

Títol: Desenvolupament d'aplicacions lliures per a telèfons mòbils usant la plataforma Android

Volum: 1

Alumne: David Casas Mangas

Director/Ponent: Antoni Soto Riera

Departament: Llenguatges i Sistemes Informàtics

Data: 22 de Gener de 2009

DADES DEL PROJECTE

Títol del Projecte: Desenvolupament d'aplicacions lliures per a telèfons mòbils usant la plataforma Android

Nom de l'estudiant: David Casas Mangas

Titulació: Enginyeria Informàtica Superior

Crèdits: 37.5

Director/Ponent: Antoni Soto Riera

Departament: Llenguatges i Sistemes Informàtics

MEMBRES DEL TRIBUNAL (nom i signatura)

President: Josep Vilaplana Pasto

Vocal: José Luis Ruiz Muñoz

Secretari: Antoni Soto Riera

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Copyright (c) 2009 David Casas Mangas. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Índex

1	Introducció	11
1.1	Motivació	11
1.2	Objectius i tasques	12
1.3	Estructura del document	13
2	Anàlisi	15
2.1	Android	15
2.1.1	Història del naixement de la plataforma Android	15
2.1.2	Característiques principals de la plataforma	17
2.1.3	Filosofia de Disseny d'Aplicacions	19
2.1.4	Particularitats d'una aplicació Android	20
2.1.5	Manifest Android	22
2.1.6	Sistemes operatius per a mòbil	23
2.1.7	Android i J2ME	25
2.2	Entorn de desenvolupament	25
2.2.1	Android Software Development Kit (SDK)	25
2.2.2	Comparació de versions	27
2.2.3	Plugin Eclipse (ADT)	27
2.2.4	Android APIs	28
2.2.5	Emulador	30
3	Primera Aplicació: Guia d'Oci de Barcelona	37
3.1	Objectius	37
3.2	Especificació i Disseny	38
3.3	Implementació	40

3.3.1	Estructura de paquets	40
3.3.2	Gestió de dades amb un Content Provider i SQLite3	41
3.3.3	Geolocalització	46
3.3.4	Utilització de la Media API	52
3.3.5	Utilització de enllaços entre aplicacions (<i>Linkfy</i>)	54
3.3.6	Serveis Web	56
3.3.7	Utilització de la interfície gràfica	67
3.3.8	Composició de components visuals propis	72
4	Segona Aplicació: Joc Senzill en OpenGL ES	81
4.1	Objectius	81
4.2	Especificació i Disseny	82
4.3	Implementació	83
4.3.1	Estructura de paquets	83
4.3.2	APIs de Gràfics	84
4.3.3	Comunicació d'emuladors online	95
5	Valoració Econòmica	103
5.1	Cost Temporal	103
5.2	Cost Econòmic	104
6	Conclusió	107
6.1	Objectius Aconseguits	107
6.2	Treball Futur	108
6.2.1	OcioBCN	108
6.2.2	BilliarDroid	108
6.2.3	Migració a la versió 1.0 de l'SDK	109
6.3	Plataforma Android	109
6.4	Documentació LaTeX	112
	Bibliografia	113
A	Instal·lació de l'entorn de desenvolupament d'Android	117
A.1	Eclipse 3.3 Europa	117

<i>ÍNDEX</i>	9
A.2 Eclipse 3.4 Ganymede	118
A.3 Java	118
B Utilització de Subversion (SVN)	121
B.1 Crear un repositori	121
B.2 Accés remot amb Apache 2.2	122
B.3 Operacions	123
B.3.1 Baixar un projecte	123
B.3.2 Actualitzar una còpia local	123
B.3.3 Enviar Modificacions	124
B.4 Subclipse	124
C GNU Free Documentation License	125

Capítol 1

Introducció

En aquest capítol d'introducció primer s'explicarà la motivació d'aquest projecte, justificada per l'aparició d'*Android* que suposa un canvi substancial en el desenvolupament d'aplicacions per a mòbils.

Tot seguit es definiran els objectius concrets del projecte per a l'anàlisi de la plataforma *Android*, amb l'explicació dels objectius de cada una de les aplicacions desenvolupades.

Per acabar, s'estructurarà i resumirà el contingut dels capítols que contindran aquesta memòria.

1.1 Motivació

En el món actual, la tecnologia està evolucionant per formar part, cada cop més, de la vida quotidiana de les persones. Actualment, la utilització d'equips embotrats en llocs de treball o l'ús de dispositius mòbils i portàtils han entrat en una dinàmica de creixement. Les proporcions de mòbils per persona van creixent anualment, la qual cosa obre un món de possibilitats a aquests tipus de dispositius a mesura que la tecnologia evoluciona.

En l'actualitat existeixen dos sistemes operatius per a mòbils que són considerats com els principals: *Windows Mobile* [30] i *Symbian* [41]. Tal i com el seu nom indica, *Windows Mobile* forma part dels sistemes operatius de *Microsoft* [29]. D'altra banda, *Symbian* va ser un producte de l'aliança entre diferents companyies de telèfons mòbils, *Nokia* y *Sony* entre les més importants, per tal d'intentar competir contra *Windows Mobile*. No obstant, el desenvolupament d'aplicacions per a aquests dos sistemes no és senzill degut a la quantitat de diferències entre els diferents dispositius que els utilitzen. Es podria dir que aquests dos sistemes s'adapten als dispositius, cosa que fa que cada aplicació desenvolupada utilitzi una implementació parcialment diferent segons el dispositiu utilitzat.

La plataforma *Android* [18] és la solució completa de programari per a dispositius mòbils que va sorgir com un intent de revolucionar el món dels dispositius mòbils per

part de l'*Open Handset Alliance* [34]. Aquesta associació és un conjunt de més de 30 companyies tecnològiques que comparteixen una mateixa visió: *construir un mòbil millor milloraria la vida d'incomptables persones*. A més a més, *Android* té l'objectiu de ser una plataforma lliure, de manera que el programari que incorpori el sistema operatiu no es diferenciï en res del desenvolupat per tercers, cosa que faria que els desenvolupadors puguin gaudir de totes les característiques del dispositiu mòbil sense límits, a més de tenir la possibilitat d'implementar aplicacions estàndards entre diferents dispositius.

Degut a aquesta imminent aparició de la plataforma *Android* en el món dels telèfons mòbils i de les oportunitats que en un principi oferirà en comparació amb la seva competència, aquest projecte estarà enfocat a l'anàlisi de les característiques d'aquesta plataforma. Per tant, analitzarem el desenvolupament d'aplicacions per a mòbils amb l'entorn de desenvolupament¹ facilitat per l'*Open Handset Alliance*.

1.2 Objectius i tasques

El projecte presenta com a objectiu principal l'anàlisi del desenvolupament d'aplicacions en la plataforma *Android* en un entorn de treball de programari lliure.

Per tal de fer l'anàlisi de les possibilitats de desenvolupament que ens ofereix aquesta plataforma, en la nova generació de telèfons mòbils, subdividirem el projecte en diferents objectius per tal d'avaluar les diferents característiques ofertes. Se sap que el desenvolupament de la telefonia mòbil està en creixement i que si encara no existeixen dispositius amb una tecnologia molt superior a l'actual, es degut al paper que hi juguen les diferents operadores i proveïdors. Per això, no s'ha cregut convenient posar límits a l'hora de generar aplicacions per a mòbils, encara que actualment aquestes aplicacions no poguessin ser utilitzades per gran part de la població.

Com tot treball científic, el nostre projecte ha de contenir una fase inicial de recerca sobre la matèria. Al tractar-se d'un entorn encara en vies de desenvolupament, aquesta fase de recerca es mantindrà constant al llarg del projecte, ja que s'hauran d'investigar els canvis involucrats en cada actualització de la plataforma. A més a més, s'analitzarà quin és el temps necessari per a que un desenvolupador sense coneixements en *Android* s'iniciï en aquest nou entorn.

El primer objectiu és avaluar una part de les *Application Programming Interfaces* (API) de l'entorn de desenvolupament d'*Android* a partir d'una aplicació que té com a principal ús ser una guia d'oci de Barcelona. Aquesta ha d'involucrar i provar les diferents llibreries per tal de poder comunicar un telèfon mòbil amb l'exterior mitjançant serveis web, ja que aquesta tecnologia ofereix una manera d'estandarditzar la comunicació entre diferents sistemes operatius i llenguatges, cosa que aporta valor en un entorn ja desenvolupat. Aquesta comunicació es farà entre el telèfon mòbil i un punt central d'informació que estarà representat per una aplicació web encarregada de gestionar la informació dels diferents locals de Barcelona, entre els quals hi hauran cinemes, res-

¹En anglès, Software Development Kit, més conegut com a SDK

taurants, hotels i llocs d'oci. A més a més, la utilització de mapes amb simulació de GPS² i la persistència i gestió dels continguts obtinguts a través d'aquests serveis web serien punts interessants per experimentar en aquesta plataforma. D'altra banda, aquesta aplicació contindrà una implementació de reproductor de vídeos online, amb l'objectiu de visualitzar els tràilers de les pel·lícules que formin part de la cartelera actual. Aquest reproductor no reproduirà els vídeos emmagatzemats al mòbil, sinó que farà ús de dades disponibles en algun servidor web, pel qual per a cada reproducció es realitzarà una connexió a Internet.

Aquesta primera aplicació utilitzarà gran part de les llibreries que actualment té disponibles *Android*. Tanmateix, *Android* té decidit donar-li un impuls a l'entorn gràfic dels mòbils similar a l'estratègia seguida per *Apple* [3] amb *iPhone* [4]. Per aquesta raó, la plataforma inclourà dues llibreries gràfiques, *SGL* [37] i *OpenGL ES* [26], especialment dedicades a gràfics en 2D i 3D respectivament. L'experimentació de gràfics 3D amb aquestes llibreries mitjançant el desenvolupament d'un joc senzill esdevindrà el segon objectiu del projecte. Aquest joc incorporarà una altra de les característiques innovadores d'*Android*, com és la comunicació *Peer To Peer (P2P)* amb altres dispositius per mitjà de la llibreria *XMPP* [57]. D'aquesta manera, s'utilitzaran dos noves característiques d'aquesta plataforma, les quals no havien estat provades en l'anterior aplicació. A més a més, existeix la possibilitat de provar de portar jocs ja existents en altres plataformes cap a *Android*, en comptes de desenvolupar un joc des de zero.

D'altra banda, la investigació de les possibilitats que ofereix l'emulador, integrat amb l'SDK d'*Android*, es situarà com un altre objectiu del projecte. Es descriuran les proves efectuades amb l'emulador i les similituds que es poden aconseguir comparant-lo amb un mòbil real. El desenvolupament de més aplicacions vindrà determinat pel temps d'investigació i desenvolupament dels objectius recentment esmentats del projecte.

Finalment, la redacció de la memòria posarà punt i final al projecte de final de carrera. Aquesta estarà estructurada de tal manera que es pugui fer una anàlisi de cada una de les aplicacions i dels objectius de que consta aquest projecte. Es considerarà aquesta redacció com a objectiu final del projecte, ja que aquí és on s'inclouran totes les conclusions respecte el nou entorn de desenvolupament d'*Android*.

1.3 Estructura del document

A continuació es farà un breu resum del contingut de la resta de capítols de la memòria.

En el primer capítol 2 es farà una introducció general de la plataforma incloent els objectius que vol aconseguir. A més a més, s'analitzarà amb detall els facilitats que aporta el entorn de desenvolupament i les diferències que destaquen entre les diverses versions del entorn que sorgeixin en el transcurs del projecte.

Posteriorment en el capítol 3, s'explicarà les passes seguides per a desenvolupar la primera aplicació, una guia d'oci per la ciutat de Barcelona. En aquest capítol s'analitzaran en detall les *Application Programming Interfaces (APIs)* utilitzades en el

²Global Positioning System

desenvolupament de l'aplicació i es descriuran les seves fases de especificació, disseny i implementació.

En el tercer capítol 4 s'analitzarà la segona aplicació desenvolupada, un joc senzill en tres dimensions per tal d'avaluar la potència gràfica que la plataforma *Android* pot adquirir. Aquest capítol contindrà una petita introducció de OpenGL ES ja que és la llibreria gràfica que inclou aquesta plataforma. D'altra banda s'analitzaran totes les etapes del desenvolupament tal i com s'ha fet anteriorment amb la primera aplicació.

Per cada una de les dos aplicacions es traurà una avaluació econòmica del desenvolupament i es farà una valoració econòmica global de tot el projecte 5. Les conclusions sobre les APIs utilitzades i la facilitat d'utilització i desenvolupament que ofereix la plataforma formaran part del últim capítol 6.

Capítol 2

Anàlisi

Aquest capítol inclou tota la informació necessària per entendre el que significa la irrupció d'aquesta plataforma en el món mòbil. S'analitzaran tots els avantatges que introdueix la plataforma *Android* per tal de poder extreure conclusions al final del desenvolupament de les diferents aplicacions que componguin el projecte.

També s'inclourà una explicació de les fases a seguir per a l'inici del desenvolupament d'aplicacions i s'analitzarà cadascuna de les utilitats que inclou l'entorn de desenvolupament, com per exemple: emulador, adb, el plugin d'*Eclipse*, etc.

2.1 Android

La plataforma Android, un projecte de l'*Open Handset Alliance*, consisteix en un sistema operatiu, middleware i les aplicacions principals de qualsevol telèfon mòbil. Entre aquestes aplicacions s'inclouen característiques típiques com són el gestor de trucades o el gestor de missatgeria, tant de correus electrònics com de missatges per mòbil. El cor del sistema operatiu està basat en un kernel 2.6 de Linux[16] encara que incorpora una ampla quantitat de llibreries exclusivament generades per a Android. *Java* ha estat el llenguatge que s'ha escollit per el desenvolupament d'aplicacions. No obstant això, l'execució d'aquestes aplicacions no es fa sobre la típica *Java Virtual Machine* de *Sun* sinó que *Android* utilitza una màquina virtual especialment implementada per a dispositius mòbils anomenada *Dalvik Virtual Machine*. Més endavant s'explicarà amb detall quines són les raons que van fer que l'*Open Handset Alliance* escollís aquesta màquina virtual.

2.1.1 Història del naixement de la plataforma Android

La història de la plataforma *Android* comença quan, al juliol del 2005, *Google* va adquirir una empresa en creixement anomenada *Android Inc.* Aquesta companyia es dedicava al desenvolupament d'aplicacions per a mòbils i, segons la revista *Business Week*[5], estaven en fases de desenvolupament d'un nou sistema operatiu per a aquests

dispositius. Va ser una adquisició sense masses detalls ni boom mediàtic, *Google* va dir què més que l'apropiació d'aquesta empresa, cercaven la incorporació de les ments dels enginyers que hi treballaven.

Poc després, al 2007, es va anunciar la creació de l'*Open Handset Alliance*, una associació de 34 grans empreses dedicada a desenvolupar estàndards per a dispositius mòbils. Entre aquestes s'inclouen *Google*, que és l'empresa que lidera i dirigeix l'associació, *NVIDIA*, *Intel*, *Motorola*, *T-Mobile* i altres companyies relacionades amb aquests tipus de dispositius. Amb l'aparició d'aquesta aliança i del telèfon mòbil d'*Apple*, l'*iPhone*, creixia la possibilitat de pensar que *Google* s'introduiria al mercat dels mòbils amb un producte similar al d'*Apple*, el qual tothom anomenava *gPhone*. No obstant això, *Google* ha intentat introduir-se d'una manera diferent i molt més amplia en aquest món i, per això, l'*OHA* va anunciar el llançament de la plataforma *Android* per mitjans de l'any 2008, una plataforma amb la qual es busca aconseguir una revolució en aquest entorn. Aquesta revolució arriba cercant la idea de que *construir un mòbil millor milloraria la vida d'incomptables persones*, ja que aquests dispositius cada cop formen més part del dia a dia de les persones. Per tal d'aconseguir-ho, tal i com s'ha comentat en el capítol anterior, la plataforma *Android*[18] funcionarà con a solució completa de programari per a qualsevol dispositiu mòbil que compleixi una sèrie de requisits, és a dir, el sistema no s'adaptarà al mòbil sinó que serà el mòbil el que s'hagi d'adaptar al sistema.

En un principi, *Google* ha anunciat que la plataforma serà una plataforma lliure basada en una llicència *Apache 2.0*. D'aquesta manera, desenvolupadors de tot el món podran gaudir del codi per tal de que cada mòbil pugui ser modelable a les necessitats de l'usuari, fent que cada dispositiu pugui arribar a ser un smartphone¹ únic. A més a més, aquesta llibertat es transmet al desenvolupament d'aplicacions, ja que no hi haurà diferències entre el programari que incorpora el sistema operatiu i el desenvolupat per tercers, donat a que les eines de desenvolupament seran les mateixes. Igualment, les aplicacions base del sistema podran ser modificades quan el codi sigui realment lliure, o substituïdes per altres aplicacions dotades de major prioritat en l'ús de les capacitats del telèfon. Una altre de les característiques que importa aquesta plataforma, tal i com s'ha esmentat abans, és la unificació del desenvolupament per a dispositius mòbils, ja que farà que un desenvolupament serveixi per als diferent dispositius que utilitzin *Android* i disposin de les capacitats o requisits necessaris. Actualment, la desunificació en el desenvolupament fa que les empreses de programari hagin de re-implementar les seves aplicacions per a diferents dispositius del mercat, cadascun amb unes característiques d'implementació diferents.

En un principi s'espera l'aparició del primer dispositiu mòbil amb *Android* a partir de finals de l'any 2008, segurament per part d'*HTC* o *Motorola*. Encara no es saben les especificacions concretes que hauria d'incorporar aquest dispositiu, el que si que és segur és que no serà necessari que tingui una pantalla tàctil. Actualment, el sistema operatiu d'*Android* ja ha pogut ser emulat en diferents dispositius amb totes les funcionalitats disponibles excepte la de telefonia mòbil. És a dir, gaudint d'un dispositiu *Android*

¹Segons la Wikipedia: Smartphone és un ordinador de butxaca que integra les funcions de telèfon mòbil, organitzador personal i sovint altres funcions de connectivitat mòbil.

com a *PDA*² en comptes de telèfon mòbil.

2.1.2 Característiques principals de la plataforma

En aquesta secció es comenten les característiques d'*Android* que l'han fet situar-se com a un rival complicat per als actuals sistemes operatius de mòbils. Les característiques que destaquen més són:

- Entorn d'aplicacions per tal de re-utilitzar i reemplaçar components
- Màquina Virtual Dalvik
- Explorador Web integrat basat en el motor WebKit
- Gràfics 2D SGL i 3D en OpenGL ES 1.0
- SQLite
- Suport de formats d'àudio, vídeo i imatges (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

L'entorn³ d'aplicacions d'*Android* és una de les principals novetats que incorpora aquesta plataforma en dispositius mòbils. L'arquitectura del sistema està dissenyada per simplificar la re-utilització de components i la substitució d'aplicacions. Això vol dir que qualsevol aplicació pot fer ús de les funcionalitats d'una altre i fins i tot, demanar els permisos adients per tal d'integrar-les dins de la propia aplicació. Això genera un entorn en el qual les aplicacions generades per tercers poden substituir les aplicacions per defecte d'*Android*, podent fer ús de tota la infraestructura del sistema.

Una de les altres característiques, amb la qual es basen les aplicacions d'*Android*, és l'ús de la *Dalvik Virtual Machine*, en comptes de la generalment utilitzada *Java Virtual Machine*. Una de les diferències principals entre aquestes dues màquines és que la *Dalvik* és una màquina basada en registres mentre que la de *Java* està basada en piles. A més a més, *Dalvik* ha estat dissenyada per a l'ús extrem de poca memòria, el qual li permet executar-se en sistemes encastats i funcionar correctament en situacions de baix subministrament elèctric. *Dalvik* té la capacitat d'executar diverses instàncies de la seva màquina virtual de manera eficient, per tal de dur a terme el propòsit de que cada aplicació corri en un procés diferent. *Google* ha comentat que *Dalvik* es va implementar per intentar resoldre el problema de fragmentació que hi havia a J2ME i dissenyar un sistema totalment lliure. Cal esmentar que aquesta màquina treballa amb binaris *.dex*, que són classes de *Java* optimitzades per dispositius mòbils mitjançant l'eina *dx*.

L'explorador web integrat en la plataforma està basat en el projecte Open Source WebKit en un intent de gaudir d'una experiència de navegació agradable gràcies a

²Personal Digital Assistant

³En anglès, *framework*

l'usabilitat i la velocitat que ofereix. Cal dir que aquest explorador inclou dos petites millores que fan que rendeixi d'una manera immillorable en petits dispositius. Primer de tot, *Google* ha introduït dos fases de renderització per capes. La primera capa carrega la pàgina sense esperar els blocs bloquejants (CCS externs, javascripts...) i fa una segona passada de renderitzat un cop té tots els elements descarregats. A més a més, es disposa d'una característica anomenada allisador de marcs⁴, que fa que un marc compost de diversos elements es sintetitzi com si es tractés d'un sol element. Amb la introducció d'aquestes dues millores s'aconsegueix una millora de velocitat de l'explorador.

Per finalitzar, es comentaran dos altres aspectes importants de la plataforma com son l'elecció de *SQLite* com a sistema gestor de bases de dades i la introducció d'unes llibreries gràfiques potents per a dispositius mòbils. *SQLite* va ser escollit com a sistema gestor de bases de dades relacionals pel poc espai que necessita. Aquest petit gestor té un disseny simple que resulta ideal per a una plataforma com Android. D'altra banda *Android* compleix l'especificació d'OpenGL ES 1.0 i probablement també complirà l'especificació OpenGL ES 1.1 per la implementació de gràfics en 3D. A més a més, incorpora un motor gràfic 2D en SGL per tal de millorar l'eficiència dels algorismes gràfics.

D'altra banda, existeixen algunes característiques que són dependents del maquinari del dispositiu, encara que algunes d'elles venen incorporades en la majoria dels mòbils actuals.

- Acceleració gràfica 3D
- Tecnologia GSM
- Connexions Bluetooth, EDGE, 3G i WiFi
- Càmera
- GPS
- Acceleròmetre ⁵

Totes aquestes característiques venen complementades per un gran entorn de desenvolupament propi, que fa que la línia d'aprenentatge per tal de generar aplicacions mòbils sigui mínima. No obstant això, la manca de documentació d'algunes de les APIs i els canvis estructurals entre versions marquen un punt negatiu en el que, segurament, serà un entorn molt usable per al desenvolupament d'aplicacions per a mòbils. A la secció 2.2 es trobarà explicat amb més detall els elements que componen aquest entorn.

⁴En Anglès, frame flattening

⁵Un acceleròmetre està compost per una sèrie de sensors, que es poden programar per executar accions segons el grau d'inclinació del dispositiu

2.1.3 Filosofia de Disseny d'Aplicacions

Un mòbil és un dispositiu de petites dimensions que ha d'oferir uns serveis determinats als usuaris. Actualment, les aplicacions desenvolupades per a aquests dispositius se semblen més al programari que s'utilitza en sistemes més grans, com els ordinadors de sobretaula, però amb una capacitat de processat molt més baixa. A més a més, l'entorn en el qual s'oferixen els serveis dels mòbils és diferent i s'esperen una sèrie de requisits a l'hora d'utilitzar-los.

Per tal de que els desenvolupadors no ensopeguin amb aplicacions inservibles en aquest entorn, *Android* proposa una sèrie de tres característiques com a base de qualsevol aplicació per a mòbils. Una aplicació ha de ser:

1. ràpida
2. interactiva
3. fluïda

Una aplicació per a mòbils és ràpida quan és eficient. En els últims temps, l'eficiència de les aplicacions per a ordinadors de sobretaula ha deixat de ser un dels factors importants en el procés de desenvolupament. Això és gràcies a l'aplicació de la llei de *Moore*, la qual indica que a mesura que passa el temps es pot empaquetar més circuits en un menor espai, cosa que implica poder tenir més potència i velocitat mantenint les mesures del xip. No obstant, en el món mòbil aquesta llei actua de manera diferent ja que l'aplicació de la llei de *Moore* implica l'aparició de dispositius amb la mateixa potència, però amb unes dimensions reduïdes o altres prestacions més importants, com un consum de bateria menor.

La interacció d'una aplicació amb l'usuari és un dels factors clau en el seu futur èxit. Aplicacions que no indiquen quin procés s'està executant o que es congelen mentre fan alguna operació costosa són exemples que poden provocar la desesperació de l'usuari en l'ús de l'aplicació corresponent. Per tal d'evitar incòmodes esperes, *Android* incorpora un mecanisme per tal d'oferir al usuari, mitjançant un missatge anomenat *ANR*⁶, la finalització del procés o aplicació que es troba en execució. La figura 2.1 mostra el missatge que envia *Android* a l'usuari. Cal esmentar que, una manera d'evitar que una aplicació doni sensació d'inactivitat és l'ús de fils⁷ al realitzar operacions costoses. D'altra banda, els factors anteriorment comentats no són suficients com per a no poder provocar una sensació d'incomoditat o desesperació a un usuari que utilitza un dispositiu mòbil. En aquests tipus de dispositius, l'espai d'interacció entre les aplicacions i l'usuari és molt petit, cosa que fa que les aplicacions hagin de respectar una sèrie de normes per a que l'usuari pugui gaudir dels serveis de la plataforma. El sistema d'*Android* ha estat dissenyat per tal de fer de un mòbil un espai on les aplicacions poden fer participar a altres constantment, fent el mòbil un espai de col·laboració entre aplicacions. Per aquesta raó, el sistema ha d'aplicar un conjunt de limitacions de seguretat per tal

⁶Application Not Responding

⁷Traducció de la paraula anglesa thread

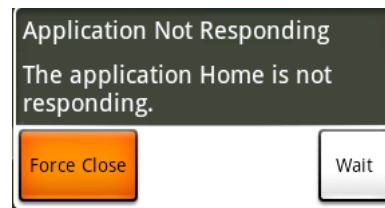


Figura 2.1: Missatge ANR d'Android

de sincronitzar-les. És per això que existeixen una sèrie de permisos, restriccions i prioritats que fan que les aplicacions no puguin fer ús del sistema a no ser que l'usuari ho hagi indicat anteriorment. Això vol dir per exemple, que una aplicació que gestioni la recepció de trucades telefòniques sempre tindrà la prioritat màxima, ja que s'encarrega de tractar la funció per a la qual van estar dissenyats aquests dispositius, i per tant és la principal aplicació del sistema. D'altra banda, *Android* ha introduït el concepte de *Content Providers* per tal de gestionar les dades entre un conjunt d'aplicacions, per tal de crear un sistema molt més integrat. Aquest concepte ve explicat a la secció 2.1.4.

2.1.4 Particularitats d'una aplicació Android

Una aplicació en *Android* es pot observar com un conjunt de blocs que interaccionen entre ells. Aquests blocs fonamentals són anomenats *Activities*, *Intent Receivers*, *Content Providers* i *Services*. Cadascun d'ells du a terme una funció diferent dins de les aplicacions de la plataforma.

Les *Activities* o Activitats representen cada una de les pantalles amb les que pot interaccionar un usuari. Les activitats del sistema *Android* tenen un cicle de vida semblant al de les pàgines d'un explorador web, les quals poden recuperar les pantalles anteriors a través d'un històric. Aquest històric es va modificant segons el consum de memòria del dispositiu, ja que en determinats moments el sistema decideix netejar parts del històric i matar els processos amb menys prioritat, fent una correcta gestió dels recursos disponibles. A la figura 2.2, trobem el cicle de vida complet d'una activitat.

Tal i com s'observa a la imatge, cada cop que es genera una nova activitat, l'anterior entra en estat *Paused* i passa a formar part de la pila històrica. Així doncs, es poden anar recuperant les aplicacions que l'usuari havia anat utilitzant amb la pulsació del botó *Back* del dispositiu. No obstant això, les pantalles que no es creguin importants donat a que la seva recuperació pot no ser útil al usuari podran ser seleccionades per tal de no guardar-les en l'anomenada pila. A més a més, existeixen activitats, anomenades *SubActivities*, que no es guarden a l'històric, donat que només interactuen amb l'activitat que les ha cridat, és a dir l'activitat pare, pel que no es considera important la seva recuperació.

Per tal de passar d'una pantalla a un altre, *Android* utilitza una classe especial anomenada *Intent* amb la qual s'especifica quina acció es vol executar. Aquests manifesten quin tipus d'acció vol executar l'usuari per tal que el sistema utilitzi les aplicacions que

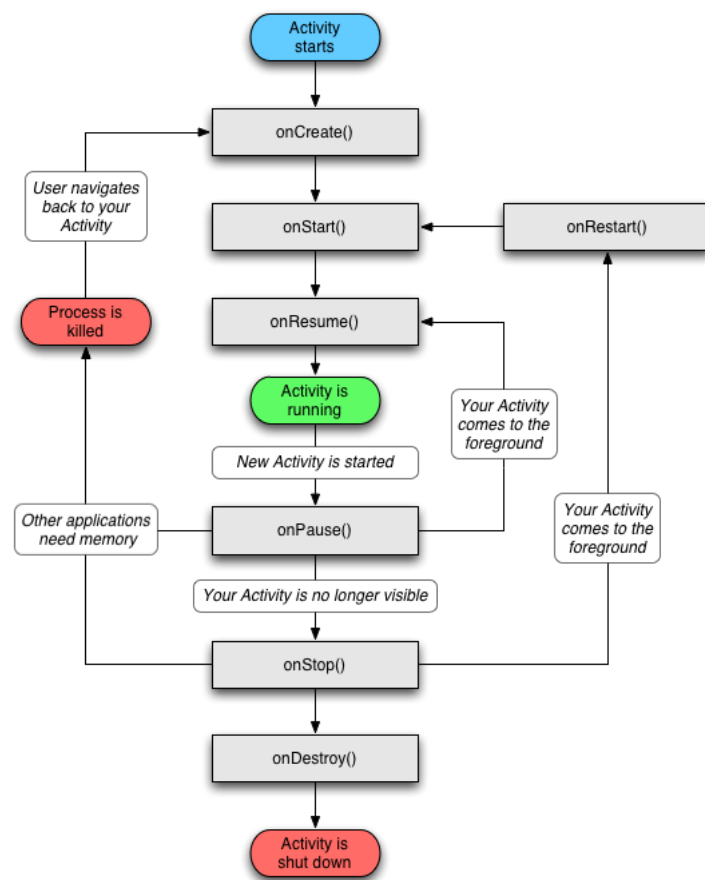


Figura 2.2: Cicle de vida d'una Activity

cregui més adients. A més a més de l'acció, aquests elements van normalment acompanyats del tipus de dades a tractar, especificant un tipus MIME[56]. Per tal d'efectuar aquestes accions, les aplicacions defineixen uns elements anomenats *IntentFilters* en un arxiu anomenat *manifest.xml* que formen una part principal de qualsevol aplicació i serveix com a definició dels seus elements principals. Els *Intent Filters* indiquen quin tipus d'accions podrà agafar cadascuna de les activitats d'una aplicació. Si dos o més *Activities* poden gestionar un mateix tipus d'*Intent*, el sistema *Android* escollirà quina és la més adequada, per mitjà d'analitzar les seves prioritats, i, en cas de no tenir preferència per cap d'elles, el sistema *Android* li donarà l'elecció a l'usuari. Tot això implica que qualsevol aplicació o activitat pot ser substituïda per una altre que declari els mateixos *IntentFilters*, però amb una prioritat igual o superior a l'aplicació que es vulgui substituir. D'altra banda, un *Intent* podria definir exactament quina activitat s'ha d'executar en un moment determinat, per exemple en les pantalles dins d'una mateixa aplicació. Per això, els *Intent Filters* han sigut utilitzats, en les aplicacions d'aquest projecte, per llençar accions externes a una aplicació, com el generar trucades i la visualització de pàgines web, i per la comunicació a través de *GTalk* entre dispositius, en la qual es necessita llençar una aplicació o activitat nova segons el missatge que s'hagi rebut.

