PFC – Xarxes de Computadors

Geoshare: Aplicació per a gestionar i compartir geolocalitzacions amb Android

Michael Vekens
Enginyeria en Informàtica

Consultor: Jordi Ceballos Villach

Data: 07-01-2013

Índex de continguts

1. Introducció

1.1. Justificació del projecte

Com s'ha pogut veure durant els últims anys, l'expansió de l'ús dels dispositius mòbils és un fet i la tendència és creixent. Les aplicacions per aquests dispositius, tant pel sistema Android com pels altres sistemes existents, són àmpliament acceptades i cada vegada augmenta el nombre d'aplicacions i d'usuaris.

Pel que fa el sistema operatiu Android en concret, s'estableix cada vegada més en el mercat i des de la seva aparició a l'any 2005 ha trobat una comunitat de nombrosos usuaris amb tendència creixent. A dia d'avui es pot dir que, en només 7 anys s'ha convertit en una competència real pels sistemes existents i ja establerts en el mercat dels sistemes operatius per dispositius mòbils com Apple iOS, RIM BlackBerry OS, Windows Mobile o Symbian OS. Actualment s'ha arribat a l'extrem que Android és el sistema amb més quota de mercat en el sector dels telèfons mòbils i amb l'aparició de la Tablet Nexus 7 ha fet un gran pas a aquest sector també. En un futur s'espera que la tendència segueixi en la mateixa direcció el qual condueix a pensar que les aplicacions en general per aquest sistema operatiu estaran en el centre del focus del sector.

El llenguatge de programació escollit per l'empresa que manté l'Android és el Java. També aquest llenguatge ha rebut molta atenció durant els últims anys i s'ha convertit en un dels llenguatges més utilitzats en molts àmbits de les tecnologies més recents (tecnologies mòbils, tecnologies web, etc.). En aquest sentit, doncs, també ajuda a pensar que l'expansió del sistema i el desenvolupament d'aplicacions del mateix s'expandirà encara més durant els propers anys.

Personalment el llenguatge Java i les tecnologies i metodologies relacionades amb aquest, són els que més he fet servir durant els estudis i la vida laboral. Això també ha estat un dels motius principals per escollir aquest sistema operatiu i no un dels altres com l'Apple iOS el qual també és molt utilitzat (tot i que menys aquí a Europa).

A part d'aquests motius més aviat tecnològics hi ha el motiu d'interès personal, ja que sempre he trobat interessant aquest sector mòbil i els sistemes i les aplicacions que hi ha a darrera. Tot i que de moment no en tinc gaires coneixements, ni com a desenvolupador ni com a usuari, aquest fet ha marcat la decisió d'escollir un projecte final de carrera en aquesta direcció. Penso que aquestes tecnologies han obert moltes portes noves i que encara no s'ha esgotat el potencial d'innovació relacionat a tot el mercat. D'aquesta manera es poden trobar molts elements nous que fins ara teològicament no eren possible o no han trobat aplicació com, per exemple, sistemes d'orientació basats en la combinació de diferents sensors (GPS i direcció), aplicacions que interactuen amb l'entorn, etc.

1.2. Descripció del projecte

L'objectiu principal del projecte és desenvolupar una aplicació pel sistema operatiu Android la qual permet gestionar i compartir les geolocalitzacions preferides dels usuaris. Basant-se en un sistema de mapes com *Google Maps* i/o *Open Street Maps*, per una banda posarà a disposició de l'usuari opcions per definir i mantenir geolocalitzacions localment en el mateix dispositiu i, per altra banda, proporcionarà funcionalitats per publicar els llocs que vulgui compartir amb altres persones i consultar els llocs publicats per altres usuaris.

Una qüestió a tenir en compte durant tot el període de disseny i implementació del projecte serà la de dissenyar-la de forma modular intentant desacoblar el màxim les funcionalitats. D'aquesta manera, en un futur, serà més fàcil incorporar noves funcionalitats a les ja existents. A més a més, això permetrà deshabilitar fàcilment determinades funcionalitats en el cas de que un dispositiu no disposa d'una tecnologia necessària per la funcionalitat.

Un altre objectiu més general seria aconseguir que l'aplicació pugui funcionar en la gran majoria de dispositius Android, el qual significa que cal tenir en compte que pot variar la versió del sistema operatiu, la mida de la pantalla, la capacitat de càlcul del dispositiu, les tecnologies que porta incorporat el dispositiu, etc.

El primer objectiu més concret seria estudiar la API del sistema Android per tenir una idea sobre quines funcionalitats proporciona i quines limitacions té. A més a més, caldrà estudiar la API de *Google Maps* per Android i el sistema d'*Open Street Maps* per decidir quin dels dos és apte per incorporar-lo a l'aplicació, o si és possible funcionar sobre qualsevol dels dos.

El segon objectiu seria configurar l'entorn de desenvolupament (amb els coneixements obtinguts en el punt anterior) per poder treballar i provar l'aplicació durant la fase de desenvolupament. Aquesta fase es realitzarà sobre un ordinador, però cal tenir en compte que el dispositiu final sobre el qual s'executarà l'aplicació serà un altre, amb unes característiques bastant diferents: un dispositiu mòbil.

El tercer objectiu serien les funcions locals del dispositiu com mostrar un mapa a la pantalla, proporcionar operacions bàsiques per interactuar amb aquest mapa, permetre definir marques per marcar els llocs preferits, guardar-les, modificar-les, etc.

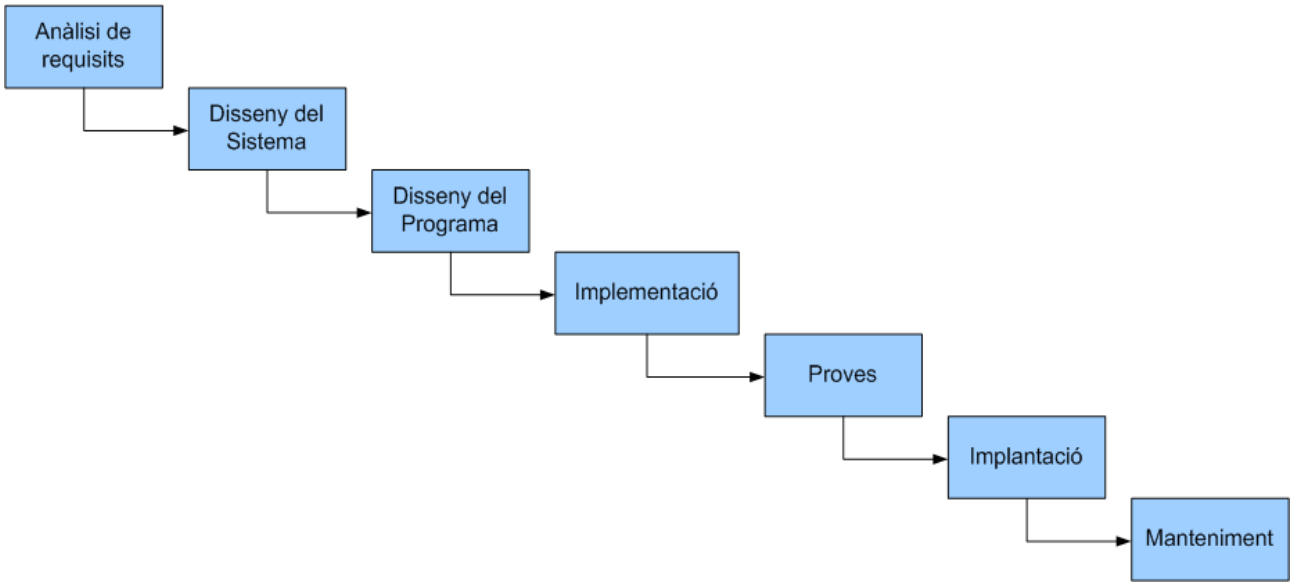
El quart objectiu seria afegir funcionalitats per poder compartir marques amb altres persones. Per aconseguir-ho caldrà implementar certa lògica en un servidor públic i comú per a tots els usuaris. A aquest servidor s'enviaran les dades a compartir i s'obtindran les dades compartides per altres.

El cinquè i últim objectiu seria provar l'aplicació en un entorn real. Tot i que ja s'hauran provat les diferents parts individualment i en conjunt en un entorn simulat (l'emulador de l'Android SDK), seria útil fer proves generals en un dispositiu real.

En resum, l'objectiu d'aquest projecte és desenvolupar un sistema per gestionar llocs geogràfics i compartir-los, a través d'un servidor públic, amb altres persones. En aplicacions d'aquest tipus la importància rau, més que en oferir una gran varietat de funcionalitats, en oferir funcionalitats pràctiques que permeten facilitar-ne l'ús. Així per exemple, tot i que mostrar els llocs definits per l'usuari en una llista i mostrar-los en un mapa són dues maneres de mostrar la mateixa informació, la segona és molt més visual i intuïtiva.

1.3. Planificació del projecte

El projecte s'ha realitzat seguint el model cascada, l'enfocament metodològic del qual ordena cada etapa del cicle de vida del programari de tal manera que l'inici de cada etapa té com a entrada el resultat de l'etapa anterior. La justificació d'utilitzar aquest mètode és que en un projecte on només hi treballa una persona no té cap avantatge desenvolupar en paral·lel cap de les tasques necessàries:



Il·lustració 1: Model cascada de l'enginyeria del programari

Anàlisi de requisits

En aquest fase s'han analitzat les necessitats dels usuaris finals i a partir de les dades resultants s'ha decidit quines són les funcionalitats més importants que haurà de tenir l'aplicació per aconseguir una eina útil.

Disseny del sistema

Durant aquesta fase s'ha definit l'estructura racional global del sistema i la manera en què es combinen les diferents parts per organitzar la implementació posterior. Això es refereix tant a l'estructura de les bases de dades com l'estructura general del codi.

Disseny del Programa

En aquesta fase es realitzen els algorismes necessaris i les eines necessàries en l'etapa de codificació. Per aquest projecte es podria dir que aquesta fase no es distingeix clarament de l'anterior i es podria dir que s'han agrupat en una fase conjunta.

Implementació

És la fase de l'implementació pròpiament dita. S'ha implementat el codi font i s'han fet proves bàsiques aïllades per assegurar el correcte funcionament. En paral·lel a aquesta fase s'ha realitzat la formació sobre el funcionament del sistema Android per poder implementar cada una de les parts que formen l'aplicació.

Proves

En aquesta fase s'han fet proves més profundes comprovant que el sistema global funciona

correctament i que compleix amb els requisits establerts a l'inici del projecte. S'han fet proves tant en un entorn simulat mitjançant un emulador com en un entorn real mitjançant un dispositiu real.

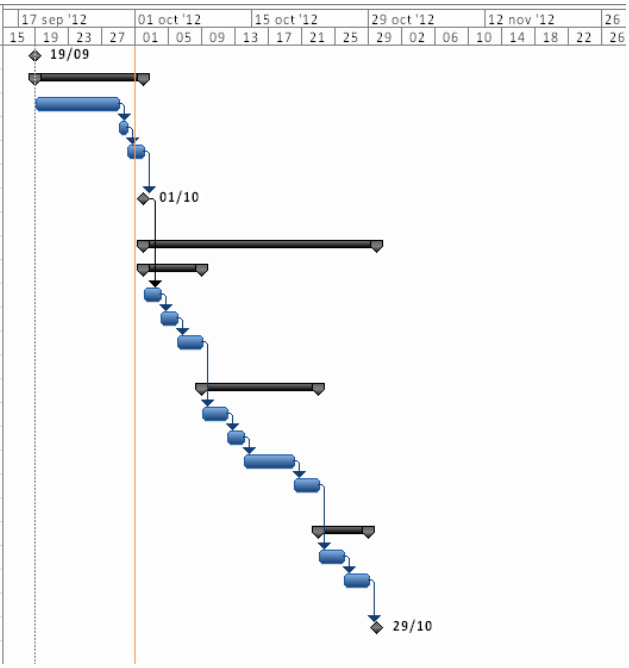
Implantació i Manteniment

El programari obtingut es posa en producció i es manté actualitzat corregint possibles errors que puguin anar apareixent o introduint millores. Durant aquest projecte només s'ha preparat el camí per aquestes dues fases intentant implementar el programari de forma que el posterior manteniment sigui el més fàcils possible. També s'ha de dir que el tipus d'aplicació desenvolupada en aquest projecte només la part del servidor s'implantaria de la manera tradicional el qual s'ha simulat implantant-la en un servidor independent de l'entorn de desenvolupament.

En cada una d'aquestes etapes s'ha anat generant la documentació necessària explicant el progrés del desenvolupament del projecte. Aquesta documentació és exposada en aquest document i els altres ja entregats explicant els resultats obtinguts i els detalls més importants.

La planificació detallada de com s'ha dut a terme el projecte és la següent:

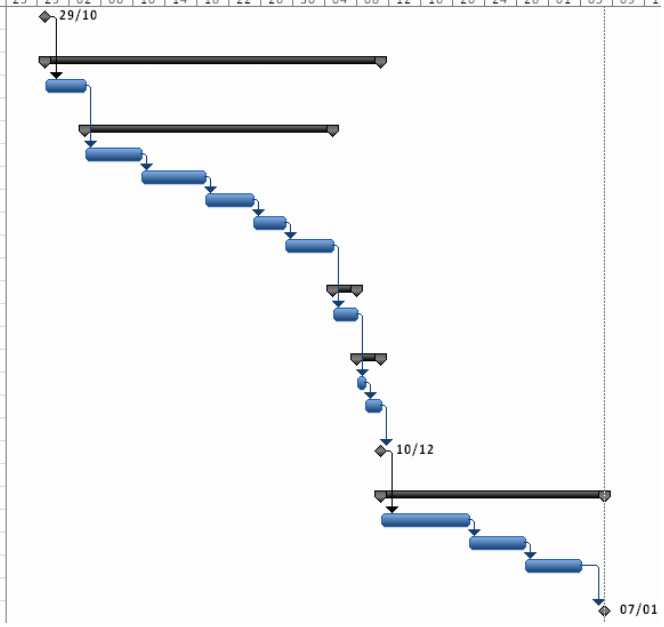
Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		inici projecte	0 días	19/09/2012	19/09/2012
2		**Pla de treball**	**13 días**	**19/09/2012**	**01/10/2012**
3		Recopilació d'informació	10 días	19/09/2012	28/09/2012
4		Instal·lació eines	1 día	29/09/2012	29/09/2012
5		Redactar pla de treball	2 días	30/09/2012	01/10/2012
6					
7		Lliurament pla de treball (PAC1)	0 días	01/10/2012	01/10/2012
8					
9		**Anàlisi funcional, Disseny Tècnic i Prototip**	**28 días**	**02/10/2012**	**29/10/2012**
10		**Analisi funcional**	**7 días**	**02/10/2012**	**08/10/2012**
11		Recopilació d'informació	2 días	02/10/2012	03/10/2012
12		Diagrama de casos d'ús	2 días	04/10/2012	05/10/2012
13		Descripció dels casos d'ús	3 días	06/10/2012	08/10/2012
14					
15		**Disseny tècnic**	**14 días**	**09/10/2012**	**22/10/2012**
16		Definició arquitectura	3 días	09/10/2012	11/10/2012
17		Descripció tecnologies	2 días	12/10/2012	13/10/2012
18		Disseny de classes	6 días	14/10/2012	19/10/2012
19		Disseny de bases de dades	3 días	20/10/2012	22/10/2012
20					
21		**Prototip**	**6 días**	**23/10/2012**	**28/10/2012**
22		Recopilació d'informació	3 días	23/10/2012	25/10/2012
23		Crear Prototip	3 días	26/10/2012	28/10/2012
24					
25		Lliurament Anàlisi, Disseny i Prototip (PAC2)	0 días	29/10/2012	29/10/2012
26					



Il·lustració 2: Fase 1 i fase 2

Les principals tasques d'aquestes fases han estat l'anàlisi funcional del sistema, el disseny tècnic i el desenvolupament d'un prototipus per tindre una idea de com s'estructurarà l'aplicació.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
25		Lliurament Anàlisi, Disseny i Prototip (PAC2)	0 días	29/10/2012	29/10/2012
26					
27		**Implementació**	**42 días**	**30/10/2012**	**10/12/2012**
28		Recopilació d'informació	5 días	30/10/2012	03/11/2012
29					
30		**Implementació i proves aïllades**	**31 días**	**04/11/2012**	**04/12/2012**
31		Interactuar amb el mapa	7 días	04/11/2012	10/11/2012
32		Gestionar llocs preferits	8 días	11/11/2012	18/11/2012
33		Compartir llocs preferits (part servidor)	6 días	19/11/2012	24/11/2012
34		Compartir llocs preferits (part client)	4 días	25/11/2012	28/11/2012
35		Indicar direcció	6 días	29/11/2012	04/12/2012
36					
37		**Proves conjuntes entorn simulat**	**3 días**	**05/12/2012**	**07/12/2012**
38		Proves entorn simulat	3 días	05/12/2012	07/12/2012
39					
40		**Proves dispositiu real**	**3 días**	**08/12/2012**	**10/12/2012**
41		Instal·lable (apk)	1 día	08/12/2012	08/12/2012
42		Proves dispositiu real	2 días	09/12/2012	10/12/2012
43					
44		Lliurament Implementació (PAC3)	0 días	10/12/2012	10/12/2012
45					
46		**Fase final**	**24 días**	**11/12/2012**	**07/01/2013**
47		Redacció memòria	11 días	11/12/2012	21/12/2012
48		Presentació - PowerPoint	5 días	22/12/2012	28/12/2012
49		Presentació - vídeo	5 días	29/12/2012	04/01/2013
50					
51		Lliurament final	0 días	07/01/2013	07/01/2013



Il·lustració 3: Fase 3

Les principals tasques d'aquesta fase són la implementació i les proves de funcionament per comprovar que l'aplicació compleix amb els requisits establerts.

2. Anàlisi del sistema

L'objectiu d'aquest projecte és desenvolupar un sistema per gestionar llocs geogràfics i compartir-los, a través d'un servidor públic, amb altres persones. En aplicacions d'aquest tipus la importància rau, més que en oferir una gran varietat de funcionalitats, en oferir funcionalitats pràctiques que permeten facilitar-ne l'ús. Així per exemple, tot i que mostrar els llocs definits per l'usuari en una llista i mostrar-los en un mapa són dues maneres de mostrar la mateixa informació, la segona és molt més visual i intuïtiva.

2.1. Funcionalitats

Un resum de les funcionalitats que haurà de tenir l'aplicació per ser pràctica per l'usuari i els punts més importants per poder-les desenvolupar.

2.1.1. Interactuar amb el mapa

Trobar una manera fàcil i intuïtiva d'interactuar amb un sistema de mapes:

- Permetre utilitzar els mapes que ofereix la API de *Google Maps* per Android.
- Permetre utilitzar els mapes que ofereix la API d'*Open Street Maps* per Android.
- Perquè sigui fàcilment utilitzable, el mapa haurà d'utilitzar la major part possible de la pantalla i adaptar-se segons el dispositiu.
- Interactuar amb el mapa: desplaçar-se pel mapa i fer *zooming* d'una manera intuïtiva per aconseguir que l'aplicació sigui fàcil d'utilitzar.
- Buscar un punt en el mapa: permetre entrar directament coordenades en diferents formats o una adreça en concret per buscar la posició en el mapa, mostrar-la i marcar-la com a preferit.
- Si el dispositiu disposa d'algun sensor geoposicional, serà possible localitzar-lo el qual permetrà mostrar en cada moment la posició actual de l'usuari en el mapa.

2.1.2. Gestionar llocs preferits

Per gestionar els llocs preferits de l'usuari, l'aplicació proporcionarà funcionalitats per posicionar, mostrar i gestionar marques:

- Posicionar marques: es permetrà posicionar marques directament sobre el mapa o entrant les coordenades o l'adreça per posicionar-lo en el lloc indicat. La informació sobre les marques es guardarà en una base de dades del dispositiu per accedir-hi posteriorment.
- Mostrar marques: les marques guardades a la base de dades es mostraran directament sobre el mapa.
- Entrar informació de marques: per qualsevol marca es podrà entrar informació relacionada a aquesta. Aquesta informació també es guardarà en una base de dades del dispositiu.
- Mostrar informació d'una marca: es permetrà clicar una marca per mostrar la informació relacionada.
- Modificar informació d'una marca: es proporcionaran opcions per modificar la informació relacionada a una marca existent.

2.1.3. Compartir llocs preferits

Permetre enviar a un servidor públic les marques que l'usuari vulgui compartir:

- Enviar la informació d'un lloc marcat per l'usuari (localització i la informació relacionada) a un servidor públic. Aquesta informació es guardarà en una base de dades del servidor.
- Rebre del servidor públic una llista de marques compartides públicament.
- Rebre del servidor públic les dades d'un lloc marcat per algun usuari (localització i informació relacionada).
- Modificar les dades d'una marca pròpia d'un usuari en el servidor públic.
- Per accedir al sistema del servidor, els usuaris hauran de registrar-se i identificar-se. El sistema haurà d'oferir aquestes funcionalitats als usuaris.

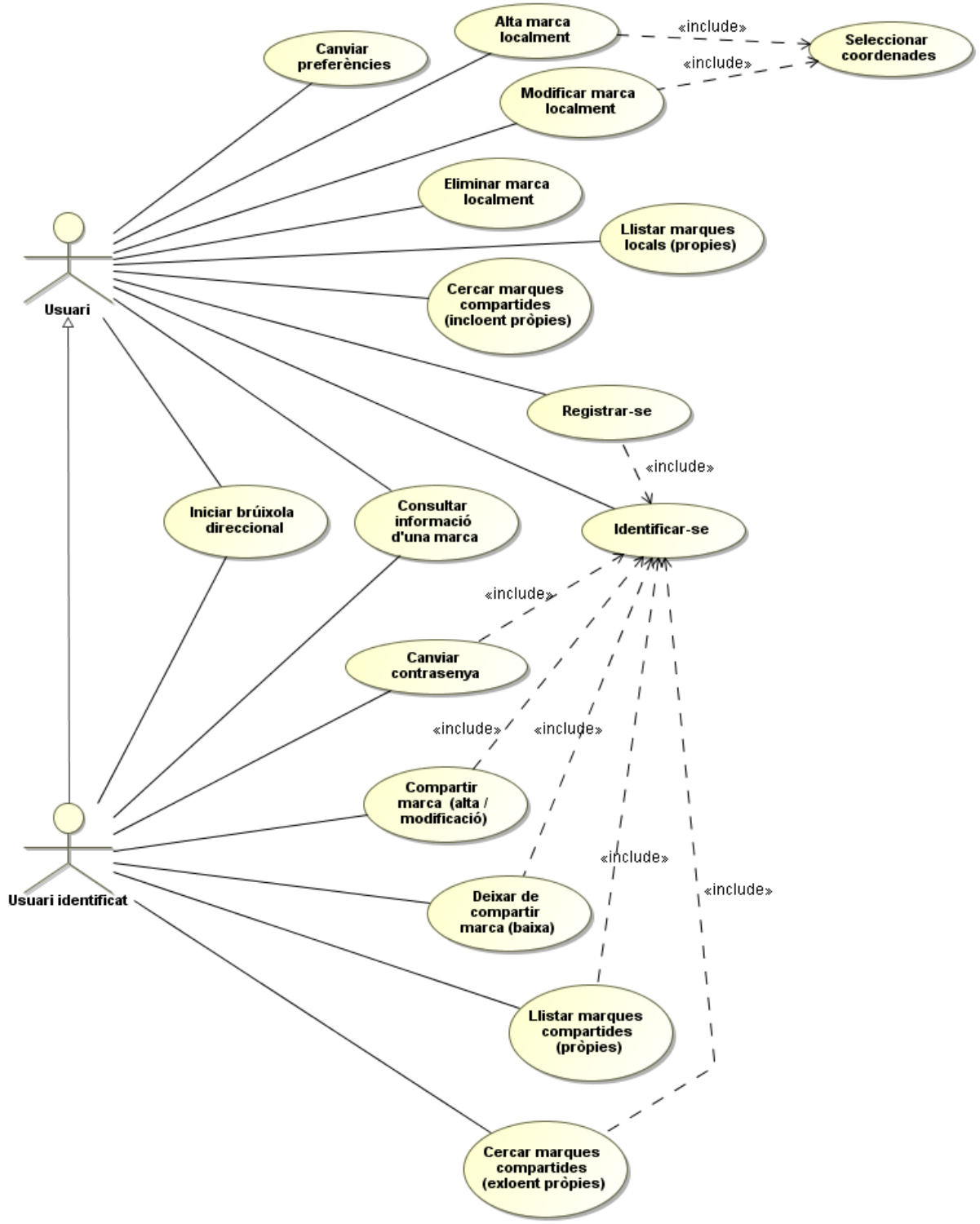
2.1.4. Indicar direcció

Un cop s'ha escollit una geolocalització pròpia, o se n'ha obtingut una d'una altra persona del servidor públic, s'indicarà a l'usuari el següent per trobar-la mitjançant GPS o *network location provider*:

- Marcar la posició del lloc escollit en el mapa.
- Indicar la direcció (sense mapa) en la qual es troba el lloc escollit i, per tant, indicar a l'usuari cap a on s'ha de moure per arribar al lloc. Aquesta opció té l'avantatge que per trobar el lloc no es necessària cap connexió a Internet.

