat per últim cop el 6 de juny de 2013].

[7] *PMD home*
`http://pmd.sourceforge.net/integrations.html#eclipse`

[consultat per últim cop el 6 de juny de 2013].

[8] *JUnit Tests*
`http://ca.wikipedia.org/wiki/JUnit`

[consultat per últim cop el 6 de juny de 2013].

[9] *Eclipse Public License*
`http://www.eclipse.org/legal/epl-v10.html`

[consultat per últim cop el 6 de juny de 2013].

[10] *Apache Software License*
`http://www.apache.org/licenses/LICENSE-2.0.html`

[consultat per últim cop el 6 de juny de 2013].

[11] *Gmapcatcher user guide*
`http://code.google.com/p/gmapcatcher/wiki/User_Guide#User_interface`

[consultat per últim cop el 10 de maig de 2013].

[12] *Gmapcatcher issue with google maps terms of use*
`http://code.google.com/p/gmapcatcher/issues/detail?id=210`

[consultat per últim cop el 6 de juny de 2013].

[13] *OSM library*
`http://osmlib.rubyforge.org/osmlib-export/rdoc/index.html`

[consultat per últim cop el 2 de febrer de 2013].

[14] *GIS file formats*
`http://en.wikipedia.org/wiki/GIS_file_formats`

[consultat per últim cop el 2 de febrer de 2013].

[15] *OsmAnd*
http://code.google.com/p/osmand/

[consultat per últim cop el 6 de Juny de 2013].

[16] *osmdroid*
http://code.google.com/p/osmdroid/

[consultat per últim cop el 6 de Juny de 2013].

[17] *OpenStreetMap wiki*
http://wiki.openstreetmap.org/wiki/Main_Page

[consultat per últim cop el 6 de Juny de 2013].

[18] *KML documentation*
https://developers.google.com/kml/documentation/?hl=ca-ES

[consultat per últim cop el 6 de Juny de 2013].

[19] *KML reference*
https://developers.google.com/kml/documentation/kmlreference

[consultat per últim cop el 6 de Juny de 2013].

[20] *GML documentation*
http://www.opengeospatial.org/standards/gml

[consultat per últim cop el 6 de Juny de 2013].

[21] *GML Application Schemas*
http://en.wikipedia.org/wiki/GML_Application_Schemas

[consultat per últim cop el 6 de Juny de 2013].

[22] *Mapnik*
http://mapnik.org/

[consultat per últim cop el 6 de Juny de 2013].

[23] *AndNav*
http://code.google.com/p/andnav/

[consultat per últim cop el 6 de Juny de 2013].

[24] *Mapsforge project home*
http://code.google.com/p/mapsforge/

[consultat per últim cop el 6 de Juny de 2013].

[25] *Mapsforge-dev googlegroups*
https://groups.google.com/forum/?fromgroups#!forum/mapsforge-dev

[consultat per últim cop el 6 de Juny de 2013].

[26] *Kivy home*
http://kivy.org/#home

[consultat per últim cop el 6 de Juny de 2013].

[27] *Cross compiling Python for Android*
http://mdqinc.com/blog/2011/09/cross-compiling-python-for-android/

[consultat per últim cop el 6 de Juny de 2013].

[28] *Android SDK*
http://developer.android.com/sdk/index.html

[consultat per últim cop el 6 de Juny de 2013].

[29] *Python for Android*
http://code.google.com/p/python-for-android/

[consultat per últim cop el 6 de Juny de 2013].

[30] *Linuxjournal Python for Android*
http://www.linuxjournal.com/article/10940?page=0,0

[consultat per últim cop el 6 de Juny de 2013].

[31] *CRYSOL Python en Android*
http://crysol.org/es/node/1516

[consultat per últim cop el 6 de Juny de 2013].

[32] *SL4A API reference*
http://code.google.com/p/android-scripting/wiki/ApiReference

[consultat per últim cop el 6 de Juny de 2013].

[33] *Android Emulator documentation*
http://developer.android.com/tools/help/emulator.html

[consultat per últim cop el 6 de Juny de 2013].

[34] *SL4A Code Structure*
http://code.google.com/p/android-scripting/wiki/CodeStructure

[consultat per últim cop el 6 de Juny de 2013].

[35] *PySide for Android*
http://thp.io/2011/pyside-android/

[consultat per últim cop el 6 de Juny de 2013].

[36] *Gmapcatcher*
http://code.google.com/p/gmapcatcher/

[consultat per últim cop el 6 de Juny de 2013].

[37] *Interesting discussion on SL4A graphics*
https://groups.google.com/forum/?fromgroups#!topic/android-scripting/qJe1dZvrpdw

[consultat per últim cop el 6 de Juny de 2013].

[38] *Google Maps Reference*
https://developers.google.com/maps/documentation/android/

reference/?hl=ca

[consultat per últim cop el 12 d'Abril del 2012].

[39] *PyQGIS documentation*
http://www.qgis.org/pyqgis-cookbook/

[consultat per últim cop el 6 de Juny de 2013].

[40] *XML data binding for Java on Android*
http://blog.tourgeek.com/2011/12/xml-data-binding-for-java-on-android.html

[consultat per últim cop el 6 de Juny de 2013].

[41] *JiBX wiki*
http://en.wikipedia.org/wiki/JiBX

[consultat per últim cop el 6 de Juny de 2013].

[42] *Stackoverflow JiBX on Android*
http://stackoverflow.com/questions/958525/jibx-on-android-or-any-other-build-time-bytecode-manipulating-library

[consultat per últim cop el 6 de Juny de 2013].