Un altre bloc són els receptors d'intents⁸. Aquests blocs són classes que s'executen com a reacció d'un esdeveniment extern, com per exemple una trucada telefònica o un missatge. Els *Intent Receivers* no disposen d'interfície gràfica i han d'interaccionar amb el *Notification Manager*, excepte en el cas de trucades, per tal de informar a l'usuari dels esdeveniments rebuts i de les accions per defecte que es poden prendre.

Els serveis o *Services* són processos de llarga duració que no fan ús interfície gràfica. Per tal de modificar un servei es necessita una activitat que es connecti al servei per tal d'informar a l'usuari del seu estat. Un servei típic podria ser el reproductor de multimèdia que pot reproduir cançons mentre que l'usuari executa alguna altre aplicació en el mòbil.

Per últim, es disposa de *Content Providers*⁹. Tal i com el seu nom indica, es tracta de gestors d'informació per tal de que les aplicacions puguin compartir informació. Aquests blocs implementen un conjunt estàndard de mètodes (inserir, eliminar..) per tal de permetre a altres aplicacions guardar i recuperar les dades gestionades per un d'aquests proveïdors. Normalment, aquests blocs s'utilitzen en conjunció amb el gestor de base de dades *SQLite3* que es troba incorporat a la plataforma *Android*. No obstant això, es pot fer ús de qualsevol sistema de gestió que es cregui convenient, tal com fitxers *XML* o altres gestors de bases de dades.

2.1.5 Manifest Android

Totes les aplicacions desenvolupades per a ser utilitzades a la plataforma *Android*, hauran d'incorporar un manifest anomenat *AndroidManifest.xml*. Aquest fitxer es troba

⁸En anglès, *Intent Receivers*

⁹En català, proveïdor de continguts

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="upc.fib.android.games.billiardroid">
    <uses-permission android:name="com.google.android.gtalkservice.permission.GTALK_SERVICE" />
    <application android:icon="@drawable/icon">
        <activity android:name=".BilliarDroidActivity" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="BilliardGameActivity">
            (. . .)
        </activity>
    </application>
</manifest>

```

Figura 2.3: Android Manifest

situat al directori arrel de cada aplicació i descriu els aspectes globals de l'aplicació desenvolupada. El manifest descriu els components de l'aplicació que el paquet exposa (Activities, Services, etc.) i les classes que els implementen. A més a més, es declara quin tipus de dades poden tractar i el moment en el es pot fer ús de cadascun dels seus components.

Un altre aspecte important d'aquest fitxer és la declaració d'*Intent Filters*, que són els filtres que indiquen quan l'aplicació pot ser llençada. *Android* genera Intents cada cop que necessita fer ús d'una aplicació del sistema, ja sigui per esdeveniments sobre el dispositiu mòbil o perquè una aplicació ho demana. D'aquesta manera, el sistema supervisa quines aplicacions tenen declarat el poder manegar el tipus determinat d'*Intent* i escolleix quina és, entre elles, la més adequada a l'*Intent* generat.

D'altre banda, aquest document serveix per a que l'aplicació, a al instal·lar-se o usar-se, demani els permisos adients per a la utilització d'un determinat element del sistema. A més a més, si es tracta d'una aplicació que necessiti seguretat i a la qual poden accedir altres aplicacions, s'hauran d'especificar quins permisos haurien d'obtenir les aplicacions que hagin d'utilitzar-la. Això es faria amb els tags `<uses-permission>` i `<permission>`, per demanar i declarar els permisos respectivament.

La figura 2.3 mostra un fragment de l'*AndroidManifest.xml* d'una de les aplicacions que formen part d'aquest projecte, s'observen les característiques abans esmentades.

S'observa que, al ser un fitxer XML[50], al principi del document es declara el tipus d'arxiu que és i el *namespace* corresponent a l'esquema que s'utilitza per a declarar les aplicacions d'*Android*.

2.1.6 Sistemes operatius per a mòbil

En l'actualitat, el món dels *smartphones*, que son telèfons mòbils d'altres prestacions amb els quals treballaria Android, es troba dominat per tres sistemes operatius. Aquests sistemes són *Windows Mobile*, *Symbian* i, el recent incorporat, *iPhone*, que encara que no és un sistema operatiu pròpiament dit, incorpora un sistema propi totalment diferent als altres dos esmentats. A part d'aquests tres, existeixen altres alternatives de

codi obert com ho serà la plataforma *Android*, anomenades *LiMo*[28] i *OpenMoko*[35]. Tanmateix, aquestes dos alternatives es troben molt lluny d'arribar a ser competidors d'aquesta plataforma ja que persegueixen una mateixa filosofia que *Android*, però tenen molt menys suport per part dels manufacturadors.

La diferència principal de *Symbian* amb els altres és que es tracta només d'un sistema operatiu per a mòbils. Tant *Windows Mobile*, com l'*iPhone* i com el recent incorporat *Android* porten un conjunt d'aplicacions base i serveis que aporten un major valor al dispositiu mòbil. *Symbian* amb la introducció de l'*iPhone* i d'*Android* deixarà de ser l'únic competidor de *Microsoft* en aquest mercat. Igualment, continuarà sent un principal competidor gràcies a que *Nokia*, que és la companyia que més mòbils ven al món amb més d'un 58% de *share*[17] al mercat a l'any 2008, n'és el principal soci. A més a més, a mitjans de 2008 *Nokia* es va fer amb la majoria de les accions d'aquest sistema operatiu i va fundar la *Symbian Foundation*[42], fundació dedicada a fer de *Symbian* un sistema de codi obert. D'aquesta manera, *Symbian* intentarà frenar els passos d'*Android* i continuar sent el principal sistema de mòbils.

Android aporta una sèrie d'avantatges que altres sistemes encara no tenen. El principal és l'entorn de desenvolupament que aporta la plataforma, que dona als desenvolupador gran facilitat d'implementació d'aplicacions per a mòbil. Aquest entorn és semblant del que *Apple* ofereix per l'*iPhone*, encara que disposen de dos grans diferències: el llenguatge de programació i les funcionalitats que poden arribar a absorbir les aplicacions de tercers. *Apple* ofereix un entorn en *C++* que facilitarà el control de la memòria per aplicacions molt pesades, com per exemple els jocs. *Android* ofereix Java, un llenguatge amb un ràpid aprenentatge ja que gestiona la memòria el propi llenguatge. D'altra banda, les aplicacions desenvolupades per *Android* podran arribar a substituir les aplicacions base del sistema sense haver de modificar el codi del sistema, ja que disposa d'eines per a poder implementar qualssevol dels serveis principals del dispositiu. L'execució de les aplicacions es basa en unes regles per escollir quina de les aplicacions disponibles té més prioritat. Cal esmentar, que *Windows Mobile* també incorpora el seu propi entorn de desenvolupament, però les aplicacions desenvolupades amb aquest no arriben a ser fàcils d'utilitzar.

Una altra de les diferències amb l'*iPhone* és la manera en que s'intenta introduir *Android* al mercat. A diferència d'*Apple*, *Google* no intenta crear el seu propi mòbil sinó que intenta introduir la seva plataforma dins dels diversos fabricants del mercat. *Google* calcula que la introducció d'*Android* reduiria el preu dels mòbils en un 10% del cost actual, ja que l'elecció d'aquest sistema de codi obert abaratiria els seus costos de fabricació. L'*iPhone*, per contra, té l'avantatge de ser l'únic dispositiu amb un sistema operatiu *MacOS* propi d'*Apple*. Això fa que *Apple* tingui el control de totes les fases de la seva producció i, per tant, disposaria de les capacitats necessàries per tal de solventar qualsevol problema.

L'arribada d'*Android* al mercat dels mòbils va generar diverses reaccions en premsa de diferents directius dels seus competidors. *Scott Horn*, que es part del equip de màrqueting de *Microsoft Windows Mobile*[14], va dir que "realment sembla que està unint a un gran grup de gent per tal de fabricar un mòbil, que es una cosa que porta fent Microsoft durant cinc anys". Per part de *Symbian*, *John Forsyth*[13] explicava

què “Symbian és realment el que té mòbils i una plataforma real, amb una riquesa construïda durant anys”.

2.1.7 Android i J2ME

Android i *J2ME* són dos coses totalment diferents. *J2ME* és un llenguatge de *Sun* que incorpora un conjunt de llibreries adequades per a l'entorn mòbil. D'altra banda *Android* és una plataforma completa que inclou tant el sistema operatiu com un entorn de desenvolupament basat en el llenguatge *Java*. A més a més, *Android* no dóna suport a les aplicacions en *J2ME*, pel que les aplicacions desenvolupades en aquest llenguatge haurien de ser portades. La raó principal és la introducció de la màquina virtual *Dalvik* en els dispositius *Android*, en contra de la típica màquina virtual *Java* de *Sun*, la qual no pot interpretar fitxers *.class*. D'altra banda, qualsevol llibreria desenvolupada en *J2ME* pot intentar portar-se *Android* gràcies a l'eina *dex* del seu entorn de desenvolupament.

Tanmateix, es pot fer una comparació entre el desenvolupament de les seves aplicacions. L'entorn ofert per *Android* amb el plugin d'*Eclipse* és similar al de desenvolupament amb *Netbeans*[32] de *J2ME*. *Android* disposa de les llibreries bàsiques de *J2ME* com *java.util* i *java.net*, encara que cadascun porten APIs totalment diferents. A més a més, les aplicacions desenvolupades per a *Android* només estaran disponibles per a dispositius amb aquest sistema, mentre que les aplicacions de *J2ME* només necessitaran que el sistema tingui una màquina virtual capaç d'interpretar els seus arxius binaris. És per això, que encara no es descarta que *Android* pugui utilitzar aplicacions en *J2ME*.

2.2 Entorn de desenvolupament

Aquesta secció està dedicada a explicar les eines que s'inclouen a l'entorn de desenvolupament d'*Android*, a més d'explicar en detall l'ús de les més importants. A l'apèndix A hi ha una descripció detallada del procés d'instal·lació del entorn. A més a més de l'entorn de desenvolupament, cal observar que es necessari tenir instal·lada una versió del entorn de desenvolupament de *Java*, ja sigui la versió 5 o 6 de *J2SE*. Això es degut a que el procés de compilació és propi del llenguatge *Java*, per mitjà del compilador *javac*, i que *Android* proporciona una eina per traduir els objectes *.class* produïts en aquest procés de compilació a objectes *.dex*. D'aquesta manera les aplicacions poden ser executades a la màquina virtual de la plataforma.

2.2.1 Android Software Development Kit (SDK)

Android inclou un conjunt de llibreries base que proveeixen la major part de les funcionalitats disponibles en las llibreries base del llenguatge de programació *Java*. A aquest conjunt de llibreries, es pot afegir algunes de les llibreries desenvolupades en *J2SE* i la majoria de *J2ME*.

Per tal de donar de flexibilitat als programadors, l'entorn de desenvolupament d'*Android* porta tot un conjunt d'eines necessàries per a poder crear i depurar les aplicacions desenvolupades en aquest. No obstant això, la majoria d'elles queden en desús si el desenvolupador utilitza l'entorn d'*Eclipse* amb el plugin ADT, que explicarem més endavant.

L'eina principal sense la qual seria impensable pensar en desenvolupar aplicacions per a telèfons mòbils és l'emulador. Aquest incorpora totes les característiques d'un mòbil d'última generació, podent-hi ajustar alguns dels aspectes del sistema, com la velocitat de connexió, la mida de la pantalla o si disposa de teclat tàctil o no. L'ús d'aquest emulador vindrà detallat al final d'aquest capítol, a la secció 2.2.5.

Existeixen dos eines per a poder configurar i depurar les dades d'aquest emulador. Aquestes són el *Dalvik Debug Monitor Service (DDMS)* i l'*Android Debug Bridge (ADB)*. Aquest segon és de molta utilitat per a usuaris de Linux, ja que permet accedir, mitjançant l'eina telnet, a un terminal de l'emulador. Això fa que es puguin instal·lar i eliminar aplicacions, dades i modificar qualsevol altre dada interna de l'emulador. D'altra banda la DDMS ens ofereix la possibilitat de depurar les nostres aplicacions i de poder interaccionar amb diferents instàncies de l'emulador d'una manera visual i interactiva mitjançant el plugin d'*Android* (ADT). Aquestes dos utilitats ens ofereixen la possibilitat de afegir i esborrar fitxers a diferents directoris del sistema operatiu del emulador.

Cada una de les aplicacions d'*Android* utilitza un procés d'execució diferent i això complica la compartició d'objectes entre diferents aplicacions de la plataforma. Per tal de poder compartir informació, els processos han de descomposar els objectes en primitives per tal de que el sistema els pugui entendre. Per tal de facilitar aquesta feina, la plataforma ens ofereix l'*Android Interface Description Language (AIDL)*. Aquesta eina és un llenguatge IDL¹⁰ utilitzat per generar el codi que fa que dos processos es puguin comunicar en un dispositiu *Android* a través de la comunicació interprocessos (IPC). El mecanisme AIDL IPC s'ha implementat per a l'ús amb interfícies, similar a CORBA[33] o a COM[31], però més lleuger. Utilitza un proxy per tal de passar informació del client la implementació del serveix que ofereix. Això significa que l'AIDL IPC és el mecanisme pel qual diferents aplicacions clients poden fer us d'un servei de la plataforma Android.

L'emulador també permet la simulació d'emmagatzemament de dades en una targeta SD simulada. Per tal de generar un fitxer que faci les funcions de targeta SD, tenim l'eina *mksdcard*, en la qual s'haurà d'indicar el fitxer que simula la targeta a l'hora d'iniciar l'emulador amb el paràmetre *-sdcard*. A l'hora de crear-los s'ha d'especificar la mida en megabytes (M) o kilobytes (K) de la manera següent:

```
mksdcard -l LABEL <size> (M|K) <file >
```

Finalment, hi ha una sèrie d'eines que queden practicament en desús si s'utilitza el plugin ADT d'*Eclipse*. La primera d'elles és una eina d'empaquetament de aplicacions en .apk, que és l'extensió dels instal·lables d'*Android*. ADT utilitza aquesta eina, anome-

¹⁰IDL és un llenguatge de definicions de interfícies, les sigles angleses d' *Interface definition language*

nada *aapt*¹¹, per generar fitxers en el moment de compilació dels projectes d'*Android*, sempre que aquesta compilació és correcta, i els guarda a la carpeta *bin* del directori on es troben els fitxers fonts. A part d'això, l'eina *aapt* té un comportament similar a qualsevol gestor de fitxers i permet llistar, crear paquets i afegir o esborrar arxius d'un paquet determinat. D'altra banda existeix també l'eina anomenada *dx* que és l'eina encarregada de traduir els fitxers executables de *Java* (.class) en arxius llegibles per la consola virtual *Dalvik* (.dex), la qual també és utilitzada per ADT en el moment de compilació a través d'*Eclipse*. Per últim, hi ha un script Python, anomenat *activityCreator.py*, que s'encarrega de generar els arxius principals d'un projecte *Android*. Aquests fitxes són també generats en la creació d'un projecte *Android* mitjançant l'IDE¹² d'*Eclipse* que en fa ús d'aquest script per a generar-los.

2.2.2 Comparació de versions

Una de les principals raons que estan fent que desenvolupadors d'arreu del món encara no hagin estat implementant programari per la plataforma *Android* és la inestabilitat que provoca el seu entorn de desenvolupament. A data de novembre de 2008, havien sortit tres versions d'aquest entorn, anomenades M3, M5 i 0.9b per la comunitat de desenvolupadors d'*Android*. En un principi, els desenvolupadors haurien d'anar basant-se en la última versió per a començar a desenvolupar aplicacions per a una nova generació de mòbils. *Google* va anunciar que no tenia pensat treure cap altre versió fins que la plataforma no estiguera al mercat. D'aquesta manera, va sortir posteriorment la versió 1.0 de l'aplicatiu, que és bàsicament igual a l'anterior versió, encara que dóna estabilitat ja que serà la versió instal·lada en els primers dispositius de la plataforma.

D'altra banda, hi ha una gran part de desenvolupadors que, durant el desenvolupament d'aquest projecte, no havien migrat el codi de les seves aplicacions a la versió M5. Normalment era per una raó principal: l'*Android Global Contest*. Els canvis més importants són la introducció de gran part dels permisos actualment existents per dotar a la plataforma de major seguretat i els canvis en el desenvolupament de gràfics 3D. Durant l'explicació de les aplicacions desenvolupades s'entrarà en detall de les coses que s'han de canviar per poder migrar jocs OpenGL de la versió M3 cap a la M5.

Cal esmentar, que les últimes versions de l'entorn de desenvolupament, no han estat analitzades en el transcurs d'aquest projecte donat que el seu llençament es va produir durant la redacció d'aquesta memòria.

2.2.3 Plugin Eclipse (ADT)

L'*Open Handset Alliance* també proporciona un plugin per Eclipse, anomenat *Android Development Toolkit*, per tal de facilitar la implementació d'aplicacions en aquest entorn de desenvolupament integrat. En el context d'aquest projecte s'han instal·lat les dues versions d'ADT, una per la versió M3 del SDK i una altre per l'M5, en l'*Eclipse* Europa

¹¹Android Asset Packaging Tool

¹²Sigles de Integrated Development Environment

3.2. D'altra banda, a la versió 3.4 d'*Eclipse* també s'ha instal·lat sense problemes. En l'apèndix A tindrem una explicació detallada de la instal·lació completa de l'entorn.

2.2.4 Android APIs

Tal i com s'ha comentat al inici del capítol, Android ofereix una sèrie d'APIs per tal de poder utilitzar amb facilitat els serveis que incorpora. En aquesta secció s'explicarà les funcionalitats i particularitats bàsiques de cadascuna d'elles, per poder entrar en més detall en els capítols posteriors.

APIs multimèdia

L'API de multimèdia proporcionada per *Android* s'encarrega de reproduir i enregistrar els diferents tipus de codificacions suportades actualment per aquest sistema. La plataforma és capaç de reproduir tant vídeo com àudio, en diferents formats. En cas de trobar-se en una trucada, la reproducció d'àudio i de vídeo seria desactivada, per tal de no interrompre l'activitat de l'usuari. En un principi, els fitxers per reproduir poden ser tant fitxers d'una aplicació, com d'un directori del sistema com fitxers reproduïbles online, és a dir, en mode *streaming*.

Per la reproducció, *Android* ofereix la classe `android.media.MediaPlayer`, amb la qual el desenvolupador només haurà de crear el *mediaplayer* amb el fitxer de l'aplicació corresponent i reproduir-lo amb el mètode *start*. Cada cop que es vulgui reproduir de nou, un cop ja ha estat creat el *MediaPlayer*, s'haurà de cridar al mètode *prepare*. En el cas de reproduir un fitxer online¹³ o des de un directori del dispositiu es crearà un reproductor amb el constructor per defecte i se li introduirà la font de la qual s'extrauran les dades a reproduir. En aquest cas, s'haurà de cridar al mètode *prepare* abans de la primera reproducció.

El cas de la gravació multimèdia és semblant a l'anterior utilitzant la classe `android.media.MediaRecorder`. Les passes a seguir seran les següents:

1. Crear la instància del *MediaRecorder*
2. Crear una instància de `android.content.ContentValues` per donar les propietats del fitxer on es guardarà el vídeo o l'àudio.
3. Crear un directori per guardar el contingut
4. Opcionalment per vídeo, veure una prèvia del que s'enregistrarà amb `MediaRecorder.setPreviewDisplay()`
5. Utilitzar el mètode `MediaRecorder.setVideoSource` per tal de especificar l'entrada de vídeo i el mètode `MediaRecorder.setAudioSource` per la d'àudio.

¹³En el cas dels vídeos online en la versió M5 SDK, només estan suportats fitxers .mp4 o .3gp que siguin capaços de reproduir-se progressivament.

6. Introduir els paràmetres de format, mida i velocitat de captura de vídeo amb *setOutputFormat()*, *setVideoSize()* i *setVideoFrameRate()*.
7. Introduir la codificació de vídeo i àudio amb *setAudioEncoder()* i *setVideoEncoder()*.

Després de configurar la gravació de la manera descrita anteriorment, ja es podria començar a enregistrar amb els mètodes *prepare()*, *start()* i *stop()*. Un cop acabat l'enregistrament s'hauria d'alliberar el *MediaRecorder* amb el mètode *release()*.

API de geolocalització

Aquesta és la primera versió d'aquesta API que segurament canviarà en un futur. L'API de geolocalització proveïda per *Android* ens facilitarà la detecció de l'existència de hardware GPS al dispositiu, ja estigui integrat al telèfon mòbil com un dispositiu exterior, i proporcionarà informació sobre la situació en la qual es troba l'usuari. Les passes a seguir són:

1. Registrar-se al servei amb `Context.getSystemService(Context.LOCATION_SERVICE)`
2. Llistar els proveïdors de localització, per saber la última localització coneguda del dispositiu.
3. Enregistrar les notificacions de actualització de posició del dispositiu per criteri o nom.
4. Enregistrar els punts d'interès en els quals es llançaran un determinat `Intent`.

En la secció 2.2.5, s'indicarà la manera de simular un dispositiu GPS en un entorn de desenvolupament Android.

API OpenGL

Android incorpora suport a l'acceleració 3D per hardware. Es pot accedir a aquesta funcionalitat a través de l'API de OpenGL ES. Aquesta API és molt similar a la desenvolupada per Sun per J2ME, la J2ME JSR239 OpenGL ES API[40], amb algunes diferències. Totes les aplicacions desenvolupades amb l'especificació OpenGL 1.3 de sobretaula podrien ser migrades a un dispositiu mòbil *Android* sense masses inconvenients.

Per tal d'utilitzar-la, bàsicament s'ha d'enllaçar l'aplicació amb un `OpenGLContext` i a partir del mètode *onDraw* de la vista, fer totes les operacions OpenGL necessàries per reproduir l'escena. En la secció 4.3.2, aquesta API serà explicada amb detall.

APIs per la utilització de serveis Google

Tal i com s'ha comentat en el capítol anterior, *Google* forma part de la Open Handset Alliance i lidera el projecte de la plataforma Android. Per aquesta raó, la plataforma disposa d'una API per tal de poder fer ús dels serveis proporcionats per aquesta companyia. Els dos serveis en els quals s'han invertit la majoria d'esforços són la utilització de *Google Maps*¹⁴ i la comunicació de dispositius a través de *GTalk*. En un principi aquesta comunicació anava a ser transmesa a través del protocol XMPP, però al no complir tots els requeriments que requerien l'estàndard d'*XMPP*, Google va decidir llençar un protocol propi de comunicació basat en l'anterior.

Aquest protocol és emprat per gestionar l'intercanvi de missatges del servei de comunicació *GTalk*, que té l'objectiu de reduir el cost econòmic d'aplicacions que tinguin necessitin intercanviar molta informació entre dispositius. Aquesta informació actualment és intercanviada a través de missatges SMS, cosa que resultaria molt costosa per a l'usuari i que faria que aquest tipus d'aplicacions s'utilitzessin només en moments de vertadera necessitat. A més a més, la comunicació a través de *GTalk* està gestionada de tal manera que el tractament de missatges es produeix a una velocitat molt superior a la que es faria amb missatges SMS actuals. D'altra banda, la velocitat de recepció dels missatges és alhora més ràpida gràcies a l'existència d'una única connexió persistent de *GTalk* en el dispositiu, què fa que el tràfic de missatges sigui molt fluït. En aquesta connexió es realitza tot el tràfic de missatges, tant els missatges entre usuaris com els missatges de control del sistema de missatgeria P2P.

D'altra banda, el servei de mapes de *Google Maps* en combinació amb la posició GPS del dispositiu, pot provocar el desenvolupament d'una quantitat d'aplicacions considerable enriquint els serveis a través de la geolocalització de l'usuari. Les eines principals per treballar amb mapes son el *MapView* i la *MapActivity*. La *MapActivity* s'encarrega de gestionar el funcionament del *MapView* i aquest ofereix tot tipus d'interacció amb l'usuari, tan visual com manual.

2.2.5 Emulador

La plataforma *Android* proporciona un emulador del sistema operatiu per tal de poder provar les aplicacions desenvolupades per aquest sistema. Aquest emulador està basat en el emulador de codi obert *qemu*[15], en concret la versió 0.8.2, el qual es capaç d'emular una arquitectura sencera. Fins ara, el desenvolupament d'aquest emulador es trobava per la versió 0.9.1, la qual corregeix alguns errors trobats en la versió utilitzada per *Android*. No obstant, aquest errors poden ser solucionats manualment modificant el codi del emulador d'*Android*, que es troba disponible online[20].

L'emulador incorpora el sistema operatiu de la plataforma amb una sèrie d'aplicacions i serveis bàsics per tal de simular les funcions típiques i el comportament d'un dispositiu mòbil. Aquestes funcionalitats poden ser gestionades i provades amb aplicacions desenvolupades en *Android* mitjançant la simulació d'events del dispositiu, com per exemple

¹⁴Actualment la majoria de dispositius mòbils de tercera generació ofereixen algun tipus de servei de mapes

missatges *SMS* o *XMPP* o la simulació de trucades al mòbil. A més a més, proveïx una serie de temes¹⁵ per tal de simular dispositius amb mides i tipus de dispositius diferents.

D'altra banda, es poden modificar diferents paràmetres de la simulació de l'emulador introduint una sèrie de variables en la seva comanda d'inicialització.

L'emulador necessita dos ports per tal de poder obtenir i modificar les seves dades. Per defecte s'utilitzen els ports 5554 i 5555, encara que per cada instància es comprova que els ports no estiguin ocupats i si ho estan es va incrementant el número de port fins a trobar un port lliure. Aquesta comprovació fa que es pugui dur a terme la comunicació entre diferents emuladors, ja que cadascun utilitzarà ports diferents. En el primer es troba la consola del emulador i en el segon trobem el servidor de l'*Android Debug Bridge*. Ambdues aplicacions es troben explicades en les seccions posteriors.

Consola de l'emulador

L'accés a la consola de l'emulador es fa mitjançant l'eina estàndard *telnet*[45]. A partir d'aquesta consola es permet modificar algun dels paràmetres de la connexió de xarxa del emulador i readreçar els ports. A més a més, també es pot configurar l'emulador per a què actui amb una latència o retard depenent del tipus de connexió, ja sigui GPRS, EDGE/EGPRS o UMS. La velocitat de pujada i baixada a Internet també es podrà modificar mitjançant el tipus de connexió o indicant aquesta velocitat de transmissió en número de kilobits per segon. Els tipus de connexions disponibles són: GSM/CSD, HSCSD, GPRS, EDGE/EGPRS, UMTS/3G i HSDPA.

D'altra banda, la consola del emulador també ens permetrà enviar i rebre trucades, dades i missatges. No obstant això, la funcionalitat de transmissió de dades mitjançant la xarxes sense fils i Bluetooth[7] encara no ha estat implementada, pel que no es podran enviar missatges ni dades amb aquest tipus de tecnologies. La simulació de trucades es farà mitjançant la comanda `call <número mòbil origen>` i l'enviament de missatges amb `sms send <número mòbil origen> <text del missatge>`. Esmentar que aquestes comandes s'executen mitjançant l'accés a la consola a través de l'eina *telnet*.

Android Debug Tool

L'eina anomenada Android Debug Tool, o ADT, és una utilitat molt versàtil que permet la instal·lació d'aplicacions i l'extracció d'informació i dades guardades al sistema operatiu de l'emulador. Existeixen quatre comandes bàsiques amb les quals es pot executar un terminal, instal·lar una aplicació, posar i extreure dades i obtenir informació del estat d'alguna de les instàncies de l'emulador. Totes aquestes característiques es troben integrades al plugin d'*Eclipse*, pel què no és del tot necessari utilitzar les comandes mitjançant la línia de comandes del sistema operatiu.

¹⁵Més coneguts com *skins*

Els paquets d'instal·lació d'aplicacions d'*Android*, amb extensió `.apk`, poden ser generats en la compilació de l'aplicació des de l'IDE d'*Eclipse* o directament amb l'eina d'empaquetament de la plataforma. Per tal de realitzar la instal·lació de l'aplicació, es seleccionarà la instància de l'emulador en el qual s'instal·larà i se li passarà la ruta del fitxer de la següent manera: `adb -d <instància16> install <ruta del paquet .apk>`. Per tal de desinstal·lar-la, no existeix cap comanda predeterminada, pel que s'haurien d'eliminar els fitxers instal·lats per l'anterior paquet, els directoris dels quals es troben a `/data`, amb el terminal d'aquesta eina. Tanmateix, es disposa del paràmetre `-wipe-data` amb el qual es podrà fer un reset del dispositiu, tornant-lo al seu estat inicial sense l'aplicació instal·lada. Es recomanable utilitzar aquest paràmetre en el cas d'observar un comportament estrany de l'emulador.

El terminal o *shell* de l'emulador es pot executar mitjançant la comanda `adb [-d <instància>] shell`. A partir d'aquí, ens trobarem en una shell UNIX reduïda amb la qual es podran executar les comandes necessàries, com per exemple crear i eliminar directoris o fitxers, canviar l'IP de les interfícies de xarxa o administrar les bases de dades en *SQLite3*. Aquest paquet es troba preinstal·lat al sistema Android com a sistema gestor de bases de dades predeterminat de la plataforma.

Per tal d'introduir i eliminar fitxers del emulador utilitzarem les comandes `pull` i `push` de la següent manera: `adb push <directori de l'ordinador> <directori de l'emulador>` i `adb pull <directori de l'emulador> <directori de l'ordinador>`. S'ha d'actuar pensant en que son dos directoris en àmbits diferents, un és un directori del sistema operatiu propi del emulador i un altre és el del sistema operatiu que utilitzi el desenvolupador.

Simulació GPS

Una de les possibles característiques dels dispositius mòbils d'*Android* és que podrien incorporar un dispositiu GPS integrat. No obstant això, qualsevol mòbil podria disposar d'aquest dispositiu mitjançant la connexió d'un aparell GPS extern. No obstant això, disposar d'un GPS integrat disminuiria el nombre de aparells utilitzats per un usuari normal. Les aplicacions desenvolupades per mitjà d'*Android* podran oferir serveis a partir de la posició del dispositiu. Per tal de poder desenvolupar aquest tipus d'aplicacions es disposa d'una simulació de GPS per mitjà d'una sèrie de rutes predeterminades per l'usuari en diferents formats.

Per defecte, l'emulador incorpora una posició predeterminada de GPS, anomenada *GPS*, sense moviment situada a San Francisco. Aquesta simulació podrà ser substituïda per un número indeterminat de simulacions, creant diferents directoris dins del sistema, el nom dels quals produirà el nom de la simulació. Un número molt gran de simulacions pot ser un inconvenient, ja que quantes més rutes es trobin introduïdes al sistema més es relentitzarà l'emulador. Aquesta lentitud és deguda a l'increment del número de

¹⁶El número de la instància del dispositiu no és necessari en cas de només executar una instància del emulador. No obstant, es pot trobar l'estat de les diferents instàncies amb la comanda `adb devices`.

càlculs per tal de trobar la posició actual de cadascuna de les simulacions introduïdes al dispositiu.

Primer de tot es necessita un fitxer de propietats per tal de que el *LocationProvider*, que serà el gestor de la localització del dispositiu. Aquest fitxer de propietats definirà els següents paràmetres booleans i enters per tal de configurar la simulació de GPS. Els valors, per defecte, dels paràmetres no especificats seran *true* pels paràmetres booleans i *0* pels enters:

- `requiresNetwork` (booleà)
- `requiresSatellite` (booleà)
- `requiresCell` (booleà)
- `hasMonetaryCost` (booleà)
- `supportsAltitude` (booleà)
- `supportsBearing` (booleà)
- `supportsSpeed` (booleà)
- `repeat` (booleà)
- `powerRequirement` (1 = baix, 2 = mitjà, 3 = alt)
- `accuracy` (mesurat en metres)
- `trackSpeed` (mesurat en km/h. Només s'utilitza en conjunció amb rutes kml.)

Les propietats *altitude*, *speed* i *bearing* només han d'estar reportades en el cas de que també estiguin definides les propietats *supportsAltitude*, *supportsSpeed* i *supportsBearing* a *true*.

Tots els arxius de configuració de GPS han d'estar situats dins d'un directori en la següent ruta: `/data/misc/location/<nom>/`, la simulació dels quals s'anomenarà amb el nom del directori final. Al demanar els serveis del servei de localització GPS al *LocationProvider*, aquest llegirà l'arxiu de propietats i posteriorment buscarà un dels següents fitxers de ruta en el següent ordre:

1. `class`
2. `kml`
3. `nmea`
4. `track`

Si es tracta d'una classe de Java, aquesta estarà instanciada i instal·lada com a proveïdor de continguts del sistema. D'altra banda, el més usual és utilitzar alguna aplicació de mapes per tal de generar un fitxer amb la ruta determinada. En aquest projecte s'ha fet servir l'aplicació *Google Earth* per a generar les rutes en fitxers *kml*, amb el següent format:

```
<longitude>,<latitude>,<altitude> <longitude>,<latitude>,<altitude>
```

Tanmateix, existeixen dues opcions més per tal de generar aquestes rutes. Una de les opcions és la generació d'un fitxer *nmea*, en concret *NMEA 0183*. Aquests fitxers tenen les rutes inserides per increments de temps. Això significa que la ruta es va generant segons el temps d'ús del sistema. Per a cada increment, existeix una posició diferent fins al final del fitxer.

L'última opció ha fer servir és utilitzar un fitxer del tipus *track* en el següent format:

```
<time> <longitude> <latitude> <altitude> <bearing> <speed>
```

El paràmetre *time* ha de començar per 0 per determinar l'inici de la ruta. A partir d'aquí, la simulació actuarà com l'anterior a partir d'increments del temps del sistema. D'altra banda, *bearing* i *speed* són dos paràmetres opcionals. El paràmetre *bearing* interpreta els graus d'inclinació des del nord i la velocitat (paràmetre *speed*) ha d'estar expressada en metres per segon.

Cal dir que qualssevol dels fitxers de ruta anomenats hauran de ser renombrats amb el nom de la seva extensió. Per exemple, un fitxer anomenat *ruta.kml* el guardarem dins del emulador del dispositiu com *kml*.

Sessió per defecte de GTalk

Per tal de poder fer ús dels serveis que proporciona *Google* en la plataforma Android, s'haurà d'introduir la informació de l'usuari tant si estem utilitzant un dispositiu real com si fem ús de l'emulador. Actualment, la introducció del compte *Gmail* de l'usuari al sistema operatiu de l'emulador es troba a l'aplicació *Dev Tools*, dins de l'opció *XMPP Settings*. Un cop introduït ja es pot disposar de les aplicacions que fan ús del servei de missatgeria *GTalk* de *Google*.

Webcam com a càmera de l'emulador

Les últimes generacions de dispositius mòbils tenen com a una de les principals característiques una o dos càmeres integrades al dispositiu. La implementació d'aplicacions per aquests dispositius encara no està massa desenvolupada i no hi ha dubte que pot ser explotat. Malauradament, la versió actual de l'emulador no permet la incorporació d'una càmera web per a simular aquest entorn.

No obstant això, existeix una solució per tal de poder simular per codi una interacció semblant. De totes maneres, aquesta solució requereix una implementació diferent a la que seria necessària per a un dispositiu real, pel que es pot considerar com una pèrdua de temps.

La solució es basa en utilitzar una aplicació de broadcasting de webcam a través d'Internet. D'aquesta manera es poden comprimir les imatges obtingudes per una càmera web i enviar-les al dispositiu *Android* mitjançant la xarxa. El dispositiu hauria d'obrir un socket per tal de poder rebre-la i d'aquesta manera poder tractar les imatges com si fos una càmera real. Aquesta opció és vàlida per desenvolupar aplicacions que tinguin com a objectiu principal tractar les imatges capturades i donar un servei al usuari.

La introducció de la sincronització del vídeo de l'emulador amb una webcam seria una millora considerable per al desenvolupament d'aplicacions per sistemes mòbils.

Capítol 3

Primera Aplicació: Guia d'Oci de Barcelona

Barcelona és una ciutat cosmopolita on cada dia apareixen locals d'oci nous, que encara es troben per explorar. És, a més a més, una de les ciutats més modernes d'Europa amb unes inversions molt grans en tecnologia i uns ciutadans molt exigents. Per tot això, l'aplicació desenvolupada vol omplir un buit als serveis de la ciutat, una forma en què els usuaris a través de la posició on es trobin puguin escollir el local on anar. Actualment, existeixen aplicacions web que ajuden a l'elecció dels locals segons una paràmetres de cerca determinat. Tanmateix, la integració de dispositius GPS en els mòbils pot incrementar el valor aportat per aquests serveis, ja que és un acte habitual que un ciutadà es trobi en una part de la ciutat quan se l'hi presenta l'ocasió de gaudir d'algun lloc d'oci. Quants cops hi ha usuaris que no gaudeixen dels serveis de la ciutat per què desconeixen totalment el lloc on es troben?

L'aplicació *OcioBCN* vol omplir aquest buit per tal de millorar la qualitat de vida dels barcelonins.

3.1 Objectius

El principal objectiu d'aquesta aplicació és analitzar les dificultats que podria presentar el desenvolupament d'una aplicació per un dispositiu mòbil Android, tot analitzant algunes de les diferents APIs que ens proporciona la plataforma.

Al ser una aplicació en la qual la informació va canviant en curts períodes de temps, es va decidir tenir un punt central d'informació a partir del qual s'actualitzés la informació necessària al dispositiu mòbil. Per aconseguir-ho s'ha fet ús d'una aplicació web en *PHP*, la qual oferirà uns mètodes remots als dispositius per actualitzar les dades de l'aplicació. A més a més, com actualment la tarifa de dades que ofereixen les operadores mòbils encara és costosa, es donarà també l'opció de descarregar un arxiu en format *XML* de la web, veieu la figura 3.1. En una situació real, aquesta aplicació web podria estar actualitzada pels mateixos llocs d'oci per tal de mantenir la publicitat del local.

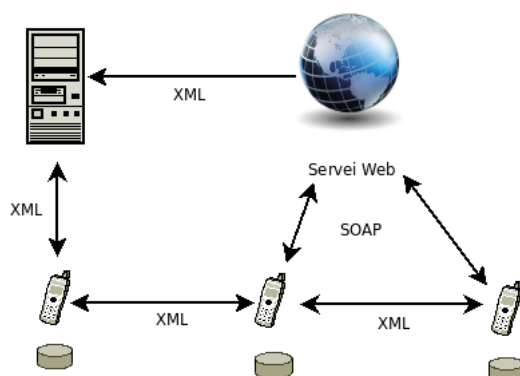


Figura 3.1: Actualització de l'aplicació

La gestió de la posició el dispositiu com l'administració de les dades obtingudes del servei web són dos dels objectius prioritaris. Tanmateix, l'aplicació també posa a prova la reproducció de vídeo i àudio a través de la retransmissió de tràilers en el dispositiu. D'altra banda, s'han incorporat una sèrie de components visuals per gaudir d'una experiència d'usuari el més còmode possible.

3.2 Especificació i Disseny

El disseny de l'aplicació no ha seguit les guies que un enginyer seguiria en un projecte de desenvolupament real. Això ha estat degut a que l'objectiu principal del projecte no és el desenvolupament d'una aplicació, sinó el de experimentar totes les possibilitats que proporciona la plataforma. D'aquesta manera les aplicacions del projecte han anat variant segons els inconvenients, les solucions que s'anaven trobant i el temps esperat de desenvolupament.

D'altra banda, es tracta d'un projecte senzill pel que el desenvolupament de totes les capes de disseny seria innecessari. No obstant això, la creació de la capa de dades en l'especificació ha sigut necessària pel desenvolupament, ja que ha servit per a generar les taules de la base de dades. La gestió d'aquestes taules per mitjà de les eines d'*Android* forma un dels objectius principals d'aquesta primera aplicació. A més a més, el treballar amb una capa de dades d'inici fa que les taules no hagin patit moltes variacions a mesura que avançava el projecte. Tanmateix, la introducció d'algunes de les noves característiques de l'aplicació han originat la introducció d'alguns dels atributs dels objectes, que posteriorment han esdevingut camps a les taules de la base de dades. A la figura 3.2, es pot observar el diagrama de classes del disseny de la capa de dades.

OcioBCN és un programa que s'encarregarà d'ajudar als ciutadans de Barcelona a trobar locals del seu interès segons la seva situació a la ciutat. Per tal de fer això, l'aplicació subdivideix els locals en les següents opcions: *bars i restaurants, hotels, cinemas i locals d'oci*

D'aquesta manera l'usuari podrà optar per veure tots els locals disponibles al seu voltant

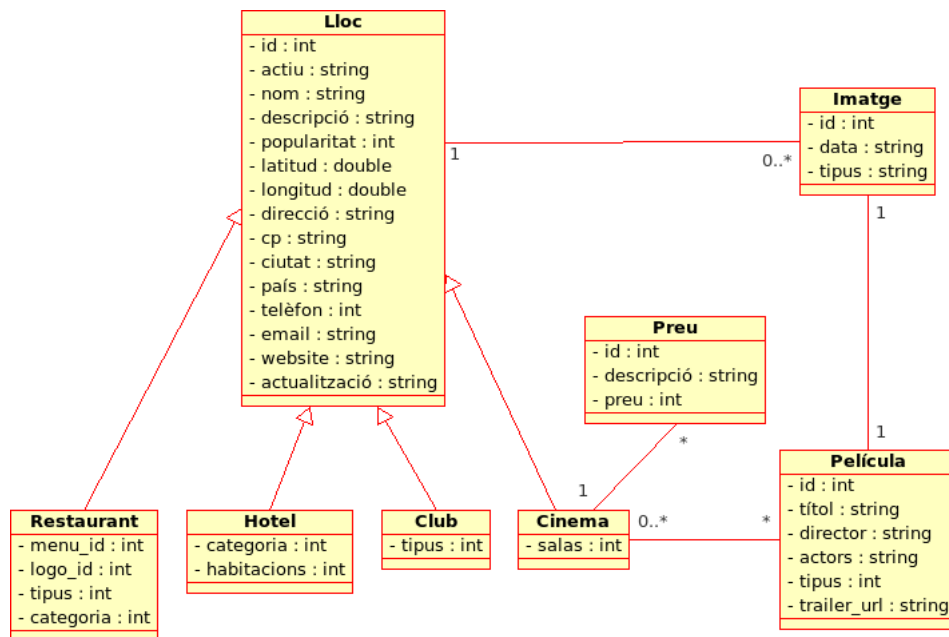


Figura 3.2: Capa de dades

a través d'un mapa o especificar quin tipus de local és en el que està interessat. L'usuari disposarà d'un número determinat de filtres per tal de seleccionar els locals segons les seves característiques. Un cop seleccionat el local, l'aplicació proporciona una simulació del recorregut necessari en cotxe a partir de diverses posicions GPS, per tal de que l'usuari arribi de manera correcta. En un principi, aquesta ruta hauria de ser per a vianants, però degut a la impossibilitat d'obtenció de coordenades GPS de la ciutat, s'ha hagut d'utilitzar les posicions proporcionades per *Android* a través de l'API de *GoogleNav*.

A més a més, s'ha utilitzat una arquitectura client-servidor per a l'actualització dels dispositius. Aquesta arquitectura dona l'avantatge de poder generar tots els processos complexos en la part servidora, per tal de que el client només mostri la informació proporcionada pel servidor incrementant el temps de resposta del dispositiu. Per tant, aquesta aplicació podria haver optat per consultar tota la informació remotament sobre un servidor allotjat en algun domini d'Internet. Això té avantatges però un principal inconvenient: la utilització de l'aplicació suposaria tenir la connexió a Internet constantment activa, cosa que com hem comentat anteriorment seria costós per a l'usuari.

En aquesta aplicació, la part servidora disposarà de tota informació actualitzada dels locals de Barcelona i es situarà com punt central d'informació. El client, d'altra banda, només haurà de demanar l'actualització dels tipus de locals en els que estigui interessat l'usuari per tal de disposar de la informació que l'usuari necessiti. D'aquesta manera un dispositiu mòbil totalment actualitzat, disposaria de la mateixa informació de la qual disposa el servidor. Una altra de les raons per les quals s'ha decidit que el dispositiu disposi de les dades en local és la utilització d'*Android* com a gestor d'informació.

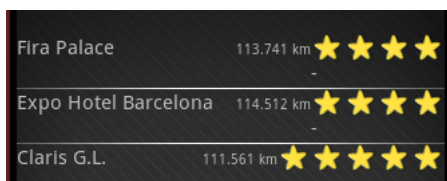


Figura 3.3: Adapters

3.3 Implementació

En qualsevol desenvolupament d'aplicacions, la part de implementació seria una part secundària a analitzar, ja que el fi no és la manera en la què es realitza l'aplicació sinó els objectius que es compleixen. L'objectiu principal d'aquest projecte és l'anàlisi de l'entorn de desenvolupament pel que aquesta part cobrarà un pes molt important dins del projecte.

En aquesta secció s'explicaran les característiques de la plataforma *Android* que han estat provades en la present aplicació i es discutiran els problemes trobats. A més a més, s'indicarà el treball d'investigació que s'ha seguit per tal d'arribar al resultat final.

3.3.1 Estructura de paquets

Aquest projecte ha estat estructurat en diferents paquets segons la funcionalitat que implementen. La majoria dels paquets parteixen de l'arrel *com.google.android.ocioBCN*, on es troben cadascuna de les pantalles principals de l'aplicació. Malgrat a ser també pantalles de l'aplicació, les *SubActivities* que mostren les pantalles amb els diferents filtres de l'aplicació es troben en el paquet *com.google.android.ocioBCN.filters*. Això és degut a que retornen un resultat a les pantalles principal. S'utilitzen per definir quin és el tipus de local que es vol trobar.

D'altra banda, en el paquet *com.google.android.ocioBCN.adapters* es troben les classes que serveixen per decorar alguns dels components de l'aplicació. Un exemple d'aquests adaptadors s'observa a la figura 3.3, on s'introdueixen imatges d'estrelles segons les categories dels hotels a la llista mostrada per pantalla. A més a més, existeix un paquet anomenat *com.google.android.ocioBCN.utils* on estan les classes que contenen mètodes estàtics que realitzen alguns dels mètodes estàndard de l'aplicació.

Les classes utilitzades per a gestionar la capa de dades es troben sota *com.google.android.ocioBCN.persistence*, encara que la classe *Updater.java* d'aquest paquet fa ús del paquet *xml* i *ws*, segons el tipus d'actualització que faci servir. Tots els objectes que s'utilitzen per a tractar el document *XML* de l'aplicació es troben al paquet *com.google.android.ocioBCN.xml*. D'altra banda, a *com.google.android.ws* es troben les classes auxiliars per accedir als mètodes de serveis web. Aquest paquet no es troba dins de l'arrel de l'aplicació, ja que són unes classes estàndard per a facilitar l'accés a qualsevol servei web en *SOAP*.

Els components gràfics d'un mapa han estat separats en un altre paquet de l'aplicació,

anomenat *com.google.android.ocioBCN.mapsUtils*. La major part d'aquests components són les capes que es dibuixen per sobre del mapa, anomenades *Overlay*.

Finalment, existeix un paquet anomenat *com.google.android.utils* que conté una sèrie de components que s'utilitzen a l'aplicació, però que podrien fer-se servir per altres aplicacions.

3.3.2 Gestió de dades amb un Content Provider i SQLite3

Android dóna la facilitat de gestionar les dades de les bases de dades de dues maneres: connexió directa a base de dades i gestor de continguts. La primera consisteix en realitzar una connexió directa a la base de dades a la qual l'aplicació tingui permisos i executar aquí totes les instruccions necessàries, tal i com es faria en una aplicació *Java* qualsevol. A més a més, proveïx una classe per a la construcció de sentències SQL correctes, per tal d'ajudar al desenvolupador a fer que l'aplicació obtingui o modifiqui les dades que cregui convenient.

D'altra banda, *Android* proporciona els gestors de continguts o *Content Providers*, que són l'opció recomenada per la plataforma. Aquest element de la arquitectura *Android*, a més de facilitar compartir dades entre aplicacions (de fet, és l'única manera de fer-ho), separa la capa de persistència de dades de les altres capes.

Per tal de gestionar les dades amb aquests gestors de continguts, primer de tot s'ha de generar una classe estàtica derivada de *SQLiteOpenHelper* que s'encarregarà de crear i actualitzar la nostra base de dades. Les diferents taules es generaran segons el model de la capa de dades dissenyat en la part de l'especificació encara que, per qüestions d'eficiència, no generarà una taula amb les dades comunes dels llocs.

En la figura 3.4 s'observa la manera com es generen les taules de la base de dades a partir del *Content Provider*, en el mètode *onCreate*.

D'altra banda, com que el número de taules ha anat augmentant a mesura que l'aplicació evolucionava, es va decidir implementar el mètode *onUpgrade* d'aquesta classe, per tal de que al fer un canvi de versió de la taula, s'eliminïn les taules de la base de dades i es generin de nou. El problema de fer això és que s'eliminaran totes les dades de la base de dades sempre que s'actualitzi la versió de la base de dades. Tanmateix, en la nostra aplicació no té sentit fer sentències SQL *alter table*, ja que es disposa de tota la informació en un punt central del qual es podrien actualitzar les dades immediatament actualitzar. Podem observar el codi final d'aquest mètode en la figura 3.5

Amb la classe de generació i actualització de la base de dades acabada, el següent pas és la creació d'una classe, derivada de la classe *ContentProvider*, per tal de realitzar les operacions que necessitem que estiguin disponibles, tant per la nostra aplicació, com per aplicacions desenvolupades posteriorment. Cadascuna d'aquestes operacions tindran associat una URI¹ per tal de poder executar-la correctament. D'aquesta manera, s'obté una facilitat d'accés sobre la base de dades molt interessant, sense necessitat d'haver de repetir un i un altre cop la mateixa sentència SQL.

¹Una URI és una cadena curta de caràcters que identifica inequívocament un recurs.

```

public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE restaurantes (_id INTEGER PRIMARY KEY,"
        + " active boolean,"
        + " name TEXT NOT NULL,"
        + " desc TEXT,"
        + " menu_id INTEGER,"
        + " logo_id INTEGER,"
        + " type INTEGER,"
        + " cat INTEGER,"
        + " popularity INTEGER,"
        + " street varchar(255),"
        + " cp varchar(10),"
        + " city varchar(50),"
        + " country varchar(50),"
        + " phone varchar(9),"
        + " email varchar(50),"
        + " website varchar(50),"
        + " updated INTEGER,"
        + " latitude INTEGER,"
        + " longitude INTEGER);");

    db.execSQL...
}

```

Figura 3.4: Mètode onCreate

```

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
        + newVersion + ", which will destroy all old data");
    db.execSQL("DROP TABLE IF EXISTS restaurantes");
    db.execSQL("DROP TABLE IF EXISTS imagenes");
    db.execSQL("DROP TABLE IF EXISTS dishes");
    db.execSQL("DROP TABLE IF EXISTS hotels");
    db.execSQL("DROP TABLE IF EXISTS clubs");
    db.execSQL("DROP TABLE IF EXISTS cinemas");
    db.execSQL("DROP TABLE IF EXISTS films");
    db.execSQL("DROP TABLE IF EXISTS cinefilm");
    db.execSQL("DROP TABLE IF EXISTS cineprice");
    onCreate(db);
}

```

Figura 3.5: Mètode onUpgrade

```

RestaurantObject elem = it.next();
elem.UpdatePosition();

int id = elem.getId();
// primer comprovem si existeix el Restaurant amb el següent id
Uri URI = Uri.withAppendedPath(
    com.google.android.ocioBCN.persistence.Restaurantes.Restaurant.RESID_URI,
    Integer.toString(id));

Cursor item = parent.managedQuery(
    URI,
    Restaurantes.RESTAURANT_PROJECTION,
    null, null);

```

Figura 3.6: Utilització de la URI

Les operacions que s'han d'implementar d'una interfície de *Content Provider*, són les operacions que substitueixen les típiques accions que s'executen directament sobre una base de dades: *select*, *insert*, *update* i *delete*. La figura 3.7 és una part de la funció *query* que executa la sentència *select*. En aquesta s'observa l'ús del condicional *switch* per a separar les diferents operacions implementades mitjançant les esmentades URIs, l'ús de les qual serà explicat més detalladament.

Cada taula de la base de dades té una relació directa amb una classe de l'aplicació. Per aquesta raó s'ha definit en cadascuna d'aquestes classes les diferents URIs com a variables constants, implementades amb cadenes de text acabades en interrogant. Per executar diferents accions sobre aquestes taules s'ha d'indicar: quin tipus d'operació és, quina és la URI que executa l'acció desitjada i una projecció de la classe on s'indiquen els camps de la taula que s'obtiniran. En cas de necessitar-ho, les URIs han d'afegir al final de la cadena els paràmetres que necessitin, separats pel símbol “&”. La figura 3.6 serveix d'exemple per escenificar la manera d'obtenir un element determinat de la base de dades sense implicar cap tipus de sentència SQL.

Android ofereix la classe *Uri* per manipular cadenes de text en aquest format determinat d'una manera fàcil i eficient. D'altra banda, per tal de poder classificar-les dins del *Content Provider*, es disposa de la classe *UriMatcher*, què, en conjunció d'una sèrie d'expressions regulars, les hi proporciona una constant identificadora, definida pel desenvolupador.

En la figura 3.8, s'observa l'anàlisi, mitjançant l'*UriMatcher*, d'una mateixa URI a la qual se li assignan dos constants diferents, *RESTAURANT* i *RESTAURANT_NAME*, depenent si aquesta URI ve acompanyada de paràmetres o no. A més a més, s'ha de tenir en compte que aquesta assignació és fa en ordre d'addició de URIs, pel qual a una URI que compleixi dos tipus de regles diferents se li assignarà la primera que estigui definida.

En teoria, la complexitat dels paràmetres no hauria d'estar limitada amb l'ús de URIs,

```

public Cursor query(Uri uri, String[] projection, String selection,
    String[] selectionArgs, String sort) {
    // Query the database using the arguments provided
    SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
    String[] whereArgs = null;
    String orderBy = "";
    List<String> elems;
    //Depenen de la URI seleccionarem la taula que volem recórrer
    int option = URLMATCHER.match(uri);
    switch (option)
    {
    case RESTAURANT_ID:case RESTAURANT:case RESTAURANT_DESC_SEARCH:
    case RESTAURANT_NAME:case RESTAURANT_WHERE_SEARCH:
        qb.setTables("restaurantes");
        switch(option){
        case RESTAURANT_ID:
            qb.appendWhere("_id=?");
            whereArgs = new String[] {uri.getLastPathSegment()};
            break;
        case RESTAURANT_DESC_SEARCH:
            qb.appendWhere("desc_like_?");
            whereArgs = new String[] {"%" + uri.getLastPathSegment() + "%"};
            break;
        case RESTAURANT_NAME:
            qb.appendWhere("name=?");
            whereArgs = new String[] {uri.getLastPathSegment()};
            break;
        case RESTAURANT_WHERE_SEARCH:
            elems = uri.getPathSegments();
            qb.appendWhere(elems.get(2));
            if(elems.size()==4)
                whereArgs = new String[]
                    {"%" + elems.get(3) + "%"};
            if(elems.size()==5)
                whereArgs = new String[]
                    {"%" + elems.get(3) + "%", "%" + elems.get(4) + "%"};
            break;
        }
        break;
    (...)
    default:
        throw new IllegalArgumentException("Unknown_URL_" + uri);
    }

    // Es mira si s'ha expressat algun tipus d'ordenació
    if (TextUtils.isEmpty(sort) && TextUtils.isEmpty(orderBy)) {
        orderBy = Restaurantes.Restaurant.DEFAULT_SORT_ORDER;
    } else {
        orderBy = sort;
    }
    // S'executa la query
    Cursor c = qb.query(mDb, projection, null, whereArgs, null, null, orderBy);
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}

```

Figura 3.7: Operacions SQL

```
URLMATCHER = new UriMatcher(UriMatcher.NO_MATCH);
URLMATCHER.addURI(Restaurantes.RESTAURANT_AUTHORITY,
    "restaurantes", RESTAURANT);
URLMATCHER.addURI(Restaurantes.RESTAURANT_AUTHORITY,
    "restaurantes/*", RESTAURANT_NAME);
```

Figura 3.8: Utilització de UriMatcher

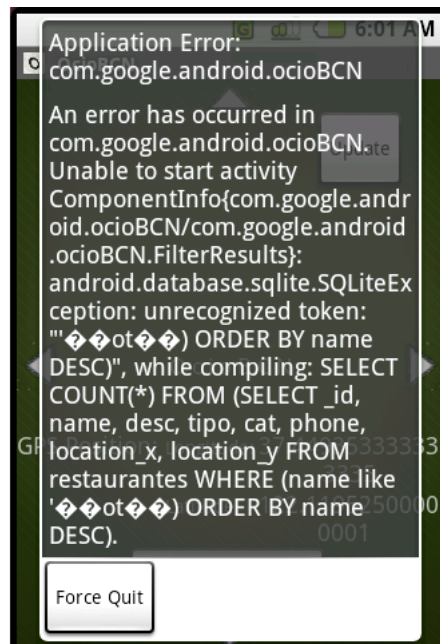


Figura 3.9: Error per introducció de caràcters especials a la sentència SQL

cosa que faria que es pogués executar qualsevol tipus de sentència. Per contrari, existeix actualment algun defecte en el pas d'algun paràmetre com es pot observar a la figura 3.9. Aquest error, es va produir per la introducció del caràcter especial “%” en la sentència per a efectuar una cerca dins d'un camp de text. Aquest caràcter sembla que no està correctament analitzat en la classe Uri, pel qual es va haver de solucionar introduint aquest caràcter especial un cop la URI ja ha estat manipulada. Aquest tractament es veu detallat a la figura 3.10.

Un cop implementat aquest proveïdor de dades propi, s'haurà de dotar a l'aplicació dels permisos per poder utilitzar-lo a l'*AndroidManifest.xml*. Això es faria especificant els proveïdors que l'aplicació pot consultar de la següent manera:

Sense aquest permís, les aplicacions que estiguessin interessades en la utilització de les dades guardades en aquest proveïdor no podrien tenir-hi accés, els hi quedaria denegat.

```

case RESTAURANT_WHERESEARCH:
    elems = uri.getPathSegments();
    qb.appendWhere(elems.get(2));
    if(elems.size()==4)
        whereArgs = new String []
            {"%" + elems.get(3) + "%"};
    if(elems.size()==5)
        whereArgs = new String []
            {"%" + elems.get(3) + "%", "%" + elems.get(4) + "%"};
    break;

```

Figura 3.10: Introducció de símbol especial %

```

<provider
    android:name=".persistence.DataProvider"
    android:enabled="true"
    android:authorities="com.google.android.ocioBCN.persistence.DataProvider" />

```

Figura 3.11: Permisos al Android Manifest

3.3.3 Geolocalització

Una de les característiques d'*Android* amb major projecció és la capacitat de localitzar la situació del dispositiu mòbil i poder oferir fàcilment una sèrie de serveis a l'usuari. La utilització de mapes i del servei de geolocalització formen una de les parts d'implementació més importants d'aquesta aplicació.

Les aplicacions que facin ús d'aquest servei necessitaran demanar una sèrie de permisos, que han de ser posteriorment acceptats per l'usuari. Actualment, en la versió M5 de l'entorn de desenvolupament, els permisos necessaris són els d'*android.permission.ACCESS_LOCATION* i *android.permission.ACCESS_GPS*, encara que això podria variar en les versions posteriors de l'entorn, tal i com ho ha fet fins el moment. Aquests permisos són assignats a través de l'*AndroidManifest.xml* i acceptats posteriorment per l'usuari al instal·lar l'aplicació. Per tal de cercar la posició de l'usuari, el primer intent és utilitzar un sistema hardware de GPS, integrat o no al dispositiu mòbil. Si no es disposara d'aquest hardware, que és el més exacte de tots, *Android* provaria de localitzar l'usuari a través d'antenes wireless i, finalment, en cas de no poder donar aquest servei, el demanaria a l'operadora pagant els serveis de connexió. Aquests dos mètodes apliquen un mètode de triangulació entre les antenes més properes, cosa que dona una posició més o menys exacta de la posició actual de l'usuari. La introducció de noves antenes produiria un increment de la precisió d'aquest servei.

La Geolocalització és un servei ofert pel sistema operatiu d'*Android*. Per tal d'usar-lo les aplicacions s'hi s'han d'enregistrar. Com l'aplicació desenvolupada treballa constantment amb la posició del dispositiu, es va decidir implementar una classe estàtica que s'associa a aquest servei a l'iniciar l'aplicació.