2.2. Diagrama de casos d'ús

En el següent diagrama es poden veure els actors i els casos d'ús de les funcionalitats de l'aplicació:



Il·lustració 4: Diagrama de casos d'ús

Podem observar-hi dos actors diferents:

- Usuaris no identificats: poden fer qualsevol operació sobre dades locals del dispositiu. També poden fer algunes operacions que accedeixen al servidor comú i permeten ser executades sense necessitat d'identificar a l'usuari.

- Usuaris identificats: la gran majoria de les operacions que contacten el servidor necessiten poder identificar l'usuari davant del sistema. Aquestes operacions només les poden executar usuaris registrats i identificats.

Per no sobrecarregar el diagrama no s'han posat les relacions entre “Consultar la informació d'una marca” (tant local com compartida) i “Llistar marques” / “Cercar marques”, on un d'aquests segons s'ha d'haver executat prèviament al primer. El mateix passa amb “Iniciar brúixola orientativa” i llistar / cercar: per escollir una marca primer s'han de llistar.

En aquesta taula es pot veure de manera resumida els casos d'ús, quin actor pot executar-lo i un identificador per la descripció detallada que hi ha a continuació:

ID	Actor	Descripció
CU-01	Usuari	Registrar-se
CU-02	Usuari	Identificar-se
CU-03	Usuari	Canviar preferències
CU-04	Usuari identificat	Canviar contrasenya
CU-05	Usuari	Alta marca localment
CU-06	Usuari	Modificar marca localment
CU-07	Usuari	Eliminar marca localment
CU-08	Usuari	Llistar marques locals (pròpies)
CU-09	Usuari	Cercar marques compartides (incloent pròpies)
CU-10	Usuari identificat	Compartir marca (alta/modificació)
CU-11	Usuari identificat	Deixar de compartir marca (baixa)
CU-12	Usuari identificat	Llistar marques compartides (pròpies)
CU-13	Usuari identificat	Cercar marques compartides (excloent pròpies)
CU-14	Usuari	Seleccionar coordenades
CU-15	Usuari	Consultar informació d'una marca
CU-16	Usuari	Iniciar brúixola orientativa

2.3. Descripció formal dels casos d'ús

A continuació es descriuen detalladament els diferents casos d'ús que es poden trobar en el sistema i la informació rellevant d'aquests. Alguns punts generals a tenir en compte:

- Les comunicacions entre dispositiu client i servidor apareixen en els mateixos casos d'ús. Tot i que es podria haver definit un actor “client dispositiu” per fer-ho, queda més clar afegint-ho directament al cas d'ús afectat.
- Quan es parla de comprovacions tant es fa referència a les comprovacions que pot fer el dispositiu Android com les que ha de fer el servidor en cas de que l'operació contacta el servidor.

2.3.1. Registrar-se

Identificador	CU-01
Nom	Registrar-se
Autor	Michael Vekens
Resum	Aquest cas mostra com es registraria un usuari en el sistema.
Actor	Usuari
Precondicions	• No hi ha cap sessió activa. • L'usuari no pot identificar-se davant del sistema (no s'ha registrat encara).
Postcondicions	• L'usuari s'ha registrat i, per tant, el sistema podrà identificar-lo. • L'usuari queda identificat (després el registrar-se, no és necessari executar explícitament el cas d'identificar-se).
Flux	1. S'inicia quan l'usuari vol accedir a una funcionalitat per la qual és necessari identificar-se. 2. El sistema demana a l'usuari un nom d'usuari, una contrasenya i la repetició de la contrasenya. 3. L'usuari omple els camps amb la informació sol·licitada. 4. El sistema comprova les dades. 5. Si la informació és correcta l'usuari queda registrat i identificat.
Fluxos alternatius	• Si les dades del pas 4 no són correctes s'avisa a l'usuari amb un missatge informatiu i el flux torna al pas 3. • En qualsevol moment, l'actor podrà accedir al cas d'ús CU-02 (Identificar-se) o finalitzar el cas d'ús de registrar-se polsant un boto. En aquests dos casos s'abandona la registració.
Inclusions	CU-14 (Identificar-se).
Extensions	Cap

2.3.2. Identificar-se davant del sistema

Identificador	CU-02
Nom	Identificar-se
Autor	Michael Vekens
Resum	Aquest cas mostra com s'identificaria un usuari davant del sistema.
Actor	Usuari
Precondicions	• No hi ha cap sessió activa. • L'usuari esta registrat com a usuari del sistema.
Postcondicions	• L'usuari s'ha identificat davant del sistema.
Flux	1. S'inicia quan l'usuari vol accedir a una funcionalitat per la qual és necessari identificar-se. 2. El sistema demana a l'usuari el nom d'usuari i la contrasenya. 3. L'usuari omple els camps amb la informació sol·licitada. 4. L'usuari confirma polsant un botó.

	5. El sistema comprova les dades. 6. Si la informació és correcta l'usuari queda identificat.
Fluxos alternatius	• Si les comprovacions de les dades del pas 5 no són correctes, s'avisa a l'usuari amb un missatge informatiu i el flux torna al pas 3. • L'actor podrà, en qualsevol moment, accedir al cas d'ús CU-01 (Registrar-se) o finalitzar el cas d'ús polsant un boto.
Inclusions	Cap
Extensions	Cap

2.3.3. Canviar preferències de l'aplicació

Identificador	CU-03
Nom	Canviar preferències
Autor	Michael Vekens
Resum	Aquest cas mostra com un usuari canviaria les preferències de l'aplicació.
Actor	Usuari
Precondicions	Cap
Postcondicions	S'han guardat i aplicat els canvis de les preferències de l'aplicació.
Flux	1. S'inicia quan l'usuari accedeix a l'opció corresponent. 2. El sistema presenta a l'usuari les diferents opcions configurables de l'aplicació. 3. L'usuari fa els canvis que s'ajusten millor a les seves necessitats. 4. L'usuari indica al sistema que vol guardar els canvis polsant un boto. 5. El sistema guarda i aplica els canvis.
Fluxos alternatius	L'usuari podrà cancel·lar les modificacions en qualsevol moment. En aquest cas no es guardarà cap canvi efectuat.
Inclusions	Cap
Extensions	Cap

2.3.4. Canviar contrasenya

Identificador	CU-04
Nom	Canviar contrasenya
Autor	Michael Vekens
Resum	Aquest cas mostra com un usuari pot canviar la contrasenya associada al seu nom d'usuari.
Actor	Usuari identificat
Precondicions	L'usuari s'ha identificat.
Postcondicions	S'ha canviat la contrasenya de l'usuari.
Flux	1. S'inicia quan l'usuari accedeix a l'opció corresponent. 2. El sistema mostra 3 camps a l'usuari: contrasenya anterior, nova

	contrasenya, repetició de la nova contrasenya. 3. L'usuari omple els camps. 4. L'usuari confirma polsant un botó. 5. El sistema fa les validacions necessàries. 6. Si la informació és correcta es reemplaça la contrasenya actual amb la nova.
Fluxos alternatius	• L'usuari pot cancel·lar l'operació fins al pas 4. En aquest cas no es guardarà cap canvi. • Si les dades del pas 5 no són correctes, s'avisa a l'usuari amb un missatge informatiu i el flux torna al pas 3.
Inclusions	Cap
Extensions	Cap

2.3.5. Alta marca nova local

Identificador	CU-05
Nom	Alta marca localment
Autor	Michael Vekens
Resum	Aquest cas mostra com un usuari pot donar d'alta una marca nova en el sistema.
Actor	Usuari
Precondicions	Cap
Postcondicions	S'ha marcat un lloc i s'ha guardat la informació relacionada.
Flux	1. S'inicia quan l'usuari accedeix a l'opció corresponent. 2. El sistema mostra a l'usuari opcions per seleccionar les coordenades i informació addicional del lloc. 3. L'usuari omple els camps. 4. L'usuari confirma polsant un botó. 5. El sistema comprova les dades que pot comprovar. 6. Si les dades són correctes, el sistema actualitza les dades de la base de dades. 7. Si les dades són correctes, el sistema actualitza la vista reflectint els canvis.
Fluxos alternatius	• Si les dades que pot comprovar el sistema en el pas 5 no són correctes, s'avisa a l'usuari amb un missatge informatiu i es torna al pas 3. • L'usuari podrà cancel·lar la creació de la marca fins el pas 4. En aquest cas no s'efectuarà cap canvi.
Inclusions	CU-14 (Seleccionar coordenades)
Extensions	Cap

2.3.6. Modificar marca localment

Identificador	CU-06
Nom	Modificar marca localment

Autor	Michael Vekens
Resum	Mostra com un usuari pot modificar les dades d'uns marca localment.
Actor	Usuari
Precondicions	Cap
Postcondicions	Les modificacions de la marca s'han guardat.
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals de la marca que vol modificar i selecciona l'opció de modificar. 2. El sistema consulta la base de dades local per obtenir la informació relacionada. 3. El sistema mostra les dades de la marca a modificar. 4. L'usuari modifica les dades que vol modificar. 5. L'usuari confirma polsant un botó. 6. El sistema comprova les dades que pot comprovar. 7. Si les dades són correctes, el sistema guarda els canvis efectuats a la base de dades. 8. Si les dades són correctes, el sistema actualitza la vista reflectint els canvis.
Fluxos alternatius	• L'usuari podrà cancel·lar la modificació de la marca fins el pas 5. En aquest cas no s'efectuarà cap canvi. • Si les dades que pot comprovar el sistema en el pas 6 no són correctes, s'avisa a l'usuari amb un missatge informatiu i es torna al pas 4.
Inclusions	Cap
Extensions	Cap

2.3.7. Eliminar marca localment

Identificador	CU-07
Nom	Eliminar marca localment
Autor	Michael Vekens
Resum	Mostra com un usuari pot eliminar una marca definida localment.
Actor	Usuari
Precondicions	Cap
Postcondicions	La marca s'ha eliminat localment.
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals de la marca que vol eliminar i selecciona l'opció d'eliminar. 2. El sistema demanarà una confirmació. 3. L'usuari confirma polsant un botó. 4. El sistema elimina la marca de la base de dades i actualitza la vista reflectint els canvis.
Fluxos alternatius	L'usuari pot cancel·lar l'eliminació de la marca denegant la confirmació del pas 2.
Inclusions	Cap
Extensions	Cap

2.3.8. Llistar marques locals (pròpies)

Identificador	CU-08
Nom	Llistar llocs locals (pròpies)
Autor	Michael Vekens
Resum	Mostra com es llisten els llocs que l'usuari ha marcat localment (totes les marques locals són del mateix usuari).
Actor	Usuari
Precondicions	Cap
Postcondicions	Cap
Flux	1. S'inicia automàticament ja que és la vista per defecte de l'aplicació. 2. El sistema consulta la base de dades local. 3. El sistema mostra tots els llocs obtinguts en un mapa.
Fluxos alternatius	
Inclusions	Cap
Extensions	Cap

2.3.9. Cercar marques compartides (incloent les pròpies)

Identificador	CU-09
Nom	Cercar marques compartides (incloent pròpies).
Autor	Michael Vekens
Resum	Mostra com un usuari pot cercar marques compartides per tots els usuaris (marques del mateix usuari incloses perquè per un usuari no identificat no es poden excloure).
Actor	Usuari
Precondicions	Cap
Postcondicions	Cap
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals d'una geolocalització qualsevol i escull l'opció de trobar marques compartides. 2. El sistema mostra a l'usuari opcions per escollir la posició d'on es vol cercar i el radi de l'àrea dins la qual vol cercar marques compartides. 3. L'usuari escull l'opció que més s'ajusta a les seves necessitats. 4. L'usuari confirma polsant un boto. 5. El sistema del dispositiu contacta el servidor comú enviant-li les dades de la consulta. 6. El sistema del servidor consulta la base de dades obtenint tots els llocs compartits que compleixen amb les dades de la consulta (marques del mateix usuari incloses). 7. El sistema del servidor retorna la llista de marques compartides. 8. El sistema del dispositiu rep la llista i la mostra per pantalla.
Fluxos alternatius	L'usuari pot cancel·lar la consulta polsant un boto en el pas 4.
Inclusions	Cap
Extensions	Cap

2.3.10. Compartir marca (alta/modificació)

Identificador	CU-10
Nom	Compartir marca (alta/modificació)
Autor	Michael Vekens
Resum	Compartir un marca definida localment (totes les marques locals són del mateix usuari) en un servidor públic. Correspondria a una alta nova si no existeix o una modificació si es comparteix una marca ja compartida.
Actor	Usuari identificat
Precondicions	L'usuari esta registrat
Postcondicions	La marca d'un lloc apareix públicament com a compartida.
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals d'una marca i escull l'opció de compartir la marca. 2. Si l'usuari no esta identificat, s'ha d'identificar (cas d'ús CU-14 Identificar-se). 3. El sistema del dispositiu contacta el servidor comú enviant-li les dades de l'alta. 4. El sistema del servidor fa les comprovacions necessàries. 5. El sistema del servidor dona d'alta les dades rebudes a la base de dades del servidor. 6. El sistema del servidor retorna una confirmació d'alta. 7. El sistema del dispositiu rep la confirmació i ho indica per pantalla a l'usuari.
Fluxos alternatius	• Si el sistema servidor en el pas 4 troba algun problema amb les dades, retornarà un missatge indicatiu pel client/usuari. • Si el servidor en el pas 5 detecta que l'identificador de la marca ja existeix, actualitza les dades de la marca (perquè ja esta donat d'alta).
Inclusions	CU-14 (Identificar-se).
Extensions	Cap

2.3.11. Deixar de compartir una marca (baixa)

Identificador	CU-11
Nom	Deixar de compartir una marca
Autor	Michael Vekens
Resum	Deixar de compartir una marca compartida prèviament (seria equivalent a donar de baixa una marca compartida).
Actor	Usuari identificat
Precondicions	L'usuari esta registrat
Postcondicions	La marca ja no apareix públicament com a compartida.
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals d'una marca compartida i escull l'opció de deixar de compartir la marca. 2. Si l'usuari no esta identificat, s'ha d'identificar (cas d'ús CU-14 Identificar-se). 3. El sistema demanarà una confirmació. 4. L'usuari confirma polsant un botó. 5. El sistema del dispositiu contacta el servidor comú enviant-li les dades de la

	baixa. 6. El sistema del servidor fa les comprovacions necessàries. 7. El sistema del servidor dona de baixa l'element identificat per les dades rebudes a la base de dades del servidor. 8. El sistema del servidor retorna una confirmació de la baixa. 9. El sistema del dispositiu rep la confirmació i ho indica per pantalla a l'usuari.
Fluxos alternatius	• L'usuari pot cancel·lar l'eliminació de la marca denegant la confirmació del pas 4. • Si el sistema servidor en el pas 6 troba algun problema amb les dades, retornarà un missatge indicatiu pel client/usuari.
Inclusions	CU-14 (Identificar-se).
Extensions	Cap

2.3.12. Llistar marques compartides (pròpies)

Identificador	CU-12
Nom	Llistar marques compartides (pròpies)
Autor	Michael Vekens
Resum	Llistar les marques compartides de l'usuari.
Actor	Usuari identificat
Precondicions	L'usuari esta registrat
Postcondicions	Cap
Flux	1. S'inicia quan l'usuari accedeix a l'opció corresponent. 2. Si l'usuari no esta identificat, s'ha d'identificar (cas d'ús CU-14 Identificar-se). 3. El sistema del dispositiu contacta el servidor comú enviant-li les dades de la consulta. 4. El sistema del servidor fa les comprovacions necessàries. 5. El sistema del servidor selecciona de la base de dades del servidor tots els llocs de l'usuari. 6. El sistema del servidor retorna una llista amb els llocs seleccionats. 7. El sistema del dispositiu rep la llista i la mostra per pantalla a l'usuari.
Fluxos alternatius	Si el sistema servidor en el pas 4 troba algun problema amb les dades, retornarà un missatge indicatiu pel client/usuari.
Inclusions	CU-14 (Identificar-se).
Extensions	Cap

2.3.13. Cerca marques compartides (excloent les pròpies)

Identificador	CU-13
Nom	Cercar marques compartides (excloent les pròpies)
Autor	Michael Vekens
Resum	Mostra com un usuari pot cercar marques compartides per altres usuaris (per un usuari identificat es poden excloure les pròpies).

Actor	Usuari
Precondicions	L'usuari esta registrat
Postcondicions	Cap
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals d'una geolocalització qualsevol i escull l'opció de trobar marques compartides. 2. Si l'usuari no esta identificat, s'ha d'identificar (cas d'ús CU-14 Identificar-se). 3. El sistema mostra a l'usuari opcions per escollir la posició d'on es vol cercar i el radi de l'àrea dins la qual vol cercar marques compartides. 4. L'usuari escull l'opció que més s'ajusta a les seves necessitats. 5. L'usuari confirma polsant un botó. 6. El sistema del dispositiu contacta el servidor comú enviant-li les dades de la consulta. 7. El sistema del servidor consulta la base de dades obtenint tots els llocs compartits que compleixen amb les dades de la consulta (marques del mateix usuari excloses). 8. El sistema del servidor retorna la llista de marques compartides. 9. El sistema del dispositiu rep la llista i la mostra per pantalla.
Fluxos alternatius	L'usuari pot cancel·lar la consulta polsant un botó fins el pas 5.
Inclusions	CU-14 (Identificar-se).
Extensions	Cap

2.3.14. Seleccionar coordenades

Identificador	CU-14
Nom	Seleccionar coordenades
Autor	Michael Vekens
Resum	Aquest cas mostra com un usuari selecciona unes coordenades.
Actor	Usuari
Precondicions	Cap
Postcondicions	Coordenades seleccionades en forma de latitud i longitud.
Flux	1. Aquest cas d'ús no s'inicia directament sinó que formarà part d'altres casos d'ús des d'on s'iniciarà aquest per seleccionar coordenades. 2. El sistema mostra a l'usuari diferents opcions per seleccionar una coordenada: entrar coordenades directament, un cercador d'adreces, la posició actual o mostrant un mapa). 3. L'usuari escull l'opció que més s'ajusta a la informació de la qual disposa. 4. Si l'usuari no ha entrat les coordenades directament en format latitud/longitud, el sistema converteix les coordenades. 5. L'usuari confirma polsant un boto. 6. S'han escollit les coordenades per processar-les.
Fluxos alternatius	Si l'usuari entra les coordenades directament per latitud i longitud en el pas 3, es salta directament al pas 5.
Inclusions	Cap
Extensions	Cap

2.3.15. Consultar la informació d'una marca

Identificador	CU-15
Nom	Consultar la informació d'una marca.
Autor	Michael Vekens
Resum	Mostra com un usuari pot consultar les dades relacionades a una marca. El funcionament és el mateix tant per marques locals com per marques compartides, el que canvia és com s'obtenen aquestes marques (el qual esta definit en altres casos d'ús).
Actor	Usuari
Precondicions	Cap
Postcondicions	Cap
Flux	1. L'usuari selecciona el lloc que vol consultar de la llista. 2. El sistema mostra la informació del lloc.
Fluxos alternatius	
Inclusions	Cap
Extensions	Cap

2.3.16. Iniciar la brúixola orientativa

Identificador	CU-16
Nom	Iniciar la brúixola orientativa.
Autor	Michael Vekens
Resum	Mostra com un usuari pot inicia la brúixola per obtenir informació sobre la direcció en que es troba una marca. El funcionament és el mateix tant per marques locals com per marques compartides, el que canvia és com s'obtenen aquestes marques (el qual esta definit en altres casos d'ús).
Actor	Usuari
Precondicions	Cap
Postcondicions	Cap
Flux	1. S'inicia quan l'usuari accedeix a les opcions contextuals d'una marca i escull l'opció corresponent. 2. El sistema calcula la direcció i la distància a que es troba la marca respecte la posició actual. 3. El sistema mostra per pantalla els resultats dels càlculs.
Fluxos alternatius	
Inclusions	Cap
Extensions	Cap

3. Disseny tècnic

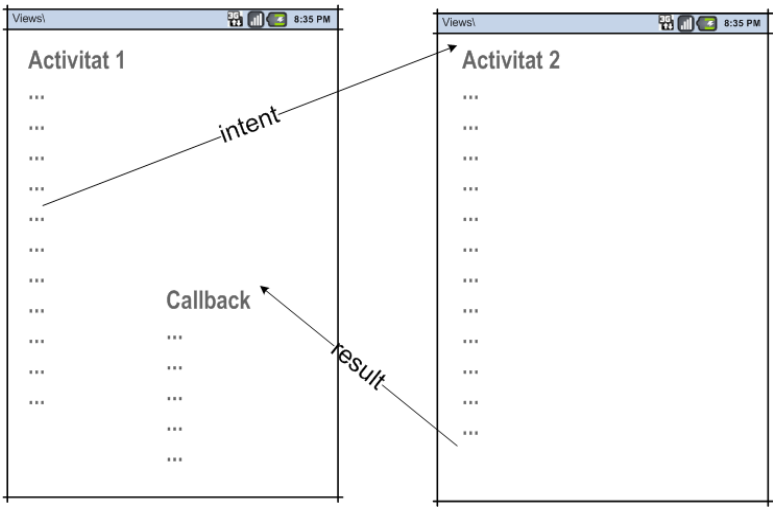
En aquesta secció es descriuen les arquitectures principals, les tecnologies utilitzades en el sistema, el disseny de les classes i de les bases de dades.

3.1. Arquitectura

3.1.1. Arquitectura modular

El mateix sistema Android condueix a desenvolupar de forma modular. La idea bàsica de les activitats definides en una aplicació Android, és que cada una només realitza una determinada tasca i forma part d'un conjunt d'activitats que interaccionen entre elles per formar l'aplicació. En aquest projecte no s'arriba a utilitzar l'extrem d'aquesta idea, el qual consisteix en interactuar amb activitats d'altres aplicacions, però sí l'interacció entre diferents activitats de la mateixa aplicació, que segueix la mateixa filosofia.

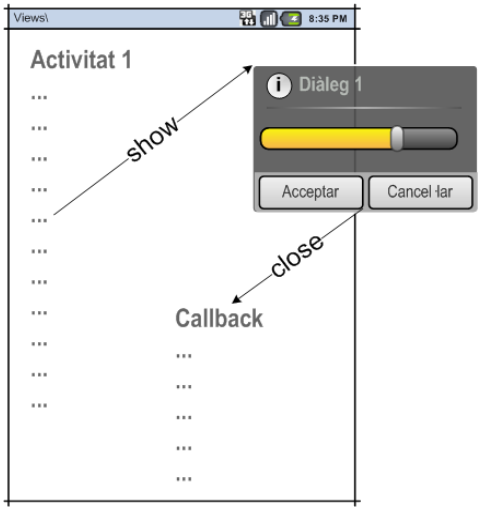
A la següent imatge es pot veure aquesta interacció entre activitats. En un punt de l'activitat 1 s'inicia l'activitat 2 i quan aquesta finalitza, es torna a la primera. És un cas simple que es podria ampliar amb moltes més activitats, per les quals es navega durant l'execució de l'aplicació:



Cal tenir en compte que Android evita el bloqueig de l'execució per evitar bloquejar la interfície gràfica de l'usuari. Per aconseguir la interacció entre activitats s'utilitzen *Callbacks* que es criden quan sigui necessari tornar al fil original. D'aquesta manera, quan l'activitat 1 crida l'activitat 2, la primera no para l'execució en el punt on es troba sinó que segueix executant-se i la segona, quan acaba, crida un *Callback* de la primera per tornar-hi.