[43] *Natvit home*
http://wiki.navit-project.org/index.php/Navit_on_Android

[consultat per últim cop el 6 de Juny de 2013].

[44] *Planet.osm extracts*
http://extract.bbbike.org/#

[consultat per últim cop el 6 de Juny de 2013].

[45] *OpenStreetMap API Reference*
http://wiki.openstreetmap.org/wiki/API_v0.6

[consultat per últim cop el 6 de Juny de 2013].

[46] *Memory management for Android Apps*
http://www.google.com/events/io/2011/sessions/memory-management-for-android-apps.html

[consultat per últim cop el 6 de Juny de 2013].

[47] *Osmosis detailed usage*
http://wiki.openstreetmap.org/wiki/Osmosis/Detailed_Usage_0.40#--sort_.28--s.29

[consultat per últim cop el 6 de Juny de 2013].

[48] *HTML5*
http://html5center.sourceforge.net/blog/What-to-Expect-from-HTML5-in-2013

[consultat per últim cop el 6 de Juny de 2013].

[49] *HTML5 geographical demos*
http://html5demos.com/geo

[consultat per últim cop el 6 de Juny de 2013].

[50] *Ultra-Easy Polygon Area Algorithm With C Code Sample*
http://alienryderflex.com/polygon_area/

[consultat per últim cop el 6 de Juny de 2013].

[51] *Pygame Subset for Android*
http://pygame.renpy.org/

[consultat per últim cop el 6 de Juny de 2013].

[52] *Android NDK Documentation*
http://developer.android.com/tools/sdk/ndk/index.html

[consultat per últim cop el 6 de Juny de 2013].

[53] *Processing OpenStreetMap data for Effective Cartography*
http://www.mapbox.com/blog/processing-osm/

[consultat per últim cop el 6 de Juny de 2013].

[54] *PythonOsmApi*
http://wiki.openstreetmap.org/wiki/PythonOsmApi

[consultat per últim cop el 6 de Juny de 2013].

[55] *Using OpenStreetMap Data with Open-Source GIS*
http://www.cartographicperspectives.org/index.php/journal/article/view/73/138

[consultat per últim cop el 6 de Juny de 2013].

[56] *Pyrender*
http://wiki.openstreetmap.org/wiki/Talk:Pyrender

[consultat per últim cop el 6 de Juny de 2013].

[57] *JiBX bindings*
http://jibx.sourceforge.net/tutorial/binding-extras.html#mixing

[consultat per últim cop el 6 de Juny de 2013].

[58] *KML opengis schema*
view-source:http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd

[consultat per últim cop el 6 de Juny de 2013].

[59] *My Tracks*
http://www.google.com/mobile/mytracks/

[consultat per últim cop el 6 de Juny de 2013].

[60] *gvSIG description*
http://www.gvsig.org/web/home/projects/gvsig-mobile

[consultat per últim cop el 6 de Juny de 2013].

[61] *[Mapnik-devel] Mapnik on Android*
http://lists.berlios.de/pipermail/mapnik-devel/2010-July/001208.html

[consultat per últim cop el 6 de Juny de 2013].

[62] *OpenStreetMap Tile usage policy*
http://wiki.openstreetmap.org/wiki/Tile_usage_policy

[consultat per últim cop el 6 de Juny de 2013].

[63] *Java Plug-in Framework*
http://jpf.sourceforge.net/

[consultat per últim cop el 6 de Juny de 2013].

[64] *Python home* http://python.org

[65] *Tango Desktop Project*
http://tango.freedesktop.org/

[consultat per últim cop el 6 de Juny de 2013].

[66] *MVC pattern in Android?*
http://stackoverflow.com/questions/2925054/mvc-pattern-in-android

[consultat per últim cop el 6 de Juny de 2013].

[67] *Android Async Task*
http://www.vogella.com/articles/AndroidBackgroundProcessing/article.html

[consultat per últim cop el 6 de Juny de 2013].

[68] *PBF Format*
http://wiki.openstreetmap.org/wiki/PBF_Format

[consultat per últim cop el 6 de Juny de 2013].

[69] *Termes de servei de google maps*
`developers.google.com/maps/terms`

[consultat per últim cop el 6 de Juny de 2013].

[70] *Issue 370 - mapsforge*
`http://code.google.com/p/mapsforge/issues/detail?id=370`

[consultat per últim cop el 6 de Juny de 2013].

[71] *Zamia project home*
`https://github.com/zamiaDroid/zamiaDroid`

[consultat per últim cop el 6 de Juny de 2013].

[72] Mark L. Murphy, *Beginning Android*. Springer-Verlag, New York, 1st Edition, 2009

ISBN-13 (pbk): 978-1-4302-2419-8 ISBN-13 (electronic): 978-1-4302-2420-4

[73] Lucas Jordan & Pieter greyling, *Practical Android Projects (chapter 5)*. Apress, 1st Edition, February 22, 2011

ISBN-10: 1430232439 ISBN-13: 978-1430232438

[74] Erik westra, *Python Geospatial Development*. Packt Publishing Ltd, 32 Lincoln Road, Olton, Birmingham, B27 6PA, UK, 1st Edition, December 2010

ISBN 978-1-849511-54-4

[75] Paul Ferrill, *Pro Android Python with SL4A*. Apress, 1st Edition, 2011

ISBN-13 (pbk): 978-1-4302-3569-9 ISBN-13 (electronic): 978-1-4302-3570-5

[76] Mark Lutz, *Programming Python*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 4th Edition, 2011

ISBN: 978-0-596-15810-1