```

try
{
    LocationManager myLocationManager =
        (LocationManager) getSystemService(LOCATION_SERVICE);
    MapUtils.setLocationManager(myLocationManager);
    gpsLocation = MapUtils.getGPSLocation();
    Log.d("GPSLocation", "("+gpsLocation.getLatitude()+",
    +gpsLocation.getLongitude()+")");
    MapUtils.setGPSEnabled(true);
}
catch(Exception e){
    showAlert("GPS_error", R.drawable.icon ,
        "The_application_could_not_start_normally." +
        "All_GPS_function_disabled",
        "ok", false);
    MapUtils.setGPSEnabled(false);
}

```

Figura 3.12: Inicialització del servei de geolocalització a l'aplicació

Aquesta classe estàtica s'anomena *MapUtils* i s'utilitzarà en cas de necessitar obtenir la posició del dispositiu. En el codi de la figura 3.12, s'observa la manera en que s'assigna el manegador del servei a la classe, per tal d'utilitzar el servei de Geolocalització. Tal i com s'ha explicat a la secció 2.2.5, per tal de desenvolupar una aplicació que utilitza geolocalització s'utilitza una simulació de ruta GPS. La posició actual de la simulació s'obté cridant al mètode *myLocationManager.getCurrentLocation("nom_del_proveïdor")*; on aquest nom serà el nom del directori on s'ha guardat la simulació.

Google Maps

Google Maps és l'eina de mapes que proporciona Android. Aquests mapes estan disponibles online, pel que cada cop que es demani una actualització del mapa es necessitarà d'una connexió d'Internet activa. En cas de no estar disponible aquesta connexió, el mapa no es podrà descarregar correctament i, per tant, no serà visualitzat.

Aquesta eina proporciona les classes *MapView* i *MapActivity*. *MapActivity* és una classe dissenyada per implementar tots els mètodes rutinaris a l'hora d'utilitzar un mapa a la interfície gràfica del dispositiu. Per tant, aquestes dues classes estan lligades entre elles.

D'altra banda, per tal de dibuixar elements per damunt del mapa, no utilitzarem cap de les anteriors classes. Aquesta capa de valor afegit vindrà representada per la classe *Overlay* i estarà associada al mapa. El codi de la figura 3.13 associa un *Overlay* creat per nosaltres a un objecte del tipus *MapView* a través del seu controlador d'*Overlays*.

Cada cop que aquest mapa es torni a dibuixar, el controlador indicarà a l'*Overlay*

```

final MyMapView mapa = (MyMapView) findViewById(R.id.mapgps_view);
MC = mapa.getController();
MC.zoomTo(15);
MapPoint current = MapUtils.getGPSLocation();
Point p = new Point(current.getLatitude(), current.getLongitude());
/* Per tal de mouren's en el mapa, utilitzem la óposici central */
MC.centerMapTo(p, true);
/* Demanar el controlador de overlays, OverlayController, del mapa. */
OverlayController myOC = mapa.createOverlayController();
/* Afegir l'Overlay que hem creat */
myOC.add(new GPSOverlay(this), true);

```

Figura 3.13: Associació d'un Overlay propi a un mapa

que s'ha de repintar. Per això és necessari implementar el mètode *draw()* de la classe generada per tal de que dibuixi els elements necessaris en aquesta capa 2D. Aquest exemple representa el dibuix del camí entre dos punts, el qual es representarà mitjançant dues imatges en format *jpeg*, indicant el punt de sortida i el d'arribada, i un conjunt de línies entre els punt GPS de la ruta. Els objectes *Overlay* es trobaran explicats amb més detall en el capítol corresponent a la segona aplicació del projecte.

Cal esmentar que el component *MapView* té un petit defecte: no detecta els esdeveniments del ratolí. Això és degut a que els mapes ja incorporen un tractament per defecte d'aquests esdeveniments, per tal de moure's sobre el mapa o utilitzar l'eina de zoom que ve integrada. Tanmateix, per tal de poder obtenir les posicions GPS del mapa amb el esdeveniment *OnClick*, es va desenvolupar un nou component anomenat *myMapView*, que desactiva els esdeveniments per defecte del mapa per poder tractar-los de manera més adient a l'aplicació. Aquest nou component es deriva de la classe *MapView* i re-implementa el mètode de pulsació sobre la pantalla, assignant-li un listener per a la notificació a les capes superiors. D'aquesta manera, s'observa la facilitat de re-utilització de components oferida per aquest sistema.

GoogleNav API

GoogleNav és una API de l'entorn de desenvolupament d'*Android* que es troba totalment indocumentada. És a dir, no se sap que passarà amb ella, si es troba en fase de proves o simplement volen anar migrant-la cap a una altre. Aquesta API en un principi, en la versió M3 SDK, contenia classes les quals han passat a formar part de l'API *GmmGeocoder*, que també es troba indocumentada.

Tanmateix, s'ha decidit utilitzar-la degut a que facilitava la classe *DrivingDirection* que calcula la ruta més curta entre dos punts utilitzant un vehicle a motor. A més a més, proporciona una enumeració dels moviments que s'haurien de fer per tal d'arribar-hi, per tal de poder indicar a l'usuari quin és el següent moviment a seguir.

Al codi de la figura 3.15, s'observa que la classe que s'encarrega de la generació de rutes,

```

public void draw(Canvas canvas , PixelCalculator pxC, boolean b) {

    super.draw(canvas , pxC, b);
    (...)
    /* Resetejem l'estat del nostre objecte Paint. */
    this.pathPaint.setStrokeWidth(4);
    this.pathPaint.setARGB(100, 113, 105, 252);
    int [] screenCoords = new int [2];
    dd = itsMapActivity.getDrivingDirections();
    MapPoint startPoint , endPoint;

    if(itsMapActivity.getStart() != null)
    {
        startPoint = itsMapActivity.getStart();
        MapPointToScreenCoords(startPoint , screenCoords , pxC);
        /* Dibuixem el punt inicial de la ruta */
        canvas.drawBitmap(PIN_START, screenCoords[0] - PIN_HOTSPOT.x,
                          screenCoords[1] - PIN_HOTSPOT.y, pathPaint);
    }
    if(itsMapActivity.getEnd() != null)
    {
        endPoint = itsMapActivity.getEnd();
        MapPointToScreenCoords(endPoint , screenCoords , pxC);
        /* Dibuixem el punt final de la ruta */
        canvas.drawBitmap(PIN_END, screenCoords[0] - PIN_HOTSPOT.x,
                          screenCoords[1] - PIN_HOTSPOT.y, pathPaint);
    }

    if (dd != null) {
        startPoint = dd.getStartPoint();
        endPoint = dd.getEndPoint();
        MapPointToScreenCoords(startPoint , screenCoords , pxC);
        Path thePath = new Path();
        thePath.moveTo(screenCoords[0] , screenCoords[1]);
        if (!dd.routeTooLong()) {
            MapPoint [] route = dd.getRoute();
            if (route == null) return;
            for (MapPoint current : route) {
                if (current != null) {
                    MapPointToScreenCoords(current , screenCoords , pxC);
                    /* Dibuixem la linea entre dos punts de la ruta */
                    thePath.lineTo(screenCoords[0] , screenCoords[1]);
                }
            }
        }
    }
    (...)
}
itsMapActivity.setWorking(false);
}

```

Figura 3.14: Mètode onDraw del objecte Overlay

```

private void startFetchDirections() {
    while(isWorking())
    {
        try {
            Thread.sleep(500);
        }
        catch (InterruptedException e) {
            Log.e("DEBUGTAG", "Overlay draw was interrupted while working", e);
        }
    }
    setFoundDirections(false);
    /* Introduim les adreces d'inici i de final.
       Son variables globals degut a que la ódirecci d'inici
       va variant segons es mou l'usuari */
    this.myDD = new DrivingDirection(start, "", end, "");
    if (this.myDD != null)
    {
        //Es dispara una ópetici i s'espera que estigui completa
        this.getDispatcher().addDataRequest(this.myDD);
        while (!GPSInteractive.this.myDD.isComplete()) {
            try {
                Thread.sleep(100);
            }
            catch (InterruptedException e) {
                Log.e("DEBUGTAG",
                    "'!MyDrivingDirectionsActivity.this.myDD.isComplete()' was interrupted.",
                    e);
            }
        }
        GPSInteractive.this.foundDirections = true;
    }
}
}

```

Figura 3.15: Mètode per aconseguir la ruta

```

Thread timer = new Thread(new Runnable() {
    public void run() {
        while (!isFinished())
        {
            setStart(MapUtils.getGPSLocation());
            startFetchDirections();
            Point point = new Point(start.getLatitude(), start.getLongitude());
            mapa.getController().centerMapTo(point, false);
            try {
                Thread.sleep(8000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
});

```

Figura 3.16: Actualització de la posició d'un usuari cada cert temps

anomenada *GPSInteractive*, només necessita la introducció de les adreces d'inici i de fi a la classe *DrivingDirection* per efectuar el càlcul dels diferents punts GPS de la ruta. Un cop els càlculs han estat executats, s'utilitza el mètode *getRoute()* d'aquesta classe, per obtenir tots els punts GPS de la ruta, els quals es dibuixaran en una capa superior del mapa, anomenada *Overlay*. D'altra banda el mètode *getTurns()* dona l'informació de la ruta en text pla, per tal de poder representar-ho per pantalla. Aquest mètode no ha estat utilitzat a l'aplicació encara que s'ha experimentat sobre els resultats retornats per tal de provar el seu correcte funcionament.

En un principi, l'aplicació volia obtenir un GPS de rutes a peu, però finalment s'ha decidit utilitzar rutes amb vehicle a motor. No s'ha trobat cap manera de poder implementar les rutes a peu, aquesta API no ofereix aquest servei de rutes. A més a més, es va intentar generar les rutes manualment a partir dels carrers de Barcelona, però també a resultat impossible obtenir les cruïlles d'aquesta ciutat. En aquesta simulació de dispositiu GPS, s'utilitza un fil, com s'observa a la figura 3.16, per tal d'obtenir la posició del usuari cada cert temps, ja que aquesta va canviant.

El temps d'espera entre càlculs podria augmentar-se o disminuir-se en funció de com funcionés en un dispositiu real, ja que s'ha d'esperar a que finalitzi el càlcul de la ruta anterior abans de calcular la nova ruta.

D'altra banda, la classe *GPSInteractive* es podria haver registrat al mateix servei de geolocalització per saber quan un usuari canvia de posició. El funcionament seria similar, encara que d'aquesta manera no faria falta crear un fil que anés actualitzant la posició. L'únic inconvenient és cada cop que es restauri o es pari l'aplicació s'hauria d'anar enregistrant i desenregistrant al servei de nou. Per tal de registrar aquesta classe, s'hauria de crear un *IntentFilter* propi per l'aplicació, per tal de fer que el servei

```
myLocationManager.requestUpdates(provider ,
    MINIMUM_TIME_BETWEEN_UPDATE,
    MINIMUM_DISTANCE_CHANGE_FOR_UPDATE,
    new Intent(MY_LOCATION_CHANGED_ACTION));
```

Figura 3.17: Manera d'obtenir el canvi de posició d'un usuari

```
class MyIntentReceiver extends IntentReceiver {
    @Override
    public void onReceiveIntent(Context context , Intent intent){
        /**
         * Executar les accions dessitjades per a tractar l'Intent
         */
    }
}
```

Figura 3.18: Estructura bàsica d'un IntentFilter

actualitzés la posició. En la figura 3.17 s'observa com es pot configurar l'aplicació per tal de que es rebin actualitzacions de la posició de l'usuari, tant per límit de temps com per distància recorreguda.

En el capítol següent, es detallarà la segona aplicació la qual fa ús d'aquests *IntentFilters*. Tot i això, en la figura 3.18 s'observa l'estructura bàsica d'un *IntentReceiver* que és l'encarregat de respondre a la recepció de l'Intent que es detalla en el filtre.

Aquesta API ha estat trobada gràcies al fòrum de desenvolupament d'*Android* a Espanya, *Android-Spa*[2], on hi ha una demostració de simulació de camins, la qual trobava el camí entre dos punts predeterminats. Aquesta ha sigut la demostració que s'ha utilitzat com a base per a crear la simulació de GPS interactiu de l'aplicació.

3.3.4 Utilització de la Media API

La utilització del reproductor multimèdia és molt senzilla i ha estat detallada en l'anterior capítol. Igualment es presentarà la utilització concreta que s'ha fet en l'aplicació i els problemes esdevinguts en el seu ús.

Streaming d'àudio i de vídeo

Tal i com s'ha indicat al capítol anterior, l'streaming d'àudio i de vídeo s'hauria de poder fer només indicant la adreça URL on s'allotja el contingut multimèdia. Aquest reproductor de vídeo streaming tindria més limitacions que el de vídeo normal, ja que, tal com s'ha esmentat en l'anterior capítol, només suporta dos formats: 3GP[52] i MP4[54].

```

mp = new MediaPlayer ();
//óInicialitzaci dels listeners del reproductor
mp.setOnBufferingUpdateListener(this);
mp.setOnCompletionListener(this);
mp.setOnPreparedListener(this);
mp.setAudioStreamType(AudioSystem.STREAMMUSIC);
//No es àrepetir constantment
mp.setLooping(0);

// Introduim quina àser la pantalla
mp.setDisplay(mPreview.getHolder().getSurface());

// Posem la URL del arxiu multimedia
Log.v(TAG, "Setting datasource to mediaplayer");
mp.setDataSource(bufFile);
mp.setDataSource(path);
// Per tal de reproduir streaming,
// s'ha de preparar íasncronament el reproductor.
mp.prepareAsync();
mp.start();

```

Figura 3.19: Ús d'streaming amb el MediaPlayer d'*Android*

El cas seria molt similar a l'anterior, canviant la ruta del arxiu en local per una adreça URL de la pàgina d'on reproduïrem els vídeos. L'aplicació està pensada per tenir un servidor de vídeos 3gp online amb els tràilers de les pel·lícules. De moment els vídeos son extrets del servidor <http://daily3gp.com/>, que disposa de vídeos 3gp amb descàrrega progressiva.

La figura *mediaplayer* és la implementació necessària teòrica per reproduir streaming en un dispositiu Android. Al contrari de l'esperat, aquesta utilització del reproductor de *MediaPlayer* dona un error indocumentat: *error=1;code=0*, el qual no especifica què és el que està fallant. La documentació proporcionada per *Google* no diu res al respecte, i el bug ha estat reportat per tal de que sigui solucionat a la següent versió del entorn de desenvolupament.

Aquest error ha fet que, de moment, es faci una simulació s'*streaming*. El que s'ha fet és guardar en un arxiu temporal les dades del vídeo que s'està descarregant per tal d'anar-lo reproduint. Com que aquest arxiu no es pot reproduir a mesura que es va utilitzant per guardar les seves propies dades, es fa ús d'un altre fitxer auxiliar que cada cop que es reproduïx totalment fa una còpia íntegra del anterior, conservant el moment de reproducció anterior per continuar reproduint-se.

```

Pattern phonePattern= Pattern.compile("\\b[0-9]+\\b");
Linkify.addLinks(phoneTV, phonePattern, "tel:");
//Linkify.addLinks(phoneTV, Linkify.PHONE NUMBERS);
Linkify.addLinks(urlTV, Linkify.WEB_URLS);
Linkify.addLinks(mailTV, Linkify.EMAIL_ADDRESSES);

```

Figura 3.20: Aplicació d'enllaços entre aplicacions Android

Problemes de reproducció

A part del problema comentat en l'anterior subsecció, trobem que en aquesta versió l'API encara es troba en un estat molt inestable. La reproducció d'àudio no és nítida i la de vídeo moltes vegades es queda penjada o simplement no funciona.

D'altra banda, es comenta que la reproducció d'àudio constant, és a dir, la simulació d'un reproductor multimèdia, fa que l'emulador es quedi penjat i no reproduï cap tipus d'arxiu multimèdia. Tots aquest problemes, fan que les proves sobre aquesta API no donin els resultats esperats, encara que al ser problemes d'execució es podria esperar a la següent versió de l'entorn per observar si s'han solucionat aquests problemes. A més a més, alguns dels problemes comentats poden ser causats per l'emulador, cosa que faria que es solucionessin en un dispositiu real.

3.3.5 Utilització de enllaços entre aplicacions (*Linkify*)

Una de les característiques normals dels mòbils és l'enllaç dels números de telèfon o dels correus electrònic a les aplicacions de comunicació del sistema operatiu. Aquesta característica, *Android* la proporciona mitjançant la classe *Linkify*². Tanmateix, aquesta classe ofereix la possibilitat d'enllaçar qualsevol tipus de text amb una activitat determinada.

Per tal de fer-ho s'utilitza la classe *Pattern* de Java, on determinarem l'expressió regular[25] adequada al text que es vol enllaçar. Tot text que correspongui a la definició de l'expressió regular estarà destacat i enllaçat amb l'esquema que s'indiqui. Com que el text seleccionat es passarà a la part posterior de la URI generada, es necessita adaptar-lo al paràmetre definit a l'*Activity* que s'encarregui de tractar-lo. Per tal de fer això disposem de la classe *TransformFilter* amb el mètode *transformUrl* que ens donarà la url definitiva que es passarà. D'aquesta manera, es pot per exemple posar que cada cop que surt la paraula *Google* en un text determinat s'enllaci amb el buscador per defecte de l'empresa: *www.google.com*.

D'altra banda, tal i com s'observa a la figura 3.20, la plataforma incorpora el tractament de correus electrònics, adreces url o qualsevol tipus de telèfon per defecte. Només s'ha d'especificar el component que volem que es tracti i quina característica és la que se li vol assignar. Per defecte *Android* incorpora les següents constants: *ALL*,

²La traducció al català de la paraula seria "Enllaçar"

EMAIL_ADDRESSES, *PHONE_NUMBERS* i *WEB_URLS* per definir els enllaços anteriorment comentats. Aquestes constants estan relacionades amb un patró per defecte associat al tipus de cadena cercada per a cada servei.

Aquestes expressions regulars per defecte, podrien no funcionar amb el text indicat a alguns components. En el cas dels números telefònics, *Android* no detecta una seqüència més llarga de 6 números seguits, la qual cosa provoca que s'hagi d'introduir un patró per enllaçar-los en una aplicació determinada. Per tal de paliar l'incorrecte funcionament per defecte dels patrons, s'ha d'introduir una expressió regular, tal com anteriorment s'ha descrit, associada amb un dels següents esquemes:

- Urls: “http:/" o “https:/"
- Números telefònics: “tel:”
- Mapes: “geo:”

El redreçament per enviar missatges a través de correus electrònics i de *SMS* encara no està suportat, encara que qualsevol aplicació que es desenvolupés per tal de complir aquest requeriment podria estar enllaçada a aquests.

Cal esmentar que aquests enllaços poden cridar a qualsevol aplicació del sistema. Per tal de fer-ho, la classe estàtica *Linkify* segueix una sèrie de passos a l'hora d'enllaçar un patró de text amb una *Activity* determinada del sistema:

1. Analitza els texts que troba dins dels components seleccionats i els transforma en enllaços composts per la URL indicada i el text seleccionat. Aquest text podrà ser transformat a través d'una classe anomenada *TransformFilter*, per tal de passar el paràmetre correcte amb la URL.
2. *Linkify* monitoritza quan un usuari fa un clic sobre un determinat enllaç.
3. Forma un *Intent* amb l'acció *VIEW* i la URI definida per el desenvolupador.
4. Busca un *Content Provider* dins del sistema que respongui a aquest tipus d'URI per obtenir més informació
5. Si té definit un *Content Provider*, torna l'objecte definit pel tipus de *MIME* i paràmetres utilitzats
6. Busca una activitat que tingui aquest *MIME* i l'acció *VIEW* definits.
7. Es llença l'*Activity* seleccionada.

Per tant, s'ha d'especificar dins del manifest d'una aplicació, les diferents URIs que són capaces de tractar les seves activitats. Un exemple d'això es troba a la figura 3.21, on es pot observar com es pot indicar al sistema quin tipus de dades són tractades per una activitat determinada. En aquest cas concret, l'activitat *MenuInfo* tracta la informació dels menús de restaurant.

```

<activity android:name="MenuInfo">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="vnd.android.cursor.item/menu"/>
  </intent-filter>
</activity>

```

Figura 3.21: Configuració d'Activity per llençar un MIME

El text el qual estaria associat a aquest tipus de *MIME* hauria de ser un identificador de menú únic de restaurant. Amb aquest identificador es cercaria tota la informació necessària al *Content Provider* definit i es llençaria l'activitat de *MenuInfo*.

3.3.6 Serveis Web

Els serveis web van sorgir davant de la necessitat d'estandarditzar la comunicació entre diferents plataformes i llenguatges de programació. Els primers intents de crear estàndards va ser la creació de *DCOM*[53] i *CORBA*[33], el primer dels quals no van tenir suficient èxit a causa de no ser independent de la implementació del venedor, *DCOM* és un protocol propi de Microsoft. Un altre problema era que feien ús de *RPC*³[55] per realitzar les comunicacions entre diferent nodes, cosa que fa que siguin poc freqüents en Internet. *SOAP*[49] va ser el primer protocol estàndard amb èxit al mercat. Aquest estàndard es va començar a crear al 1999, i a és un dels protocol més utilitzats en l'actualitat per fer serveis web. Cal observar que les carències en seguretat de *SOAP* fan que *CORBA* sigui més utilitzat en intranets d'empreses. A més a més *CORBA* és més ràpid.

L'accés a serveis web des d'*Android* és una de les parts importants de l'aplicació, ja que d'aquesta manera es pot provar que una aplicació de la plataforma pot funcionar com una eina client de grans aplicacions, on els serveis web són una de les tecnologies que està cobrant més força. En un principi, la idea era trobar un servei web sobre locals d'oci del món que ja estigues disponible a Internet. No obstant això, davant la impossibilitat de trobar-ne un de vàlid, es va decidir implementar un servei propi en *PHP*[48]. Aquesta tecnologia va ser escollida perquè la finalitat principal era disposar d'un servei web senzill per tal de provar els dispositius Android, i *PHP* era un llenguatge que podia resultar més senzill per implementar-lo.

Simple Object Access Protocol (SOAP)

Dins dels serveis web existeixen diferents protocols de transmissió d'informació. En aquest cas, s'ha optat per treballar amb serveis Web sobre el protocol *SOAP*.

³Remote Procedure Call

SOAP és un protocol estàndard, creat per *Microsoft* i *IBM* entre d'altres, que defineix la manera en que dos objectes de diferents processos poden comunicar-se a través d'un intercanvi de dades en *XML*. És utilitzat normalment sobre el protocol *HTTP*., ja que gràcies a les seves propietats, és un protocol àmpliament difós i es troba molt poc restringit per tallafocs. Tanmateix, el protocol *SOAP* no va ser creat per funcionar sobre *HTTP*, pel que podria funcionar per altres protocols de comunicació, com *FTP* o *SMTP*.

Aquest protocol es va dissenyar per la comunicació entre diferents aplicacions, independentment de la plataforma on s'executin i del llenguatge de programació amb el qual han estat implementades. Al ser un protocol basat en l'estructura d'un document *XML*, és fàcilment extensible.

D'altra banda, un dels pocs inconvenients que presenta és el haver de fer un tractament dels caràcters especials abans de la transmissió. Tot caràcter especial que s'hagi de tractar en un arxiu *XML* normal, haurà de ser manipulat de la mateixa manera en el cas de transmissió de dades a través de *SOAP*.

Implementació d'un Web Service SOAP en PHP

A partir de la versió 5.0 de *PHP*, aquest llenguatge de marques ha introduït suport, en la seva llibreria estàndard, a la comunicació mitjançant serveis web en *SOAP*. Les versions anteriors utilitzaven el paquet *NuSOAP*[39] per tal d'implementar o utilitzar serveis web en *SOAP 1.1*.

Primer de tot, s'ha de realitzar un anàlisi dels casos d'ús del servei web per tal de poder-los descriure en un document *WSDL*⁴. Aquest llenguatge està basat en el llenguatge *XML* i serveix com a descripció de les operacions que implementa o a les quals es pot accedir des del servei web. A més a més, indicarà al client quines són les interfícies que proveeix el servei web i els tipus de dades que són necessaris per a la utilització del mateix. A la versió electrònica es troba el *WSDL* del servei web d'aquesta aplicació.

Un cop descrit el servei web, és el moment d'implementar la part servidora de l'aplicació. Aquesta constarà d'una sèrie de formularis per introduir les dades dels locals de Barcelona. Aquestes han estat guardades en una base de dades d'*SQLite* amb les mateixes taules que l'aplicació de la plataforma Android. No s'explicarà gaire aquesta implementació, ja que no forma part de l'objectiu del projecte, només es descriurà què ha fet falta per a la creació d'aquest servei web.

Encara que *PHP* és un llenguatge no tipificat, els tipus complexes de dades s'han de definir per tal de poder tractar-los. D'aquesta manera, per a cada tipus complex s'ha creat una classe en *PHP*, especificant tots els atributs anteriorment descrits en el document *WSDL*. A més a més, s'ha de generar una classe que actuï de servidor de les operacions que ofereix aquest servei web. Un cop les operacions han estat implementades s'ha de enregistrar com s'observa a la figura 3.22. Els tipus de dades complexes s'enregistren en un vector de tuples, introduint com a primer argument el nom del tipus descrit en

⁴El Web Service Descriptor Language, o *WSDL*, és un llenguatge *XML* que proveix un model per descriure serveis web.

```

$wsdl="ocioben.wsdl";

$classmap = array("Restaurant" => "Restaurant",
                 "DishMenu" => "Dish",
                 "Hotel" => "Hotel",
                 "Cinema" => "Cinema",
                 "CinePrice" => "CinePrice",
                 "CineFilm" => "CineFilm",
                 "Film" => "Film",
                 "Club" => "Club",
                 "Image" => "Image",
                 "ImageAsoc" => "ImageAsoc",
                 "ArrayOfImages" => "array",
                 "ArrayOfImageAsoc" => "array",
                 "ArrayOfRestaurant" => "array",
                 "ArrayOfHotel" => "array",
                 "ArrayOfCinema" => "array",
                 "ArrayOfCinePrice" => "array",
                 "ArrayOfCineFilm" => "array",
                 "ArrayOfFilms" => "array",
                 "ArrayOfClub" => "array",
                 "ArrayOfDishes" => "array",
                 "doImagesUpdateResponse" => "Array",
                 "doRestaurantUpdateResponse" => "Array",
                 "doCinemaUpdateResponse" => "Array");

$server = new SoapServer($wsdl, array("uri" => "urn:ociobcn",
                                     "classmap" => $classmap));

$server->setClass("Server");
if($_SERVER["REQUEST_METHOD"]=="POST"){
    $server->handle();
}

```

Figura 3.22: Creació d'un servei web SOAP en PHP

el document *WSDL* i com a segon l'objecte *PHP* que el tractarà. Finalment, s'indica quin és el document *WSDL* que descriurà el servei web i s'associarà el namespace i el vector la descripció dels tipus complexes.

La manera més fàcil de depurar els serveis web és la impressió de les peticions i les respostes en un navegador. Això és gràcies a que els serveis web en *SOAP* treballen a partir de documents *XML*, pel qual el navegador ens mostraria un document *XML* amb la petició del client i la resposta del servidor.

A la figura 3.23, observem com a la implementació del client no fa falta associar tots els tipus de dades. Això és degut a què només fa falta associar els que s'utilitzen en les peticions que es facin servir. En la setena línia s'observa la crida al mètode *doRestaurantUpdate*, que s'encarrega de retornar totes les actualitzacions de restaurants que s'han fet a partir d'una data en concret. Les dos línies següents, servirien només per a depurar el codi, ja que imprimarien la petició i la resposta del servidor dins de la

```

$wsdl="ociobcn.wsdl";

$classmap = array("Restaurant" => "Restaurant",
                  "DishMenu" => "Dish",
                  "ArrayOfRestaurant" => "array",
                  "ArrayOfDishes" => "array",
                  "doRestaurantUpdateResponse" => "Response");

$cliente=new SoapClient($wsdl,
                       array("uri" => "urn:ociobcn",
                              "classmap" => $classmap,
                              "trace" => 1,
                              "exceptions" =>1));

try {
    $from = date("d/m/y_H:i:s");
    $resultado = $cliente->doRestaurantUpdate($from);
    $BGCOLOR="#DEDEDE";
    $restaurants= array_pop($resultado);
    $dishes = array_pop($resultado);
}

```

Figura 3.23: Utilització d'un servei web SOAP en PHP

mateixa pàgina web.

Cal esmentar que el retorn de les funcions de la part servidora són sempre del tipus *array*, on s'inclouen cadascun dels arguments retornats. A la figura 3.23, s'observa com s'extreuen dos objectes d'aquest vector resultant. Això és degut a que el mètode retorna diverses *arrays* d'objectes, cadascuna amb un vector d'un tipus complex diferent.

Enviament de binaris per SOAP

Tal i com s'ha comentat anteriorment, *SOAP* hereta del protocol *XML* i, per tant, també hereta els seus caràcters especials. És per això que la informació d'un arxiu binari no pot ser enviada a través d'aquest protocol, ja que la seva codificació pot contenir caràcters protegits del protocol. Així doncs, per tal d'enviar aquesta informació s'haurà de codificar, per evitar la introducció d'aquests caràcters especials, utilitzant la codificació Base64.

La codificació Base64 es basa en l'ús de caràcters *US-ASCII*, que no són accentuats, per a codificar qualsevol tipus d'informació mitjançant un codi de 8 bits. La principal utilització d'aquesta codificació es dona en l'enviament de dades adjuntes als protocols de correu electrònic, els quals van estar dissenyats, en un principi, per l'enviament de text.

El concepte de codificació Base64 consisteix en l'ús de 4 caràcters imprimibles per a

codificar 3 bytes, és a dir, un total de 24 bits. En aquesta codificació s'utilitza un alfabet de 64 caràcters, els quals van ser escollits per ser universalment llegibles, i perquè no tenen cap significat especial dins dels protocols de correu electrònic.

- ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz12345678-9+/-.

Per tal de garantir la compatibilitat amb els sistemes de correu electrònic, les dades en base64 es formatejan amb retorns de carro per a que cap línia sigui major de 76 caràcters.

En la tecnologia utilitzada en la creació del servei web, la codificació d'imatges en Base64 es faria de la següent manera:

```
$string = base64_encode($row->data);
```

El problema d'aquests enviaments d'arxius binaris és la sobrecàrrega que provoquen. En una codificació en base64 existeixen 6 bits de dades útils per cada caràcter, el que farà incrementar en gairebé un 33% el volum de les dades a enviar. En aquesta aplicació, s'ha fet servir aquest procediment en la descàrrega d'imatges, encara que pot ser més útil buscar alternatives que no sobrecarreguin els dispositius dels usuaris.

Utilització de Web Services amb Android

Amb la part servidora dels serveis web implementada, el següent pas és implementar l'accés des d'*Android*. La plataforma en la versió M5 de l'SDK, no porta cap tipus de suport per la implementació de clients de Serveis Web, com caldria esperar. *Android* disposa d'un traductor de arxius .class, compilats en *Java*, a fitxers dex per tal de reproduir-los en la consola virtual del sistema, la consola *Dalvik*. Això significa, que gairebé qualsevol llibreria implementada en *Java* podria ser importada a *Android*.

La primera llibreria provada per accedir als serveis web des d'*Android* va ser una llibreria distribuïda per *Apache*[46] anomenada *Axis*[47]. És una llibreria de *J2SE* que al intentar traduir-la a un arxiu .dex, dona la següent excepció: "Conversion to *Dalvik* format failed with error 1". Aquest bug de conversió, reportat en el *Issue 458* a la pàgina principal d'*Android*, pot no tenir solució, ja que *Google* ja adverteix que no totes les classes *Java* poden ser traduïdes a *Dalvik*. Igualment, al ser una llibreria de *J2SE* se suposa que consumirà un nombre més elevat de recursos que una desenvolupada per exemple en *J2ME*.

Tanmateix, es va trobar una altre llibreria anomenada *KSOAP*[38], actualment en la versió 2.0. En la seva pàgina *sourceforge* trobem que "KSOAP és una llibreria de clients de serveis web en *SOAP* construïda per a entorns *Java*, com *Applets* o aplicacions *J2ME*". Aquesta definició dona la impressió de que *KSOAP* pot ser una llibreria per *Android*, ja que aquest porta un entorn *Java* similar al de *J2ME*.

Després de diverses proves, es van trobar dues classes en un blog[9] d'un desenvolupador que faciliten la implementació de la connexió amb el *webservice*, anomenades

AndroidServiceConnection i *AndroidHttpTransport*. La primera és una classe derivada de la classe de la llibreria *KSOAP ServiceConnection*, que delimita la connexió i introdueix els mètodes d'entrada i de sortida de dades. *AndroidHttpTransport* en canvi, implementa la crida a les operacions del servei web, per tal de no haver d'introduir cada cop els paràmetres de configuració de *ksoap2*.

Tal i com s'ha fet en la part servidora, és necessària la definició dels tipus complexos de dades per tal de que la llibreria els pugui tractar de la manera adequada, reservant l'espai de memòria adient al tipus de dades bàsics que els componen. Per tal de fer això, la llibreria *KSOAP* proporciona la classe *SoapObject*, de la qual derivaran els objectes complexos del nostre servei. En aquests s'han d'implementar quatre mètodes: *getProperty*, *setProperty*, *getPropertyInfo* i *getPropertyCount*.

Els dos últims mètodes indiquen el tipus de les diferents propietats del objecte i el número de propietats que té, respectivament. Els tipus de les propietats suportats es troben dins la classe *PropertyInfo* i són: *STRING_CLASS*, *LONG_CLASS*, *INTEGER_CLASS*, *BOOLEAN_CLASS*, *VECTOR_CLASS* i *OBJECT_CLASS*.

Qualsevol altre tipus hauria de ser tractat com a un objecte *SoapPrimitive* i, a partir d'aquest, transformar-lo al tipus corresponent. Un exemple és el cas del tipus *Double* el qual no està suportat. Per tal de fer-ho, s'haurà de transformar a *string* el tipus *SoapPrimitive* per després passar-ho a *Double*, tal i com s'observa en la figura 3.24

Un cop es tenen els tipus complexos implementats, ja es poden realitzar les crides a les operacions del servei web. Per tal de realitzar la crida, s'utilitzarà un objecte del tipus *AndroidHttpTransport* i es cridarà a una de les operacions disponibles al servidor, a través del mètode *call* i el nom del mètode *SOAP* i un objecte *SoapSerializationEnvelope*, que contindrà tot el conjunt d'atributs de la petició, com a paràmetres. Dins d'aquest "envoltori" es troba un objecte *SOAP* on s'introduiran els paràmetres del mètode amb *request.addProperty()*. A més a més, s'han d'introduir quines són les classes que tractaran cada tipus de dades complex amb la funció *addMapping*. Els dos primers paràmetres de la funció han de coincidir amb el *namespace*⁵ del servei web i el nom del tipus especificat al *WSDL* de l'aplicació, respectivament. En el cas de que algun dels paràmetres del tipus complex no existís, o estigués mal, especificat, el programa donaria un error en l'anàlisi de la resposta del servei web. Tot això s'observa en el mètode *doRequest* de la classe *Updater*, tal i com s'observa a la figura 3.25. Cal esmentar que aquesta classe s'encarrega d'actualitzar l'aplicació tan per sigui per servei web com per l'anàlisi d'un fitxer de la memòria del dispositiu. Aquesta distinció es fa mitjançant el paràmetre *mode*, que indica el mode d'actualització escollit per l'usuari.

Un cop s'ha efectuat la crida es reben els resultats en un *SoapObject* que s'ha d'interpretar com un vector amb els resultats. Encara que sembli estrany, la resposta es rep a través de l'atribut *bodyIn* de l'objecte *SoapSerializationEnvelope*, ja que el *bodyOut* és la petició que hem introduït anteriorment.