En quant a les activitats, Android defineix un mètode estàndard a seguir per aconseguir aquesta interacció (per facilitar el compartiment d'activitats entre diferents aplicacions). Pels diàlegs, en canvi, no ho defineix ja que aquests no solen ser compartits entre diferents aplicacions. Per evitar la creació de dependències directes entre classes i incrementar la modularitat, s'ha decidit utilitzar un mecanisme semblant pels diàlegs de l'aplicació que s'esta desenvolupant: es crearan mètodes de *Callback* per tornar l'execució al fil original. L'exemple més significatiu és el de tancar un diàleg ja que el sistema Android, en el moment d'obrir un diàleg, tampoc no bloqueja l'execució del codi fins a tancar-lo, amb lo qual obliga al desenvolupador a utilitzar una alternativa com un *Callback* que s'executa en tancar el diàleg.

A la següent imatge es pot veure com funcionaria aquesta interacció amb els diàlegs:

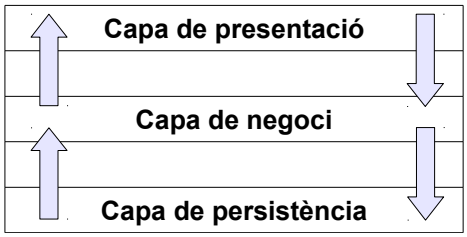


Més endavant, a l'apartat de disseny de classes, es poden veure les interaccions entre les activitats i els diàlegs que componen l'aplicació i els mètodes de *Callback* juntament amb els valors enviats per aquests.

3.1.2. Arquitectura multicapa

Seguint la filosofia d'Android també s'aconsegueix que cada un dels mòduls tingui certa estructura multicapa. L'avantatge de separar-ho en capes diferents és que facilita el manteniment de l'aplicació. S'hi poden trobar les tres capes següents:

- Capa de presentació: que defineix la interfície gràfica i majoritàriament es defineix en fitxers XML.
- Capa de negoci: la qual es programa en Java i és on es faran els càlculs i el processament de dades.
- Capa de persistència: és la capa responsable d'interrogar i modificar la base de dades i es programa en Java.



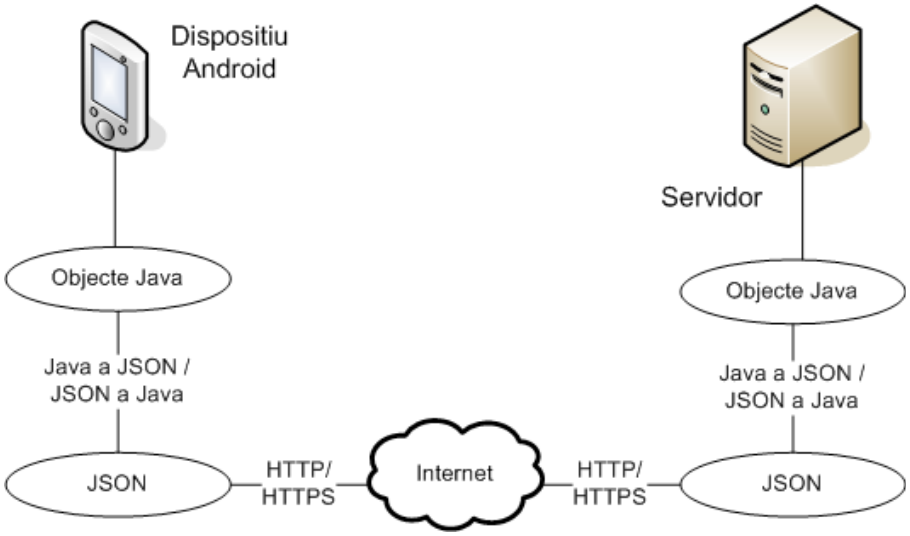
El mateix Android ofereix funcions pels missatges de connexió entre la capa de negoci i la capa de presentació. Els missatges entre la capa de negoci i la capa de persistència seran objectes del tipus *Placemark* com es veurà més endavant a l'apartat de disseny de classes per la persistència.

3.1.3. Arquitectura client - servidor

La comunicació entre el dispositiu Android i el servidor públic és del tipus *client – servidor*, que és l'arquitectura que estem més acostumats a utilitzar a Internet: el servidor proveeix una serie de serveis i espera a les peticions dels clients. En el cas de l'aplicació a desenvolupar, el servidor conté una serie de Servlets que esperen a que un client inicii la comunicació. El servidor/Servlet, en rebre la petició analitza les dades d'aquesta i construeix la resposta dinàmicament per tornar el

resultat al client. Com a dades d'entrada i de sortida dels Servlets s'utilitzaran classes comunes als dos costats (el client i el servidor) i, per tant, els dos han de conèixer. La transmissió dels missatges sempre serà a través d'Internet i mitjançant el protocol HTTP per transmissió de dades no confidencials i HTTPS per la transmissió de dades confidencials (com contrasenyes). Els objectes Java instanciats a partir de les classes d'intercanvi no es poden transmetre directament i serà necessari transformar-los en una estructura de dades compatible, que en aquest cas serà text pla estructurat seguint la notació de JSON.

En el següent diagrama es pot veure el funcionament de la comunicació entre client i servidor:



Il·lustració 5: Comunicació entre el dispositiu client i el servidor

3.2. Tecnologies

En aquest apartat s'expliquen les tecnologies utilitzades i els motius pels quals s'han escollit. Cal tenir en compte que el sistema de desenvolupament és diferent al sistema un cop en producció i les complicacions que aquest fet pot generar. Les proves es faran sobre un entorn simulant les condicions finals (amb l'emulador de l'Android SDK) i, si és possible, algunes proves sobre un dispositiu mòbil real.

3.2.1. Eines de desenvolupament

Pel que fa el desenvolupament, el llenguatge de programació utilitzat per aquest sistema operatiu (Android) és el Java, però sempre seguint les regles que imposa l'Android SDK, amb les funcionalitats que aquest ofereix i les limitacions que imposa.

IDE

El desenvolupament amb l'Android SDK es pot fer sobre diferents plataformes per línia de comandes o integrat en un IDE, en aquest cas es farà sobre un sistema Linux i utilitzant l'IDE Eclipse. S'ha escollit aquest IDE per la possibilitat d'integrar fàcilment totes les eines de desenvolupament de l'Android SDK.

Servidor

Durant el desenvolupament s'utilitzarà un Tomcat versió 7 que s'executarà localment dins de l'entorn de desenvolupament (eclipse) com a contenidor dels Servlets. S'ha escollit aquest contenidor perquè és un dels més utilitzats i suportats i perquè és fàcil integrar-lo a la resta de les eines de desenvolupament.

Emulador Android

Per fer proves durant el desenvolupament de l'aplicació s'utilitzarà l'emulador que incorpora el kit de desenvolupament de l'Android.

3.2.2. Servidor

Una part del sistema necessita accedir a un servidor públic i compartit per tots els usuaris. La tecnologia utilitzada en aquest servidor és la següent:

Servlets

Els Servlets respondran les peticions dels clients amb la informació sol·licitada per aquests. S'ha escollit perquè pel sistema en qüestió és el més apropiat ja que s'ajusta a les necessitats d'aquest:

- Funcionament petició – resposta, que és el que es necessita en aquest cas.
- Funciona sobre HTTP/HTTPS, el qual significa que es minimitzen problemes com, per exemple, configuracions de ports.
- Pot funcionar amb qualsevol contenidor de Servlets.
- Llibertat en la definició dels missatges que s'intercanvien.

Contenidor de servlets

Com s'ha comentat, durant el desenvolupament s'utilitzarà un Tomcat versió 7 però teòricament, a l'entorn de producció, es podrà utilitzar qualsevol contenidor web que suporti les funcionalitats necessàries.

Base de dades

La base de dades utilitzada en el servidor serà MySQL perquè en aquesta gamma de preus (gratis) és una de les més potents i més utilitzada. Tot i així, com que s'utilitzarà un llenguatge SQL estàndard per interrogar la base de dades, un canvi posterior a un altre sistema de base de dades no hauria de suposar gaires complicacions.

3.2.3. Client

La tecnologia utilitzada en el client és la següent:

Sistema operatiu

La tecnologia final sobre la qual s'executarà l'aplicació resultant d'aquest projecte serà un Android a partir de la versió 2.2 (que correspon al nivell 8 i posteriors de l'API d'Android). S'ha decidit suportar qualsevol versió començant per la 2.2 perquè la mateixa documentació d'Android ho recomana per arribar a la majoria d'usuaris (concretament es parla d'un 93%). Tot i que ja s'ha publicat la versió 4.1, la majoria dels *Smartphones* que es venen avui en dia, encara funcionen amb la versió 2.3.

Dispositiu

Majoritàriament seran telèfons mòbils però també podrien ser *Tablets* o inclús ordinadors amb el sistema Android instal·lat (per emulador o una versió per processadors d'ordinador). Per poder ajustar l'aplicació al màxim als diferents tipus de pantalles d'aquests dispositius, s'ha decidit fer ús de les funcionalitats que ofereix Android per fer front a aquesta diversitat.

Base de dades

S'utilitzarà la base de dades SQLite perquè és la que els dispositius d'Android ja porten incorporats per defecte. Aquesta base de dades té l'avantatge que és poc pesant en quant a

espai i a processament, el qual són factors importants a tenir en compte en un dispositiu mòbil.

GPS i/o network location provider

Per la localització geogràfica s'utilitzarà tant el GPS, pels avantatges en la precisió i la cobertura, com la tecnologia de *network location provider*, pels avantatges de la seva rapidesa i el consum d'energia en comparació amb el sensor de GPS.

3G o Wi-Fi

Per les operacions que contacten un servidor web i per poder obtenir els mapes que es visualitzen, és necessària una connexió a Internet. Sempre que sigui possible s'utilitzarà Wi-Fi pels avantatges que té respecte 3G (cost i velocitat) i, si no és possible, s'intentarà utilitzar aquest segon per l'avantatge de cobertura que té respecte el primer.

Mapes

Per mostrar el mapa s'utilitzarà la API de Google Maps i la d'Open Street Maps perquè tots dos són sistemes molt complets.

Magnetòmetre o giroscopi

Segons quin proporciona el dispositius, s'utilitzarà un o altre per obtenir informació sobre la direcció. Tot i que també s'utilitzarà un sistema de mapes on es podrà veure en quina direcció es troba un punt en concret, s'ha decidit fer servir la informació orientativa per complementar-ho ja que té l'avantatge que no necessita cap connexió a Internet.

3.2.4. Comunicació

Per la comunicació entre el dispositiu Android i el servidor públic es faran servir les següents tecnologies:

Protocol de transport

S'utilitzarà HTTP i HTTPS perquè permeten fer tot el que cal pel sistema en qüestió i no fer-lo més complexe del que és necessari:

- Es minimitzen problemes com, per exemple, configuracions de ports.
- Llibertat en la definició dels missatges que s'intercanvien.

Missatges d'intercanvi

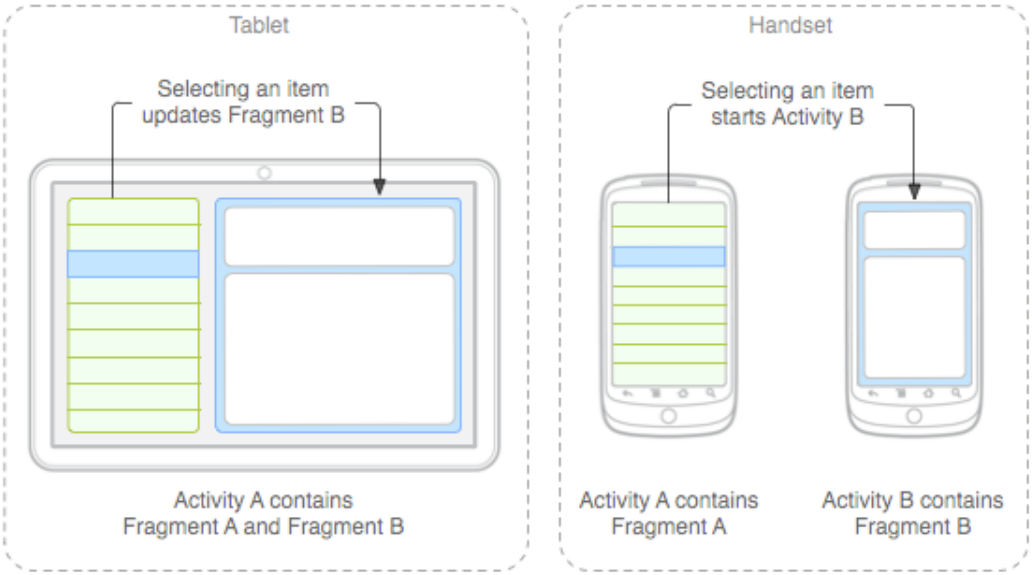
S'utilitzarà JSON per serialitzar/deserialitzar objectes Java i poder-los intercanviar entre el client i el servidor. Té l'avantatge que es manté cert ordre en l'intercanvi de missatges però la quantitat de dades addicionals a transmetre és relativament petita (comparat amb altres, com per exemple XML). Això és un punt important a tenir en compte perquè Internet en els dispositius mòbils no és el mateix que Internet a casa, tant pel que fa la velocitat com pel cost econòmic (si és per 3G).

3.3. Disseny de classes

3.3.1. Activitats i diàlegs

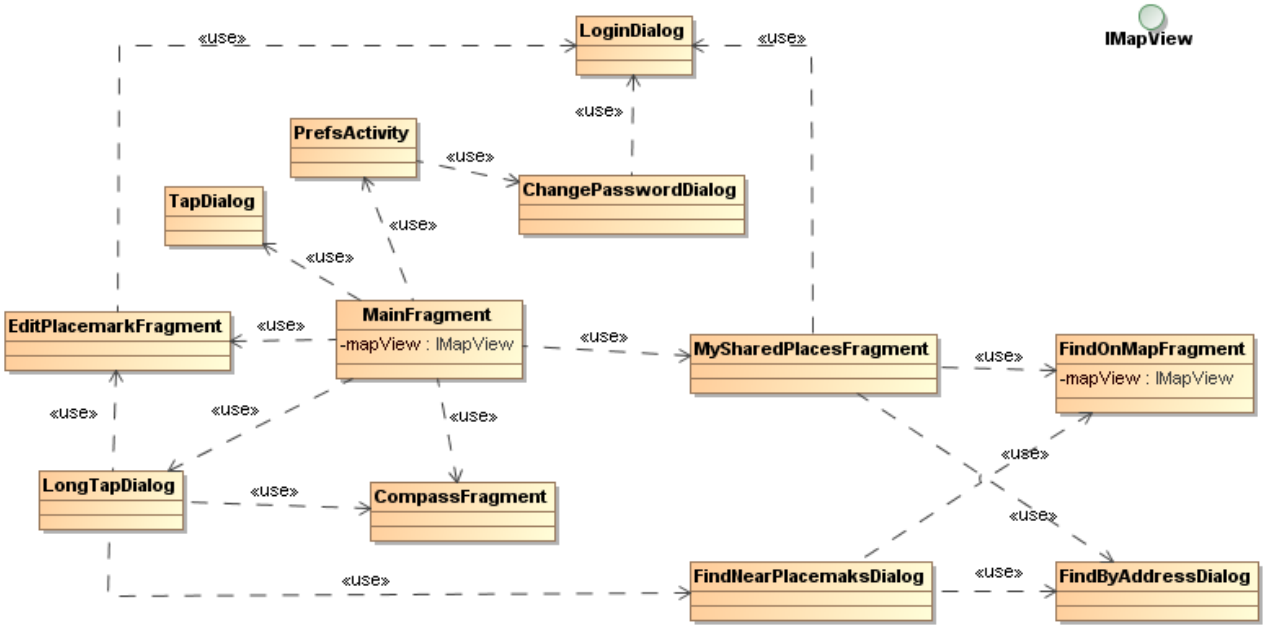
Com s'ha comentat a l'apartat de Tecnologies, el sistema Android pot funcionar sobre una amplia gamma de dispositius amb prestacions diferents. Per aquest motiu s'ha decidit fer ús de les funcionalitats que ofereix Android per fer front als diferents tipus de pantalles d'aquests dispositius. El que es pretén inicialment és que tot el que s'implementarà estigui preparat per poder suportar aquesta funcionalitat. Això afecta el disseny de les classes en quant a les activitats i els diàlegs que tindrà l'aplicació. Totes les activitats, en comptes d'heretar directament de la classe d'activitats, hereten d'una classe que suporta fragments. D'aquesta manera es facilitarà l'adaptació

de l'aplicació a tots els tipus de pantalles, organitzant els fragments de maneres diferents com es pot veure a la següent imatge:



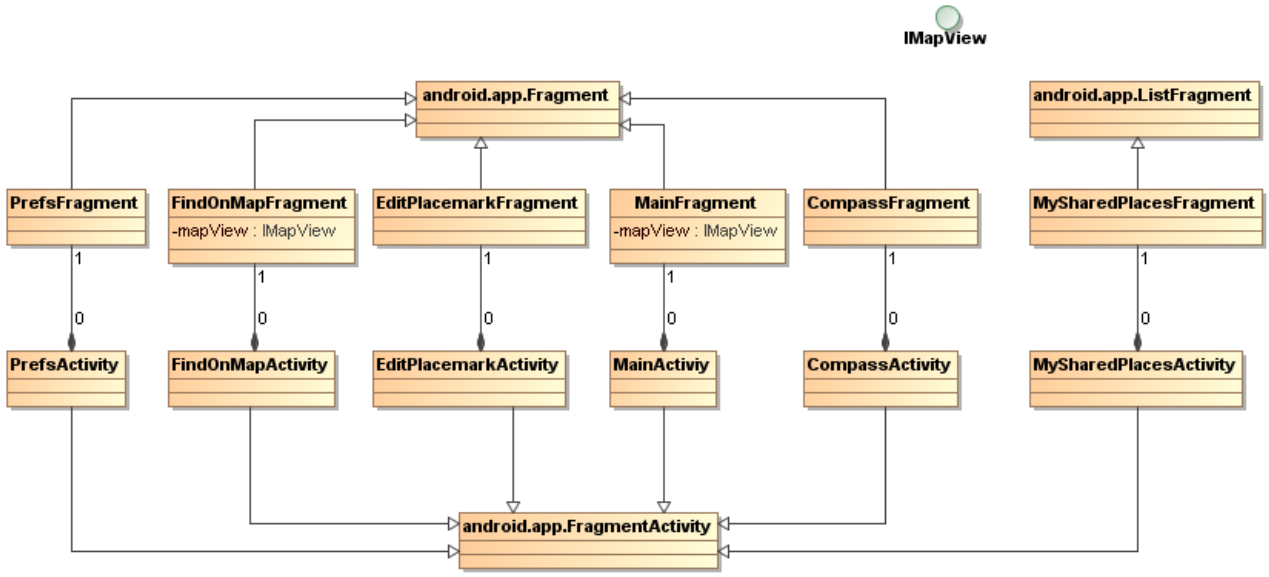
Il·lustració 6: Dispositius amb pantalles diferents poden organitzar el contingut de manera diferent

En el següent diagrama es pot veure quines activitats i diàlegs formaran part de l'aplicació i la relació entre ells. També s'ha indicat l'atribut *IMapView* en les classes corresponents (*MainFragment* i *FindOnMapFragment*) el qual serà el mapa d'un dels dos tipus suportats (Google Maps o Open Street Maps) com s'explica més endavant. En aquest primer diagrama no té gaire importància el comportament fragmentat ja que seria molt semblant sense suportar-lo:



Il·lustració 7: Relacions entre activitats i diàlegs

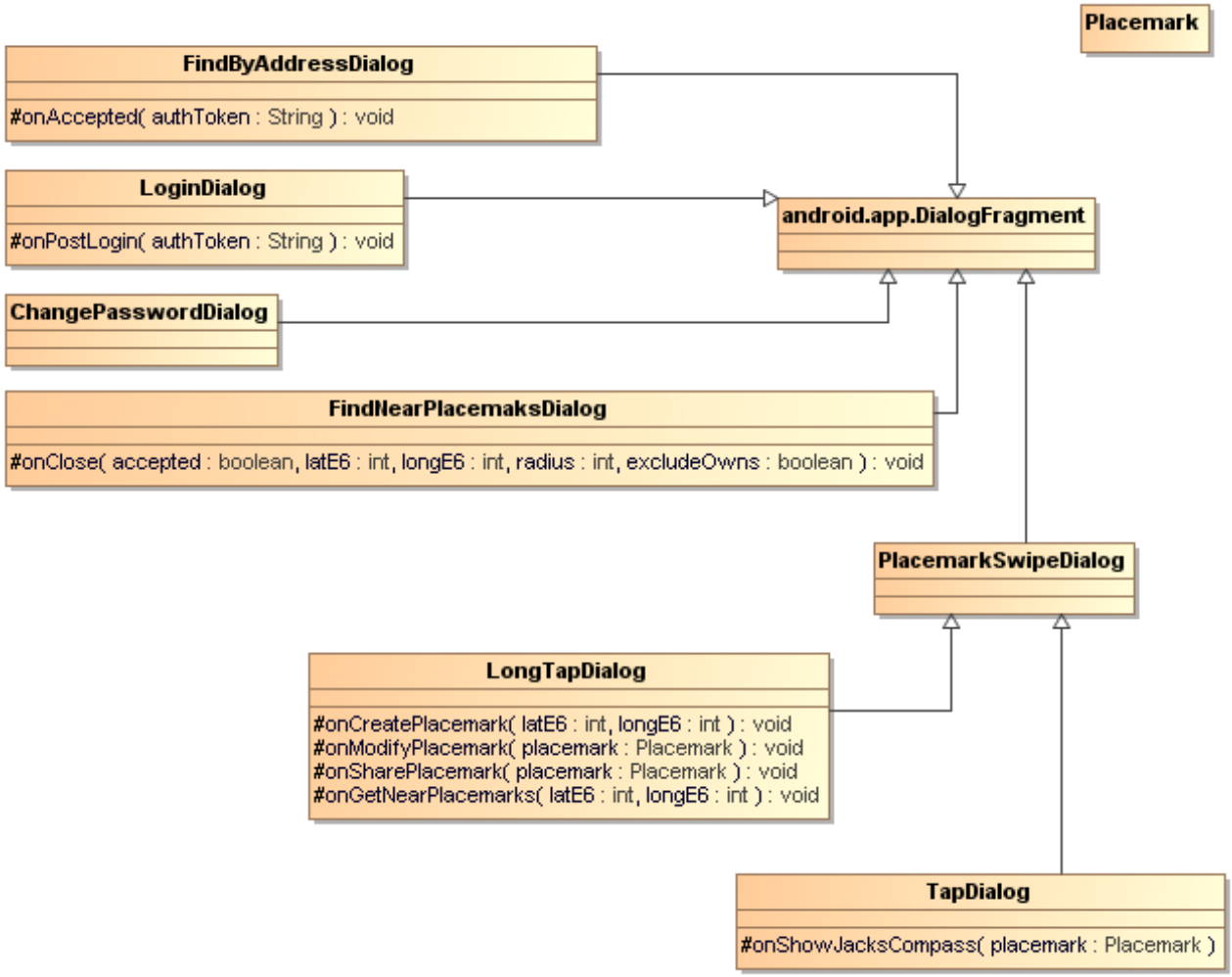
A continuació es mostra un diagrama on s'han aïllat les activitats per veure com s'implementaran per suportar el comportament fragmentat. Totes les activitats hereten d'una classe amb suport per fragments i s'utilitzen des d'una activitat normal:



Il·lustració 8: Herència de les activitats

En el diagrama que es mostra a la il·lustració següent s'han aïllat els diàlegs per veure com s'implementen per suportar el comportament fragmentat. Tots els diàlegs hereten d'una classe que suporta el comportament fragmentat. En aquest diagrama també s'han indicat els mètodes de *Callback* que proporcionaran els diferents diàlegs. Aquests *Callbacks* definits tenen les següents funcions:

- *FindByAddressDialog.onAccepted()*: es cridarà quan es tanca el diàleg acceptant.
- *LongTapDialog.onCreatePlacemark()*: es cridarà en escollir l'opció de crear una marca.
- *LongTapDialog.onModifyPlacemark()*: es cridarà en escollir l'opció de modificar una marca.
- *LongTapDialog.onSharePlacemark()*: es cridarà en escollir l'opció de compartir una marca.
- *LongTapDialog.onGetNearPlacemarks()*: es cridarà en escollir l'opció de trobar marques compartides en una àrea.
- *LoginDialog.onPostLogin()*: es cridarà un cop ha acabat l'identificació en el servidor i s'ha obtingut un *authToken*.
- *FindNearPlacemaksDialog.onClose()*: es cridarà en tancar el diàleg (quan l'usuari confirma o cancel·la les dades entrades).