```
resultRequestSOAP = (SoapObject) envelope.bodyIn;
```

⁵El namespace serveix com a context dels objectes que conté. Mitjançant aquest concepte es poden separar diferents conceptes entre objectes

```

public void getPropertyInfo(int arg0, Hashtable arg1, PropertyInfo arg2) {
    switch(arg0)
    {
        case 0:
            arg2.type = PropertyInfo.INTEGER CLASS;
            arg2.name = "MenuId";
            break;
        case 1:
            arg2.type = PropertyInfo.INTEGER CLASS;
            arg2.name = "DishId";
            break;
        case 2:
            arg2.type = PropertyInfo.STRING CLASS;
            arg2.name = "Name";
            break;
        case 3:
            arg2.type = Double.class;
            arg2.name = "Price";
            break;
        case 4:
            arg2.type = PropertyInfo.INTEGER CLASS;
            arg2.name = "Type";
            break;
    }
}
public void setProperty(int arg0, Object arg1) {
    switch (arg0) {
        case 0:
            setMenu_id(((Integer) arg1).intValue());
            break;
        case 1:
            setDish_id(((Integer) arg1).intValue())
            break;
        case 2:
            setName((String) arg1);
            break;
        case 3:
            String value = ((SoapPrimitive) arg1).toString();
            setPrice(new Double(value).doubleValue());
            break;
        case 4:
            setType(((Integer) arg1).intValue());
            break;
        default:
            break;
    }
}
}

```

Figura 3.24: Implementació de la classe SoapObject que representa l'objecte plat

```

public void doRequest(int mode, String METHOD NAME, boolean images)
{
    try {
        SoapObject request = new SoapObject(NAMESPACE, METHOD NAME);
        if (mode != 4 ) {
            request.addProperty("from", data[mode]);
            request.addProperty("downloadImages", images);
        }
        SoapSerializationEnvelope envelope =
            new SoapSerializationEnvelope(SoapEnvelope.VER1
            envelope.setOutputSoapObject(request);
        switch(mode)
        {
            case 0:
                envelope.addMapping(NAMESPACE, "Restaurant", RestaurantObject.class);
                envelope.addMapping(NAMESPACE, "DishMenu", DishObject.class);
                break;
            case 1:
                envelope.addMapping(NAMESPACE, "Hotel", HotelObject.class);
                break;
            case 2:
                envelope.addMapping(NAMESPACE, "Cinema", CinemaObject.class);
                envelope.addMapping(NAMESPACE, "Film", FilmObject.class);
                envelope.addMapping(NAMESPACE, "CinePrice", CinePriceObject.class);
                envelope.addMapping(NAMESPACE, "CineFilm", CineFilmObject.class);
                break;
            case 3:
                envelope.addMapping(NAMESPACE, "Club", ClubObject.class);
                break;
        }
        envelope.addMapping(NAMESPACE, "Image", ImageObject.class);
        envelope.addMapping(NAMESPACE, "ImageAsoc", ImageAsocObject.class);
        AndroidHttpTransport androidHttpTransport = new AndroidHttpTransport(URL);
        SoapObject resultRequestSOAP = null, aux= null;
        androidHttpTransport.call(METHOD NAME, envelope);
        (. . .)
    }catch (IOException e) {
        e.printStackTrace();
    } catch (XmlPullParserException e) {
        e.printStackTrace();
    }
}
}

```

Figura 3.25: Implementació del mètode doRequest

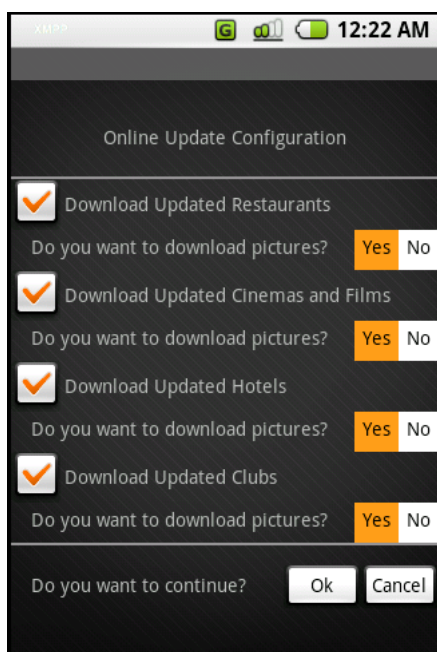


Figura 3.26: Operacions a realitzar

Per tots els mètodes que utilitza l'aplicació, el servei web retornarà *arrays* d'objectes cosa que fa que s'utilitzi el nom d'aquest per saber de quin tipus són els objectes. Un cop se sap el tipus d'*array* al qual correspon, s'agafa cadascun dels seus elements i es tracten com l'objecte definit anteriorment. Abans de guardar-los en la base de dades, s'introduiran en una llista per reduir les transaccions d'*SQLite* i reduir el temps en l'anàlisi de la resposta del servei web. Un cop totes les dades es trobin en les diferents llistes d'objectes s'aniran guardant a la base de dades.

D'altra banda, per tal de poder seleccionar quins locals s'han d'actualitzar, es disposa d'un formulari per poder-ho especificar. A més a més, també existeix l'opció de no descarregar imatges, ja que, com s'ha especificat anteriorment, a mesura que la seva mida sigui més gran anirà fent més lent el procés d'actualització. En la figura 3.26 es poden observar les opcions anteriorment esmentades.

Les operacions per actualitzar l'aplicació poden ser lentes, segons la connexió que tingui establerta l'usuari i la quantitat de dades a descarregar. Per tal de que l'aplicació sigui "reactiva", les operacions d'actualització, ja sigui a través del tractament d'un arxiu *XML* o de l'accés a serveis web, sempre es realitza en un fil apart, per tal de poder interaccionar amb la interfície visual en el fil principal. En la figura 3.27 s'observa el diàleg que es mostra a l'usuari, per tal de que sàpiga que l'aplicació està treballant en l'actualització.

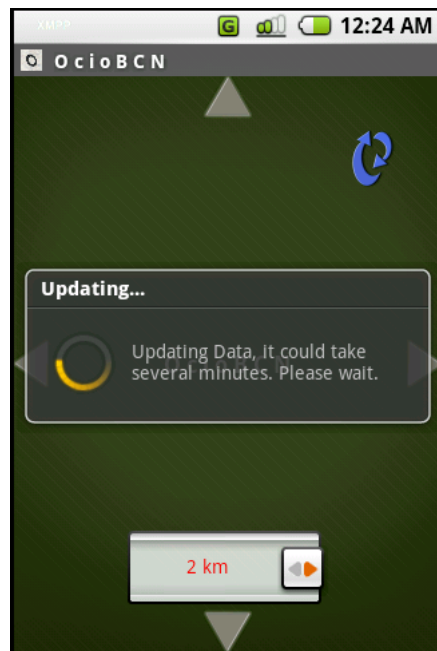


Figura 3.27: Barra de progrés de l'actualització

Reproducció d'imatges rebudes mitjançant serveis web

La reproducció de les imatges obtingudes mitjançant el servei web és un punt interessant a tractar. Les dades d'aquestes imatges venen codificades en Base64 per les raons que s'han explicat anteriorment. D'aquesta manera un cop descarregades, s'han de descodificar i guardar en el format que es desitgi. En el cas d'aquesta aplicació, es va decidir guardar les imatges a la base de dades de l'aplicació. Es guarden descodificades per tal de ocupar el mínim espai possible en el dispositiu.

Cada imatge disposarà de dos identificadors, els quals indiquen el tipus de local assignat i l'identificador del local en concret al qual pertanyen. D'aquesta manera, cada local podrà mostrar les fotos que es tenen guardades quan es selecciona la secció d'imatges.

Com s'observa en la figura 3.28, les imatges del local queden guardades en un *array* de *Drawables* per tal de poder treballar amb els diferents components de la vista. Al ser guardada a la base de dades, s'ha de generar la imatge a partir d'una cadena de *bytes* mitjançant un *ByteArrayInputStream*. Després aquests bytes es passen a un arxiu temporal amb l'objectiu final de guardar-lo a la memòria del dispositiu.

En la vista final, tal i com s'observa a la figura 3.29, destaquen dos elements principals: *ImageSwitcher* i *Gallery*.

La *Gallery* serà utilitzada per a reproduir, en miniatura, totes les imatges guardades del local per tal de poder anar seleccionant quina imatge és la que es vol veure maximitzada. Es volia poder seleccionar la foto mitjançant un clic sobre la imatge desitjada, però l'esdeveniment de *mouseClicked* no pot ser capturat per l'objecte *Gallery*, pel que els elements aniran reproduint-se seqüencialment mitjançant les tecles de direcció del

```

private int object id, type;
private Drawable[ ] mImages;
private String[ ] mDescription;
private void loadImages() {
    Uri query=Uri.withAppendedPath(Image.OBJECT_URI,
                                   Integer.toString(object id));
    query= Uri.withAppendedPath(query, Integer.toString(type));

    Cursor aux = managedQuery(query,
                              Images.ASOC_PROJECTION,
                              null,
                              null);
    mImages=new Drawable[aux.count()];
    mDescription=new String[aux.count()];
    int i=0;
    while (aux.next())
    {
        int image id= aux.getInt(1);
        query = Uri.withAppendedPath(Image.IMAGE_URI,
                                     Integer.toString(image id));
        Cursor image = managedQuery(query,
                                    Images.IMAGE_PROJECTION,
                                    null, null);

        image.first();
        String imagedata= image.getString(
                           image.getColumnIndex(Image.DATA));
        ByteArrayInputStream input =
            new ByteArrayInputStream(imagedata.getBytes());
        mImages[i] = Drawable.createFromStream(input,
                                               "/tmp/tmpimage");
        mDescription[i]=
            image.getString(image.getColumnIndex(Image.DESCRPTION));
        i++;
    }
}

```

Figura 3.28: Càrrega d'imatges

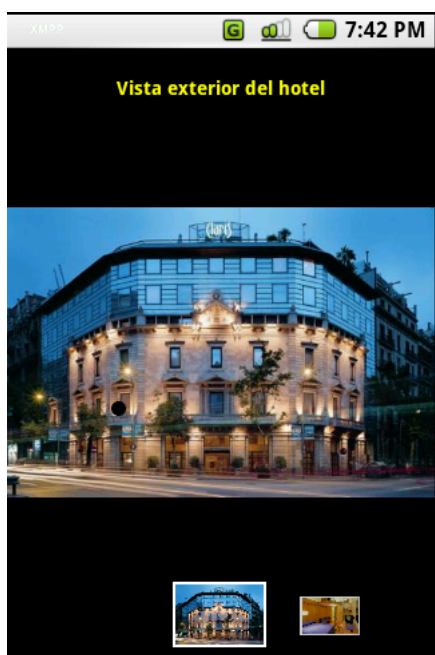


Figura 3.29: Reproducció d'imatges

dispositiu. El focus es situarà sempre a la *Gallery*, pel que cada cop que un element estigui seleccionat canviarà la imatge seleccionada al *Switcher*. A més a més, durant les diferents proves que s'han executat sobre l'aplicació s'ha observat que existeix la possibilitat d'anar passant les imatges de la galeria arrossegant-les, cosa que fa canviar també la imatge de l'altre component gràcies a la implementació d'un *listener* sobre l'element seleccionat. Aquesta funcionalitat estava ja implementada dins del component *Gallery*.

D'altra banda, l'*ImageSwitcher* és un component una mica complex el qual reproduceix imatges en un espai i ofereix la possibilitat de succedir-les mitjançant animacions. Aquestes característiques fonamentals de l'objecte s'han de definir en la seva creació. Tal i com s'observa en la inicialització del component de la figura 3.30, en l'aplicació han quedat definides les animacions d'entrada i sortida d'imatges del component.

En l'ús d'aquest component, es necessita implementar una factoria de vistes, anomenada *ViewSwitcher.ViewFactory*, per tal de generar les vistes que es visualitzen en el seu espai gràfic. En aquesta implementació, s'ha dissenyat la vista per tal de que la imatge ocupi la posició central del component, sobre un fons negre. L'única funció d'aquesta factoria que s'ha d'implementar és *makeView*, que quedaria de la següent manera:

Amb tot això, ha quedat implementada la galeria de fotos personalitzada per l'aplicació.

3.3.7 Utilització de la interfície gràfica

La composició de les interfícies utilitzades per la nostre aplicació es pot arribar a fer de dos maneres diferents.

```

desc=(TextView) findViewById(R.id.image_description);
mSwitcher = (ImageSwitcher) findViewById(R.id.image_switcher);
mSwitcher.setFactory(this);
mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
    android.R.anim.slide_in_bottom));
mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
    android.R.anim.slide_in_top));
Gallery g = (Gallery) findViewById(R.id.image_gallery);
g.setAdapter(new ImageAdapter(this));
g.setSelectorSkin(getResources().getDrawable(android.R.drawable.box));
g.setOnItemSelectedListener(this);
g.setOnItemClickListener(this);

```

Figura 3.30: Inicialització dels components Gallery i ImageSwitcher

```

public View makeView() {
    ImageView i = new ImageView(this);
    i.setBackgroundColor(0xFF000000);
    i.setScaleType(ImageView.ScaleType.FIT_CENTER);
    i.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
        LayoutParams.FILL_PARENT));
    return i;
}

```

Figura 3.31: Funció makeView

La més habitual es basa en la redacció d'un document *XML* per interfície, on es declararan tots els components de cada una de les activitats amb els atributs que es considerin necessaris. Per tal de situar els elements en una posició concreta, es disposa de diversos tipus de patrons o disposicions⁶ per tal de facilitar la feina de posicionament.

Existeixen un total de cinc *layouts*, els quals es poden combinar entre sí per ajustar-se a les necessitats dels desenvolupadors: *LinearLayout*, *RelativeLayout*, *TableLayout*, *AbsoluteLayout* i *FrameLayout*.

Encara que es poden utilitzar tots cinc, els que realment s'utilitzen són els tres primers. El *FrameLayout* s'utilitza per a omplir un espai de la pantalla en el qual es situaran elements, els quals encara no es disposen en aquell moment, però que d'altra banda se sap quin és l'espai que ocuparan, i l'*AbsoluteLayout* s'utilitza per a situar els components en funció de les coordenades de la pantalla, sense poder utilitzar marges. El *FrameLayout* aporta també la possibilitat de ser utilitzat per situar algun component visual que vagi canviant. Aquest *layout* anirà dibuixant un component sobre l'altre de tal manera que pot arribar a donar sensació d'activitat. Els components haurien de ser de la mateixa mida per tal de què no quedessin restes del component anterior.

El *LinearLayout*, en canvi, és un dels *layouts* més utilitzats. Aquest s'encarrega d'alinear a cada fila o columna un únic component fill segons la seva orientació sigui vertical o horitzontal, respectivament. La manera en que s'alineen els components ve especificada en l'atribut *android:orientation* del document *XML* on es defineix. D'altra banda, aquest tipus de *layout* té el concepte de gravetat⁷, atribut que permet que l'únic component del *layout* es situï en una posició determinada dins de la fila o columna

Existeix també un atribut, anomenat *android:layout_weight* que li dona una característica especial al *LinearLayout*. Aquest atribut indica quin pes específic té un component sobre el *layout*, la qual cosa determinarà la seva mida i l'espai on es troba situat. Si, per exemple, tenim tres components en un *LinearLayout* horitzontal i dos d'ells tenen una mida específica, pot interessar que el tercer component s'ajusti a la resta de l'espai de pantalla. Això fa que l'aplicació s'adapti a la pantalla de cada dispositiu de mòbil concret. El valor per defecte si no s'especifica aquest atribut és zero (figura 3.32)

El *RelativeLayout*, en canvi, situa els seus components respecte altres components de la pantalla. El component pare és el mateix *layout*, amb el qual es poden centrar, verticalment y horitzontalment, qualsevol objecte que s'introdueixi. Les relacions entre components s'avaluen en una sola passada, pel que els components que estiguin referenciats per altres components hauran d'estar especificats al davant d'aquests. Aquesta característica impedeix la realització de qualsevol relació circular. D'altra banda, en cas de que algun component del *layout* no tingui relacions en el document, el *RelativeLayout* actuarà com un *LinearLayout* vertical i anirà situant seqüencialment els elements a l'inici pantalla.

Per últim, el *TableLayout* és una *layout* que distribueix la pantalla en files y columnes. El component principal d'aquest *layout* és el *TableRow*, que actua de *LinearLayout* horitzontal amb una sèrie de característiques pròpies de taula. Tots les columnes entre

⁶En anglès son anomenats layouts

⁷La gravetat d'un component indica on es situaran els seus components fills.



<LinearLayout

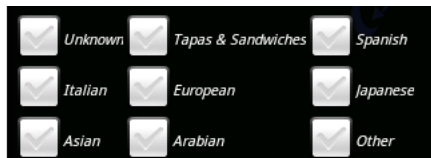
```

xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:layout_weight="2">
<TextView android:layout_width="wrap_content"
  android:layout_height="wrap_content" android:text="Category:"
  android:padding="3dip" />

<Spinner android:id="@+id/hot_filter_spinner"
  android:layout_width="wrap_content" android:layout_height="60px"
  android:layout_weight="2" android:textSize="8dip" />
<Spinner android:id="@+id/hot_filter_spinner2"
  android:layout_width="wrap_content" android:layout_height="60px"
  android:layout_weight="2" android:textSize="8dip" />
</LinearLayout>

```

Figura 3.32: LinearLayout horitzontal



<TableLayout

```

xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content" android:stretchColumns="true"
android:layout_weight="2">
<TableRow>
  <CheckBox android:id="@+id/res_unknown_check"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:textStyle="italic"
    android:textSize="11sp" android:text="@string/res_unknown_check" />
  <CheckBox android:id="@+id/res_tapas_check"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:textStyle="italic"
    android:textSize="11sp" android:text="@string/res_tapas_check" />
  <CheckBox android:id="@+id/res_spanish_check"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:textStyle="italic"
    android:textSize="11sp" android:text="@string/res_spanish_check" />
</TableRow>
(...)
</TableLayout>

```

Figura 3.33: TableLayout

```

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    * android:layout_height="wrap_content">
    <LinearLayout android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingTop="15px">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="15px"
            android:text="Cartelera" android:padding="5px"
            android:textColor="@color/solid_yellow"
            android:layout_gravity="center" />
        <ListView android:id="@+id/cine_films"
            android:layout_width="fill_parent"
            * android:layout_height="wrap_content" />
    </LinearLayout>
</ScrollView>

```

Figura 3.34: ScrollView amb subelements amb scroll

les files d'un mateix *TableLayout*, s'alineen per tal de donar la composició d'una taula. En aquest tipus de *layouts*, les columnes es poden tractar amb el mètode *setColumnCollapsed()*, per tal de que no surtin per pantalla en un moment determinat. A més a més, existeixen dos atributs amb una utilitat similar: *shrinkable* i *stretchable*. Ambdós fan que les columnes s'adaptin a l'espai lliure disponible a pantalla. El primer ajusta les columnes a l'espai disponible a la taula i el segon utilitza l'espai disponible dins dels seu objecte pare per ampliar-les. Els filtres de l'aplicació, mostren un exemple d'aquests *layouts* a la figura 3.33.

Error típic ScrollViews

Existeix un error que dona molts problemes als desenvolupadors fins que se n'adonen de la realitat del problema. A *Android* existeixen com a mínim dos components, *ListView* i el *GridView*, que porten definit per defecte un *scroll* vertical automàtic. Això fa que quan s'inserten multitud d'elements en aquestes vistes, no faci falta indicar quan es vol introduir un objecte del tipus *scroll*.

Ara bé, el problema ve donat quan un d'aquests components és subelement d'un *layout* superior que té definit un *scroll* vertical per tal de no perdre la informació inserida en la vista. *Android* facilita un argument, anomenat *wrap_content*, per tal d'ajustar el component a la informació que conté redimensionant-lo segons la mida de la pantalla. Aquest és l'origen de les confusions dels usuaris.

En l'exemple de la figura 3.34, s'observa com els components *ScrollView* i *ListView* utilitzen aquest argument per tal d'adaptar la seva longitud a la mida de la pantalla. No obstant, quan s'intenta executar el desenvolupador es trobaria amb un error del

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <LinearLayout android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingTop="15px">
        <LinearLayout android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
            (. . .)
        </LinearLayout>
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="15px"
            android:text="Cartelera" android:padding="5px"
            android:textColor="@color/solid_yellow"
            android:layout_gravity="center" />
        <ListView android:id="@+id/cine_films"
            android:layout_width="fill_parent"
            android:layout_height="40px" />
    </LinearLayout>
</ScrollView>

```

Figura 3.35: Layout de la interfície d'informació d'un cinema

tipus:

- ListView can't have UNESPECIFIED size.
- GridView can't have UNESPECIFIED size.

Un missatge d'error que pot confondre, però que clarament indica què és el que està passant. El problema es dona perquè existeixen dos components niats que incorporen un *ScrollView*. D'aquesta manera, aquesta pantalla té un contingut ambigu, ja que no se sap quin dels dos components té prioritat respecte l'altre. Per tal de solucionar-ho, sempre es fixa la mida del component fill, per tal de poder tenir dos components dels tipus *ScrollView* niats tal i com s'observa a la figura 3.36. Aquesta solució és l'adoptada en la reproducció de la informació de cinemes i pel·lícules. La figura 3.35 és una part del *layout* d'una d'aquestes interfícies.

Es pot trobar una descripció de tots els atributs que es poden introduir en els diferents documents de *layout* a la pàgina oficial d'*Android*[21].

3.3.8 Composició de components visuals propis

Una de les coses que més crida l'atenció d'*Android* és la capacitat real de poder crear interfícies gràfiques vistoses sense massa dificultat. De fet, en el manual introductori a



Figura 3.36: ScrollViews niats

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <declare-styleable name="TextOnlyButton">
    <attr name="textColorNotFocused" format="integer" />
    <attr name="textColorFocused" format="integer" />
  </declare-styleable>
</resources>
```

Figura 3.37: Atributs definits per al textButton

la plataforma Android, existeix una part dedicada als components gràfics disponibles i a les possibilitat de crear-ne de nous adaptats a les necessitats del desenvolupador.

En aquesta secció, es presenten els components que són propis de l'aplicació i, per tant, les diferents maneres de generar components visuals.

TextButton

Per tal d'ocupar el menor espai possible amb els botons, s'ha decidit crear botons només textuais. Això vol dir, que els hi assignarem una propietat per tal de que al ser enfocats canviïn de color, però conservant l'aspecte d'etiqueta de text. Cal dir que existeixen diverses maneres de fer-ho, cadascuna amb les seves característiques.

En aquest cas, es defineixen primerament dos nous atributs del component visual que donaran el color del text quan hagi agafat el focus i quan no el tingui. Aquests atributs, figura 3.37 s'han de declarar al fitxer *attr.xml* del directori *res* de la nostre aplicació.

```

private void init(AttributeSet attrs) {
    Resources.StyledAttributes a =
        getContext().obtainStyledAttributes(attrs, R.styleable.TextOnlyButton);
    if (a.getString(R.styleable.TextOnlyButton_textColorNotFocused) != null
        && a.getString(R.styleable.TextOnlyButton_textColorFocused) != null) {
        notFocusedTextColor =
            a.getColor(R.styleable.TextOnlyButton_textColorNotFocused, 0xFF000000);
        focusedTextColor =
            a.getColor(R.styleable.TextOnlyButton_textColorFocused, 0xFF000000);
    } else {
        throw new RuntimeException("Valid colors (e.g. #ffffff) must be
            passed to this class via the XML parameters:
            pj:textColorNotFocused & pj:textColorFocused.");
    }
}
}

```

Figura 3.38: Inicialització dels atributs propis de `TextButton`

```

public void onDraw(Canvas canvas) {
    if (isFocused()) {
        setTextColor(focusedTextColor);
    } else {
        setTextColor(notFocusedTextColor);
    }
    super.onDraw(canvas);
}
}

```

Figura 3.39: Mètode que dibuixa el botó de text

Un cop declarats els atributs, s'ha de crear la implementació del botó derivant-lo de la classe `Button`. D'aquesta manera ja tindrà implementat tota la funcionalitat del botó. En realitat, l'únic que canvia respecte el component `Button` és l'aspecte visual. En la creació del botó, es fa una crida al mètode `init`, el qual disposarà de tots els atributs introduïts en el arxiu de `layout` on s'indiquen els seus components, inclosos els atributs definits exclusivament en aquesta aplicació. En aquest mètode, es guardaran a memòria els valors dels atributs que hem declarat anteriorment. A la figura 3.38 es pot veure la manera per implementar-lo.

Com el fons d'aquest tipus de botons es transparent, el mètode `onDrawBackground` no s'implementarà, és a dir, s'implementarà aquesta funció sense cridar al mètode de la classe pare. D'aquesta manera, només s'ha d'assignar quin és el color del text per a cada estat del botó en el mètode `onDraw`, figura 3.39

Imatges que fan de botó

La creació de botons d'imatge es realitza d'una manera totalment diferent a la creació del component `TextButton` descrita anteriorment. En realitat, existeix una classe en la

llibreria per defecte d'*Android*, anomenada *ImageButton*, que fa les funcions de botó amb una imatge inserida al damunt. El problema es que el que es busca realment és una imatge que varii segons el tipus d'esdeveniment que rebí.

Per tal de fer això, tot el que es necessita es troba al directori de *res*. Dins d'aquest directori, es troba el directori *drawable* on es guarden totes les imatges utilitzades per la aplicació. Un cop creades les imatges per a cada tipus d'esdeveniment es guardaran en aquest directori. En l'exemple de la figura 3.40 es troben l'estat normal, pressionat i enfocat d'un dels botons de l'aplicació. El següent pas és associar cadas-



Figura 3.40: Estats del botó

cuna d'aquestes imatges amb els estats predefinitos de la classe *Button* de la llibreria estàndard d'*Android*. Els estats en els que un component pot estar són: *state_single*, *state_first*, *state_middle*, *state_last*, *state_focused*, *state_window_focused*, *state_checkable*, *state_checked*, *state_active*, *state_selected* i *state_pressed*

En el cas d'aquest tipus de botó, s'ha considerat important la combinació dels estats *state_focused* i *state_pressed*, la combinació dels quals tindran associada la imatge determinada com a imatge de fons. L'arxiu de selecció quedaria com a la figura 3.41. En aquest s'indiquen les condicions que s'han de complir per tal de escollir una o una altra imatge. Aquestes condicions són els diferents estats que el botó ha de tenir, per exemple en el cas de estar enfocat no només el botó ha de tenir el focus de la finestra, sinó que a més a més no ha d'estar pressionat.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_focused="true"
        android:state_pressed="false"
        android:drawable="@drawable/my_downbutton_background_focus" />
  <item android:state_focused="true"
        android:state_pressed="true"
        android:drawable="@drawable/my_downbutton_background_pressed" />
  <item android:state_focused="false"
        android:state_pressed="true"
        android:drawable="@drawable/my_downbutton_background_pressed" />
  <item android:drawable="@drawable/my_downbutton_background_normal" />
</selector>
```

Figura 3.41: Document XML per seleccionar la imatge d'un botó

Per tal d'utilitzar aquest tipus de botons, s'ha de definir un botó, de la classe *Button*, en l'arxiu *layout* corresponent i indicar-li quin arxiu té definit el fons del botó per mitjà del seu estat amb la propietat *android:background*. S'assignarà el nom del fitxer on hem guardat la selecció anterior sense la seva extensió, tal i com s'observa a la figura 3.42. A més a més observem com *Android* té definides les seves carpetes de recursos per tal de poder accedir des de els arxius de definició d'interfícies, en aquest cas s'observa l'accés

al directori *drawable* de l'objecte mitjançant la comanda *@drawable*.

```
<Button android:id="@+id/club"
    android:layout_centerHorizontal="true"
    android:layout_alignParentBottom="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/down_button"
    android:nextFocusUp="@id/cinema" />
```

Figura 3.42: Definició d'un ImageButton propi

MapView

Tal i com s'ha esmentat en la secció 3.3.3, s'ha generat un mapa per tal de modificar les funcions que venen implementades per defecte al component *MapView*. Aquest mapa introduirà la propietat de ser polsable per tal d'obtenir una posició en concret del mapa. No seria necessari obtenir la posició d'un lloc d'oci mitjançant el mapa, ja que existeix una imatge superposada al mapa la qual té una posició dins la pantalla del dispositiu. És a dir, el dispositiu ens podria indicar les coordenades on s'ha pitjat i mitjançant aquestes trobar el local que l'usuari a indicat. Tanmateix, la posició retornada s'utilitza per saber quins locals estan al voltant de la posició polsada, per treure informació com s'observa a la figura 3.45.

A més a més, la funcionalitat de redimensionament també ha sigut re-implementada. En aquest component aquesta funcionalitat no s'executa polsant durant un llarg període de temps sobre la pantalla, sinó que s'utilitza un component propi per tal de poder visualitzar-la constantment a la pantalla. El component utilitzat com a zoom, que es pot observar en la figura 3.43, serà explicat a la següent secció.

El component derivat de la classe *MapView* inclou un nou *listener* per tal de proporcionar la posició GPS que s'ha polsat sobre el mapa. Aquesta posició no és 100% precisa, encara que s'apropa el suficient a la realitat. El *listener* genera un esdeveniment cada cop que el mapa detecta un esdeveniment de pantalla del tipus *MotionEvent.ACTION_DOWN*. Tal i com es veu a la figura 3.44, s'agafa les coordenades en pixels del mapa on s'ha polsat i es transforma a una posició GPS aproximada. El component de mapa per defecte, proporciona la posició GPS real en el seu punt mig, pel que es fàcil l'obtenció del punt real.

D'altra banda, el següent codi mostra que per tal de desactivar la funció de zoom s'ha hagut de re-implementar la funció *onLongPress*. En comptes d'augmentar o disminuir el mapa, el component centrarà el mapa en la posició que hagi estat polsada durant un període llarg de temps.



Figura 3.43: Mapa inicial amb els locals propers

```

public boolean onTouchEvent(MotionEvent ev) {
    double x pos, y pos;
    switch (ev.getAction())
    {
    case MotionEvent.ACTION_DOWN:
        x pos = ev.getX();
        y pos = ev.getY();
        Point clickPosition = getGpsPosition(x pos, y pos);
        listener.onClick(this, new MapPoint(clickPosition.getLatitudeE6(),
            clickPosition.getLongitudeE6()));

        break;
    default:
        //no es canvia cap altre esdeveniment
        break;
    }
    return super.onTouchEvent(ev);
}

```

Figura 3.44: Re-implementació del mètode onTouchEvent del MapView

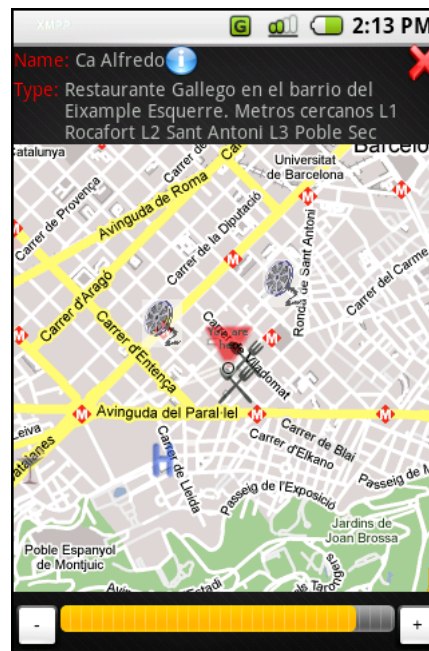


Figura 3.45: Mapa amb informació sobre el local desitjat

Horizontal Slider

Aquest component es pot observar a la part baixa de les figures 3.43 i 3.45 i és utilitzat per a reproduir la funcionalitat de zoom del mapa. La idea va estar extreta del fòrum *Anddev*[1], modificada per a que actuï com l'aplicació necessita. El component formava part d'un manual d'us de modificació de components gràfics i, en aquest cas, ha estat utilitzat com a Zoom dels mapes de la nostra aplicació. És un component derivat de la classe *Progress Bar* al qual se l'hi ha afegit el tractament del esdeveniment de pressió y moviment. D'aquesta manera, polsat sobre una posició de la barra, és pot obtenir un valor que, a l'aplicació, funcionarà com a paràmetre de zoom.

A més a més, aquest component genera un nou esdeveniment per tal de poder-ho tractar en les interfícies gràfiques en les quals s'utilitzi. Això es realitza afegint-li un nou *listener*, el qual es cridarà cada cop que hi ha hagut un esdeveniment de pressió o moviment sobre la barra del component, com es pot observar a la figura 3.47

```
public boolean onTouchEvent(MotionEvent event) {
    int action = event.getAction();
    if (action == MotionEvent.ACTION_DOWN ||
        action == MotionEvent.ACTION_MOVE)
    {
        float x mouse = event.getX() - padding;
        float width = (getWidth() - 2*padding);
        int progress = Math.round((float) getMax()*(x mouse/width));
        this.setProgress(progress);
    }
    return true;
}
```

Figura 3.46: Re-implementació del mètode onTouchEvent de l'HorizontalSlider

```
private OnProgressChangeListener listener;
public interface OnProgressChangeListener {
    void onProgressChanged(View v, int progress);
}
public synchronized void setProgress(int progress) {
    if (progress <1) progress = 1;
    super.setProgress(progress);
    if (listener != null) listener.onProgressChanged(this, progress);
}
```

Figura 3.47: Implementació d'un nou listener

Capítol 4

Segona Aplicació: Joc Senzill en OpenGL ES

En aquest capítol s'explicarà el desenvolupament de la segona aplicació desenvolupada en aquest projecte. En aquesta s'utilitzarà la llibreria gràfica 3D OpenGL ES desenvolupada per *Android* i la llibreria per comunicar dos dispositius mitjançant el servei de *GTalk Client*, anomenada *GTalk API*, que utilitza un protocol semblant a l'XMPP.

Aquesta segona aplicació intenta ser un senzill joc de Billar amb gràfics 3D. El joc serà jugat sempre per dos jugadors ja sigui en un mateix dispositiu o mitjançant una connexió a Internet amb diferents dispositius. D'altra banda, no es contemplarà la introducció de cap mena d'intel·ligència artificial per la implementació d'un jugador controlat per la CPU.

4.1 Objectius

L'objectiu d'aquesta segona aplicació és provar dos de les APIs que no han estat utilitzades en la guia de Barcelona. El primer objectiu era provar l'entorn gràfic que ofereix la plataforma. Per tal de fer-ho, es va decidir el desenvolupament d'un joc senzill 3D, al qual es pogués introduir algun element per provar la capa 2D. D'altra banda, com actualment estan molt de moda els jocs online, el segon objectiu és introduir aquesta faceta de manera que es pugui jugar a *BilliarDroid* des de diferents dispositius. Aquesta característica també s'incorpora per tal de provar la comunicació *GTalk* que incorpora Android.

Un objectiu secundari, el qual sorgeix amb la implementació de la comunicació mitjançant el servei *GTalk*, és la utilització de múltiples emuladors sobre la mateixa màquina. D'aquesta manera s'observaran els problemes que poden sorgir.

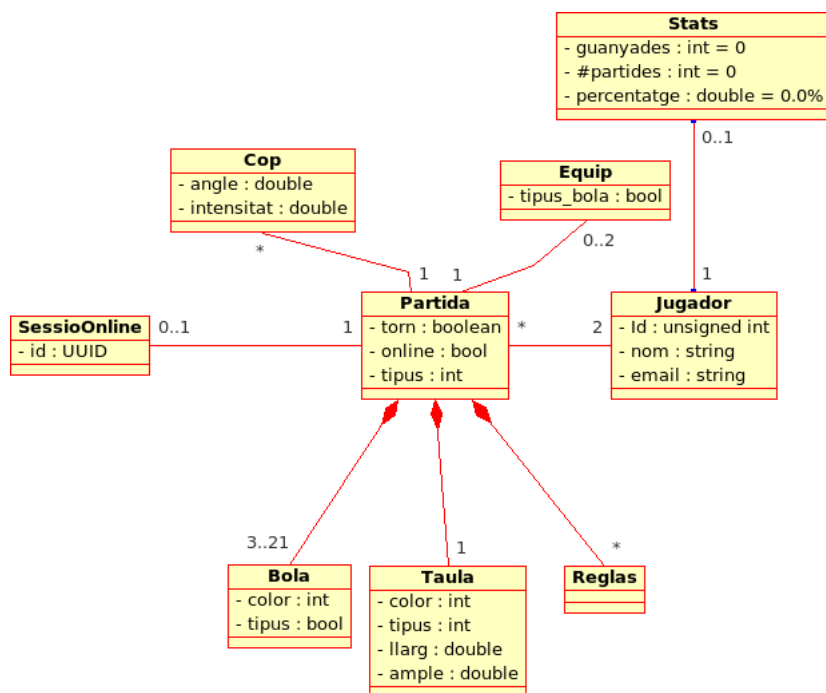


Figura 4.1: Diagrama de classes de BilliardDroid

4.2 Especificació i Disseny

Tal i com s'ha explicat en la secció 3.2 d'aquesta memòria, la metodologia de desenvolupament d'aplicacions no ha sigut el principal objectiu d'aquest projecte.

Una partida està composta d'un conjunt de regles segons el joc, un conjunt definit de boles i una taula d'una mida específica. A més a més, en una partida de billar sempre hi han de participar dos jugadors, els quals podran tenir una sèrie d'estadístiques segons les partides jugades i guanyades.

Segons la partida que s'estigui jugant existiran uns components variables. Si la partida és online, és a dir, la partida no es jugarà en dos dispositius diferents, existirà un identificador de partida per tal de poder enviar les accions de cada usuari. A més a més, segons les regles de la partida, existirà un tipus de bola que serà lligat al torn de la partida.

En la figura 4.1 observem el diagrama de classes d'aquesta aplicació. En aquesta s'observen els components anteriorment esmentats de la partida, la qual esdevé punt central de l'aplicació.

En el món del billar, existeixen moltes variacions en la manera de jugar, cadascuna amb un conjunt de regles diferents i una quantitat diversa de boles. En aquesta aplicació es va decidir que existirien les més típiques: bola 8, bola 9, *Snooker* i de bandes.

L'aplicació *BilliardDroid* constarà d'una sèrie de pantalles per tal de definir els jugadors i els tipus de partida que tindrà el joc. Aquest usuari en un principi, haurien

d'estar registrats per portar un control d'estadístiques que, en una versió posterior de l'aplicatiu, es podria enllaçar amb algun tipus de *ranking online*. El joc farà ús de la pantalla tàctil o del teclat per tal de generar el moviment del pal de billar. Es podrà moure també la bola blanca en cas de que el jugador contrari hagi sigut penalitzat per incomplir les normes del joc. El tir, d'altra banda, estarà representat per una barra de càrrega d'intensitat i un botó, per tal de fer un ús més fàcil de l'aplicatiu.

D'altra banda, el joc desenvolupat tindrà la capacitat de poder jugar online mitjançant el servei de missatgeria de Google. Per tant, per tal de jugar online en aquesta primer versió serà necessària l'obtenció d'un compte *Gmail*. Tanmateix, *XMPP* té l'avantatge de no necessitar un únic servidor, pel que es podria investigar la utilització del mòbil que inicialitza la partida com a part servidora de la comunicació.

4.3 Implementació

En aquesta aplicació s'ha volgut provar principalment dos parts de l'entorn de desenvolupament d'*Android*: la llibreria gràfica, tant 2D com 3D, i la comunicació entre dos emuladors mitjançant la connexió a Internet.

La primera secció està dedicada a tota la part de gràfics, mentre que la segona estarà dedicada als protocols de comunicació emprats per el desenvolupament de la comunicació.

4.3.1 Estructura de paquets

Aquesta aplicació es troba dividida en tres paquets diferents. Aquests paquets són temporals ja que es podrien crear de nous a mesura que l'aplicació es va implementant.

Primer de tot, la arrel del codi del projecte es troba a *upc.fib.android.games.billardroid*. En aquest paquet principal trobem totes les classes que implementen cadascuna de les *Activities* i algun dels seus components. Tanmateix aquest components no implementen cap tipus d'escena OpenGL, ja que totes les classes que utilitzen aquesta llibreria es troben sota el paquet *upc.fib.android.games.billardroid.opengl*.

D'altra banda, al paquet *upc.fib.android.games.data* trobem les classes que representarien el model de dades de l'aplicació. En aquest paquet queda per definit alguna classe del model. De moment es poden trobar classes del tipus partida, anomenada *BilliardGame.java*, o jugador, *Player.java*, entre altres.

Per últim, es disposa del paquet *upc.fib.android.games.billardroid.utils* que conté totes aquelles classes que contenen funcions que poden ser utilitzades generalment dins de l'aplicació.

4.3.2 APIs de Gràfics

Android intenta donar un impuls al entorn gràfic dels dispositius mòbils. La facilitat aportada per tal de poder generar les interfícies de les aplicacions se li suma la capacitat de desenvolupar aplicacions de gràfics 2D i 3D. Tal i com s'ha esmentat en el capítol d'anàlisi de la plataforma, a la secció 2.2.4, el desenvolupament 2D es fa sobre una llibreria gràfica anomenada SGL[37] i la 3D en OpenGL ES 1.0[26]. Existeix també una implementació de l'API d'*OpenGL ES 1.1*, encara que de moment no està suportada completament.

Aquesta part de l'aplicació ha costat bastant de ser desenvolupada per dues raons principals:

1. La poca experiència en desenvolupament usant OpenGL.
2. Les diferències entre la versió M3 i la M5 d'aquesta API en l'SDK d'*Android*.

El desenvolupament en OpenGL és diferent a un desenvolupament normal. Al treballar amb matrius s'ha d'anar amb molt de compte amb les instruccions que es van afegint, ja que la modificació de la matriu de modelatge pot provocar la representació d'una escena que no és l'esperada. En les següents seccions es farà una petita introducció a OpenGL.

D'altra banda, les diferències entre la versió M3 i la versió M5 no haurien d'haver estat una complicació ja que l'aplicació va ser desenvolupada després de la sortida de l'última versió utilitzada. El problema és que la majoria de desenvolupadors encara utilitzen la versió M3 SDK i la majoria de demostracions estan en aquesta versió. Això fa que tots els problemes que s'han trobat s'hagin hagut de solucionar sense cap tipus d'ajuda, ja que la documentació no és gaire extensa.

No obstant això, la demostració de *Google* anomenada *AndroidGlobalTime* i el codi del joc *MonolithAndroid*, que es troba disponible a Internet, han ajudat a solucionar alguns dels problemes trobats. Totes dues aplicacions han estat desenvolupades amb una llicència *Apache License 2.0*, la qual és compatible amb la versió 3 de *GPL*.

El component `BilliardGameSurfaceView`

Un dels canvis entre la versió M3 i la M5 de l'entorn de desenvolupament, és l'ús obligatori del *SurfaceView* com a component predestinat a contenir els objectes o escenes *OpenGL*. En un principi, es podia utilitzar components del tipus *View* en la reproducció d'escenes *OpenGL*, cosa que ha canviat amb la introducció de la nova versió. De fet, aquest canvi no havia estat actualitzat en la pàgina oficial de *Android*, on es trobava que per tal de desenvolupar una aplicació *OpenGL* s'havien de seguir tres passes:

1. Escriure una subclasse pròpia de *View*
2. Obtenir un manegador del *OpenGLContext*, que proveix l'accés a la funcionalitat *OpenGL* de la plataforma

```

gl.glBindTexture(GL10.GL_TEXTURE_2D, tex);
gl.glTexImage2D(GL10.GL_TEXTURE_2D, 0,
                GL10.GL_RGBA, width, height, 0,
                GL10.GL_RGBA,
                GL10.GL_UNSIGNED_BYTE, bb);
gl.glTexParameterx(GL10.GL_TEXTURE_2D,
                  GL10.GL_TEXTURE_MIN_FILTER,
                  GL10.GL_LINEAR);
gl.glTexParameterx(GL10.GL_TEXTURE_2D,
                  GL10.GL_TEXTURE_MAG_FILTER,
                  GL10.GL_LINEAR);

```

Figura 4.2: Inicialització de textures

3. Escriure els mètodes necessaris per representar l'escena en el mètode *onDraw()* de la subclasse *View* generada

A aquestes passes se li hauria d'afegir l'obligació d'utilitzar components del tipus *SurfaceView*, en comptes de *View*. Cal esmentar que la documentació oferta a la pàgina oficial de la plataforma no es troba constantment actualitzada, cosa que pot provocar alguna confusió a l'hora d'utilitzar algunes de les seves APIs

La classe *BilliardGameSurfaceView*, s'encarrega de gestionar tots els esdeveniments que es generin al dispositiu, tant per la pantalla tàctil com els esdeveniments de teclat. No obstant, encara que aquesta classe conté la capa on es dibuixarà l'escena 3D *OpenGL*, no s'encarregarà de dibuixar-la. Per tal de fer això, existeix una classe auxiliar, anomenada *BilliardGameThread*, que s'executa en un fil apart i s'encarrega d'aquesta feina. Això es degut a dos raons principals. La primera es que separar la capa de dibuix de l'escena es pot obtenir una abstracció de la capa de dibuix, la qual es pot modificar per a que funcioni de diferents maneres amb diferents interfícies. Tanmateix, la raó principal d'aquesta decisió va ser l'excepció, de punter nul, que llençava la plataforma *Android* durant la càrrega de textures al seu context. Aquesta excepció no se sap a què era deguda, ja que només passava al utilitzar el mètode de càrrega de les textures en 2D i no s'especificava la raó per la qual no funcionava correctament, simplement l'aplicació deixava de funcionar.

Aquesta solució es va adoptar gràcies, en part, a l'observació i investigació sobre el codi d'un joc lliure anomenat *MonolithAndroid*, en el qual la càrrega de textures es realitza en un fil secundari. En la secció 4.3.2 s'explicarà amb més detall la representació d'escenes 3D.

El component gràfic *BilliardGameSurfaceView* hereta directament de la classe *SurfaceView* i, tal i com s'ha observat en el codi de la figura 4.3, és l'encarregat d'inicialitzar el fil d'*OpenGL*. Depenent de l'esdeveniment del component pare que rebí el component desenvolupat s'inicia o es demana la parada del fil. S'utilitza el esdeveniment de *SurfaceChanged*, ja a l'última fase d'inicialització d'aquest component es genera aquest esdeveniment, esperant d'aquesta manera a que el component estigui correctament inicialitzat.

```

public void surfaceChanged(SurfaceHolder holder ,
    int format, int width, int height) {
    glThread = new BilliardGameThread(getContext(), layer2d, this);
    glThread.start();
}

public void surfaceDestroyed(SurfaceHolder holder) {
    glThread.requestExitAndWait();
    glThread = null;
}

```

Figura 4.3: Tractament dels esdeveniments de la SurfaceView

Per últim comentar que els esdeveniments que siguin capturats per aquest component, són tractats i passats al fil secundari, per tal de que aquest modifiqui la escena segons els esdeveniments rebuts.

Gràfics 2D - Overlays

Per tal de la reproducció d'elements gràfics en 2D s'utilitza la llibreria d'*Android* que implementa SGL que es troba dins del paquet *android.graphics*. SGL s'ha utilitzat per tal d'afegir contingut textual que informi de l'estat d'una partida. Aquests gràfics 2D també ha estat utilitzats en l'aplicació anterior en la part visual dels mapes, encara que no han estat explicats específicament. És per aquesta raó que aquest secció contindrà explicacions relacionades amb l'aplicació anterior.

El paquet *android.graphics* proveeix eines per dibuixar gràfics 2D en baix nivell tals com *canvas*, filtres de colors, punts i rectangles. L'API permet dibuixar aquests components directament sobre la pantalla utilitzant els component gràfics anomenats *Overlays*¹.

En aquesta aplicació, aquest paquet s'ha utilitzat per a generar una classe que s'encarregarà de reproduir tota la informació necessària de la partida. S'ha nombrat *BilliardGameOverlay* per tal de descriure quin tipus d'element és, encara que es podria haver nombrat *BilliardGameInfo*, indicant el tipus de servei que ofereix. El mètode més important de les classes que deriven d'*Overlay*, com en totes les classes que involucren gràfics, és el mètode *onDraw*, el qual permetrà treballar amb el canvas del component.

Qualsevol dels objectes que es dibuixin en aquestes capes necessita quatre elements per a ser representat. Primer de tot necessita un mapa de bits on ser dibuixat, el qual serà la capa desenvolupada anteriorment. També necessita un objecte que manegui totes les crides al mètode de dibuix *draw*, que és l'anomenat *canvas*. Per últim es necessita disposar de la primitiva a dibuixar i de l'estil i els colors amb els quals es dibuixarà. El paquet de grafics 2D disposa d'una classe anomenada *Paint* que ofereix la possibilitat d'assignar aquestes propietats a la primitiva escollida.

En l'aplicació *BilliardDroid*, s'ha dibuixat la primitiva de *Text* dos cops amb una matei-

¹La traducció al català és capa d'adalt.

```

statusTextPaint1 = new Paint ();
statusTextPaint2 = new Paint ();
statusTextPaint1.setARGB(200, 255, 0, 0);
statusTextPaint1.setTextSize(14);
statusTextPaint2.setTextSize(14);
statusTextPaint2.setARGB(255, 128, 128, 128);

```

Figura 4.4: Inicialització d'objectes Paint

```

public void drawString(Canvas canvas, String str, int x, int y)
{
    canvas.drawText(str, x+1, y+1, statusTextPaint2);
    canvas.drawText(str, x, y, statusTextPaint1);
}

```

Figura 4.5: Dibuix de text ombrejat a la capa 2D

xa cadena de caràcters per tal de donar un aspecte de superposició i ombra. D'aquesta manera, s'utilitzen dos objectes Paint que representaran cadascuna d'aquestes primitives amb un color diferent, ja que per fer l'efecte de ombra no es necessita res més. En la figura 4.4, s'observa com es fa la inicialització d'aquests objectes, tots dos de la mateixa mida però amb diferents colors.

Un cop ja hem inicialitzat l'estil en el qual volem dibuixar el nostre text, només fa falta indicar la posició i la cadena de caràcters que s'escriurà. S'ha generat una funció, anomenada *drawString* (figura 4.5), per tal d'escriure les cadenes de caràcters amb ombra, utilitzant els estils esmentats anteriorment.

Les altres primitives que han estat utilitzades en l'anterior aplicació, la guia de Barcelona, són rectangles, punts i línies entre dos punts. El paquet *android.graphics* els representa com *Rect*, *Point* i *Path*, respectivament.

Per exemple, a l'hora de dibuixar un camí amb la classe *Path* s'ha de especificar el tipus d'estil que es necessita entre aquests tres: *FILL*, *FILL_AND_STROKE* i *STROKE*.

L'estil anomenat *stroke*² marca la línia entre els punts donats, que seria el que serviria per a dibuixar un camí entre dos punts. D'altra banda i tal com el seu nom indica, l'estil *fill*³ emplenarà una àrea definida pels vèrtexs especificats. Per defecte s'utilitza aquest estil i, per tant, es pot fer servir aquesta classe com a eina de dibuix de figures donant una sèrie de vèrtex d'aquestes. Aquesta classe pot dibuixar tant línies rectes com circulars per tal d'establir la figura desitjada. Aquest paquet d'*Android* ofereix també la possibilitat d'afegir diferents objectes i textos durant el dibuix de la ruta. En el cas de la primera aplicació del projecte, el camí va quedar dibuixat un cop la ruta ja estava calculada per la classe *DrivingDirections*, tal i com s'observa en la figura 4.6.

A més a més, en aquestes capes 2D també es poden introduir imatges amb el mètode

²En català, acariciar

³En català, emplenar

```

for (MapPoint current : route) {
    if (current != null) {
        MapPointToScreenCoords(current, screenCoords, pxC);
        thePath.lineTo(screenCoords[0], screenCoords[1]);
    }
}
this.pathPaint.setStyle(Paint.Style.STROKE);
/* Dibuixem la ruta al canvas */
canvas.drawPath(thePath, this.pathPaint);

```

Figura 4.6: Dibuix de la ruta

DrawBitmap(). Aquest mètode també ha estat emprat en l'aplicació anterior, on es van incloure imatges per tal d'indicar la situació de l'usuari respecte els locals del seu voltant. S'identificarà cadascun dels local amb una imatge diferent dins del mapa. Per tal de situar aquestes imatges sobre el mapa es necessiten la posició de la cantonada sud-oest de cadascuna de les imatges. Com que en aquest cas la imatge consisteix en un clau que indicarà el inici i final de la ruta sobre un mapa, es necessita saber quina és la coordenada, en pixels, de la punta del clau per tal de situar-la a la posició desitjada del mapa. Tal i com s'observa a la figura 4.7, l'eix de coordenades de la pantalla d'un dispositiu *Android* es troba situat a la seva part superior esquerra, per tant s'haurà de restar la posició de la punta del clau a la posició on es vol que estigui apuntant. La figura 4.8 representa la manera de representar les imatges 2D mitjançant l'Objecte canvas del *Overlay*.

Gràfics 3D - OpenGL ES

En la part gràfica 3D, *Android* ha apostat per implementar una API que doni la possibilitat d'utilitzar la llibreria gràfica *OpenGL ES 1.0*. Aquesta API està integrada al paquet *javax.microedition.khronos.opengles*, del qual podem trobar més informació a la pàgina del projecte *OpenGL de Khronos*[26].

Tal i com el grup *Khronos* indica a la seva pàgina web, *OpenGL ES* és una API multiplataforma per la reproducció de gràfics 2D i 3D en sistemes encastats, com dispositius de mòbils, electrodomèstics o cotxes. És un conjunt ben definit de l'*OpenGL* d'escriptori, en concret seria compatible amb la versió 1.3, que creen una interfície de baix nivell flexible i potent entre el software i la capa d'acceleració gràfica hardware. *OpenGL ES* inclou perfils *Common* i *Common-Lite* per sistemes de punt flotant i de punt fix. En la plataforma *Android*, la versió 1.1 d'*OpenGL ES* s'emfatitza en la part d'acceleració gràfica de la API mentre que la 1.0 s'enfoca en activar implementacions a través només de la capa de software. La versió 1.1 de la llibreria és compatible cap en darrera amb la 1.0.

En aquesta aplicació, s'ha fet ús de la part d'*OpenGL ES 1.0*, que és la que principalment està suportada per *Android*. La raó per la qual no s'ha utilitzat la versió 1.1 amb acceleració gràfica és la poca documentació existent sobre la disponibilitat d'aquesta i


```
mGLContext = new OpenGLContext(
    PERFORMDEPTHTEST ? OpenGLContext.DEPTH_BUFFER : 0);
```

Figura 4.9: Creació del context OpenGL

que, en qualsevol cas, no dóna la sensació d'estar acabada de moment.

S'ha generat una classe del tipus *Thread*, per tal de executar tota la representació gràfica 3D, anomenada *BilliardGameThread*. Aquesta classe es troba dins del paquet *upc.fib.android.billiardroid.opengl*, acompanyada de tots els objectes 3D implementats per representar l'escena. Per tal de portar a terme la representació, el primer pas és obtenir el context d'*OpenGL* i crear alguns dels objectes a representar per pantalla. En el nostre cas

Un cop creat, el *thread* es llança amb el mètode *run* de la figura 4.10. Aquesta funció s'encarrega primerament d'enllaçar el context OpenGL a un component visual de la pantalla, derivat del tipus *SurfaceView*. Poc després, carrega totes les textures necessàries per a reproduir l'escena. Aquesta carrega és un procés molt lent, que incrementa la seva lentitud a mesura que existeixen més imatges a carregar. La fase final del mètode és una bucle on s'actualitza la vista 2D explicada anteriorment i es dibuixa la capa 3D. Per tant, es realitza una actualització continua de les capes 2D i 3D, el que farà que l'estat de la partida estigui sempre actualitzat.

Tal i com s'ha comentat anteriorment, la càrrega de textures ha sigut la principal causa de la creació d'un fil separat únicament per OpenGL. Això és degut a que cada cop que s'intentava carregar una imatge en el context l'aplicació donava una excepció, de manera que semblava que no s'hagués inicialitzat correctament la funcionalitat d'*OpenGL*. Finalment, la càrrega de les textures de les boles de billar ha quedat de la següent manera:

Com s'observa en el codi de la figura 4.12, es necessita indicar quins són els bits de la textura, en aquest cas una imatge rectangular en format PNG[22][figura 4.11]. Aquestes textures es guarden a memòria un cop carregades, per tal de millorar la velocitat de representació de l'escena. Per tal d'accedir a elles, es guarda cadascuna de les imatges en un vector de punters, representats com enters, on es podrà obtenir cada una de les imatges anteriorment carregades. Finalment, els mapa de bits d'aquestes imatges estaran assignats als vèrtexs de la figura d'una esfera a través d'un vector que indica quin vèrtex correspon a cada part de la figura.

A més a més, per tal de especificar les propietats que tindrà la textura s'utilitza el mètode *glTexParameterx* o *glTexParameterf*, on s'introdueixen la dimensió de la textura (1D, 2D o 3D), el nom del paràmetre a especificar i el seu valor. Els paràmetres que es poden ajustar en una textura són els següents: *GL_TEXTURE_MIN_FILTER*, *GL_TEXTURE_MAG_FILTER*, *GL_TEXTURE_MIN_LOD*, *GL_TEXTURE_MAX_LOD*, *GL_TEXTURE_BASE_LEVEL*, *GL_TEXTURE_MAX_LEVEL*, *GL_TEXTURE_WRAP_S*, *GL_TEXTURE_WRAP_T*, *GL_TEXTURE_WRAP_R*, *GL_TEXTURE_BORDER_COLOR* i *GL_TEXTURE_PRIORITY*.

```
public void run()
{
    // Lliguem el context amb el component
    mContext.makeCurrent(view.getHolder());
    GL10 gl = (GL10) (mContext.getGL());
    if (mTextures) loadBallTextures(gl);
    init(gl);
    while (!done)
    {
        if (isAnimating())
        {
            layer2d.postInvalidate();
            drawOpenGLScene(gl);
            mContext.post();
        }
        else
        {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    // Lliberem tots els objectes OpenGL
    mContext.destroy();
}
```

Figura 4.10: Mètode d'inicialització i repintat OpenGL



Figura 4.11: Textura de la bola número 8

```

balltextures=new int[numBalls];
gl.glGenTextures(numBalls, balltextures, 0);
BilliardBall[] balls = BilliardGame.getBalls();
for (int i=0 ; i<numBalls;i++)
{
    Bitmap bmp = BitmapFactory.decodeResource(context.getResources(),
        balls[i].getTextureId());
    ByteBuffer bb = GLUtils.extract(bmp);
    int tex = balltextures[i];
    int width = bmp.getWidth();
    int height = bmp.getHeight();
    gl.glBindTexture(GL10.GL_TEXTURE_2D, tex);
    gl.glTexImage2D(GL10.GL_TEXTURE_2D, 0,
        GL10.GL_RGBA,width, height, 0,
        GL10.GL_RGBA,
        GL10.GL_UNSIGNED_BYTE, bb);
    gl.glTexParameterx(GL10.GL_TEXTURE_2D,
        GL10.GL_TEXTURE_MIN_FILTER,
        GL10.GL_LINEAR);
    gl.glTexParameterx(GL10.GL_TEXTURE_2D,
        GL10.GL_TEXTURE_MAG_FILTER,
        GL10.GL_LINEAR);
}

```

Figura 4.12: Càrrega de textures OpenGL

Aquests paràmetres es troben explicats dins de la pàgina oficial d'*OpenGL*, encara que no es poden trobar a la pàgina d'*Android*. A més a més, la plataforma defineix tot una sèrie de constants per a treballar amb aquests valors en les classes *GL10* i *GL11*, dependent de la versió d'*OpenGL ES* que s'estigui emprant.

Gràfics 3D - GLUtils

OpenGL no és la única llibreria que trobem en la representació d'escenes. *GLU* és una llibreria que fa estalviar molta feina a la hora de treballar amb *OpenGL*. Aquesta conté funcions gràfiques de més alt nivell, que permeten realitzar operacions més complexes en *OpenGL*.

Android en la seva documentació oficial de la versió M5 SDK explicava que el paquet de *GLU* havia estat eliminat per tal d'estabilitzar la plataforma. Per això, es va intentar generar l'escena del joc sense la seva utilització. La dificultat va venir a l'hora d'implementar una de les funcions incloses en aquesta llibreria, anomenada *gluLookAt*, que situa la càmera en un punt de l'escena i, a partir del seu un vector normal, indica la quina direcció en la que s'enfocarà la càmera. L'especificació d'aquesta funció es pot trobar Internet[27] i, per tant, es va intentar implementar a partir d'aquesta i de diferents codis en C++[12].

Aquesta implementació no va sortir endavant, encara que segurament sigui a causa

d'alguna transformació mal configurada. Tanmateix, es va cercar per Internet més informació d'aquesta llibreria i en un exemple de la plataforma *Android* es va trobar la utilització del paquet *android.opengl.GLU*, la qual cosa no estava documentada en aquell moment⁴. A partir d'aquest moment, la implementació de l'escena va donar un salt en quan a velocitat de progrés.

Cal esmentar que aquesta llibreria no incorpora totes les utilitats disponibles normalment en aquest paquet, com per exemple, el dibuix de esferes o rectangles, que hagués sigut d'utilitat en el nostre joc. Els vèrtexs i tota l'informació de la esfera s'han extret de la demostració de *Google* anomenada *AndroidGlobalTime*, on s'utilitza una esfera amb un mapa mundi dibuixat que va voltant segons el horari que s'especifica. Es tracta d'una aplicació demostrativa amb llicència *Apache 2.0*, que il·lustra tot el que es pot arribar a fer amb aquesta plataforma de Google.

Grafics 3D - Objectes 3D

Una de les particularitats de l'API d'*Android* és que la introducció de les dades dels objectes 3D a representar en fa amb objectes del tipus *Buffer*. D'aquesta manera, tots els punts d'interès com vèrtexs, colors, índexs, etc. vindran representats per un determinat tipus de buffer segons el número de bits que es necessiti per a representar la quantitat d'elements que existeixen.

Per tal de dibuixar l'objecte existeix un tipus de buffer diferent per a cadascun dels atributs necessaris per a la seva representació. Primer de tot, s'ha d'especificar un vector de vèrtexs al qual hem anomenat *mVertexBuffer*, en el qual es situaran les diferents coordenades xyz dels vèrtexs dels objectes. En el cas de ser un objecte circular, com per exemple les boles de billar, una major quantitat de vèrtex donarà una millor definició de l'objecte. Això és degut a la impossibilitat de dibuixar corbes en gràfics per ordinador.