Il·lustració 9: Callbacks i herència dels diàlegs

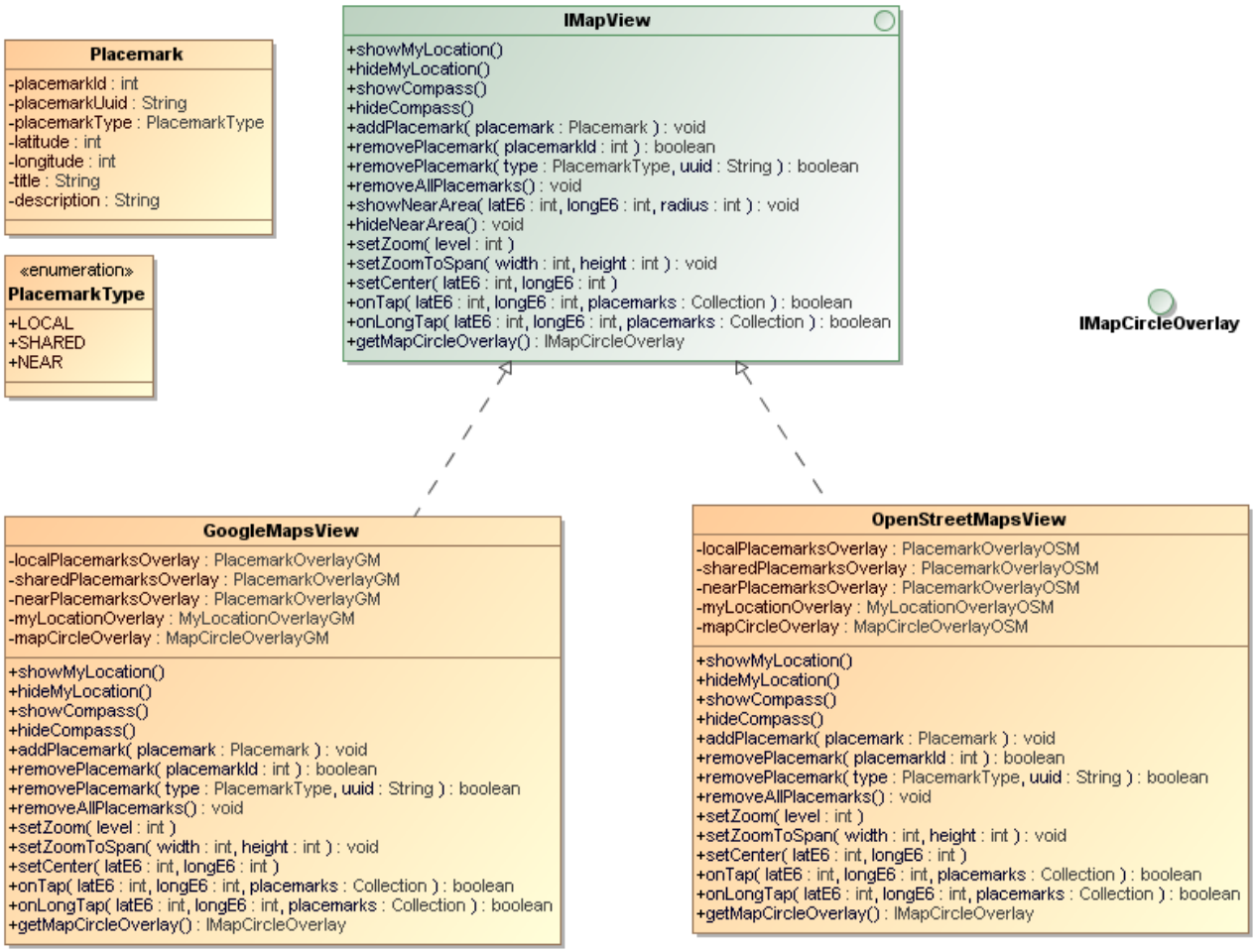
3.3.2. MapViews

Els dos tipus de mapes suportats per aquesta aplicació són els mapes de Google Maps i els d'Open Street Maps. Tot i que les dues llibreries proporcionen mètodes i classes semblants, no són exactament iguales i no implementen interfícies comunes perquè són desenvolupades per entitats independents. Per aquesta aplicació, s'ha decidit fer dues vistes diferents implementant una interfície comuna per abstreure la implementació individual de cada una de les vistes. D'aquesta manera les dues vistes es podran utilitzar de la mateixa manera a la resta de l'aplicació.

En el diagrama que hi ha a continuació es pot veure la interfície comuna i les dues vistes. En cada una de les dues vistes s'hi poden observar els següents atributs:

- *localPlacemarksOverlay*: contindrà les marques locals de l'usuari que s'estan mostrant en cada moment.
- *sharedPlacemarksOverlay*: contindrà les marques compartides de l'usuari (obtingudes del servidor) que s'estan mostrant en cada moment.
- *nearPlacemarksOverlay*: contindrà les marques resultants d'una cerca de marques properes (obtingudes del servidor).

- *myLocationOverlay*: marca la posició actual del dispositiu.
- *mapCircleOverlay*: és el tipus de capa que marca una àrea.

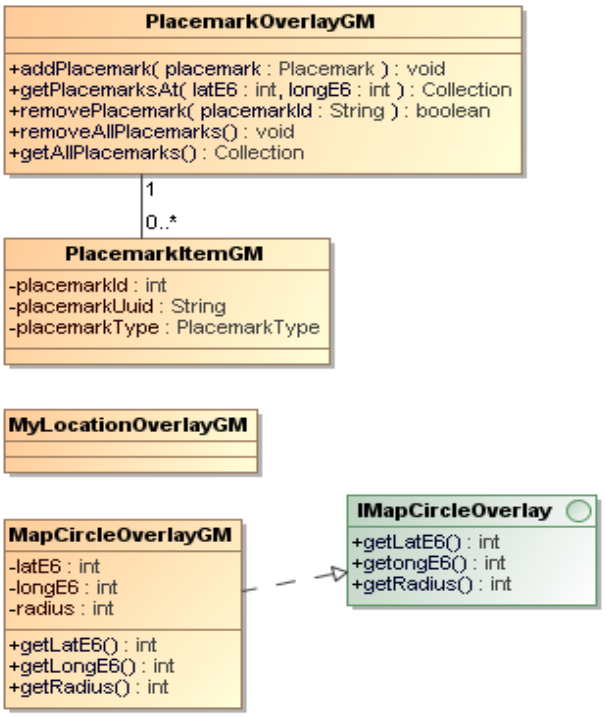


Il·lustració 10: Les dues vistes de mapes que suportarà l'aplicació

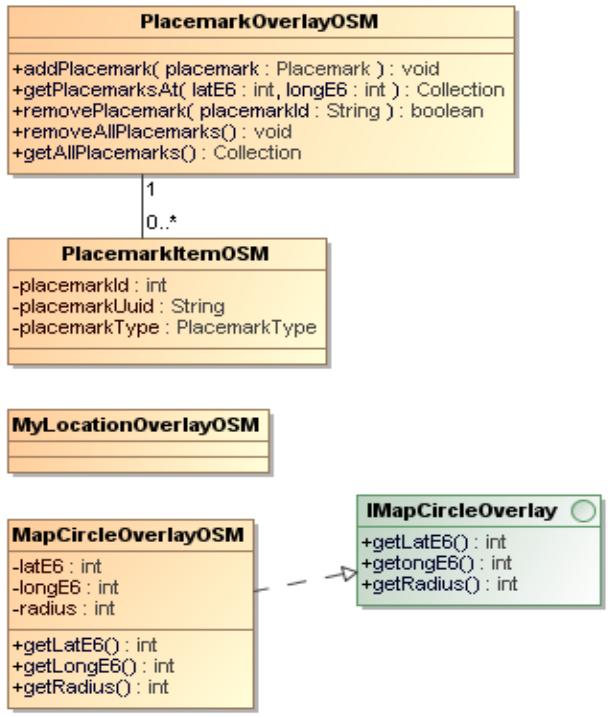
3.3.3. Overlays

El mètode que s'utilitza per mostrar informació en el mapa, és crear capes que s'hi superposen. Per poder mostrar la informació rellevant per l'aplicació que s'esta desenvolupant són necessàries les tres capes següents:

- *PlacemarkOverlay*: és el tipus de capa que mostra els llocs marcats, per fer-ho conté una llista de marques (representat per l'associació amb *PlacemarkItem*). Com s'ha explicat a l'apartat anterior de *MapViews*, les dues vistes (*GoogleMapsView* i *OpenStreetMapsView*) defineixen tres atributs d'aquest tipus.
- *MyLocationOverlay*: és el tipus de capa que mostra la posició actual del dispositiu.
- *MapCircleOverlay*: és el tipus de capa que marca una àrea.



Il·lustració 11: Capes per la vista de Google Maps



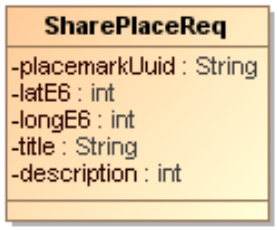
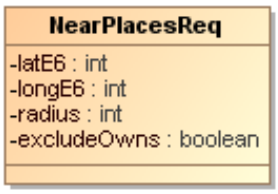
Il·lustració 12: Capes per la vista d'Open Street Maps

3.3.4. Servlets

Els dispositius Android faran peticions a un servidor públic i comú per tots els usuaris per consultar o modificar les dades que emmagatzema. S'implementen una seria de Servlets que s'executaran en el servidor per respondre a aquestes peticions. Com s'ha comentat a l'apartat de Tecnologies, s'utilitzen objectes Java serialitzats en format JSON com a missatges d'intercanvi entre el client i el servidor. Cada un d'aquests objectes Java exten un objecte base que defineix els paràmetres comuns entre tots ells. Les classes que tenen la funció de de sol·licitud (request) extenen *BaseReq* i les de resposta (response) extenen el *BaseResp*:

La classe *BaseReq* defineix un atribut *authToken* que identifica l'usuari que fa la petició, en el cas que la petició necessita identificació. *BaseResp* defineix un atribut *msgError* que contindrà un missatge informatiu si les dades proporcionades per la petició no són correcte o si es produeix un error en el servidor.

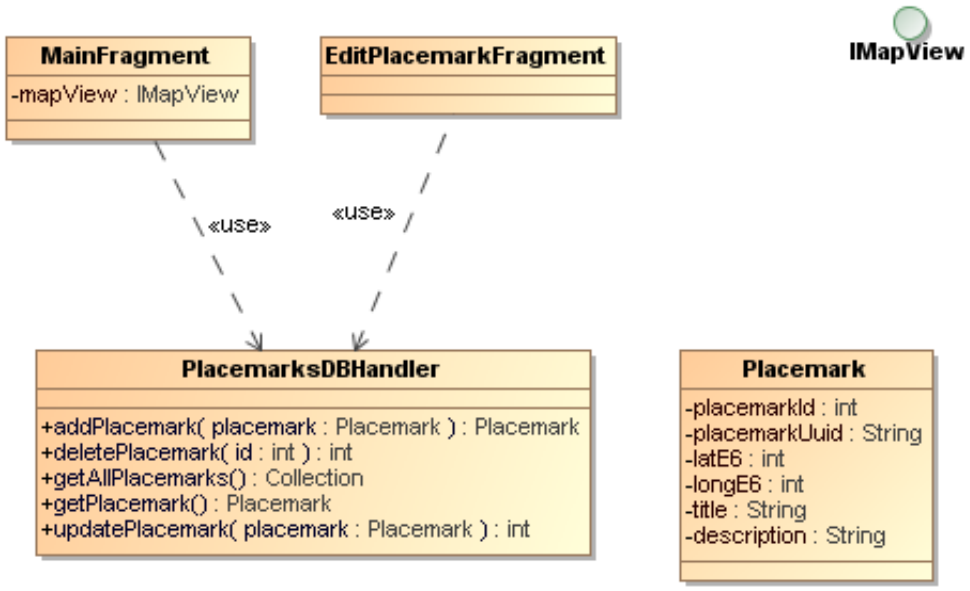
Les classes dels diferents Sevlets i les classes per l'intercanvi de dades entre client i servidor són les següents:

Servlet	Descripció	Objecte request	Objecte response
SharePlace	Servlet que es crida per compartir públicament una marca de l'usuari.		
UnsharePlace	Servlet que es crida per deixar de compartir públicament una marca de l'usuari.		
GetMySharedPlaces	Servlet que es crida per obtenir una llista de totes les marques pròpies compartides per l'usuari.		
GetNearPlaces	Servlet que es crida per obtenir marques compartides públicament (per qualsevol usuari).		
CreateAccount	Servlet que es crida per registrar un usuari nou.		

Login	Servlet que es crida per identificar un usuari.		
ChangePassword	Servlet que es crida per canviar la contrasenya de l'usuari.		

3.3.5. Persistència

Els dos fragments *MainFragment* i *EditPlacemarkFragment* es troben a la capa de negoci i són els únics que accediran a la capa de persistència per consultar o modificar la base de dades. A part d'aquestes dues classes, en el diagrama següent es pot veure la classe *PlacemarksDBHandler*, la qual forma part de la capa de persistència, i la classe *Placemark* que representa una marca i en aquest context té el rol de missatge entre la capa de persistència i la capa de negoci.



Il·lustració 13: Classes que intervenen en la persistència

3.4. Bases de dades

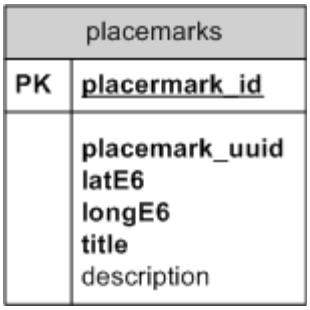
Les dues bases de dades (la del dispositiu Android i la del servidor) són bastant independents en el sentit que la base de dades local del dispositiu desconeix l'estat de la informació de la base de dades del servidor i al reves igual (no necessàriament han d'estar sincronitzades). El motiu d'això és que aquesta informació podria quedar inconsistent fàcilment (per exemple, si l'usuari utilitza dispositius diferents amb el mateix nom d'usuari per modificar les dades compartides). El camp que les relaciona és l'identificador del lloc (*placemark_uuid*).

També cal remarcar que els identificadors locals de les marques (*placemark_id*) són numèrics incrementals que els identifiquen de forma única localment. En el servidor públic, en canvi, no poden ser-ho perquè si l'usuari comparteix marques des de dispositius diferents podrien aparèixer problemes de duplicació de claus (un mateix usuari pot tenir marques amb identificadors iguals en dispositius diferents però no en el servidor). Per aquest motiu, les marques de llocs tenen un altre identificador universal (UUID) que s'utilitzarà per identificar un lloc compartit.

3.4.1. Dispositiu Android

La base de dades del dispositiu Android consta d'una única taula on es guarden totes les marques definides localment per l'usuari. No s'hi guarda cap informació relacionada amb l'estat de compartició de la marca (si s'ha compartit o no) pels problemes que s'han comentat anteriorment.

```
PLACEMARKS (
        placemark_id INT,
        placemark_uuid CHAR(36),
        latE6 INT,
        longE6 INT,
        title TEXT,
        description TEXT)
```



3.4.2. Servidor

Les tres taules del servidor emmagatzemen el següent:

- A la taula users es guarda la informació dels usuaris registrats.
- A la taula shared_places es guarden els llocs compartits. Si un lloc no esta compartit, directament no apareix a la taula.
- A la taula auth_tokens es guarden *tokens* d'identificació que es donen als usuaris que s'identifiquen perquè no s'hagin d'identificar per cada petició que fan (seria equivalent a mantenir una sessió).

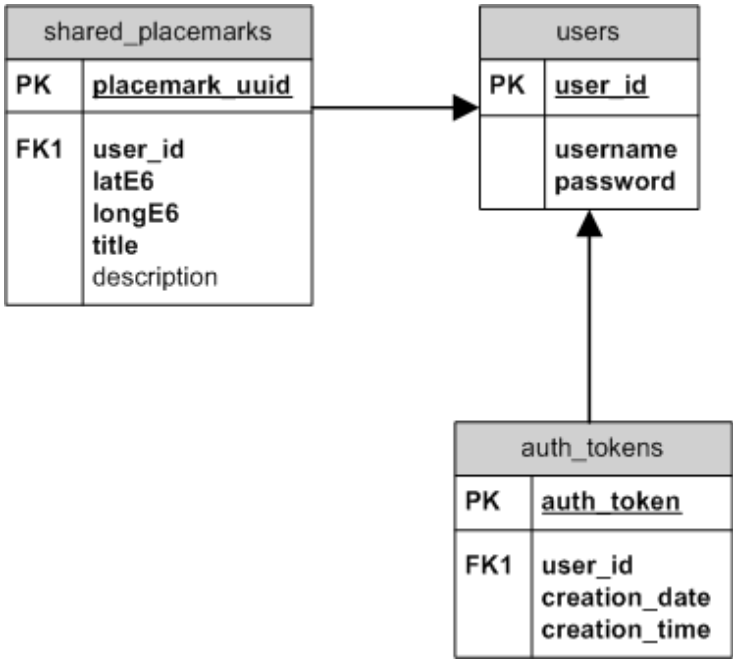
```
USERS (
        user_id INT,
        username TEXT,
        password TEXT)


SHARED_PLACEMARKS (
        placemark_uuid CHAR(36),
        user_id INT,
        latE6 INT,
        longE6 INT,
```

```
        title TEXT,
        description TEXT)
user_id es clau forana a users.

AUTH_TOKENS (
        auth_token CHAR(36),
        user_id INT,
        creation_date CHAR(8),
        creation_time CHAR(6))
user_id es clau forana a users.
```

L'esquema de la base de dades és el següent:



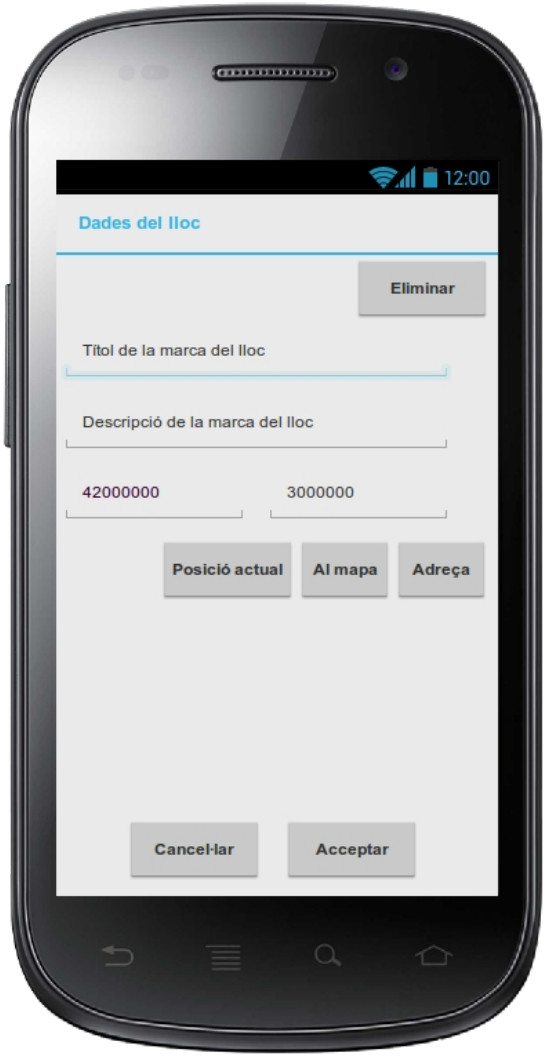
Tenint en compte la informació dels dos punts anteriors, es pot veure que serà possible definir una marca localment que no estarà compartida (el qual és clarament desitjat) però també seria possible tenir una marca compartida sense tenir-la localment (el qual podria semblar menys lògic). Aquest fet es podria evitar per codi però s'ha decidit que serà una característica vàlida per donar més llibertat als usuaris del sistema. Això afecta algunes parts de la resta de l'aplicació ja que l'usuari haurà de tenir alguna manera d'accedir als seus llocs compartits, encara que s'hagin eliminat localment. Com ja s'ha vist en el disseny de classes i com es veurà en el prototipus, s'ha tingut en compte afegint una opció per llistar i operar sobre les marques pròpies compartides independentment de l'estat local de la marca.

4. Prototipus

El prototipus desenvolupat durant la fase de disseny del sistema tenia la funció d'una primera aproximació de la interfície gràfica i la distribució de les opcions de l'aplicació. Inicialment contenia el disseny de cada una de les pantalles que es poden trobar a l'aplicació desenvolupada. Donat que el prototipus complet ja es va entregar en el document corresponent i com que en el document actual ja es disposa de les pantalles finals de l'aplicació real que es mostraran més endavant, a continuació només es mostren les pantalles principals el prototipus original.