Un altre dels vectors necessaris és el que conté la unió entre els vèrtexs de l'objecte, anomenats índexs. L'ordre d'inserció dels índexs influiran en el dibuix de la figura. Amb aquest dos vectors definits, ja es podria representar un objecte en OpenGL ES d'*Android*. No obstant existeixen altres vectors que ajuden a fer una representació molt més realista, com són el de colors, les normals i textures.

Un cop es tenen tots els vectors complets, s'han de transformar a buffers de dades per a ser tractats posteriorment per OpenGL. Els vectors estaran guardats en buffers de tipus de dades diferents segons la seva mida. En la figura 4.13 hi ha un exemple de com tractar diferents mides de vectors.

A partir d'aquí, només falta implementar el mètode *onDraw* de l'objecte, en el que farà falta especificar quins són els paràmetres a dibuixar segons les característiques de l'objecte que es vulgui dibuixar. Per tal de dibuixar un objecte amb colors, s'ha definit un conjunt de colors, en format RGB, per cada vèrtex amb els quals es donarà el color general de l'objecte. És bo indicar quines són les normals de cada vertex per tal, per una banda, de dibuixar correctament els colors de les cares i, per l'altre, indicar quines

⁴En la nova versió de l'entorn de desenvolupament, s'indica que aquest paquet es troba disponible

```

private void allocate(int[] vertices, int[] texcoords, int[] normals,
    int[] colors) {
    ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * INT BYTES);
    vbb.order(ByteOrder.nativeOrder());
    mVertexBuffer = vbb.asIntBuffer();
    mVertexBuffer.put(vertices);
    mVertexBuffer.position(0);
    if ((texcoords != null) && emitTexCoordinates) {
        ByteBuffer tbb = ByteBuffer.allocateDirect(texcoords.length
            * INT BYTES);
        tbb.order(ByteOrder.nativeOrder());
        mTexCoordBuffer = tbb.asIntBuffer();
        mTexCoordBuffer.put(texcoords);
        mTexCoordBuffer.position(0);
    }
    if ((normals != null) && emitNorm) {
        ByteBuffer nbb = ByteBuffer.allocateDirect(normals.length
            * INT BYTES);
        nbb.order(ByteOrder.nativeOrder());
        mNormalBuffer = nbb.asIntBuffer();
        mNormalBuffer.put(normals);
        mNormalBuffer.position(0);
    }
    if ((colors != null) && emitCol) {
        ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length
            * INT BYTES);
        cbb.order(ByteOrder.nativeOrder());
        mColorBuffer = cbb.asIntBuffer();
        mColorBuffer.put(colors);
        mColorBuffer.position(0);
    }
}

```

Figura 4.13: Funció per tractar els vectors de dades d'*OpenGL*

```
gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);
gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);
gl.glNormalPointer(GL10.GL_FIXED, 0, mNormalBuffer);
gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
gl.glColorPointer(4, GL10.GL_FIXED, 0, mColorBuffer);
```

Figura 4.14: Assinació dels buffers a la representació

```
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, mTexcoordBuffer);
gl.glEnable(GL10.GL_TEXTURE_2D);
```

Figura 4.15: Textures OpenGL

son les cares que es veuen del objecte. El codi de la figura 4.14 serveix per a dibuixar qualsevol tipus d'objecte d'*OpenGL*. Malgrat això, els buffers i la matriu de projecció distingirien cadascun dels objectes de l'escena.

Si es desitja introduir un objecte amb textures, s'haurà d'especificar quin és el buffer que indica la manera de situar-les al objecte. El buffer de colors seria innecessari en aquest cas, ja que les textures són més prioritàries que els colors. En la figura 4.15 es té les instruccions a utilitzar en cas d'introduir textures en un objecte.

Finalment, es crida al mètode *drawElements*, figura 4.16 per tal de que l'objecte es dibuixi per pantalla. En aquest mètode s'indica quin són els índexs, elements que com anteriorment s'ha comentat serveixen per unir els vèrtexs. Si no s'introdueixen correctament, es podria generar una imatge o representació de l'objecte inesperada.

Resultat obtingut

El resultat obtingut després de la introducció de les capes 3D i 2D es pot observar a la figura 4.17. En aquesta s'observa que l'escena no es troba totalment definida, tanmateix l'objectiu d'aquesta aplicació no era tant el crear un joc sinó l'ús de les llibreries gràfiques de les que disposa *Android*.

4.3.3 Comunicació d'emuladors online

En aquesta secció indicarà els passos a seguir per tal de poder comunicar dos emuladors mitjançant la connexió a Internet. Es començarà per descriure el protocol *XMPP*,

```
gl.glDrawElements(GL10.GL_TRIANGLES, mIndexBuffer.capacity(),
GL10.GL_UNSIGNED_SHORT, mIndexBuffer);
```

Figura 4.16: Dibuix dels vectors anteriorment insertats



Figura 4.17: Escena 3D del billar

protocol base de les comunicacions de la llibreria de *GTalk* incorporada en la plataforma. Posteriorment, s'inclourà una descripció de la llibreria i es descriuran totes les passes per tal d'aconseguir una correcta comunicació entre emuladors.

D'altra banda, s'inclourà una explicació del gestor de notifikacions, ja que es una eina molt utilitzada en la recepció d'esdeveniments externs, que en aquesta aplicació vindran representats pels missatges entre dispositius.

Protocol XMPP

XMPP són les sigles angleses de *eXtensible Messaging and Presence Protocol*. Es tracta d'un protocol obert i ampliable basat en *XML*, originalment creat per a la seva utilització en missatgeria instantània. Al ser un protocol basat en *XML* hereta totes les característiques quant a adaptabilitat i senzillesa.

A diferència d'altres protocols propietaris, XMPP al ser un protocol estàndard obert es troba totalment documentat i s'insta a utilitzar-lo en qualsevol projecte. A més existeixen servidors i clients lliures, com per exemple el servidor *jabber*[24] que es utilitzar per moltes aplicacions de missatgeria.

El principal avantatge que aporta aquest protocol és que proveeix una comunicació descentralitzada. Això vol dir que qualsevol pot engegar el seu propi servidor XMPP, sense que les dades hagin de passar per un servidor central. D'altra banda, és considerat un protocol segur, ja que els servidor XMPP poder estar aïllats de la xarxa pública *Jabber*, allotjada a *jabber.org* i es poden introduir sistemes de seguretat robustos, com SASL[36] i TLS[51]. El port estàndard dels servidors XMPP és el 5222.

A part d'aquests protocols de seguretat, la *XMPP Standards Foundation* posa a disposició servidors amb funcions d'autoritat de certificació a *xmpp.net* per tal d'impulsar els sistemes de xifrat. Encara que *Gtalk* no és un protocol *XMPP* estàndard, s'observarà la gran flexibilitat que ofereix aquest protocol alhora d'extreure les funcionalitats més comunes.

El desavantatge més gran que aporta el protocol és la sobrecàrrega de dades de presència, cosa que fa que s'estiguin estudiant altres protocols semblants per reduir aquest problema. Al ser basat en *XML*, totes les dades binàries haurien de ser transmèses amb un altre protocol, o hauran de ser codificats en base64, tal i com s'ha fet anteriorment amb el protocol *SOAP* a la secció 3.3.6.

Per tal d'identificar els usuaris de la xarxa *Jabber* es té un identificador únic anomenat *JID*, *Jabber ID*. El *JID* està estructurat com a una adreça de correu electrònic, un nom d'usuari i la direcció DNS del lloc de residència de l'usuari. A més permet als usuaris introduir-se al sistema de missatgeria des de diverses localitzacions, gràcies a la introducció del lloc des d'on es connecta l'usuari. El servidor permet especificar una cadena de text, coneguda com a recurs, cosa que serà utilitzada en els servidors de *Gmail* per tal d'identificar sessions de dispositius *Android*. En aquest cas, els usuaris connectats des d'un dispositiu *Android* a la xarxa de *GTalk* s'identificaran dins del servidor amb el text "android" com a recurs del *JID*.

GTalk API

Tal i com s'ha explicat en el capítol 2, aquesta API va estar introduïda per permetre una comunicació instantània entre dispositius fluida i poc costosa. Aquest servei, *Gtalk*, inclou notificació de presència, pel qual un usuari pot determinar quan un altre usuari es troba en línia. A més a més, *Android* inclou un model programàtic similar al de l'enviament de SMS, és a dir, el remitent crida a un mètode per enviar un missatge i el receptor el rep mitjançant una implementació d'*IntentReceiver*. El servei *GTalkService* manté una connexió persistent per tenir un temps de resposta molt més ràpid que el del servei SMS, permetent una millor experiència d'usuari.

El sistema actualment manté una única sessió *GTalk* en el dispositiu, pel que tota la informació és carregada en una única connexió. Això significa que l'usuari del mòbil, no podrà utilitzar diferents comptes *GTalk* a la vegada des del seu dispositiu.

En aquesta aplicació, existeix una classe estàtica anomenada *OnlineGameSession*, que s'encarrega de la gestió de l'estat de la sessió, inicialitzant-la en el cas de que no ho estigui. A més a més, també s'encarrega de la gestió dels missatges, tant de l'enviament com de la rebuda de missatges amb la llibreria de *GTalk*.

Per tal d'utilitzar el servei correctament, actualment s'ha d'introduir el nom d'usuari i la contrasenya en la part de *DevTools* del sistema operatiu. Aquesta característica canviarà en la seva distribució final i se suposa que s'integrarà dins de la part de configuració dels dispositius.

Un cop es té el servei, es necessita una interfície per tal de poder-lo utilitzar. A la

```

public static void InitSession(Application app)
{
    application=app;
    application.bindService((new Intent()).setComponent(
        GTalkServiceConstants.GTALK_SERVICE_COMPONENT),
        mConnection, 0);
}
private static ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className,
        IBinder service) {
        IGTalkService GTalkService = IGTalkService.Stub.asInterface(service);

        try {
            mGTalkBilliardSession = GTalkService.getDefaultSession();
        } catch (DeadObjectException ex) {
            Log.e(LOG_TAG, "caught " + ex);
        }
    }
}

public void onServiceDisconnected(ComponentName className) {
    // Es crida quan la ósessi es tanca de forma inesperada
    mGTalkBilliardSession = null;
}
}

```

Figura 4.18: Utilització de la interfície de gTalk

figura 4.18 hi ha els dos mètodes que s'han d'implementar en aquest tipus d'interfícies: *onServiceConnected* i *onServiceDisconnected*. El mètode *onServiceConnected* es crida quan la connexió amb el servei *GTalk* ha estat establida, donant l'objecte per a comunicar-se amb el servei demanat. Aquesta comunicació es fa mitjançant una interfície *IDL*. D'altra banda, el mètode *onServiceDisconnected* es crida quan la sessió es tanca i, normalment, s'utilitza per tancar correctament les connexions en el cas d'una finalització inesperada de la comunicació.

A partir d'aquí, només fa falta indicar quin tipus de missatge s'enviarà i adjuntar-li les dades adequades. En el procés d'enviament no fa falta indicar quin és el remitent, ja que, com s'ha esmentat anteriorment, el dispositiu utilitza una mateixa connexió per tot, per tant el remitent estarà determinat pel dispositiu.

En el codi de la figura 4.19, s'observa que el destinatari té afegit un codi al final. Encara que el destinatari hauria de ser únicament una adreça de correu electrònic, actualment la comunicació, sense la introducció d'aquest sufix, no s'executa correctament i es perden missatges. Aquest codi forma part del *JID* comentat anteriorment, es suposa que serà l'especificació del lloc des d'on es connecta l'usuari. L'error que la plataforma llença quan no s'introdueix aquest codi és "*can't find the full JID for xxx@gmail.com*".

Cal esmentar que aquesta API ha estat eliminada de la nova versió del entorn de desenvolupament. Diversos problemes de seguretat, com el poder extreure totes les dades d'un usuari durant una trucada, han fet que l'*Open Handset Alliance* hagi decidit no

```

private static void SendMessage(String dest, Intent intent)
{
    try {
        mGTalkBilliardSession.sendDataMessage(
            dest+"/androidwCxwXphYj3JM", intent);
    } catch (DeadObjectException ex) {
        Log.e(LOG TAG, "caught " + ex);
        mGTalkBilliardSession = null;
        application.bindService((new Intent()).setComponent(
            GTalkServiceConstants.GTALK_SERVICE_COMPONENT),
            mConnection, 0);
    }
    catch (Exception e) {
        Log.e("BilliardDroid", e.getMessage().toString());
    }
}

```

Figura 4.19:

incloure l'API dins de la versió final de la plataforma. Tanmateix, se suposa que en alguna versió posterior es solucionaran aquests problemes i es podrà utilitzar novament aquesta llibreria. D'altra banda, la solució seria utilitzar alguna altre llibreria de J2ME, que funcionés amb Android, per tal de comunicar diferents dispositius a través de la xarxa. Aquesta llibreria podria ser, per exemple, una llibreria de comunicació mitjançant *XMPP*.

Format dels missatges entre dispositius - Descripció del protocol

Per tal de que la comunicació entre dispositius fos fructuosa a l'hora d'intercanviar informació sobre les partides, s'ha generat un protocol propi de comunicació. En aquest protocol existeixen quatre tipus de missatges diferents, els quals informen l'usuari de l'estat de la partida.

- BILLIARDROID_START_SESSION
- BILLIARDROID_ACCEPT_SESSION
- BILLIARDROID_CANCEL_SESSION
- BILLIARDROID_GAMEPLAY

A l'hora de començar una partida, l'usuari amfitrió genera un identificador únic de partida anomenat *sessionID*. Aquest identificador es genera a partir de la classe *UUID* de Java, la qual genera un nombre hexadecimal únic que es transmet com si fos una cadena de caràcters. Un cop generat aquest identificador, el dispositiu envia un missatge del tipus *BILLIARDROID_START_SESSION* al destinatari escollit per l'usuari, indicant el seu nick i els paràmetres de la partida que ha escollit. Un cop fet això, l'usuari es posarà

en espera, fins rebre una notificació del destinatari. Existeix un temps màxim d'espera de tres minuts, a partir d'aquest moment el dispositiu cancel·la la partida per mitjà d'un missatge de cancel·lació de la sessió, del tipus *BILLIARDROID_CANCEL_SESSION* i es demanarà a l'usuari que es posi en contacte amb el destinatari o que esculli un altre usuari.

Per un altra banda, el destinatari rebrà una notificació a la barra de notifiacions del seu dispositiu informant de que un altre usuari intenta convidar-lo a una partida de billar. Si accepta, haurà d'enviar un missatge del tipus *BILLIARDROID_ACCEPT_SESSION* amb el nick amb el qual se'l reconeixerà. En cas contrari, si no accepta la invitació, enviarà un missatge de cancel·lació indicant el mateix número de sessió que l'altre usuari havia creat. És obligatori la indicació del número de sessió sempre que s'envii un missatge. Això fa que es descartin els missatges d'altres usuaris que s'introdueixin durant la comunicació entre els dos participants del joc (usuaris malintencionats o errors). En cas d'acceptar la participació en una nova partida durant el transcurs d'una altra, s'enviarà un missatge de cancel·lació al usuari de la partida anterior i un d'acceptació a l'usuari de la nova partida.

Un cop les dues parts es troben en contacte, compartint una sessió amb un mateix identificador, comença el joc. El joc es gairebé independent de la comunicació, ja que cada dispositiu farà els càlculs corresponents per a dibuixar l'estat de la partida en cada moment. Per tant, l'únic que es necessita es saber qui és l'usuari que està jugant en un instant determinat, i quina intensitat i direcció agafarà la bola blanca. Cal esmentar, el torn s'especifica en el moment en que s'inicia la partida i, per tant, no formarà part d'aquests tipus de missatges.

Actualment, la part del joc que està implementada no ha arribat a ser jugable, pel que els botons d'intensitat i direcció encara no formen part de l'aplicació. Tanmateix, es podrà observar com es canvia d'usuari a mesura que cadascun dels jugadors prem el botó *Shot*. A més a més, s'ha implementat que la bola blanca vagi voltant per la pantalla depenent de qui estigui tirant, per donar sensació de moviment.

Notificacions al sistema

Android incorpora una sèrie de restriccions per tal de que determinats esdeveniments no puguin arrancar aplicacions. Això és degut a que si no fos d'aquesta manera l'usuari podria ser interromput en l'ús concret del seu dispositiu, per un esdeveniment no desitjat. L'únic esdeveniment que té prioritat és la recepció de trucades, ja que és l'objectiu principal dels telèfons mòbils.

Tanmateix, *Android* disposa d'un component per tal de poder avisar a l'usuari dels esdeveniments que es van generant al seu sistema. Aquest component s'anomena *Notification Manager*, és a dir, el gestor de notifiacions, i es troba amagat a la part superior de les pantalles dels dispositius. Aquest notificador permet a l'usuari executar les accions per defecte que un esdeveniment tingui definida. Per tal d'enviar una notificació al sistema, s'ha d'introduir l'identificador de l'aplicació que el notifica i s'ha d'especificar un Intent amb l'acció per defecte que s'executarà. En el exemple de la figura 4.20, s'ob-

```

protected void showNotification(Context context, Bundle extras, String message) {
    NotificationManager nm = (NotificationManager)
        context.getSystemService(context.NOTIFICATION_SERVICE);
    Intent contentIntent = new Intent(context, OnlineGameAlertActivity.class);
    if (extras != null)
    {
        extras.putString("message", message);
        contentIntent.putExtras(extras);
    }
    Intent appIntent = new Intent(Intent.VIEW_ACTION, Sms.Inbox.CONTENT_URI);
    Notification notif = new Notification(
        context, // context de 'l'aplicacio
        R.drawable.icon_small, // icona
        message, // descripcio
        System.currentTimeMillis(), // temps 'darribada
        "BilliarDroid", // titol
        message, // missatge ampliat
        contentIntent, // accio a executar prement el missatge
        R.drawable.icon, // icona de 'l'aplicacio
        context.getText(R.string.app_name), // nom de 'l'aplicacio
        contentIntent // accio a executar a étravs de 'licona
    );
    // Despres de 100ms vibrar durant 250ms, esperar 100ms i
    // tornar a vibrar durant 500ms
    notif.vibrate = new long[] { 100, 250, 100, 500 };
    nm.notify(R.string.app_name, notif);
}

```

Figura 4.20: Enviament d'una notificació al sistema

servarà la creació d'una notificació del joc de billar en la qual es preguntarà a l'usuari si vol jugar un tipus determinat de partida.

D'altra banda, sembla ser que, de moment, l'única manera de eliminar-los és a través de codi en Java. Cada notificació té associat un identificador de grup o d'aplicació amb el qual es poden cancel·lar les notificacions que no interressi mostrar, tal i com es fa en l'exemple de la figura 4.21.

Tal i com es troba actualment implementat, en el cas de que una notificació de l'aplicació fos cancel·lada, també es cancel·larien totes les altres notificacions creades per aquesta aplicació. Això fa que l'usuari hagi d'escollir la notificació que sigui adequada en cada moment, ja que, un cop escollida aquesta notificació, les altres notificacions quedaran eliminades. La solució a aquest problema seria assignar-li a cada notificació, el nom de la sessió a la qual pertanyen.

Comunicació entre dos emuladors a la mateixa màquina

Comunicar dos emuladors per tal de fer proves en la mateixa màquina normalment no funciona amb total normalitat. A l'executar dos emuladors per defecte desde l'IDE

```
private void cancelNotifications ()
{
    NotificationManager nm =
        (NotificationManager) getSystemService (NOTIFICATION_SERVICE);
    nm.cancel (R.string.app_name);
}
```

Figura 4.21: Cancel·lació de notifikacions

d'*Eclipse*, un dels dos emuladors no guarda les dades de l'usuari. Això provoca problemes a l'hora d'introduir el compte *Gmail* de l'usuari en un dels emuladors i, per tant, dóna problemes al connectar-se amb la connexió a *gTalk*.

El millor que es pot fer és arrancar els emuladors des de la *shell* de Linux amb la comanda *emulator* amb diferents directoris de dades d'usuari. Això es faria utilitzant la comanda *./emulator -datadir userdata/* des del directori on s'hagin instal·lat les eines que proveeix la plataforma. D'aquesta manera, l'emulador crearà una imatge on guardarà totes les dades de configuració del dispositiu i les aplicacions instal·lades.

Un cop generades les dues instàncies de l'emulador, s'ha d'instal·lar l'última versió del joc *BilliarDroid* en cada una de elles i indicar quin és el compte *GTalk* a utilitzar en cadascuna. Aquesta solució, de vegades dóna problemes que no deixen inicialitzar el compte *GTalk* correctament, cosa que fa que s'hagi de repetir el procés fins que el registre es completi correctament. Tanmateix, es té constància que l'última versió de l'emulador conté millores que fan que les proves amb múltiples emuladors no donin cap tipus de problemes.

Capítol 5

Valoració Econòmica

Aquest capítol inclourà una valoració, tant econòmica com temporal, del desenvolupament del projecte. Al ser un projecte d'anàlisi d'una plataforma, aquests costos no tindran una importància significativa i serviran per demostrar quin temps s'ha estat invertit en ell.

5.1 Cost Temporal

A continuació es detallarà el conjunt d'hores que s'han dedicat a cada una de les tasques.

Tasca	Hores
Estudi Plataforma	50
Tutorialització i Demos	40
Disseny d'Aplicacions	12
Primera Aplicació	248
Estructura Aplicació	16
Content Provider	64
Implementació Servidor WS	40
Creació Client WS	32
Reproducció Multimèdia	48
Linkify	8
Proves	40
Segona Aplicació	158h
Interfície	4
OpenGL ES i Demos	110
Protocol GTalk	24
Proves	20
Documentació	220
TOTAL	728

El temps d'estudi de la plataforma és el temps que es va dedicar a llegir tota la documentació que ofereix la plataforma i a fer la instal·lació de tots els elements necessaris per al desenvolupament de les aplicacions del projecte. A més a més, en aquest període també s'ha inclòs els temps emprats en les primeres reunions amb el director del projecte.

Després de l'estudi de la plataforma, hi ha hagut un període d'experimentació amb les diferents parts de la plataforma. En aquesta part s'han desenvolupat petites aplicacions a partir de tutorials i demostracions dels diferents fòrums que s'han anat utilitzant. Aquesta experimentació ha servit per a enfocar les possibles aplicacions a desenvolupar en el projecte. Aquest disseny ha estat efectuat en paral·lel amb l'estudi de la plataforma *Android* i el període de aprenentatge. A partir dels diferents dissenys es van escollir les dos aplicacions que constitueixen el projecte.

Les seccions de desenvolupament de les aplicacions estan detallades per a cada part de l'aplicació. Tanmateix, a la segona aplicació s'observa una fase de demostracions que van servir per a estudiar el desenvolupament d'escenes 3D en *OpenGL*. En el temps desenvolupament de la guia de Barcelona, no s'ha inclòs el disseny de les icones d'aplicació, ja que es considera que no forma part rellevant del projecte.

Per finalitzar, el temps de documentació està format pel temps de cerca d'informació i redacció de la memòria. A més a més, també s'hi ha inclòs el temps d'aprenentatge de l'entorn *LaTEX*, que forma part dels requisits del projecte.

5.2 Cost Econòmic

En aquest apartat s'efectua una estimació del cost econòmic del projecte, tenint en compte el nombre d'hores dedicades a les tasques del apartat anterior. Per fer els càlculs s'utilitzarà un cost de 45 euros per l'hora d'Analista Programador, que es el perfil que hem decidit assignar al desenvolupador del projecte.

Tot i que les aplicacions no estan realment acabades, s'indicarà el cost de cadascun dels desenvolupaments del projecte sense tenir en compte el procés d'aprenentatge de la plataforma. El cost de les tasques de documentació sobre *Android* i de l'escriptura de la memòria estaran assignats al apartat *Altres* del requadre posterior.

Part del projecte	Preu
Primera Aplicació	8680
Segona Aplicació	5530
Altres	11870
TOTAL	26080

El cost total indica la quantitat de diners necessària per tal d'assolir els objectius d'aquest projecte d'anàlisi de la plataforma *Android*. En aquest cas, aquest projecte

ha obtingut un cost final de 26080 euros.

Capítol 6

Conclusió

En aquest capítol s'explicarà quins objectius han estat aconseguits dins del marc del projecte i quin serà el treball futur per tal d'arribar de acabar les aplicacions desenvolupades durant el seu termini. Tanmateix, es farà una conclusió final dels avantatges reals que proporciona la plataforma Android, dins del marc del món dels dispositius mòbils.

6.1 Objectius Aconseguits

Aquest projecte d'investigació i anàlisi de la plataforma *Android* ha assolit la gran majoria dels objectius que es van fixar al principi.

Per una part, s'ha realitzat un treball d'investigació constant sobre la plataforma i la forma de paliar els problemes es poguessin presentar. Primer de tot, s'ha utilitzat la documentació proporcionada per *Google* com a guia per al desenvolupament de les diferents aplicacions del projecte. Tanmateix, les investigacions extretes per mitjà d'altres desenvolupadors en els fòrums de desenvolupament d'*Android* han aportat un valor considerable. S'ha de donar les gràcies a aquests fòrums i als blogs per explicar experiències en el desenvolupament en aquesta plataforma, com *Android-Spa*[2] i *Anddev*[1], ja que han aportat solucions a alguns dels problemes trobats. En el primer d'ells, s'ha participat activament per tal d'ajudar a altres desenvolupadors.

D'altra banda, la part dels objectius de la primera aplicació ha estat totalment acomplida. L'aplicació s'ha desenvolupat utilitzant totes les APIs que s'havien comentat als objectius, la gran majoria de les ofertes per la plataforma. A més a més, s'han afegit llibreries J2SE i J2ME, totalment externes a l'entorn d'*Android*, per a comprobar el grau d'adaptabilitat de la plataforma amb diferents llibreries de l'entorn Java. A més a més, l'accés a serveis web obre la possibilitat de què els dispositius *Android* puguin obtenir informació a través d'altres plataformes i llenguatges.

Les APIs d'*OpenGL ES* i *GTalk*, primerament anomenada *XMPP API*, han estat provades en la segona aplicació. En la part de gràfics, a més, s'ha pogut provar la integració d'interfícies amb tres tipus de components: la capa 3D, la capa 2D i altres components

visuals. D'altra banda, per intercanviar informació entre diferents dispositius s'ha creat un protocol de comunicació propi de l'aplicació a partir de la comunicació *XMPP de GTalk*.

Per finalitzar, comentar que no s'ha migrat cap aplicació de J2ME a *Android* per manca de temps. No obstant com s'ha comentat en el primer capítol, s'ha trobat un projecte de final de carrera[6] que marcava les passes a seguir per tal de migrar aquest tipus d'aplicacions.

6.2 Treball Futur

En aquesta secció s'explicarà tant el treball a realitzar en cadascuna de les aplicacions desenvolupades en aquest projecte com les característiques de la nova versió de l'entorn de desenvolupament de la plataforma *Android* que falten per provar.

6.2.1 OcioBCN

L'aplicació *OcioBCN* per a la plataforma *Android* està pràcticament acabada. Faltaria acabar de solucionar el problema de l'*streaming* de vídeos, que en principi pot estar solucionat en la nova versió de l'entorn de desenvolupament. D'altra banda, la introducció d'una gestió de rutes a peu seria una característica desitjable per a aquest tipus d'aplicació.

Tanmateix, falta millorar l'aplicació web per introduir les dades dels locals. Aquestes dades podrien ser extretes d'alguna pàgina web de locals d'oci, la qual estigués associada amb l'aplicació desenvolupada i oferís uns serveis web per actualitzar-la. A part d'una millora de la interfície i d'usabilitat, seria bo introduir un servidor de vídeos *streaming*. Com la principal idea d'aquest servidor és la reproducció dels tràilers de les pel·lícules al mòbil, serà necessari obtenir les llicències que siguin necessàries per a reproduir aquest tipus de vídeos legalment. Tanmateix, es pot esperar a que les pròpies productores generin aquests vídeos i així enllaçar-los amb l'aplicació.

6.2.2 BilliarDroid

Aquesta aplicació, al ser la segona del projecte ha estat una mica limitada de temps. Això significa que encara hi ha gran part de l'aplicació que no ha estat implementada. Bàsicament s'ha analitzat l'ús de *Android* per al desenvolupament d'un joc i faltaria desenvolupar tota la dinàmica del joc.

Primerament s'hauria de completar el disseny 3D per tal de dotar de major realisme al joc, que actualment està format per una superfície rectangular que fa de taula i un conjunt de boles amb textures, cosa que, de moment, es podria quedar d'aquesta manera. El modelatge 3D és una feina bastant complicada si no s'ha treballat un temps amb ella, pel que es podria intentar aconseguir un dissenyador gràfic que treballés amb

aquesta part del joc. D'altre banda caldria implementar tots els moviments de les boles de billar, és a dir la simulació dels xocs elàstics entre múltiples boles.

La darrera part és potser una de les més atractives dins d'un joc: la intel·ligència artificial del joc. És segurament la part més complicada del desenvolupament de qualsevol joc, encara que va lligada al nivell de dificultat que es vol obtenir. Per tal de desenvolupar-la, s'hauria de pensar quin tipus d'algoritme utilitzar, amb diferents heurístics segons el nivell de joc.

6.2.3 Migració a la versió 1.0 de l'SDK

Durant la redacció d'aquesta memòria, l'*Open Handset Alliance* ha tret dos versions, entre elles la versió definitiva del entorn de desenvolupament d'*Android*, versió 1.0. D'aquesta manera, els desenvolupadors hauran de basar-se en aquesta versió per tal de lliurar les primeres versions de les seves aplicacions.

Per aquesta raó, s'hauria de migrar ambdues aplicacions a la nova versió, modificant tot allò que estigui desfasat. A més a més, s'haurien de provar les noves característiques de la nova versió com:

- Previsualització gràfica de les interfícies en *XML*
- Eina per generar icones de 9 polsades
- Provar l'API de *MediaPlayer* que ha estat millorada
- La nova API de Wifi

Amb tot això es podria considerar que l'anàlisi de la plataforma quedaria finalitzada. El primer mòbil de la plataforma, va sortir el 22 d'octubre de l'any 2008 i és distribuït per *T-Mobile*[43]. Aquest mòbil és anomenat *T-Mobile G1*[44].

6.3 Plataforma Android

En aquest projecte de final de carrera, ha quedat provat que si totes les expectatives es compleixen, *Android* pot tenir un començament brillant en el món dels dispositius mòbils. Al ser *Java* la tecnologia escollida, ha fet que desenvolupadors d'arreu del món que ja estaven acostumats a treballar en aquest llenguatge desenvolupin i provin els aspectes que ofereix la plataforma. A més a més, el període d'adaptació és curt, només fa falta seguir els tutorials de la seva pàgina oficial per entendre les particularitats que pot tenir una aplicació en aquests tipus de dispositius. D'altra banda, la capacitat de la plataforma per absorbir llibreries externes desenvolupades amb *Java*, sobretot les desenvolupades per a *J2ME*, dona una projecció immillorable a *Android*.

També ha quedat demostrat que encara que tot són bones sensacions, encara falta molt per millorar. S'ha de considerar que és una plataforma que encara es troba en fase de

desenvolupament i això fa que sigui totalment inestable. Això dona certs problemes, ja que els desenvolupadors són, en realitat, *betatesters* i per tant saben que les aplicacions que realitzin poden quedar anul·lades en la sortida de noves versions. Aquest és el cas, per exemple, de les aplicacions que han estat desenvolupades utilitzant la llibreria de *GTalk*. Aquesta API ha estat, de moment, suprimida per qüestions de seguretat en la nova versió de l'entorn de desenvolupament i actualment és segur de que no es podrà utilitzar en els primers dispositius *Android* del mercat.

A més a més, la falta d'implementació d'una API per poder utilitzar Bluetooth fa que *Android* prescindeixi d'una de les característiques de comunicació més utilitzada en aquests dispositius. Actualment, la majoria dels dispositius no disposen d'interfícies wifi, pel que la tecnologia Bluetooth pot ser, a més a més, un medi de comunicació amb altres plataformes i/o dispositius.

Un dels altres aspectes negatius, és la falta de documentació adequada i actualitzada. S'ha arribat a trobar durant el transcurs del projecte informació sobre les APIs que no s'adequava a la versió del moment. És per això que alguna de les APIs ha estat provada a base de prova i error.

Tanmateix, tot i aquests petits inconvenients, *Android* ha donat als desenvolupadors un sistema amb el qual existeix una gran facilitat per al desenvolupament d'aplicacions. Actualment, només *Apple* ha aconseguit donar una plataforma de desenvolupament completa per al seu sistema. La principal diferència entre aquestes plataformes és que la plataforma d'*Apple* disposa d'un únic mòbil al mercat, l'*iPhone*, cosa que fa que els desenvolupament no s'hagin d'adaptar a diferents models de dispositius. *Android* ha intentat entrar d'una altra manera en aquest món, ha volgut crear un sistema amb el qual una mateixa aplicació serveixi per a una gran quantitat de mòbils.

D'altra banda, a falta de saber quin serà el conjunt de software finalment utilitzat, *Android* aporta suficients llibreries com a per a que tant l'usuari com el desenvolupador pugui gaudir d'una grata experiència amb la plataforma. A més a més, les millores incorporades en l'última versió poden solucionar alguns dels errors d'aquestes llibreries trobats durant el desenvolupament, com per exemple la reproducció de vídeos mitjançant *streaming*.

Les llibreries ofertes per l'entorn de desenvolupament d'*Android* ofereixen funcionalitats per a desenvolupar gairebé qualsevol tipus d'aplicació per a mòbil. En el context del projecte, només s'ha afegit una llibreria externa, *KSOAP*, per a accedir a serveis web. Tanmateix, això no es considera un inconvenient sinó un avantatge, ja que *Android* ha sigut capaç d'adaptar una llibreria *J2ME* al seu entorn propi. Segons la documentació[18], *Android* seria compatible amb gairebé totes les llibreries de *J2ME* i algunes llibreries de *J2SE*. Cal esmentar que primer es va intentar adaptar la llibreria *AXIS* de *J2SE* sense èxit i, per això, es va optar per utilitzar *KSOAP*.

La introducció de la llibreria d'*OpenGL ES* en el món dels mòbils és un dels aspectes més importants. Caldrà veure quin rendiment poden obtenir en aquests dispositius, ja que encara no s'ha disposat de cap demostració real. D'altra banda, l'energia consumida per aquests tipus d'aplicacions pot ser un punt negatiu de la plataforma, ja que sempre és important tenir en compte quina és la principal necessitat dels usuaris d'a-

quest dispositiu: mantenir el mòbil encès per poder rebre i emetre trucades. Tanmateix, si aquest no és un inconvenient, les aplicacions en les quals s'utilitzin gràfics experimentaran un gran canvi, ja que fins ara la majoria estaven desenvolupades en *J2ME*. A més a més, qualsevol desenvolupador que ja hagi utilitzat la tecnologia *OpenGL* no tindrà cap tipus de problema en utilitzar la llibreria d'*Android*, la qual se suposa que anirà creixent per integrar totes les funcionalitats d'aquesta tecnologia que no estan disponibles actualment, com les *glutills*.

Durant el projecte s'ha tingut l'oportunitat de parlar amb un desenvolupador de jocs per *iPhone*. El desenvolupador ha comentat, que la principal avantatge de desenvolupar jocs per a aquest dispositiu és la possibilitat de poder implementar la representació de l'escena amb *OpenGL*, tal i com es podrà fer amb *Android*. Malgrat això, en aquest cas l'*iPhone* pot ser superior a *Android* ja que el seu entorn de desenvolupament fa ús del llenguatge *C++* per a la implementació de les seves aplicacions, cosa que permet portar el control de la memòria utilitzada. *Android* al utilitzar *Java* perd aquest control als seus dispositius. Aquest serà un dels grans inconvenients a l'hora de desenvolupar un joc per aquesta plataforma. D'altra banda, caldrà veure si la gestió de memòria del dispositiu és millor que l'esperada.

A més a més, el plugin *ADT* d'*Eclipse* ofereix una sèrie d'eines per a facilitar el desenvolupament. Malgrat això, durant el transcurs del projecte s'ha trobat a faltar un editor visual d'interfícies per a les aplicacions. El disseny de les interfícies s'efectua a través de documents *XML* pel que l'única manera de comprobar el resultat és executant l'aplicació. Tanmateix, el disseny manual d'aquestes interfícies resulta més fàcil que amb la llibreria *Swing* de *Java*, en la qual es realitza tot mitjançant codi font. Cal esmentar que en la última versió del plugin, desenvolupat només per a la versió 1.0 de l'*SDK*, s'ha introduït un editor que ajuda a generar el document de *layout* de les finestres de la aplicació.

D'altra banda, *Android* ens ofereix també un emulador per a provar les aplicacions. Aquesta eina és molt útil, encara que té un gran defecte: consumeix molts recursos de l'ordinador. A més a més, el procés d'engega d'aquest pot tenir una durada de 10 minuts segons la màquina que s'estigui utilitzant. En el context d'aquest projecte es va utilitzar un portàtil *Intel*[23] *Pentium Centrino 1.5* amb 1Ghz de memòria *RAM*. D'altra banda actualment no disposa de simulació de *Bluetooth* i la seva capacitat de gràfics no ve adequada a la potència de la targeta gràfica de la qual disposi el desenvolupador, pel que serien dos millores considerables. Actualment, la reproducció de gràfics a l'emulador fa ús d'una simulació *OpenGL* mitjançant únicament el processador.

Un dels problemes que, segurament, tindrà *Android* és la incompatibilitat amb aplicacions *J2ME*, dos entorns que utilitzen el llenguatge *Java* però cadascun amb les seves particularitats. El problema sorgeix en el fet de que hi ha moltes aplicacions ja desenvolupades en *J2ME* que haurien de ser migrades a la tecnologia proporcionada per *Android*. Aquesta feina pot ser no massa complicada en el cas d'una sola aplicació, pel contrari, pot ocupar moltes hores de treball en cas de tenir-hi múltiples aplicacions.

Tot i això, aquesta plataforma pot estandarditzar el món del mòbils i, segurament, farà que aquest grup de desenvolupadors canviï d'opinió, sobretot si els usuaris es decanten

per utilitzar dispositius Android. De moment el primer mòbil del mercat no es troba en estoc per l'elevada demanda, faltirà veure quin grau d'acceptació tindran els usuaris en aquest nou entorn.

6.4 Documentació LaTeX

La memòria d'aquest projecte ha estat redactada en *LaTeX*. Aquest llenguatge de marques permet la creació de documents científics molt estructurats sense la necessitat de donar format a cada part. *LaTeX* introdueix una sèrie de paquets que ajuden a fer aquesta tasca, només indicant quin tipus d'element s'està utilitzant en cada moment i quin tipus de document es vol crear. Tanmateix, és un llenguatge que requereix un aprenentatge per tal d'utilitzar totes les seves funcionalitats correctament, pel que s'ha dedicat temps del projecte a documentar-se. A més a més, s'ha utilitzat l'editor *Kile* per tal de facilitar la redacció.

LaTeX ha sigut de gran utilitat per referenciar de bibliografia i altres parts del document. Un dels inconvenients, no s'ha pogut determinar la situació de les figures al text, cosa que fa que normalment les figures es situïn al final d'una part del document. He de donar les gràcies a *Antoni Soto i Riera*, director d'aquest projecte, per la ajuda que m'ha prestat en els diferents problemes en aquest entorn.

Per concloure, la redacció amb *LaTeX* dona un format de document estàndard segons una sèrie de paràmetres, pel que qualsevol text científic o professional hauria d'estar redactat amb aquesta eina.

Bibliografia

- [1] Android Development Community. Anddev.org, 2007. <http://www.anddev.org/> [Online; accessed 11-September-2008].
- [2] Android-Spa Team. Android-Spa, 2008. <http://www.android-spa.com/> [Online; accessed 11-September-2008].
- [3] Apple Inc. Apple, 2008. <http://www.apple.com> [Online; accessed 11-June-2008].
- [4] Apple Inc. iPhone, 2008. <http://www.apple.com/iphone> [Online; accessed 11-June-2008].
- [5] Ben Elgin - BusinessWeek. Google Buys Android for Its Mobile Arsenal, 2005. http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm [Online; accessed 11-June-2008].
- [6] Malin Berggren. *Media Browser on Android Platform*. 2008.
- [7] Bluetooth SIG, Inc. Bluetooth, 2008. <http://www.bluetooth.com/bluetooth/> [Online; accessed 11-January-2009].
- [8] Canonical Ltd. Ubuntu Linux, 2008. <http://www.ubuntu.com> [Online; accessed 11-June-2008].
- [9] chitgoks. Chitgoks Blog, 2008. <http://chitgoks.blogspot.com/2008/03/android-and-web-services.html> [Online; accessed 11-June-2008].
- [10] CollabNet. Subclipse, 2008. <http://subclipse.tigris.org/> [Online; accessed 11-January-2009].
- [11] CollabNet. Subversion, 2008. <http://subversion.tigris.org/> [Online; accessed 11-January-2009].
- [12] cplusplus.com. Welcome to cplusplus.com, 2008. <http://www.cplusplus.com/> [Online; 11-September-2008].
- [13] Darren Waters - BBC. Symbian dismisses Google Android, 2008. <http://news.bbc.co.uk/1/hi/technology/7082414.stm> [Online; accessed 15-January-2009].
- [14] Dieter Bohn. Microsoft Responds to Android: Meh., 2007. <http://www.wmexperts.com/microsoft-responds-android-meh> [Online; accessed 15-January-2009].

- [15] Fabrice Bellard. QEMU, 2008. <http://bellard.org/qemu/> [Online; accessed 29-July-2008].
- [16] Free Software Foundation Inc. GNU, 2007. <http://www.gnu.org> [Online; accessed 11-June-2008].
- [17] GetJar.com. Manufacturer Market Share, 2004-2008. <http://stats.getjar.com/statistics/> [Online; accessed 15-January-2009].
- [18] Google. Android, 2008. <http://code.google.com/android/> [Online; accessed 11-June-2008].
- [19] Google. Android, 2008. <http://dl-ssl.google.com/android/eclipse/> [Online; accessed 11-June-2008].
- [20] Google. Android Emulator, 2008. <http://android.googlecode.com/files/android-emulator-m3-rc37.tar.bz2> [Online; accessed 11-June-2008].
- [21] Google. android.R.styleable, 2009. <http://code.google.com/intl/es-ES/android/reference/android/R.styleable.html> [Online; accessed 11-June-2008].
- [22] Greg Roelofs. Portable Network Graphics (PNG), 1995-2008. <http://www.libpng.org/pub/png/> [Online; accessed 20-July-2008].
- [23] Intel Corporation. Welcome to Intel, 2008. <http://www.intel.com/> [Online; accessed 11-January-2009].
- [24] JabberES. jabberes - Mensajería Instantánea Libre, 2003-2007. <http://www.jabberes.org/> [Online; accessed 11-June-2008].
- [25] Jan Goyvaerts. Regular Expressions Info, 2003-2008. <http://www.regular-expressions.info/> [Online; accessed 11-June-2008].
- [26] Khronos Group. OpenGL ES, 2008. <http://www.khronos.org/opengles/> [Online; accessed 11-June-2008].
- [27] Khronos Group. The Khronos Group, 2008. <http://www.khronos.org/> [Online; accessed 11-June-2008].
- [28] LiMo Foundation. LiMo Foundation, 2009. <http://www.limofoundation.org/> [Online; accessed 15-January-2009].
- [29] Microsoft. Microsoft, 2008. <http://www.microsoft.com/es/es/default.aspx> [Online; 11-September-2008].
- [30] Microsoft. Windows Mobile, 2008. <http://www.microsoft.com/windowsmobile/en-us/default.mspx> [Online; accessed 11-September-2008].
- [31] Microsoft Corporation. COM: Component Object Model Technologies, 2009. <http://www.microsoft.com/com/default.mspx> [Online; accessed 11-January-2009].

- [32] Netbeans. Netbeans, 2008. <http://www.netbeans.org/> [Online; accessed 11-June-2008].
- [33] Object Management Group, Inc. Corba, 1997-2009. <http://www.corba.org/> [Online; accessed 11-January-2009].
- [34] Open Handset Alliance. Open Handset Alliance, 2008. <http://www.openhandsetalliance.com> [Online; accessed 11-June-2008].
- [35] Openmoko. Openmoko - Open. Mobile. Free., 2008. http://wiki.openmoko.org/wiki/Main_Page [Online; accessed 15-January-2009].
- [36] SASL. SASL, 2009. <http://asg.web.cmu.edu/sasl/> [Online; accessed 11-June-2008].
- [37] Scott McMillan. SGL - A 3D Scene Graph Library, 2001. <http://sgl.sourceforge.net/> [Online; accessed 11-June-2008].
- [38] SourceForge, Inc. kSOAP 2, 1999-2009. <http://ksoap2.sourceforge.net/> [Online; accessed 11-January-2009].
- [39] SourceForge, Inc. NuSOAP - SOAP Toolkit for PHP, 1999-2009. <http://sourceforge.net/projects/nusoup/> [Online; accessed 11-January-2009].
- [40] Sun Microsystems. JSR-000239 JSR-000239 Java™ Bindings for OpenGL ES API , 1995-2004. <http://jcp.org/aboutJava/communityprocess/final/jsr239/index.html> [Online; accessed 11-June-2008].
- [41] Symbian. Symbian OS, 2008. <http://www.symbian.com/> [Online; accessed 11-September-2008].
- [42] Symbian Software Limited. Symbian Foundation, 2008. <http://www.symbianfoundation.org/> [Online; accessed 15-January-2009].
- [43] T-Mobile USA, Inc. T-Mobile, 2008. <http://www.t-mobile.com> [Online; accessed 11-January-2009].
- [44] T-Mobile USA, Inc. T-Mobile G1 with Google - Bronze, 2008. <http://www.t-mobile.com/shop/phones/Cell-Phone-Detail.aspx?cell-phone=T-Mobile-G1-with-Google-Bronze> [Online; accessed 11-January-2009].
- [45] Telnet.org. Telnet.org, 1998-2008. <http://www.telnet.org> [Online; accessed 11-January-2009].
- [46] The Apache Software Foundation. The Apache Software Foundation, 2008. <http://www.apache.org/> [Online; accessed 19-August-2008].
- [47] The Apache Software Foundation. Web Services - Axis, 2008. <http://ws.apache.org/axis/> [Online; accessed 11-June-2008].
- [48] The PHP Group. PHP:Hypertext Preprocessor, 2008. <http://www.php.net> [Online; accessed 11-June-2008].

- [49] W3C. XML Protocol Working Group - Simple Object Access Protocol, 2008. <http://www.w3.org/2000/xp/Group/> [Online; accessed 11-June-2008].
- [50] W3C®. Extensible Markup Language (XML), 1996-2003. <http://www.w3.org/XML/> [Online; accessed 11-January-2009].
- [51] Wikipedia. Transport Layer Security, 2003-2007. http://es.wikipedia.org/wiki/Transport_Layer_Security [Online; accessed 11-June-2008].
- [52] Wikipedia. 3GP, 2008. <http://es.wikipedia.org/wiki/3GP> [Online; accessed 11-January-2009].
- [53] Wikipedia. Distributed Component Object Model, 2008. http://es.wikipedia.org/wiki/Distributed_Component_Object_Model [Online; accessed 11-January-2009].
- [54] Wikipedia. MPEG-4, 2008. <http://es.wikipedia.org/wiki/MPEG-4> [Online; accessed 11-January-2009].
- [55] Wikipedia. Remote procedure call, 2008. http://en.wikipedia.org/wiki/Remote_procedure_call [Online; accessed 11-January-2009].
- [56] Wikipedia. Multipurpose Internet Mail Extensions, 2009. <http://es.wikipedia.org/wiki/MIME> [Online; accessed 11-January-2009].
- [57] XSF. XMPP Standards Foundation, 2008. <http://www.xmpp.org/> [Online; accessed 11-June-2008].

Apèndix A

Instal·lació de l'entorn de desenvolupament d'Android

Aquest apèndix indica les passes a seguir per tal de posar en funcionament l'entorn de desenvolupament d'Android segons la versió que s'utilitzi de l'IDE d'*Eclipse*.

De totes maneres, el primer pas és descarregar la versió que es desitgi utilitzar de l'*Android SDK* de la pàgina oficial de la plataforma i desempaquetar-ho en un directori de la màquina. Dins d'aquest directori es trobaran totes les eines i les llibreries necessàries per tal de fer servir la plataforma. A més a més, també es pot trobar la documentació que farà servir l'IDE d'*Eclipse* per a guiar al desenvolupador a implementar les aplicacions.

Un cop es té el SDK instal·lat a la màquina, el següent pas és la instal·lació de l'IDE amb el qual treballarem, en aquest cas Eclipse. Durant el projecte s'ha utilitzat dos ordinadors diferents cadascun amb una versió diferent d'*Eclipse* pel que es subdividirà aquesta secció en dos parts.

A.1 Eclipse 3.3 Europa

Durant el transcurs d'aquest projecte, s'ha utilitzat una distribució *Ubuntu*[8] com a sistema operatiu. Aquesta distribució utilitza un repositori, per tal de mantenir una serie de paquets de software adaptats al seu propi entorn. En aquests es troba la versió *Eclipse Europa*, la qual s'ha instal·lat utilitzant la utilitat *apt* del sistema.

- apt-get install eclipse

Aquesta eina necessita permisos d'administrador per tal de fer-la servir.

El plugin d'*Android* per *Eclipse* porta una sèrie de requisits que poden o no estar incorporats en la versió que l'usuari estigui utilitzant. ADT necessita el plugin de *Web Standard Tools (WST)* per funcionar i aquest a la vegada necessita les eines de GEF,

EMF i XSD. Per això, abans de poder instal·lar el plugin de desenvolupament d'*Android* s'hauria de:

1. Instal·lar el plugin XSD (Schema Infonet Model) de les eines de modelament d'*Eclipse*.
2. Instal·lar l'EMF (Eclipse Modeling Framework).
3. Instal·lar les eines de web estandard o WST.

Aquestes instal·lacions es poden fer descarregant els *plugins* directament al directori *plugins* del directori on s'hagi instal·lat *Eclipse* o, més fàcilment, utilitzar l'eina d'actualització proveïda per aquest IDE. Aquesta eina d'actualització és capaç d'informar dels requeriments del plugin. En cas de que tot estigui correcte es podrà iniciar el procés d'instal·lació.

Un cop *Eclipse* està configurat, es passa a instal·lar el plugin d'*Android*. En aquest cas, s'utilitzarà l'eina d'actualització ja que és l'única manera que ofereix la plataforma *Android* per a instal·lar aquest plugin. Primer de tot, s'ha d'introduir l'adreça de descàrrega de la versió que es vol instal·lar, en aquest projecte es va fer ús de la versió M5[19], com s'observa a la figura A.1. Després s'ha de seleccionar els diferents paquets que conté el plugin i acceptar les seves llicències d'ús.

Després de instal·lar el plugin serà necessari reiniciar l'*Eclipse* per a que tots els canvis tinguin efecte. Un cop fet això, només caldrà introduir la ruta on es troba les eines d'*Android* en l'opció de menú **Windows->Preferences**, com s'observa a la figura A.2.

A.2 Eclipse 3.4 Ganymede

Aquesta versió d'*Eclipse* incorpora per defecte els requisits del plugin d'*ADT*, pel que no s'hauran de instal·lar. En aquest cas, utilitzem una segona forma d'instal·lar aquest IDE en una distribució lliure *Ubuntu*. Només caldrà descarregar un arxiu comprimit amb els fitxers de l'aplicació, descomprimir-ho al directori escollit i donar-li els permisos d'usuari adients per a la seva utilització. Un cop aquí, s'instal·larà el plugin *ADT* per mitjà de l'actualitzador de software de l'IDE.

Per finalitzar, caldrà indicar quin és el camí per a utilitzar les eines proporcionades per *Android* i el camí on es troben les llibreries de *J2SE*, sense les quals no es podria compilar les aplicacions desenvolupades.

A.3 Java

A més a més s'ha d'instal·lar el kit de desenvolupament de *J2SE JDK5* o *JDK6*, ja que amb el *JRE* no és suficient per a compilar les aplicacions. D'altre banda, cal esmentar que *Android* no és compatible amb el compilador de *GNU* per a *Java*, anomenat *GCC*.

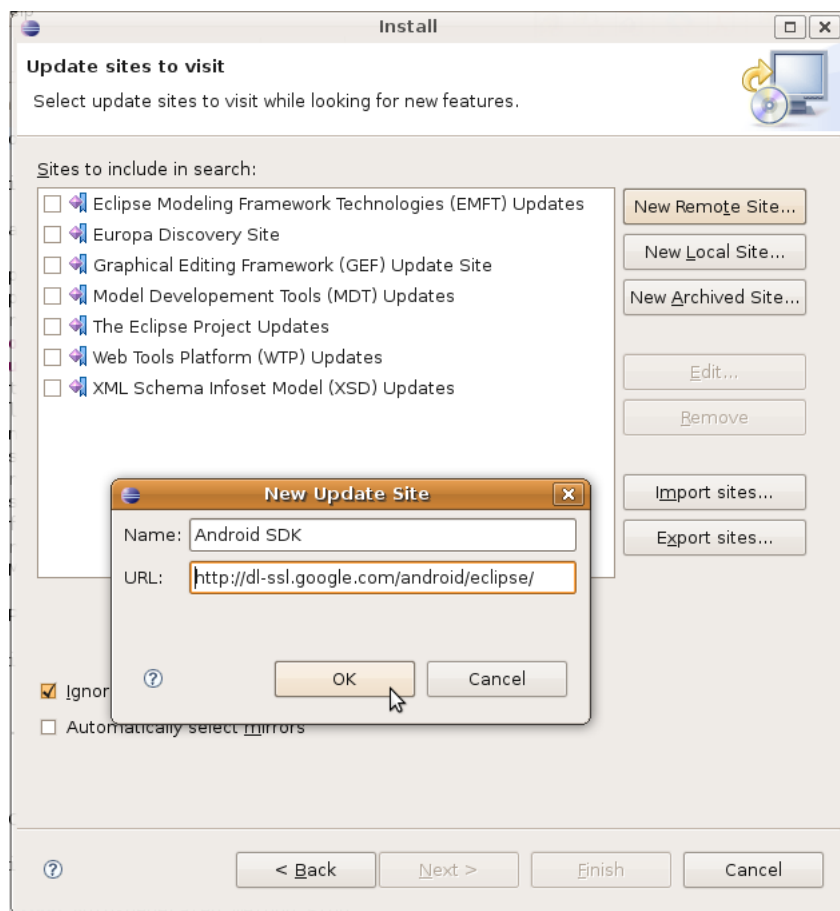


Figura A.1: Introducció de l'adreça de descàrrega d'ADT

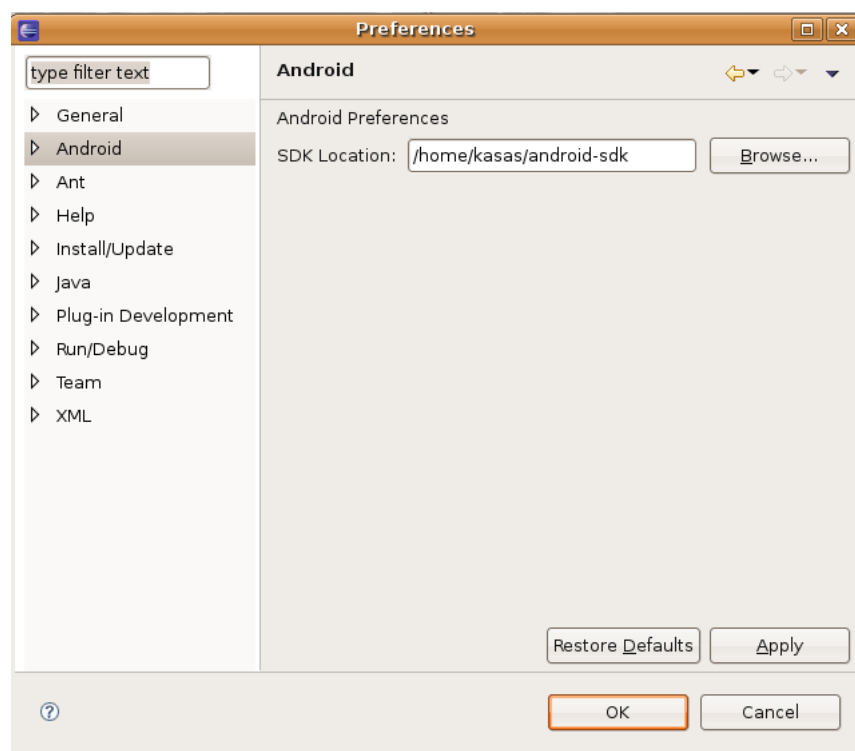


Figura A.2: Introducció de la ruta de les eines de desenvolupament d'Android

Apèndix B

Utilització de Subversion (SVN)

Subversion[11] és un sistema de control de versions. És un software que administra l'accés a un conjunt de fitxers i que manté un historial de canvis realitzats. El control de versions és una pràctica molt utilitzada per a control del codi font d'un programa, sobretot si hi han diversos grups de desenvolupadors del mateix.

Normalment, existeixen en una còpia mestre en un repositori central i un programa client amb el qual cada usuari sincronitza la seva còpia local. Això permet compartir els canvis sobre un mateix conjunt de fitxers. A més a més, el repositori enregistra els canvis realitzats per cada usuari i permet tornar enrere en cas de necessitat.

L'instal·lació de svn depèn de sistema operatiu amb el qual es treballa, encara que normalment utilitza paquets autoinstal·lables, que faciliten la feina al usuari.

B.1 Crear un repositori

El primer pas i més important a l'hora de gestionar un control de versions, és la creació del repositori principal. Aquest es genera en el servidor des de el qual es connectaran les aplicacions clients. El repositori és un arbre de directoris on Subversion emmagatzema tots els fitxers i els seus canvis.

Per a generar un repositori buit s'utilitza l'eina *svnadmin*:

- Creem el directori: `mkdir -p "path"`
- Creem el repositori: `svnadmin create "path"`
- Donem permisos: `chown -R www-data:www-data "path"`

Totes les transaccions que fem sobre aquest repositori tindran una sèrie de propietats:

1. Atòmica
2. Resultats consistents

3. Independents d'altres transaccions (independents)
4. Efecte permanent

Mai s'ha d'operar directament sobre el repositori si es vol conservar en el millor estat el control de versions instal·lat.

B.2 Accés remot amb Apache 2.2

Per tal de tenir accés al repositori des de totes les màquines de la xarxa, s'ha de donar gestió a través d'un servidor d'aplicacions. En aquest projecte s'ha escollit *Apache 2.2* com a servidor d'aplicacions per *Subversion*.

Per a que *Apache* transformi operacions *HTTP* dels clients de *Subversion* en operacions sobre el repositori, es necessària la instal·lació del mòdul *libapache2-svn*. Aquest disposa de moltes configuracions possibles segons el nivell de seguretat que es vulgui aplicar. No obstant això, ha estat configurat amb les mínimes restriccions de seguretat donat que és un repositori amb el que només es treballa a través de la xarxa local.

Per tal de configurar *Apache* com a servidor de *Subversion*, s'ha de configurar el fitxer `/etc/apache2/mods-available/dav_svn.conf`, en el qual s'ha d'introduir la directiva *Location* que per defecte es troba comentada. La qual quedaria de la següent manera:

```
<Location /repos>
  DAV svn
  SVNPath /var/local/repositorio
  AuthType Basic
  AuthName "Subversion Repository"
  AuthUserFile /etc/subversion/passwd
  <LimitExcept GET PROPFIND OPTIONS REPORT>
    Require valid-user
  </LimitExcept>
</Location>
```

S'observa que en la configuració, s'ha introduït una mesura de seguretat: la autenticació bàsica. Aquesta mesura no és massa segura, ja que fa una codificació de les contrasenyes en base64, cosa que amb qualsevol *sniffer*¹ es podria capturar i descriptar. No obstant això, com el repositori s'utilitzarà dins d'una intranet, ens servirà com identificació dels usuaris que introdueixen modificacions al codi.

Per tal d'introduir aquests usuaris, s'utilitza la comanda *htpasswd2*, de la manera següent:

```
htpasswd2 -c /etc/subversion/passwd USER
```

Després de fer tota aquesta configuració, s'ha de reiniciar el servidor *Apache* per carregar tots els canvis efectuats.

¹Un packet *sniffer* és un programa de captura de trames de xarxa

B.3 Operacions

En aquesta secció s'anomenaran les operacions més utilitzades durant el transcurs del projecte. S'ha de pensar que les aplicacions han estat desenvolupades per una sola persona, cosa que fa que les revisions de modificacions siguin les mínimes.

B.3.1 Baixar un projecte

Per tal de baixar un projecte s'utilitzarà la comanda *checkout*. Es seguiran les següents passes:

1. Creem el directori on es vol mantenir el projecte.
2. Es situa la terminal en la ruta del directori
3. S'executa la següent comanda: *svn checkout *url_projecte**

B.3.2 Actualitzar una copia local

Un projecte del qual es té una copia local s'ha d'actualitzar per mantenir-ho al corrent de les implementacions d'altres desenvolupadors. Hi han tres tipus de actualitzacions possibles:

1. Directoris amb recursivitat: *svn update*
2. Directori actual: *svn -N update*
3. Fitxer concret: *svn update *nom_fitxer**

Al hora d'actualitzar poden generar-se conflictes en cas de que dos desenvolupadors hagin tractat una mateixa zona del codi. D'aquesta manera, s'haurà de revisar i modificar el codi actualitzat i pujar-lo de nou al repositori. Per tal de fer-ho, *svn* proporciona tres fitxers la copia modificada en local, la copia de l'altre desenvolupador i una barreja entre les dues.

D'altre banda, es poden recuperar versions anteriors del codi font en cas de errors en les modificacions efectuades, o simplement per conservar diverses versions de l'aplicació. Tanmateix, per tal de conservar diverses versions s'hauria d'utilitzar una branca nova del repositori per tal de poder solucionar els bugs i enviar-los al repositori sense necessitat de molestar la versió més actual. Per tal d'utilitzar una revisió anterior del codi, es farà:

1. Mirar la revisió que es vol recuperar amb *svn log*
2. Actualitzar la copia local: *svn update -rx*, on x és el número de la revisió.

B.3.3 Enviar Modificacions

Els canvis en arxius de codi en *SVN* són atòmics, això vol dir que no es barrejaran amb els arxius allotjats en el repositori. Això vol dir que serà una bona pràctica la introducció de comentaris cada cop que es pugen canvis al repositori. Això farà que es puguin trobar més fàcilment les correccions efectuades. Abans de fer un *commit*, és important que el codi utilitza l'última versió disponible al repositori, ja que si no *SVN* no deixarà enviar-les. Per tal de pujar-ho es poden efectuar les següents operacions:

1. Tots els arxius modificats: `svn commit -m 'missatge'`
2. Arxius modificats a diferents directoris: `svn commit -m 'missatge' dir1 dir2 ...`

Els comentaris no són obligatoris, pel que es podria obviar el paràmetre del missatge.

B.4 Subclipse

Subclipse és un plugin d'*Eclipse* per al manteniment de repositoris *SVN*. Aquesta eina permet executar les comandes bàsiques d'aquest tipus de repositori, sobre un repositori base principal.

L'última versió d'aquest plugin es troba a la seva pàgina oficial[10], encara que no va poder ser utilitzada a l'última versió d'*Eclipse*, la versió 3.4, durant la realització del projecte. D'aquesta manera, al utilitzar aquesta versió, es va tenir que fer el manteniment del repositori mitjançant la *shell* d'*Ubuntu*.

Appendix C

GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms

of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF

produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be

added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.