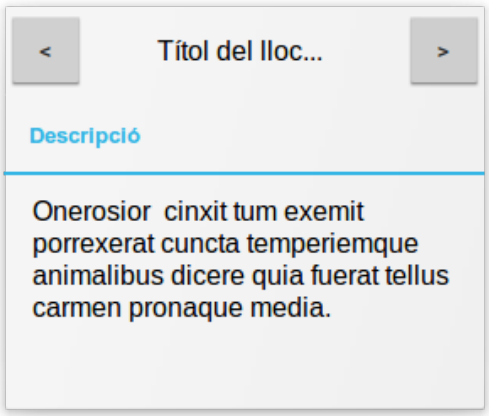
Pantalla principal i pantalla d'edició de marca

Il·lustració 14: Activitat pantalla principal

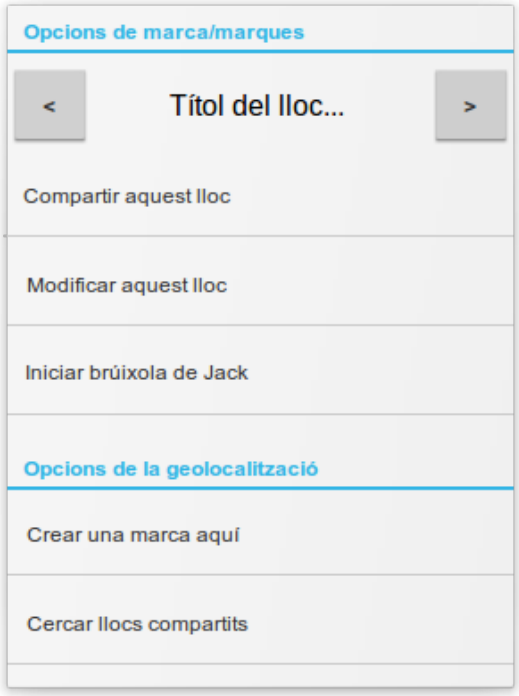


Il·lustració 15: Activitat de modificació d'una marca existent

Diàleg informatiu de marca i diàleg contextual de marca i geoposició



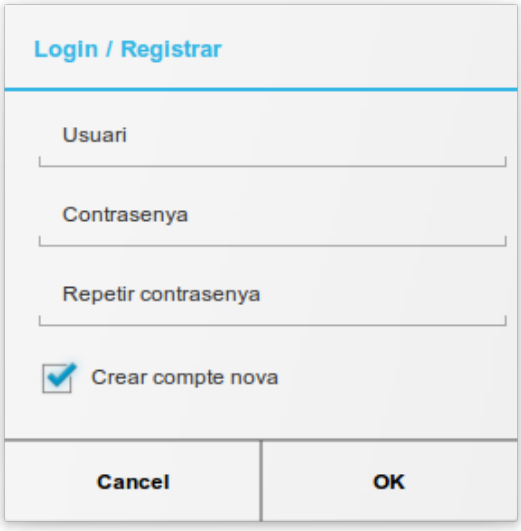
Il·lustració 16: Diàleg d'informació d'un lloc marcat



Il·lustració 17: Diàleg contextual d'una marca

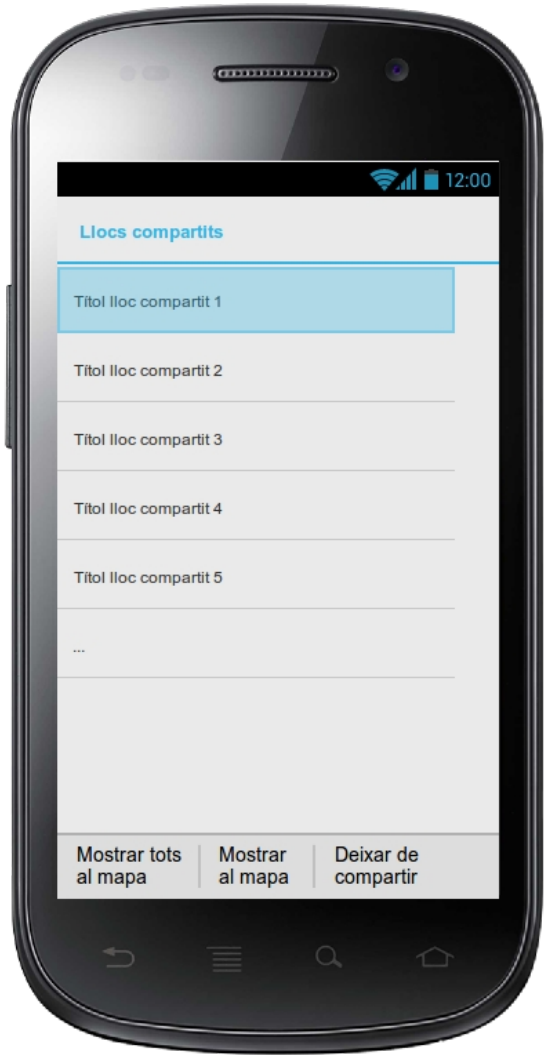
Diàleg de cerca per carrer i diàleg de registre / identificació

Il·lustració 18: Diàleg de resultats múltiples de cerca de posició per adreça



Il·lustració 19: Diàleg d'usuari nou

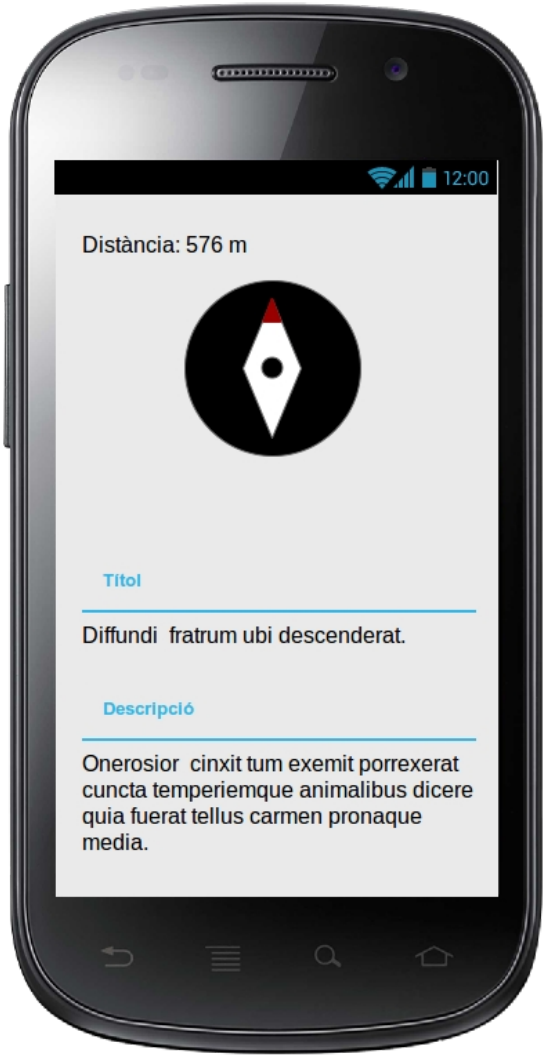
Llista de llocs propis compartits i diàleg de cerca de llocs compartits



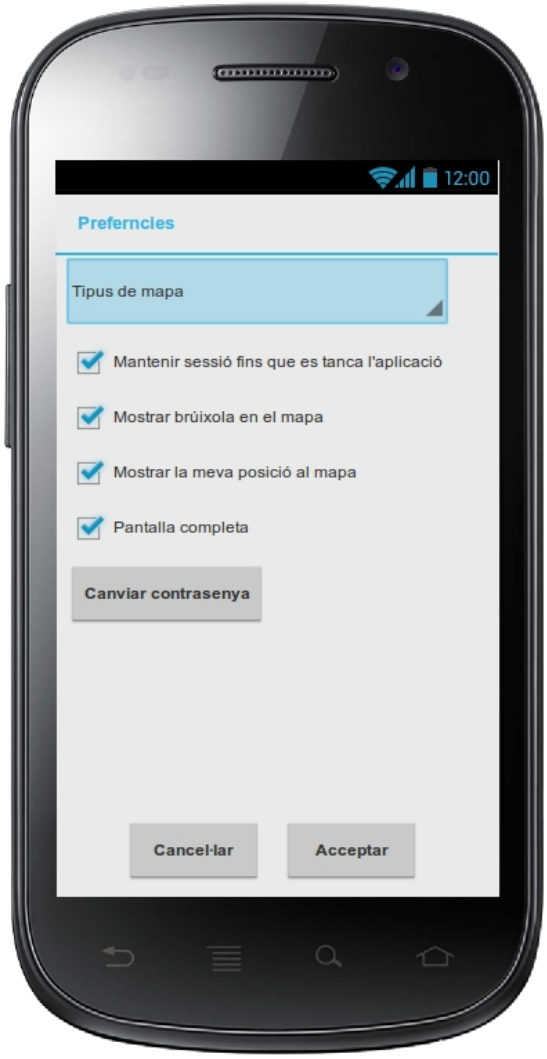
Il·lustració 20: Activitat amb la llista de llocs compartits propis

Il·lustració 21: Diàleg per escollir la posició i el radi de l'àrea

Brúixola orientativa i preferències de l'aplicació



Il·lustració 22: Activitat brúixola



Il·lustració 23: Activitat de preferències

5. Implementació

Durant la implementació de l'aplicació del projecte s'han hagut de prendre decisions i s'han trobat problemes causats per les limitacions o, en alguns casos, errors del sistema base que s'han hagut de resoldre. A continuació s'explicaran els punts més importants d'aquesta fase d'implementació.

5.1. Estructura de projectes

Per l'aplicació desenvolupada s'han creat dos projectes diferents. El primer per la part client que es podrà instal·lar i executar en un dispositiu Android i el segon és el que s'executarà en el servidor public per rebre i respondre les peticions dels dispositius client.

5.1.1. Projecte pel Client

A partir del projecte definit per la part client, al final de tot el procés es crearà un paquet APK estàndard del sistema Android. Aquest paquet contindrà tota la informació necessària (codi, llibreries, imatges, etc.) per possibilitar la instal·lació de l'aplicació en un dispositiu que funciona amb aquest sistema operatiu. En aquest apartat s'expliquen les principals característiques d'aquest projecte.

Fitxer *AndroidManifest.xml*

Tots els projecte Android tenen un fitxer anomenat *AndroidManifest.xml* on es defineixen les propietats bàsiques de l'aplicació. Pel projecte actual s'hi ha definit el següent:

- La versió mínim a suportar i la versió contra la qual es compila:

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="16" />
```

  Com s'ha explicat anteriorment, la versió mínim que es vol suportar és la versió 8 de l'API d'Android ja que d'aquesta manera funcionarà per la majoria de dispositius que es poden trobar al mercat avui en dia. Segons la documentació, es recomana compilar contra la última versió publicada pels desenvolupadors del sistema que actualment és la versió 16.

- Permisos que necessita l'aplicació per poder funcionar:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

  El permís d'Internet és necessari ja que l'aplicació ha de poder accedir a Internet tant per baixar els mapes com per contactar el servidor comú. Els dos permisos "*coarse location*" i "*fine location*" són els permisos per obtenir accés a la posició actual del dispositius que és necessària per algunes *opcions de l'aplicació. Els dos permisos "access network state"* i *" write external storage"* són necessaris per l'open street maps. Tot i que no s'explica en

detall perquè són necessaris, el primer segurament és per actuar en funció de l'estat de la i el segon (permís d'escriure a un emmagatzemament extern) segurament és per poder fer cache dels mapes baixats.

- Els mapes de Google Maps no són un element estàndard d'Android (alguns dispositius no ho suporten per defecte). Per aquest motiu és necessari definir explícitament que aquesta aplicació fa ús d'aquesta llibreria:

```
<uses-library android:name="com.google.android.maps" />
```

- Finalment, en el fitxer es defineix quines activitats composen l'aplicació i quina és l'activitat principal. Aquesta activitat principal és la que s'obrira al clicar la icona corresponent que apareixerà el lloc on hi ha totes les aplicacions instal·lades del dispositiu.

Recursos

Qualsevol recurs que es farà servir en algun punt de l'aplicació, es localitzarà en una estructura de carpetes organitzada. En aquesta carpeta es troben els XMLs que defineixen la interfície gràfica de l'aplicació, els textos, les imatges, etc.

Les imatges es divideixen en quatres categories i segons la resolució de la pantalla del dispositiu mòbil, Android escull una imatge d'aquestes categories. Per l'aplicació actual s'han inclòs imatges de dues mides diferents:

- *Drawables-hdpi* per resolucions altes.
- *Drawables-mdpi* per resolucions mitjanes.

Per les dues restants (*drawables-ldpi* i *drawables-xhdpi*) no s'han definit imatges explícites i s'agafarà una de les dues categories anteriors en funció de la que s'adapta millor.

Tot i que tots els textos es podrien posar directament a l'XML de la interfície gràfica, s'ha seguit el mètode recomanat per Android de definir els textos en un fitxer XML a part. De moment no té gaire importància ja que l'aplicació només estarà disponible en un idioma, però en un futur facilitaria la traducció de l'aplicació a altres idiomes afegint un fitxer XML per cada idioma que es vulgui suportar.

Llibreries

A part de les llibreries que Android inclou per defecte, s'han afegit les següents llibreries de les quals s'utilitzen algunes de les funcionalitats que ofereixen:

- android-support-v4-r10-googlemaps.jar: s'utilitza pel funcionament de Fragments en versions inferiors a l'API 11 tal com s'explica a l'apartat de Fragments.
- gson-2.2.2.jar: s'utilitza per serialitzar i deserialitzar les peticions i respostes de la comunicació entre client i servidor tal com s'explica a l'apartat de comunicció amb el servidor.
- osmdroid-android-3.0.8.jar: és la llibreria que ofereix les funcionalitats per la gestió de mapes d'Open Street Maps.
- slf4j-android-1.5.8.jar: és una llibreria per funcionalitats de *log* que no és fa servir directament però és requerida per la llibreria *osmdroid*.

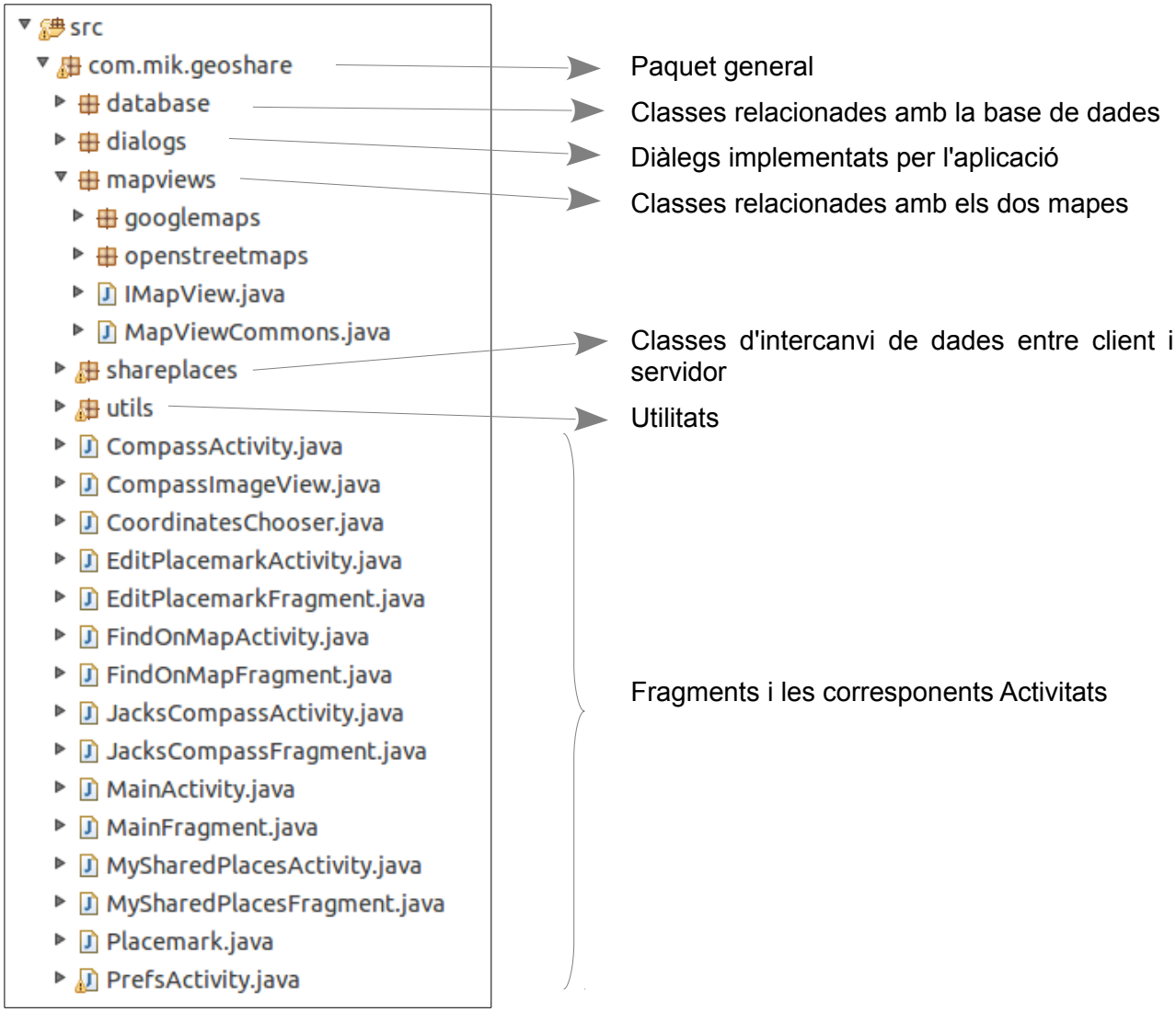
Fitxer de propietats

El fitxer de propietats del client (*geoshare.properties*) conté les principals configuracions pel funcionament d'aquesta part de l'aplicació. En aquest cas s'han definit propietats referents a:

- La URL a traves de la qual es pot accedir als Servlets del servidor.
- La IP i el port per defectes que s'utilitzen per les connexions normals i les connexions segures en cas de no especificar-los a les preferències de l'aplicació.
- La *api-key* requerida perque l'aplicació tingui accés als mapes de Google Maps.
- La contrasenya del *keystore* que conté el certificat de la comunicació per HTTPS (només la contrasenya del *keystore*, no la del certificat).

Codi font

Per organitzar el codi font de l'aplicació s'ha creat una jerarquia que s'estructura en els següents paquets agrupant-lo per funcionalitats:



5.1.2. Projecte Servidor

El projecte del servidor és un projecte web a partir del qual, al final del procés es crearà un mòdul WAR que es podrà desplegar en qualsevol servidor web compatible. En aquest apartat s'expliquen les principals característiques d'estructuració d'aquest projecte web:

Fitxer *web.xml*

Es defineix un fitxer *web.xml* en el qual es descriuen tots els Servlets accessibles i com localitzar-los. Pel Servlet *SharePlace* de l'aplicació, per exemple, es definiria de la següent manera el lloc on es troba la classe del Servlet:

```
<servlet>
     <servlet-name>SharePlace</servlet-name>
     <display-name>SharePlace</display-name>
     <description></description>
     <servlet-class>com.mik.shareplaces.SharePlace</servlet-class>
</servlet>
```

El corresponent *mapping* del mateix Servlet seria el següent i descriu per quin nom s'hi podrà accedir:

```
<servlet-mapping>
     <servlet-name>SharePlace</servlet-name>
     <url-pattern>/SharePlace</url-pattern>
</servlet-mapping>
```

D'aquesta manera el Servlet serà accessible a traves de:

http://ip:port/SharePlaces/SharePlace

El *SharePlaces* és el nom del projecte que al final també serà el nom del mòdul WAR que es desplegarà en el servidor web i el *SharePlace* és el nom del Servlet en concret.

Llibreries

A part de les llibreries que s'inclouen per defecte pel fet de tractar-se d'un projecte web, s'han inclòs les següents llibreries per algunes funcionalitats que s'han fet servir:

- gson-2.2.2.jar: igual que per la part client, s'utilitza per serialitzar i deserialitzar les peticions i respostes de la comunicació entre client i servidor.
- jasypt-1.9.0.jar: s'utilitza per encriptar les contrasenyes dels usuaris tal com s'explica a l'apartat de Seguretats.

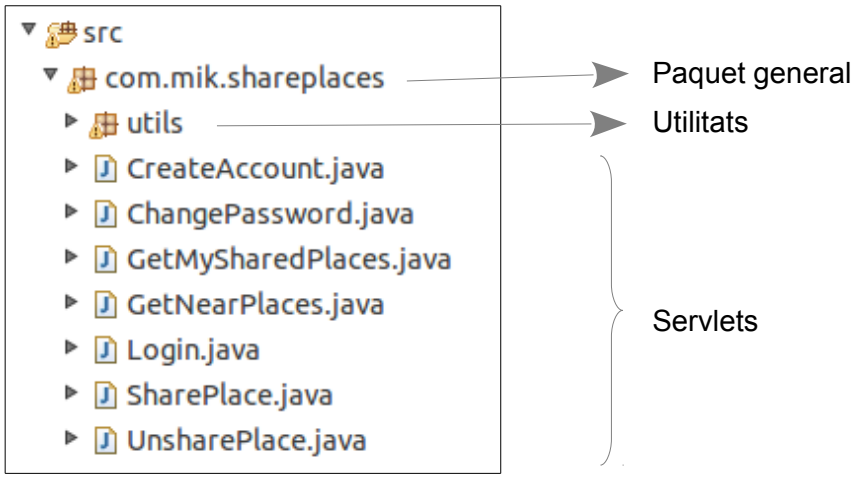
Fitxer de propietats

El fitxer de propietats del servidor (*shareplaces.properties*) conté les principals configuracions pel funcionament d'aquesta part. En aquest cas només és necessària una propietat: la contrasenya de

la base de dades.

Codi font

Gràcies a J2EE, la lògica de com es maneguen els Servlets i tot el que comporta queda amagada als programadors. D'aquesta manera, aquests es poden concentrar en el que realment volen fer, que en aquest cas és el codi dels Servlets en si. Per tant, el codi font d'aquest projecte són, bàsicament, els diferents Servlets que composen la part del servidor del projecte i algunes utilitats. S'estructura en els següents paquets:



5.1.3. Utilitats

Moltes operacions necessiten ser executades des de molts llocs diferents. Per no duplicar codi s'han creat classes d'utilitats que implementen el codi compartit i es criden des dels llocs on es necessita. Cal tenir en compte que les utilitats que es comparteixen entre servidor i client s'han implementat en el client per l'estructura dels projectes explicada anteriorment (el projecte del servidor veu el client però no al reves).

Servidor

- DateUtils: conté utilitats relacionades amb dates i hores com la conversió entre formats (a la base de dades, per exemple, es guarden com a cadenes de text en format AAAAMMDD).
- DBUtils: conté utilitats relacionades amb la base de dades com obrir i tancar connexions.
- ServerUtils: conté utilitats generals del servidor com el tractament d'excepcions o escriure la resposta HTTP.
- ValidationUtils: conté utilitats relacionades amb les validacions que el Servlets fan sobre els paràmetres d'entrada.

Client

- GeoUtils: conté utilitats relacionades amb les localitzacions geogràfiques com la conversió entre formats, els càlculs de distància o l'orientació entre punts, etc.
- JsonUtils: conté utilitats relacionades amb la serialització / deseralització (aquesta classe és compartida entre client i servidor).

- PropertiesUtils: conté utilitats per operar amb el fitxer de propietats.
- ServerUtils: conté utilitats relacionades amb la comunicació amb el servidor (veure Comunicació amb el servidor).
- StringUtils: conté utilitats relacionades amb les cadenes de text.
- ValidationUtils: conté utilitats relacionades amb les validacions que pot fer el dispositiu localment.

A més a més, s'ha creat una classe compartida entre client i servidor anomenada *Constants* que conté totes les constants que s'utilitzen en algun punt de l'aplicació (màxim de registres a seleccionar, radi de la terra, etc.).

5.2. Interfície gràfica d'usuari

Aquesta part només afecta l'aplicació client ja que la part del servidor d'aquest projecte no proporciona cap interfície gràfica perquè no és necessària.

5.2.1. Consideracions generals

Pel que fa la definició de la interfície gràfica, en general es fa mitjançant fitxers XML. L'avantatge d'això és que es manté separada la definició de la interfície gràfica de la resta del codi. Però cal notar, en aquest punt, que qualsevol element o propietat que es pot definir en el fitxer XML, també es pot definir per codi Java mitjançant els mètodes que exposen les classes corresponents. Pel projecte actual, en general s'ha definit tota la part de la interfície gràfica mitjançant aquests fitxers XML però s'ha avaluat en cada cas quina és la millor opció pel futur manteniment del codi.

Durant la definició de les pantalles s'ha intentat seguir les recomanacions i bones pràctiques tenint en compte l'usabilitat i la gran varietat de dispositius amb prestacions diferents que existeixen (mides de pantalles, teclats físics o virtuals, etc.). Així els diàlegs d'Android, per exemple, ja proporcionen els botons d'acceptar i cancel·lar, el desenvolupador únicament ha d'implementar el codi que cal executar en clicar el botó. Com que aquests botons per defecte apareixen en un ordre concret (primer el botó d'Acceptar i en segona posició el de Cancel·lar) per la resta de l'aplicació s'ha seguit el mateix patró (per exemple a l'activitat d'editar una marca).

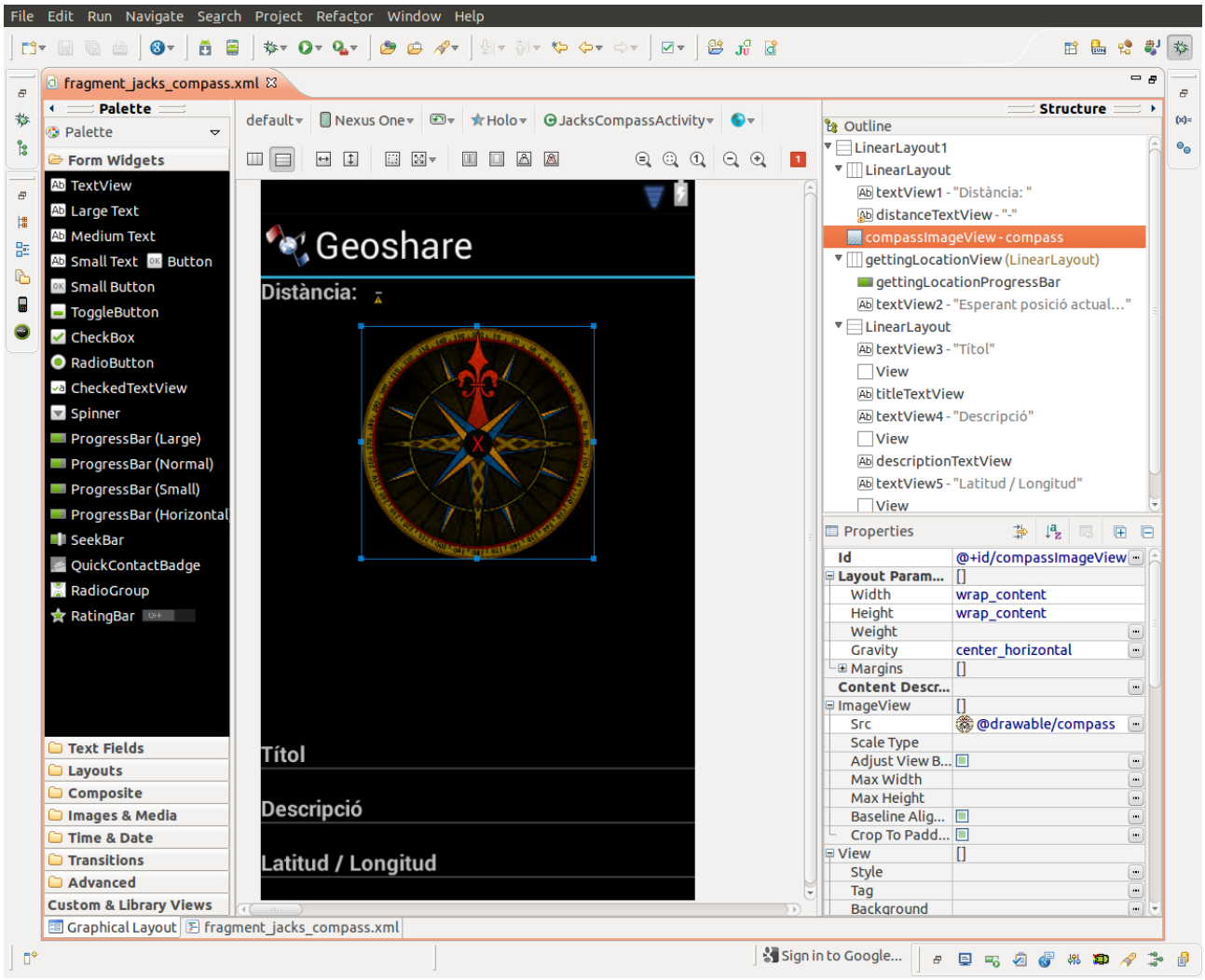
Cal tenir en compte que la major part del disseny ve definit pel sistema (igual que es coneix de molts altres sistemes operatius). Gracies a això és possible canviar el tema a les configuracions del sistema operatiu i totes les aplicacions segueixen el mateix estil definit (colors, tipus de botons, etc.). Per fomentar la integritat entre aplicacions, Android proporciona maneres que permeten referenciar estils (els colors per exemple). D'aquesta manera es pot aconseguir que els estils s'adapten en funció de la versió d'Android i del tema escollit per l'usuari. Inicialment s'ha desenvolupat l'aplicació fent ús d'aquestes referència i els resultats a l'emulador de l'Android SDK eren satisfactoris. Però a les primeres proves en un dispositiu real s'han trobat problemes que impossibilitaven utilitzar l'aplicació la qual es tancava llençant excepcions. Després de moltes proves, s'ha vist que els problemes eren causats per aquests estils referenciats i, tot i que podria ser un problema específic del dispositiu mòbil (Samsung Galaxy S i9000, versió 2.2 d'Android), s'ha decidit que el risc de que pugui passar amb altres dispositius és massa elevat i s'ha optat a no utilitzar aquestes referències.

5.2.2. Edició d'interfície gràfica

El plugin de desenvolupament d'Android que s'integra a la eina de desenvolupament utilitzada durant aquest projecte (eclipse) proporciona un editor visual per dissenyar les interfícies gràfiques

de l'aplicació. El resultat és un fitxer XML que defineix i organitza tots els elements que es mostraran en pantalla. També és possible modificar aquest XML manualment i en molts casos ha estat necessari retocar-lo per obtenir el resultat desitjat.

Per tindre una idea general, a la següent imatge es mostra aquest editor dins de l'entorn de desenvolupament:



Il·lustració 24: Editor visual d'interfícies gràfiques d'Android

A la part del mig es pot veure el fragment de la brúixola orientativa amb una llista d'elements de control insertables a l'esquerra. A la dreta els elements que formen la pantalla a la dreta i les propietats de l'element seleccionat.

Problemes trobats

Algunes pantalles no es podien editar per un problema de l'editor que s'ha pogut solucionar afegint `android:inputType="textNoSuggestions"` als camps de text. Sembla que hi ha algun *bug* amb l'editor si no es desactiva explícitament la opció de propostes textuals. Per l'aplicació final es podria tornar a activar per tots els camps.

5.3. Fragments

Com s'ha comentat anteriorment en el disseny tècnic, s'ha implementat l'aplicació seguint la metodologia de *Fragments* que ofereix Android per possibilitar una futura adaptació de l'aplicació en funció de la pantalla del dispositiu. D'aquesta manera es defineixen fragments de la interfície de l'usuari els quals es podran combinar de forma diferent dins d'activitats diferents. Es podria definir, per exemple, una activitat per un telefon mòbil amb un únic fragment com a contingut i una altra activitat per *tablets* combinant fragments. Això condiciona la implementació en quant a que les classes principals han d'heretar de la classe *Fragment* d'Android i tots els diàlegs han d'heretar de la classe *FragmentDialog*.

Aquest fet entra en conflicte amb un altre punt important dels requisits de l'aplicació, el de suportar versions a partir de l'API 8, ja que la funcionalitat dels fragments s'ha introduït amb la API 11 d'Android. Afortunadament els desenvolupadors d'Android han considerat que els *Fragments* són prou importants com per crear una llibreria externa, que es pot incloure explícitament en projectes per versions anteriors a l'API 11, per proporcionar les classes necessàries. Aquesta llibreria es troba al repositori oficial de l'SDK Manager (*support-android-v4-r10*) d'on es pot baixar per incloure'l al projecte.

Malauradament no han tingut en compte o era impossible tindre en compta que, com s'explicarà més endavant, les activitats que contenen mapes de Google Maps han d'heretar d'una activitat especial anomenada *MapActivity*. El conseqüent problema és que no podran heretar de les dues classes al mateix temps. La solució d'aquest problema ha estat utilitzar una altra llibreria (*support-android-v4-r10-googlemaps*) que té en compte els dos casos. Aquesta llibreria no es troba en el repositori oficial però es basa en la versió oficial.

Un altre problema trobat és que per les activitats de preferències (explicades més endavant) no existeix cap *Fragment* corresponent. Si que existeix a l'API 11 però en aquest cas no serveix ja que per suportar versions inferiors s'esta obligat a utilitzar la llibreria de support (ni la oficial ni la modificada per funcionar amb Google Maps ho tracten). Les dues possibles opcions alternatives eren les següents:

- Tractar les preferències com una activitat normal. Tindria com a avantatge que es pot definir com un *Fragment* però tindria el desavantatge de no seguir amb les recomanacions d'Android d'utilitzar l'activitat *PreferenceActivity* per les preferències (el qual ajuda a uniformitzar la configuració entre totes les aplicacions).
- Tractar les preferències com una activitat individual i no com un fragment. Té el desavantatge que sempre haurà de ser una activitat individual que no es podrà combinar amb altres fragments però té l'avantatge de permetre la uniformització per part d'Android d'aquest tipus d'activitats.

Considerant que les preferències potser són un cas especial d'activitat i és acceptable no poder combinar aquesta activitat amb altres, s'ha optat per la segona opció: la part de les preferències és una activitat individual i no un fragment.

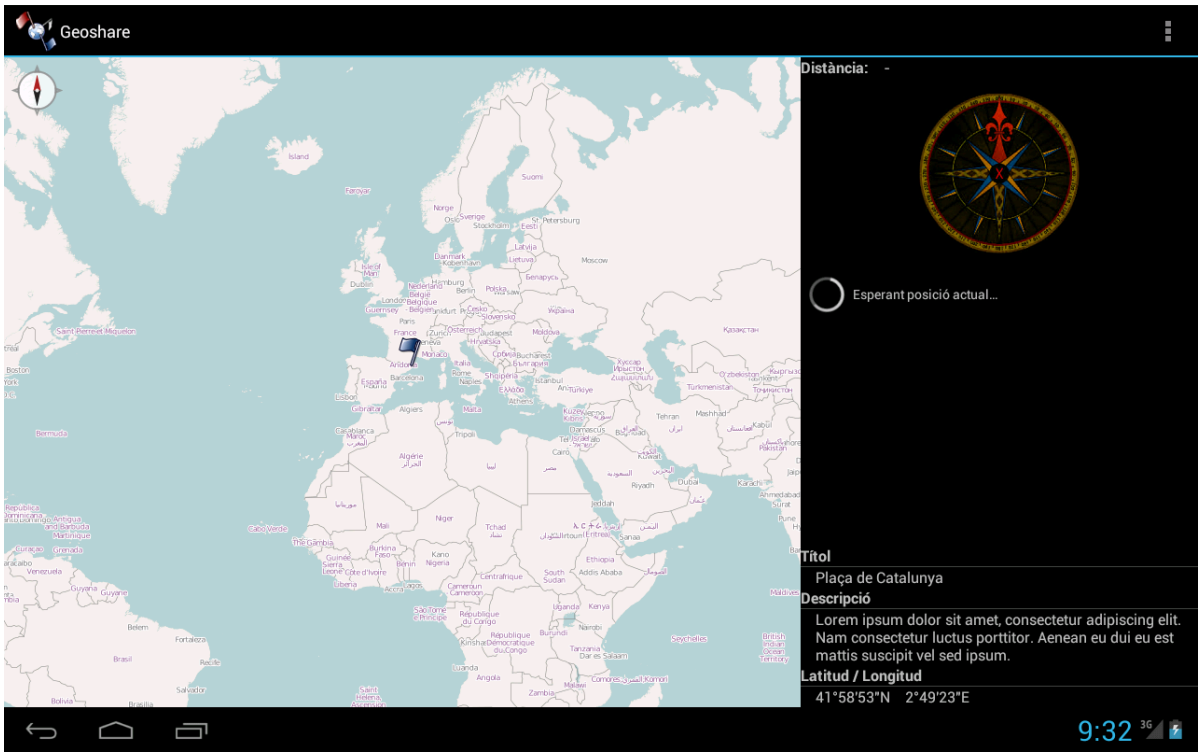
El fet d'utilitzar fragments també afecta la manera d'implementar algunes funcionalitats com el codi que s'executa en clicar un botó. Per facilitar la implementació del codi d'accions dels botons, Android permet definir a l'XML de la capa de presentació el nom del mètode que s'ha d'executar al clicar el botó i automàticament fa el *binding* amb el mètode que el desenvolupador defineix al codi.

```
// A l'XML
android:onClick="clic"

// Al codi Java
public void clic(View view)
{
      // ...
}
```

El problema és que aquests mètodes han d'estar definits a la classe que hereta d'*Activity* el qual significaria que per dues activitats diferents (una per un mòbil i una altra que combina fragments per un *tablet*) s'hauria de duplicar el codi. Inicialment s'ha fet servir aquesta opció de *binding* però finalment, per evitar la duplicació de codi en aquests casos, s'ha optat a implementar el mètode *onClick()* del botó directament per codi i no a través de la definició a l'XML.

Tot i que, pel que fa els fragments, el propòsit general d'aquest projecte només era preparar l'aplicació per poder-la adaptar en un futur, s'ha fet alguna prova emulant un dispositiu amb una pantalla més gran simulant una *tablet*, per exemple:



Il·lustració 25: Funcionament dels fragments

5.4. Pantalla principal

La part més important de l'aplicació són les marques de l'usuari posicionades en un mapa d'un dels dos sistemes de mapes suportats. Per tant, el més lògic és que a la pantalla principal es mostri directament el mapa amb les marques definides per l'usuari. Com ja s'ha explicat en el disseny tècnic, les dues llibreries utilitzades són desenvolupades per entitats independents i per integrar-les en el projecte actual, s'ha decidit implementar una interfície comuna pròpia amagant les implementacions en concret de cada un dels dos tipus.

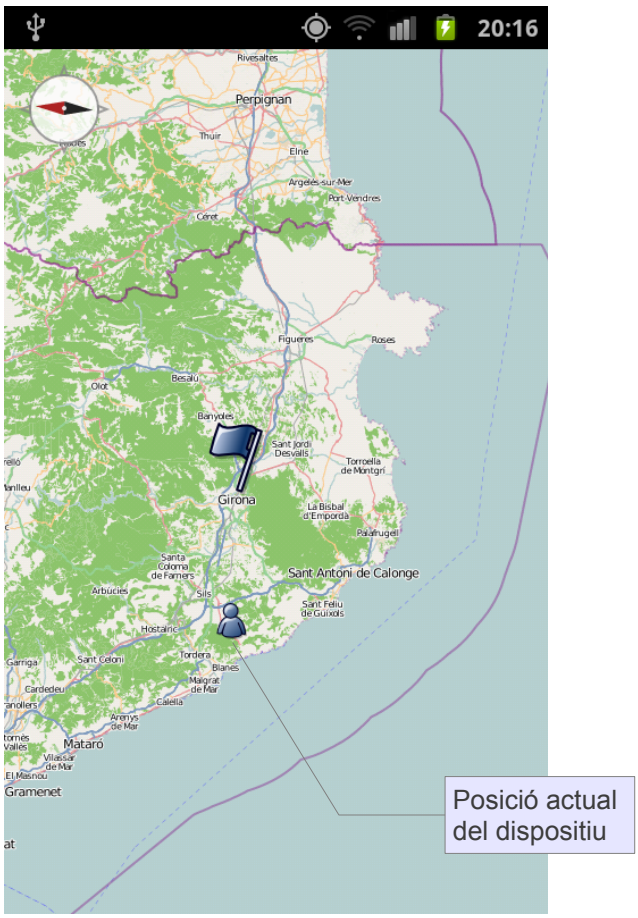
Totes les activitats d'Android per defecte tenen un títol i, en general, no s'hauria de modificar aquesta particularitat per no confondre l'usuari. En el cas de la pantalla principal, però, s'ha decidit prescindir-ne per guanyar espai pel mapa ja que es consider un avantatge important i no problemàtic, tenint en compte que es tracta d'una activitat prou diferenciada com per no seguir la norma general. Això només es pot aplicar a dispositius amb botó físic pel menú ja que els altres mostren un botó virtual per accedir-hi a la part superior. Aquest botó virtual desapareix juntament amb la barra del títol i per aquest motiu no es pot amagar la barra en aquests dispositius.

Independentment del tipus de mapa que l'usuari decideixi utilitzar, les opcions suportades que permeten navegar pel mapa del sistema desenvolupat són les següents:

- Desplaçament pel mapa arrossegant-lo.
- Opcions de *zoom* mitjançant:
  - *Pinch-zoom* per fer *zoom-in* o *zoom-out,*
  - Tap doble en un punt per fer *zoom-in* en aquest punt o
  - utilitzant els botons que apareixen a la part inferior fent tap simple en un punt buit (sense marques).

A les següents imatges es pot veure l'activitat principal mostrant el mapa mitjançant els dos sistemes de mapes suportats:

Il·lustració 26: Google Maps



Il·lustració 27: Open Street Maps

Les dues llibreries utilitzades (per Google Maps i per Open Street Maps) s'ocupen de les funcionalitats més bàsiques a l'hora de mostrar els mapes. Això inclou baixar les rajoles individuals de la zona que s'esta mostrant com també funcionalitats bàsiques de navegació i zooming del mapa.

5.4.1. Sobre les operacions geogràfiques

Avui en dia se sap que la terra no és plana sinó una esfera i, mirant més en detall, es pot afirmar que és més aviat forma d'el·lipsoide i tampoc no és un el·lipsoide perfecte. Sobretot el fet de que no s'esta treballant sobre un pla sinó sobre un pla corbat que forma l'esfera (o l'el·lipsoide) fa que tots els càlculs geogràfics es compliquen més del que inicialment es podria pensar. Com que en aquest projecte no es tracta d'estudiar com realitzar càlculs geogràfics, s'han consultat fonts externes per obtindre les fórmules necessàries adaptant els càlculs, tant pel que fa la traducció al llenguatge de programació com per les necessitats de l'aplicació. Les principals fonts consultades es poden trobar a l'apartat de fonts d'informació d'aquest document i tenen en compte els punts més importants comentats per aconseguir una precisió acceptable i uns resultats satisfactoris pel propòsit de l'aplicació desenvolupada.

Tots els càlculs realitzats s'han fet en el format indicat per les fórmules utilitzades però els resultats finals, tant per l'intercanvi de missatges entre mètodes o entre client – servidor com en el moment de fer-los persistents, s'ha fet servir el format latitudE6 i longitudE6 que consisteix en multiplicar la latitud/longitud per 1E6. El motiu d'això és que tant la llibreria de Google Maps com la de l'Open Street Maps utilitzen aquest format per tots els paràmetres d'entrada i sortida i d'aquesta manera es segueix el mateix criteri.

5.4.2. Mapes de Google Maps

La possibilitat de mostrar mapes de Google Maps no és un atribut estàndard de tots els dispositius que funcionen amb aquest sistema operatiu sinó que és un component opcional però la majoria de dispositius ho permeten. Per tenir accés a aquest sistema de mapes és necessari que el desenvolupador de l'aplicació adquireixi una clau (anomenada *api-key*) que s'ha d'inidicar en el moment d'instanciar el mapa. A partir d'aquesta clau, Google la pot identificar i fer les comprovacions necessàries per deixar accés, o no, al sistema de mapes. Aquesta clau es pot obtenir sense problemes de la pàgina oficial i inicialment també es pot obtenir una clau de proves que només serveix pel desenvolupament de l'aplicació (en mode *debug*). Per la utilització final, o sigui, en el moment de crear el paquet APK que es podrà instal·lar als dispositius, cal obtenir una clau diferent (per entorn productiu).

Igual que tots els elements que es poden afegir en una activitat Android (botons, camps de text, labels, etc.), l'element de mapa (*MapView*) hereta de la classe classe *View* (o *GroupView*). Això significa que teòricament es pot utilitzar igual que tots els altres elements en qualsevol activitat. Però a la pràctica això no és del tot cert per Google Maps ja que, a diferència dels altres *Views*, l'element de mapa té la particularitat que només es pot utilitzar en una activitat especial pels mapes (*MapActivity*). Aquest fet per si sol no comporta cap problema però sí en combinació amb *Fragments* com s'ha explicat a l'apartat de Fragments. No es troba gaire informació del perquè és necessari heretar de *MapActivity* i, segons algunes fonts, amb versions més noves ja no és obligatori, però esta comprovat que actualment els mapes deixen de funcionar si no es segueix aquesta norma.

Principals avantatges i inconvenients respecte Open Street Maps

- Mapes molt actualitzats pel que fa la xarxa de carreteres ja que a darrera hi ha una empresa amb interessos.

- Encara no es poden trobar molts dels camins fora de la xarxa oficial de carreteres asfaltades.
- Tot i que a la pàgina oficial s'esmenta que Google Maps fa *caching*, no s'explica més en detall com activar-ho ja que per defecte sembla que no ho fa. Moltes altres fonts comenten que això legalment no és possible amb Google Maps.

Principals problemes trobats

- El problema comentat anteriorment d'utilitzar un mapa amb Fragments.
- Només es pot crear una única instància per cada activitat, el qual no és un problema directa ja que, en principi, només és necessària una instància. El problema afecta en el cas de canviar de tipus de mapa (entre Google Maps i Open Street Maps). S'ha volgut fer que aquest canvi entre tipus de mapes es pugui fer en viu (sense haver de reinicialitzar l'aplicació) però això significa que es creen instàncies noves quan es canvia de tipus (que el *garbage-collector* de Java pot anar netejant). Per solucionar el problema, es manté una referència a la primera instància que es crea dels mapes de Google Maps (d'aquesta manera es reutilitza la instància si cal). L'inconvenient d'això és que hi ha un mapa que no s'esta veient però al qual es té una referència (el *garbage-collector* no ho podrà netejar). De totes maneres, aquesta situació només es dóna si durant la sessió actual s'ha canviat de tipus de mapa. La gran majoria de casos només hi haurà una instància que serà la del tipus de mapa especificat l'última vegada que s'ha canviat. Aquest problema no es troba amb Open Street Maps.

5.4.3. Mapes d'Open Street Maps

La llibreria que permet utilitzar aquests mapes en aplicacions Android ha intentat seguir l'estructura de la llibreria de Google Maps en quant a classes i mètodes. Però en alguns casos es poden trobar diferències significatives, o bé perquè no es segueix el mateix ritme de desenvolupament o bé perquè no sempre és possible mantenir aquest paral·lelisme.

En aquest cas no és necessària cap clau per poder accedir al sistema i, per tant, es pot utilitzar directament a l'aplicació. A diferència de Google Maps, en aquest cas es pot definir la mida de les rajoles individuals d'imatges que composen el mapa global. La mida d'aquestes rajoles afecta directament a la qualitat i al rendiment: amb més rajoles millora qualitat però disminueix el rendiment. Per l'aplicació actual s'ha decidit utilitzar una mida de 256, que sembla una bona relació qualitat/rendiment.

Principals avantatges i inconvenients respecte Google Maps

- No hi ha una empresa com Google a darrera i els mapes podrien no estar actualitzat fins a l'últim racó del món.
- Hi col·labora una gran comunitat ja que esta obert a tothom. Això és un avantatges sobretot per camins fora de la xarxa de carreteres asfaltades.
- S'ha comprovat que una àrea ja visitada es pot visitar encara que no hi hagi connexió Internet i, per tant, no hi ha cap problema per fer *caching* dels mapes.

Principals problemes trobats

S'ha trobat algun *bug* i algun mètode no accessible (per estar declarat com a privat) però

necessari per poder implementar algunes de les opcions necessàries per l'aplicació actual. Per solucionar-ho s'han fet algunes modificacions a la llibreria original i, per tant, la llibreria utilitzada és un recompilació de la versió oficial modificada lleugerament.

5.4.4. Menú principal

En aquest apartat s'expliquen les opcions que es mostren al accionar el botó "*Menú*" dels dispositius Android. Cal remarcar que segons la documentació d'Android aquest botó és obsolet i que els dispositius nous ja no el tenen, sinó que es basen en un funcionament diferent que consisteix en una barra d'opcions que apareix a la part superior de l'aplicació. De totes maneres, sempre poden accedir al menú d'alguna altra manera i en aquesta aplicació s'ha fet servir el sistema antic ja que es volen suportar versions antigues per les quals aquest sistema nou és desconegut.

A les següents imatges es pot veure com es mostraria el menú amb la versió 2 d'Android (per la qual és la manera normal de mostrar el menú) i amb una versió 4 (la qual mostra el menú però només per ser compatible cap enrere):

Il·lustració 28: Menú Android versió 2.x

Il·lustració 29: Menú Android 4.x

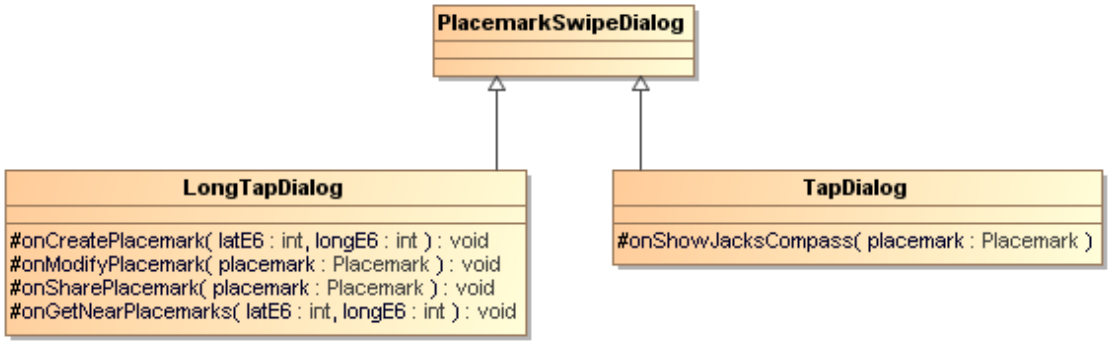
Les opcions que s'hi troben són les següents:

- Crear marca. La diferència entre aquesta opció i la mateixa del menú contextual que s'explica més endavant, és que la posició en que es vol crear la marca es defineix posteriorment (no s'omplen automàticament els camps de les coordenades).
- Reinicialitzar el mapa: reinicialitza el mapa tornant a obtenir les marques locals. Per tant, tots els llocs obtinguts del servidor s'esborren del mapa local i s'amaga una possible àrea marcada dins la qual s'han cercat llocs propers.
- Trobar llocs propers. La diferència entre aquesta opció i la mateixa del menú contextual és que la posició en que es volen cercar marques es defineix posteriorment (no s'omplen automàticament els camps de les coordenades).
- Marques pròpies compartides: obre una activitat nova amb opcions sobre les marques pròpies compartides a traves del servidor públic.
- Preferències: obre una activitat nova amb les opcions de configuració de l'aplicació.
- Ajuda: obre un diàleg d'ajuda amb els significat dels símbols i opcions possibles.

5.4.5. Diàlegs contextuals

Aquests diàlegs, tot i que contenen informació contextual del mapa, no són menús de context típics d'Android (com el menú contextual de la llista de llocs propis compartits). El motiu és que en un menú de context típic d'Android només es poden tenir opcions molt bàsiques però en el cas actual és necessari poder tenir funcionalitats més complexes com una imatge en el títol, oferir alguna manera de desplaçament per la llista de marques i canviar les opcions dinàmicament.

El diàleg informatiu d'una marca i el diàleg d'opcions contextuals tenen en comú que els dos accepten una llista de marques com a entrada i permeten a l'usuari iterar per aquesta llista per mostrar la informació corresponent per pantalla. S'ha decidit crear una classe comuna (`PlacemarkSwipeDialog`) entre aquests dos diàlegs per centralitzar aquestes funcionalitats:



Si en una mateix posició hi ha més d'una marca (o marques en posicions molt properes), el diàleg carrega cadascuna d'elles i l'usuari pot canviar entre elles fent *Swipe*. El motiu d'afegir aquesta opció és que molts dispositius mòbils tenen pantalles petites que fan difícil seleccionar una marca entre diverses marques molt properes. A la part inferior s'indica quantes marques hi ha en el punt clicat i quina s'esta mostrant:

Il·lustració 30: Primer de 5 seleccionat

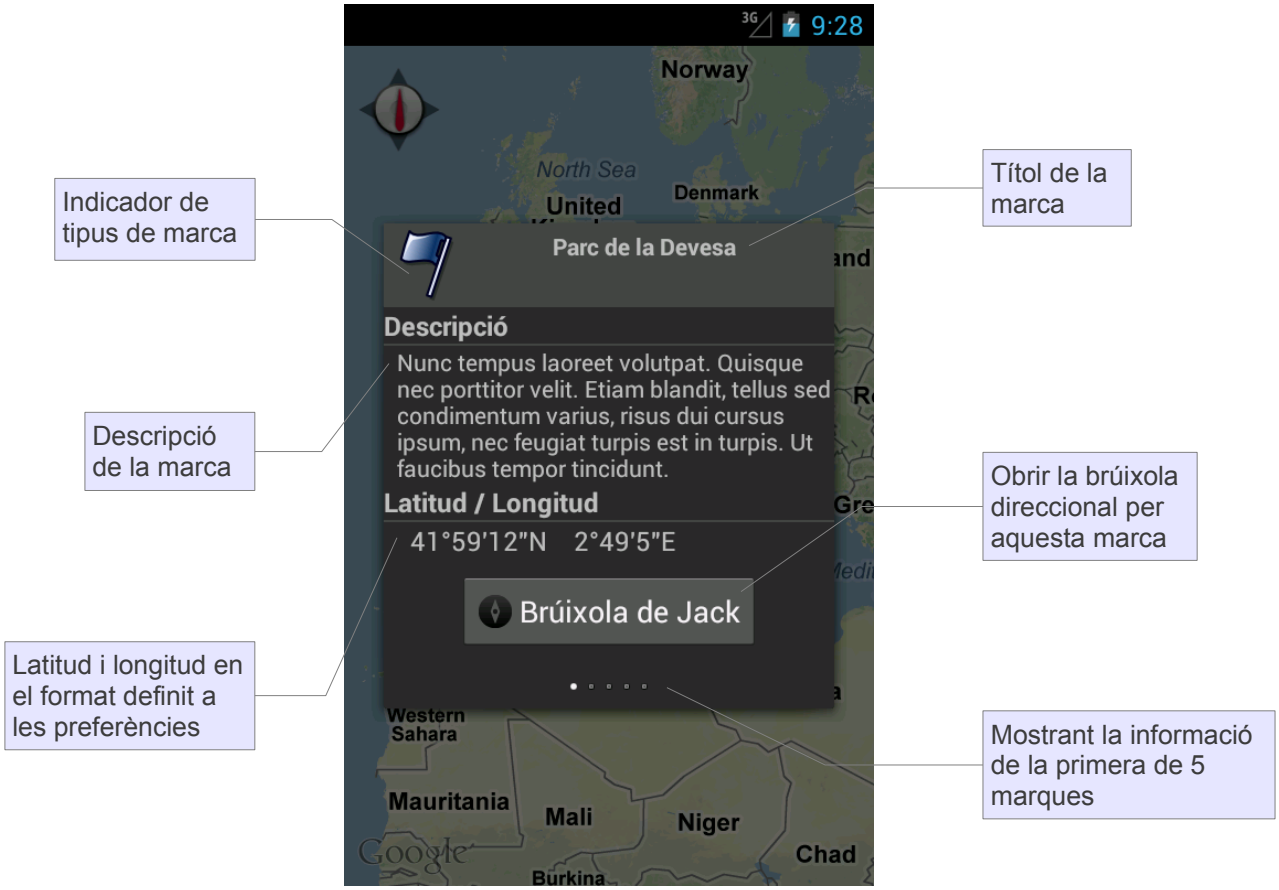
Aquests indicadors s'afegeixen dinàmicament ja que és en el moment de clicar les marques quan se sap quantes marques hi ha en el lloc seleccionat. Tot i així, la part gràfica es defineix en un fitxer XML igual que la resta de l'aplicació i, posteriorment, per codi Java *s'inflen* (com ho anomena Android) tants indicadors com són necessaris i s'afegeixen a un contenidor també definit a l'XML de la pantalla.

Per millorar l'experiència d'ús de l'aplicació s'ha afegit un efecte d'animació en el moment de navegar per marques (*Swipe*) del diàleg. D'aquesta manera l'usuari obté un *feedback* millor de l'acció ja que en cas contrari només canviaria el contingut del diàleg i, sobretot en el cas del diàleg d'opcions contextuals, podria passar desapercebut.

5.4.6. Diàleg informatiu de marca

Aquest diàleg mostra la informació relacionada a la marca: títol, descripció, coordenades i l'opció per accedir a la brúixola orientativa. Tot i que inicialment podria semblar que les coordenades no tenen cap utilitat per l'usuari (per això ja hi ha la posició visual en el mapa), si que es mostren com

a informació addicional (per si es volen apuntar en un paper, per exemple). Proporciona les opcions d'*anterior* i *següent* explicades al punt anterior, si en el lloc es troben varies marques.

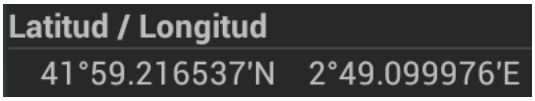


La bandera que es pot observar a dalt a l'esquerra del diàleg indica el tipus de la marca i canvia en funció de si es tracta d'una marca pròpia local, una marca pròpia compartida (obtinguda del servidor) o una marca compartida no pròpia (o no identificada com a pròpia com seria el cas en cercar marques properes incloent les pròpies):

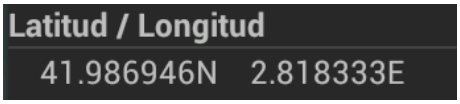
Igual que qualsevol altra coordenada que apareixen a l'aplicació, els valors de latitud i longitud d'aquest diàleg, es mostren en el format que l'usuari ha definit a les preferències de l'aplicació:

- DMS (Graus, minuts i segons):

- MinDec (Graus i minuts):



- DegDec (Graus Decimals):



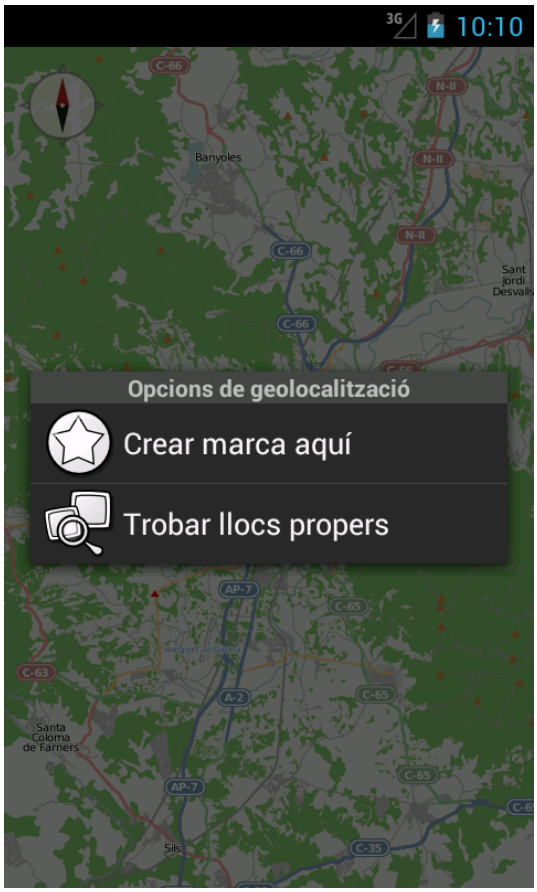
5.4.7. Diàleg d'opcions contextuals

El diàleg d'opcions contextuals proporciona opcions referents a la marca o les marques del lloc i a la geolocalització en general. S'estructura en les tres seccions següents:

- El títol del lloc.
- Opcions del lloc.
- Opcions de la geolocalització (no de la marca).

Les opcions de la geolocalització es mostren en qualsevol cas però el títol i les opcions de la marca només si en el punt clicat es troba una o varies marques locals. En aquest cas només afecta les locals perquè les opcions d'aquest menú només són vàlides per aquest tipus de marques. Això també significa que l'indicador de tipus de marca (la bandera a dalt a l'esquerra) sempre serà el mateix però s'ha decidit afegir-lo igualment per evitar confondre l'usuari (ja que el diàleg d'informació de marca també el té).

Cal notar que en aquest cas l'opció de desplaçament entre múltiples marques és referent a les opcions de les marques i no a les opcions de la geolocalització. Per aquest motiu, l'indicador de marques múltiples no apareix a la part inferior del diàleg sinó a la part inferior de les opcions de les marques.



Il·lustració 31: Opcions de modificació de geolocalització

Il·lustració 32: Opcions de modificació de marca i geolocalització

Les opcions de geolocalització són les següents:

- Crear marca nova: permet crear una marca nova igual que l'opció equivalent del menú principal, amb la diferència que els valors de les coordenades on es vol crear la marca s'omplen de forma automàtica en funció de la geolocalització clicada per obrir el diàleg.
- Trobar llocs propers: permet cercar llocs propers igual que l'opció equivalent del menú principal. Com en el punt anterior, els valors de les coordenades s'omplen automàticament.

Les opcions d'una marca són les següents:

- Compartir marca: inicia el procés de compartició d'una marca i tot el que comporta (identificació, comunicació amb el servidor, etc.). S'ha decidit afegir aquesta opció directament al menú ja que es considera prou important tenint en compte el tipus d'aplicació que s'esta desenvolupant.
- Modificar marca: aquesta opció obre el diàleg de manteniment de la marca. Aquest manteniment també conté un botó per eliminar la marca completament. S'ha decidit fer-ho d'aquesta manera i no afegint aquesta opció directament al menú per no sobrecarregar-lo.

5.5. Alta, baixa, modificació

L'activitat que gestiona les altes, les baixes i les modificacions és la mateixa. Aquesta activitat s'inicia des de l'activitat principal i un cop acaba, retorna a la mateixa activitat principal. El mecanisme que Android ofereix per fer-ho és inicialitzant un *Intent* del qual s'espera un resultat. A aquest *Intent* es poden passar paràmetres d'entrada que en aquest cas són la latitud i la longitud per inicialitzar els camps corresponents:

```
Intent intent = new Intent(getActivity(), EditPlacemarkActivity.class);
intent.putExtra("latitude", latE6);
intent.putExtra("longitude", longE6);
startActivityForResult(intent, CREATE_PLACEMARK);
```

Un cop l'usuari accepta o cancel·la l'activitat s'executa un mètode de *callback* a l'activitat original:

```
public void onActivityResult(int requestCode, int resultCode, Intent
data)
```

En aquest *callback* és on s'especifica que fer amb les dades que es retornen. En aquest cas és necessari actualitzar el mapa per reflectir-hi els possibles canvis que ha fet l'usuari (una marca nova, una marca canviada de posició, etc.).

Cal notar en aquest punt que qualsevol modificació que es fa sobre les marques només afectarà les marques localment. Per aconseguir actualitzar una marca pròpia del servidor, es pot tornar a compartir la mateixa marca el qual es detecta per tractar-ho com a modificació. Per eliminar una marca del servidor cal accedir a l'opció de “marques pròpies compartides” on es proporciona una funcionalitat per deixar de compartir una marca.

A les dues il·lustracions següents es pot veure aquesta pantalla de l'aplicació:

Il·lustració 33: Alta marca nova

Il·lustració 34: Modificació de marca

Com es pot observar, el botó d'eliminar una marca existent només es mostra quan s'està modificant una marca. Si s'està creant una marca nova, aquest botó s'amaga.

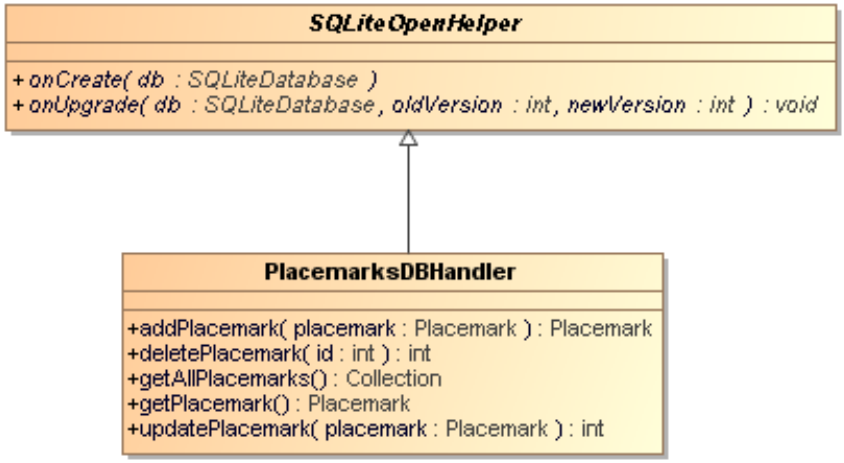
5.6. Base de dades

Per implementar la gestió de la base de dades SQLite del dispositiu, s'ha fet servir una classe anomenada *SQLiteOpenHelper* que proporciona l'Android. Aquesta classe és abstracta i obliga a implementar dos mètodes per aconseguir el comportament desitjat:

```
public abstract class SQLiteOpenHelper {
     public abstract void onCreate(SQLiteDatabase db);
     public abstract void onUpgrade(SQLiteDatabase db, int oldVersion,
                                                       int newVersion);
}
```

El mètode *onCreate* conté la part responsable de crear la base de dades i és cridat el primer cop que s'obre l'aplicació en el dispositiu en que s'ha instal·lat. El mètode *onUpgrade* és cridat cada vegada que és necessari actualitzar l'estructura de la base de dades (no el contingut). Per saber si cal actualitzar l'estructura, és necessari proporcionar un identificador de versió en el moment d'instanciar la classe *SQLiteOpenHelper* (o, millor dit, la classe que hereta d'aquesta). Si la versió ha canviat, s'actualitza la base de dades.

La classe resultant que hereta d'aquesta classe *SQLiteOpenHelper* i implementa els dos mètodes comentats és la següent:



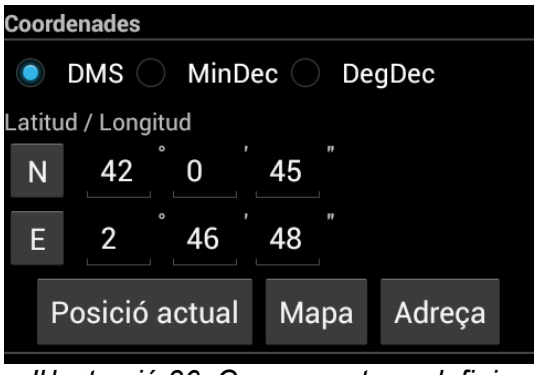
Il·lustració 35: Diagrama de classes de gestió d'SQLite

Els mètodes tenen les següents funcions:

- addPlacemark: dona d'alta una nova marca a la base de dades. El valor de retorn és la marca inserida a la base de dades.
- deletePlacemark: elimina la marca identificada per *id*. Aquest *id* és l'identificador local (no l'identificador universal compartit amb la base de dades del servidor). Retorna el nombre de registres eliminats.
- getAllPlacemarks: obtenir una llista de totes les marques guardades localment.
- getPlacemark: obtenir la marca identificada per *id* (també és l'identificador local).
- updatePlacemark: actualitzar les dades de la marca. Retorna el nombre de registres actualitzats.

5.7. Definir posicions

Tant l'activitat d'alta/modificació/baixa com el diàleg per trobar marques properes proporcionen a l'usuari 4 opcions per definir una posició. El component dissenyat per fer-ho és el següent:



Il·lustració 36: Component per definir coordenades

Aquest component i les corresponents funcionalitats que es veuran a continuació, és el mateix a nivell d'implementació i es reutilitza a les dues pantalles esmentades.

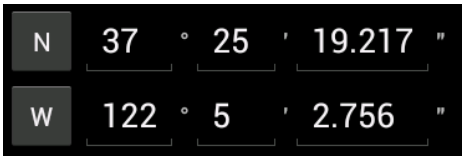
5.7.1. Definir posició manualment

Per definir una posició, a nivell conceptual es divideix l'esfera que representa la terra en 360°. Cada espai es divideix en dues meitats, *North (N)* i *South (S)* en el cas de latitud i *East (E)* i *West (W)* en el cas de la longitud. Els rangs són els següents:

- Latitud:
  - 90 N = 90
  - 90 S = -90
- Longitud:
  - 180 W = -180
  - 180 E = 180

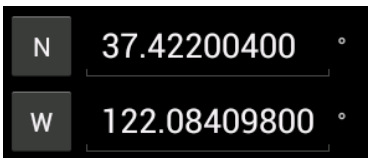
Per poder entrar els valors d'aquests rangs s'han definit tres conjunts de camps diferents, un conjunt per cada format de coordenada permès:

- DMS (decimals, minuts i segons):



- MinDec (graus i minuts decimals):

- DegDec (graus decimals):



L'aplicació permet especificar de quina meitat de l'esfera es tracta mitjançant els dos botons que es mostren a les imatges anteriors. Els valors d'aquests botons canvien a mesura que s'hi clica i mostren els quatre valors possibles en cada cas:

- Per la latitud són *North* (N o +) i *South* (S o -).
- Per la longitud són *West* (W o -) i *East* (E o +).

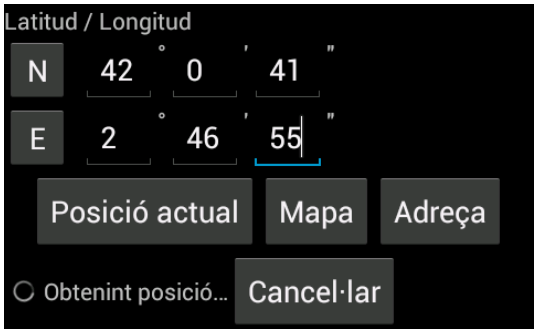
A l'hora d'entrar una coordenada es pot escollir el format mitjançant les tres opcions que apareixen a la part superior (DMS, MinDec o DegDec). Per defecte apareix seleccionada l'opció definida per l'usuari a les preferències de l'aplicació:

Els formats suportats i alguns exemples de valors que es permeten entrar són els següents:

- DMS (graus, minuts i segons), per exemple:
  - W87°43'41"
  - W79°58'36"
  - -87°43'41”
  - 12°34'56.78”
- MinDec (graus i minuts decimals), per exemple:
  - N40°26.7717'
  - W79°56.93172'
  - -79°56.93712'
- DegDec (graus decimals), per exemple:
  - 41.729600
  - N41.729600
  - -12.345678

5.7.2. Definir posició per GPS o Network Location Provider

El botó “*Posició actual*” obté la posició actual del dispositiu omplint així de forma automàtica els dos camps de latitud i longitud. Com que aquesta opció pot trigar una estona fins localitzar completament el dispositiu, es mostra un missatge indicatiu d'espera a sota indicant-ho fins que s'ha localitzat. Aquest missatge també proporciona un botó de cancel·lació que permet cancel·lar-ho en qualsevol moment. De totes maneres, no bloqueja la interfície i, per tant, l'usuari pot continuar editant els altres camps.



Il·lustració 37: Obtenint posició actual

Mentre s'està obtenint la informació dels satèl·lits per localitzar el dispositiu via GPS, es va actualitzant la posició mitjançant la informació que proporciona la xarxa de mòbils (*Network Location provider*). Aquesta posició no és tant precisa com la del GPS però disponible més ràpidament. Per aquest motiu, si esta disponible, s'utilitza complementariament però es considera que la precisió és important per aquest tipus d'aplicació i, per tant, un cop esta disponible la posició per GPS, s'utilitza aquesta i es descarta l'altre.

Un cop s'ha obtingut la posició definitiva s'alliberen els sensors deixant d'escoltar la posició dels dos proveïdors (GPS i/o *Network Location Provider*) perquè el dispositiu pugui procedir a apagar-los per estalviar energia si ho considera necessari.

5.7.3. Definir posició al mapa

La manera més fàcil i intuïtiva segurament és definir directament al mapa de l'activitat principal en quin punt es vol crear una marca (fent "*tap llarg"* per obrir el diàleg contextual) però per modificar una marca existent és útil aquesta opció. L'activitat que s'obre conté el mateix *IMapView* que l'activitat principal (la qual al final és una instància de *GoogleMapsView* o *OpenStreetMapsView*):

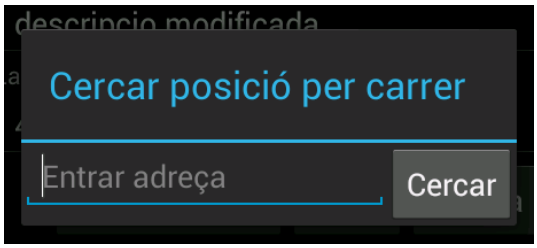
Il·lustració 38: Definir posició al mapa

A diferència de l'activitat principal, aquesta activitat no implementa els mètodes per obrir els menús contextuals del mapa ja que en aquest cas no és necessari. Si que implementa, en canvi, el "*tap simple*" però no per obrir un diàleg, sinó perquè l'usuari pugui indicar d'aquesta manera el lloc escollit. Totes les altres opcions de navegació (navegar, zoom, etc.) estan disponibles igual que a l'activitat principal.

5.7.4. Definir posició per adreça

Aquesta funcionalitat fa ús d'un servei que ofereix Google i ve inclòs en el mateix sistema de mapes: el *Geocoder*. Permet convertir una coordenada tipus (latitud, longitud) en una adreça de carrer i al revés. En aquest cas només s'ha fer servir la opció de convertir una adreça a coordenades (latitud, longitud) i s'executa de forma asíncrona ja que les dades es consulten a una font externa a través d'Internet. El resultat retornat pot ser una única localització (si les dades entrades per l'usuari són prou específiques) o una llista de resultats (si els criteris de cerca podrien ser vàlids per més d'un lloc). En aquest segon cas es mostra una llista amb els 5 primers resultats trobats perquè l'usuari pugui escollir l'adreça a la qual es referia. Si entre aquests 5 no es troba la que l'usuari esta buscant haurà d'entrar més informació per limitar la cerca. Un cop escollit el lloc,

el diàleg es tanca i el resultat és retornat a l'originador per omplir els camps latitud i longitud amb les coordenades obtingudes.



Il·lustració 39: Diàleg de cerca d'adreça

Il·lustració 40: Resultats de la cerca

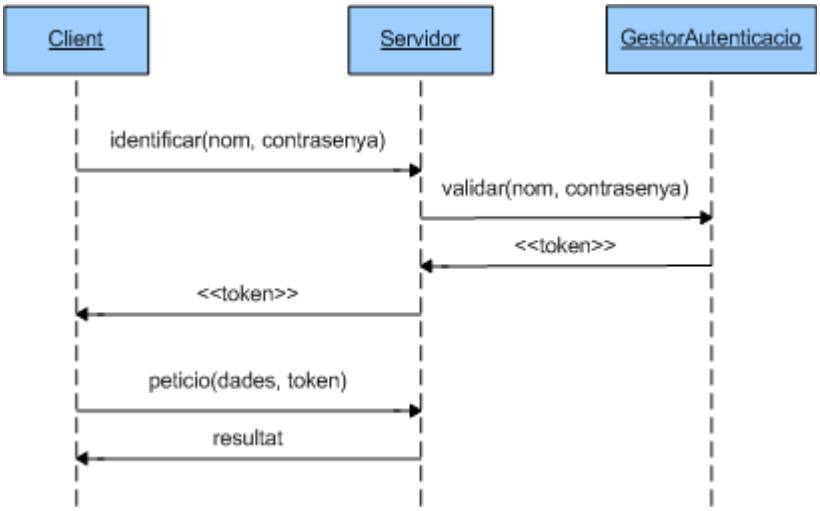
Tot i que aquest servei l'ofereix Google, no esta lligat al Google Maps. Això significa que funciona independentment del tipus de mapa que s'esta visualitzant en cada moment (Google Maps o Open Street Maps).

5.8. Identificació d'usuaris

El primer cop que un usuari vol executar una operació que requereix identificar-lo com a usuari del sistema, automàticament es presenta el diàleg d'identificació d'usuari en el qual s'ha d'entrar el nom d'usuari i la contrasenya. Un cop enviat i validat pel servidor, aquest mateix retorna un *token* aleatori (un UUID) el qual identifica l'usuari de forma única. Aquest UUID és una cadena de caràcters formada per 36 valors (32 alfanumèrics i 4 guions). Un exemple d'aquest identificador seria el següent:

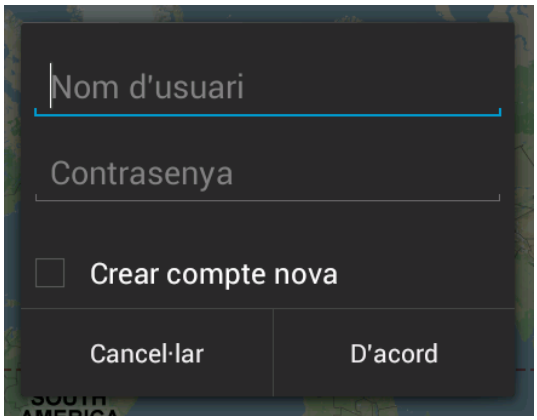
```
02d1b896-83c9-46f8-b998-9fa0db6b92c8
```

A partir d'aquest moment, per evitar que l'usuari necessiti identificar-se explícitament per cada petició al servidor, l'aplicació client enviarà aquest *token* juntament amb totes les peticions posteriors per identificar-lo. D'aquesta manera l'usuari només s'ha d'identificar una única vegada i per les peticions posteriors l'identificació és automàtica gracies al *token*. Per seguretat s'ha definit que aquests *tokens* caduquen al cap de 24 hores i, per tant, el sistema podrà identificar-lo de forma automàtica com a màxim un dia. Passat aquest dia l'usuari s'haurà de tornar a identificar.

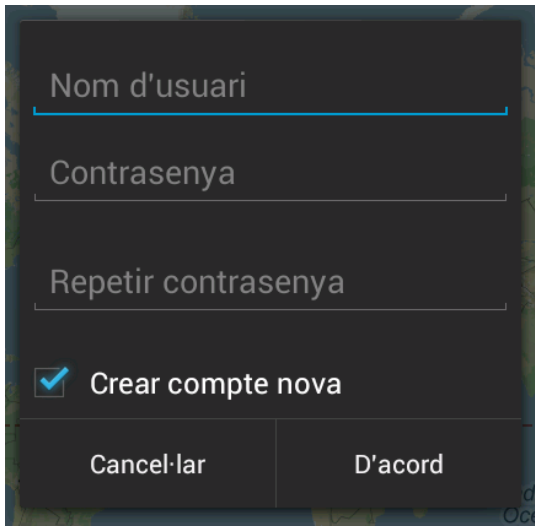


Il·lustració 41: Procés d'identificació d'usuari

Si l'usuari encara no esta registrat al sistema, primer de tot haurà de registrar-se. En una aplicació web es demanaria una adreça de correu de l'usuari la qual s'hauria de comprovar abans de permetre el registre però tot aquest procés és poc pràctic mitjançant un dispositiu mòbil. Per aquest motiu, s'ha decidit mantenir el registrament d'usuaris el més senzill possible, el qual comporta que les dues opcions d'identificació i registre d'usuari són molt semblants. Per l'usuari l'única diferència és que, a l'hora de registrar-se, ha d'indicar la contrasenya dues vegades. S'ha afegit un *Checkbox* al diàleg el qual, en activar o desactivar-se afegeix o treu dinàmicament el camp addicional del diàleg (repetir contrasenya).



Il·lustració 42: Identificació d'usuari



Il·lustració 43: Registre d'usuari

A nivell d'implementació les dues funcionalitats comparteixen molt de codi en la part client ja que es troben definides en un mateix diàleg. Això també té l'avantatge que aquest diàleg es pot cridar sempre que és necessària l'identificació de l'usuari, independentment si l'usuari ja s'ha registrat alguna vegada on no. El codi que hi ha a darrera, en canvi, si que varia bastant en funció d'aquesta opció ja que el Servlet que al final es crida pel *login* és diferent del que es crida per crear una compte nova. De totes maneres, el resultat retornat pel servidor en qualsevol dels dos casos els servidor és el *token* explicat anteriorment mitjançant el qual es pot identificar a l'usuari.

Tal com s'explica a l'apartat de seguretats, tant la identificació de l'usuari com el procés de registrar un usuari nou es fa a través de connexions segures HTTPS. Les altres peticions es fan a través d'HTTP normal.

Problemes trobats

Un problema trobat dels diàlegs en general, és que Android per defecte tanca els diàlegs d'alerta (del qual hereten tots els diàlegs d'aquesta aplicació) però en alguns casos això no és el comportament desitjat ja que no s'hauria de tancar fins que s'ha validat que les dades entrades són correctes. Per solucionar-ho ha calgut sobreescriure alguns mètodes dels diàlegs per canviar el comportament per defecte.

5.9. Llocs propis compartits

Aquesta activitat permet a l'usuari obtenir una llista de tots els seus llocs compartits del servidor comú. La llista resultant es mostra en un fragment del tipus *ListFragment*. Aquesta tipus de fragment és proporcionat per Android, igual que l'activitat equivalent *ListActivity*, per facilitar la creació d'activitats que consisteixen en una llista. A la llista d'aquesta funcionalitat es mostren cada un dels llocs compartits indicant el títol i la descripció de cada un. Tot i que no s'hi mostren les altres dades (com les coordenades), aquestes també s'obtenen amb la petició inicial per evitar haver de fer més peticions posteriorment per obtenir dades addicionals. Cal tenir en compte que funciona a través d'Internet i d'aquesta manera no es fan més peticions de les necessàries.

El mateix Android proporciona una manera per mostrar un contingut a la llista mentre aquesta és buida. En aquest cas s'ha fet ús d'aquesta propietat que es defineix mitjançant el fitxer XML que descriu l'activitat/fragment per indicar que no n'hi ha elements.

Il·lustració 44: Llista buida de llocs propis compartits

Il·lustració 45: Llista de llocs propis compartits

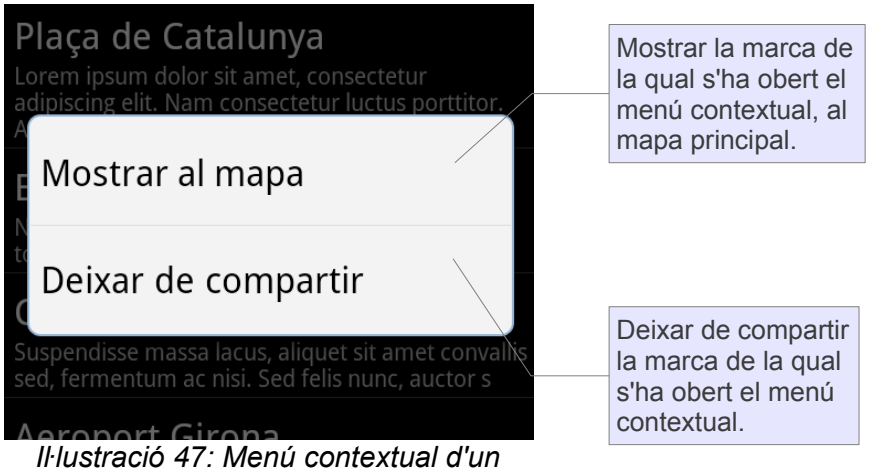
El menú principal d'aquest fragment que s'obre al accionar el botó "*Menú*" del dispositiu Android conté dues entrades diferents (cal tenir en compte els canvis que Android introdueix a la versió API 11 explicats anteriorment). Aquestes opcions són les següents:

- Mostrar tots al mapa: es tanca l'activitat actual per tornar a l'activitat principal mostrant al mapa tots els elements de la llista. Aquesta opció no fa cap petició addicional perquè ja es tenen totes les dades necessàries.
- Recarregar la llista: permet recarregar la llista, el qual torna a fer la petició al servidor per obtenir els llocs compartits propis.

Il·lustració 46: Menú de l'activitat

A part del menú principal de l'activitat, les llistes permeten definir un menú contextual per cada un dels elements individuals de la llista. En aquest cas, a diferència del menú contextual del mapa explicat anteriorment, és un menú contextual real. Les opcions que s'hi poden trobar són les següents:

- Mostrar al mapa: es tanca l'activitat actual per tornar a l'activitat principal mostrant al mapa l'element en concret. Aquesta opció no fa cap petició addicional perquè ja es tenen totes les dades necessàries.
- Deixar de compartir: permet indicar que l'usuari vol deixar de compartir una marca en concret. Després d'acceptar una confirmació de l'usuari, es crida el Servlet corresponent que indica al servidor que es vol eliminar la marca i, un cop ha acabat el procés, s'actualitza la llista del client per reflectir el canvi.



Il·lustració 47: Menú contextual d'un element de la llista

5.10. Cercar llocs propers

La opció de cercar llocs propers és un diàleg d'Android en el qual l'usuari pot escollir una coordenada com a centre i un radi per definir l'àrea dins la qual vol cercar marques compartides. Si s'accedeix a aquest diàleg a traves del menú contextual del mapa, els valors de latitud i longitud

s'ompliran de forma automàtica en funció del punt del mapa en qual ha clicat l'usuari. En cas contrari, accedint pel menú principal, no es poden omplir aquests camps i l'usuari ho haurà d'especificar mitjancant una de les maneres explica a Definir posicions. Un cop accepta les dades entrades, aquestes s'envien al servidor el qual retorna una llista amb les marques trobades.

Il·lustració 48: Diàleg de cerca de llocs propers compartits

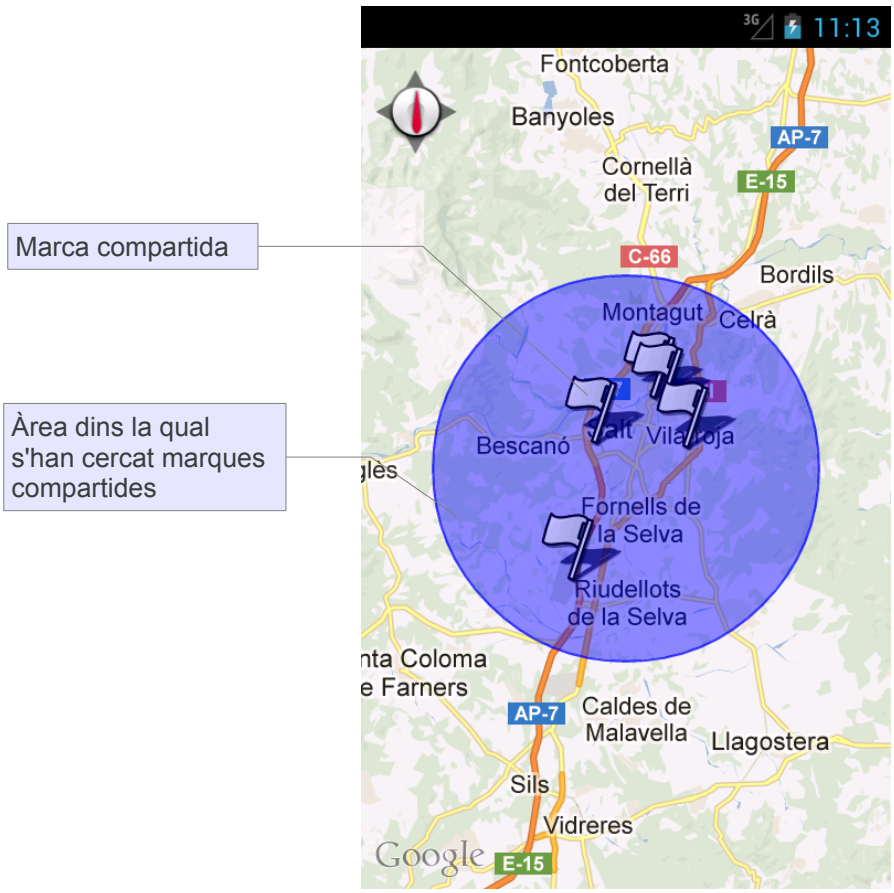
Il·lustració 49: Opcions de radi disponibles al Spinner

Després d'acceptar les dades d'entrada, l'aplicació del dispositiu client centra el mapa en la coordenada escollida i amplia la zona tant com sigui possible sense que l'àrea surti del tros visible a la pantalla. Per calcular aquest nivell de zoom és necessari obtenir els punts extrems de la rodona que defineix l'àrea de cerca. Les fórmules utilitzades són les següents traduïdes a Java (la primera per la latitud i la segona per la longitud):

$$\varphi_2 = \text{asin}(\sin(\varphi_1)*\cos(d/R) + \cos(\varphi_1)*\sin(d/R)*\cos(\theta))$$
$$\lambda_2 = \lambda_1 + \text{atan2}(\sin(\theta)*\sin(d/R)*\cos(\varphi_1), \cos(d/R)-\sin(\varphi_1)*\sin(\varphi_2))$$

on φ és la latitud, λ la longitud, d la distància (radi de l'àrea), θ l'orientació i R el radi de la terra. Tot i que el radi de la terra varia en funció del punt en que es troba la marca, en qualsevol operació de l'aplicació s'ha fet servir una constant de 6.371 Km ja que pel propòsit d'aquesta aplicació la precisió dels resultats obtinguts és més que suficient.

A la següent il·lustració es pot veure l'àrea marcada, centrada i ampliada mostrant tots els llocs trobats que compleixen estar dins d'aquesta àrea:



Problemes trobats

Els problemes més importants trobats en el desenvolupament d'aquesta funcionalitat són els següents:

- Inicialment s'ha volgut fer servir un element d'Android anomenat *NumberPicker* ja que sembla la opció més pràctica a l'hora d'escollir el radi de l'àrea. Però aquest element es va introduir a l'API 11 i, per tant, no esta disponible en versions anteriors. S'ha decidit fer servir un altre element anomenat *Spinner* que per aquest propòsit també és adequat i si que esta disponible per les versions anteriors. Té el desavantatge que cal fer un clic explícit més per mostrar les opcions a escollir però té l'avantatge que, de les possibles opcions a escollir, mostra un subconjunt més gran a la vegada (aprofitant tot l'espai de la pantalla disponible).
- Canviant l'orientació de la pantalla del dispositiu en el moment de tenir el *Spinner* desplegat, l'aplicació finalitzava llençant una excepció. Sembla que aquest problema és un *bug* del sistema que s'ha pogut solucionar amb un *workaround*.

5.11. Brúixola orientativa

La brúixola orientativa és una eina que permet a l'usuari orientar-se a l'hora de trobar una marca en concret en el món real. S'indica la direcció en que es troba la marca seleccionada i la distància que falten per arribar-hi. Aquesta distància s'indica en quilòmetre si en falten més de 2 i en metres si en falten menys. L'avantatge d'aquesta funcionalitat és que, a diferència dels mapes, no necessita cap connexió a Internet per funcionar. D'aquesta manera també és operativa en dispositius que no disposen de 3G o en llocs on no hi ha cobertura de telefònica. A part d'això, sobretot en jocs de camp obert com el *geocaching*, és més interessant trobar un punt sense tenir indicacions exactes de quin és el camí que s'ha d'agafar per arribar-hi.

La funcionalitat en si és una activitat Android que rep com a dades d'entrada la informació de la marca que s'esta buscant. Aquestes dades (el títol, la descripció i les coordenades) es mostren directament en camps de text normals aplicant la formatació corresponent en el cas de les coordenades (segons definit a les preferències). Les altres dades que es mostren en aquesta activitat (la brúixola en si i la distància), en canvi, s'han de calcular a partir de les coordenades rebudes com a paràmetres d'entrada i la posició actual del dispositiu.

Les dues il·lustracions següents són captures de pantalla d'un dispositiu real ja que, tal com s'explica a l'apartat de proves, els sensors utilitzats per la brúixola només funcionen en un dispositiu real:

Il·lustració 50: Esperant posició actual del GPS

Il·lustració 51: Distància i direcció calculades mitjançant la posició

Igual que qualsevol altra coordenada que apareixen a l'aplicació, els valors de latitud i longitud

d'aquesta activitat, es mostren en el format que l'usuari ha definit a les preferències de l'aplicació.

La brúixola apunta al nord mentre s'esta obtenint la posició actual del dispositiu que és necessària per calcular la rotació respecte la marca. En el moment en que s'obté, desapareix el missatge d'espera i s'actualitzen les dades (distància i direcció de la brúixola). A continuació s'expliquen els càlculs realitzats per obtenir aquestes dades:

Distància entre dos punts

Les dades d'entrada són les coordenades de la marca i la posició del dispositiu que s'obté mitjançant el *Network Location Provider* i el GPS un cop esta disponible. Els càlculs aplicats per obtenir la distància entre els dos punts són els següents (fórmula *haversine*):

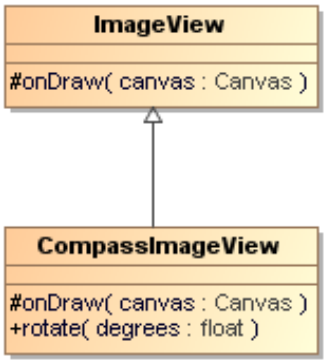
$$a = \sin^2(\Delta\varphi/2) + \cos(\varphi_1).\cos(\varphi_2).\sin^2(\Delta\lambda/2)$$
$$c = 2.\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R.c$$

on φ és la latitud, λ la longitud i R el radi de la terra (6.371 Km com s'ha comentat anteriorment).

Per actualitzar la distància restant, aquest càlcul s'executa cada vegada que la posició del dispositiu canvia (la velocitat d'actualització solen ser qüestions de segons). El resultat obtingut és la distància entre la marca escollida per l'usuari i la posició actual del dispositiu. Cal tenir en compte que aquesta distància és en línia recta i serveix com a valor orientatiu per a l'usuari.

Brúixola

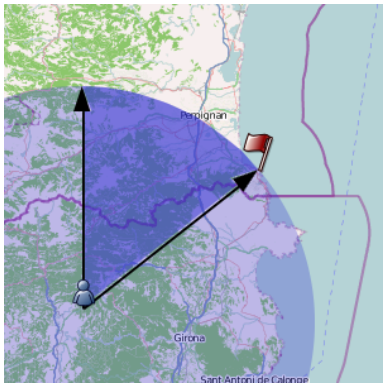
La brúixola en el fons és una imatge que es va girant en funció de la rotació del dispositiu. En aquest cas, l'animació resultant no necessita gaire processament de càlcul (només és rotar una imatge a pocs *frames* per segon). Segons la documentació d'Android la millor opció per casos d'aquest tipus és extendre un *View* i sobreescriure els mètodes necessaris. En aquest cas s'ha extés la classe *ImageView* d'Android sobreescrivint el mètode *onDraw* per tindre en compte un valor de rotació cada cop que es pinta la imatge. Aquesta rotació es pot modificar mitjançant el mètode *rotate* que s'exposa públicament a l'exterior i que, cada cop que canvia, repinta la imatge cridant el mètode *postInvalidate* de la classe *View* actualitzant així la rotació de la imatge en pantalla.



Il·lustració 52: Diagrama de classe

Per calcular en quants graus cal girar la imatge per correspondre's amb la rotació real del dispositiu respecte la marca es fa el següent:

1. Mitjançant les coordenades de la marca i la posició del dispositiu es calcula a quants graus es troba la marca respecte la posició actual.



Il·lustració 53: Rotació respecte la marca

La fórmula utilitzada per calcular-ho és la següent:

$$\theta = \operatorname{atan2}(\sin(\Delta\lambda).\cos(\varphi_2), \cos(\varphi_1).\sin(\varphi_2) - \sin(\varphi_1).\cos(\varphi_2).\cos(\Delta\lambda))$$

Els dos tipus de sensors utilitzats per obtenir la posició són:

- GPS_PROVIDER: que detecta la posició mitjançant GPS.
- NETWORK_PROVIDER: que detecta la posició mitjançant la xarxa telefònica mòbil.

S'ha decidit que la posició obtinguda pel primer sensor, sempre i quan esta disponible, té prioritat sobre la del segons ja que és més precisa.

2. S'obté la rotació del dispositiu mitjançant sensors del mateix dispositiu (acceleròmetre i magnetòmetre). En aquest punt no s'utilitza cap fórmula sinó els mètodes que ofereix l'Android per obtenir una matriu de rotació / inclinació. Els dos tipus de sensors utilitzats per calcular aquesta matriu són:

   - TYPE_ACCELEROMETER: que mesura l'acceleració.
   - TYPE_MAGNETIC_FIELD: que mesura la força del camp magnètic.

   Els valors que proporcionen aquests dos sensors són els paràmetres d'entrada a partir dels quals Android calcula la matriu de rotació / inclinació.

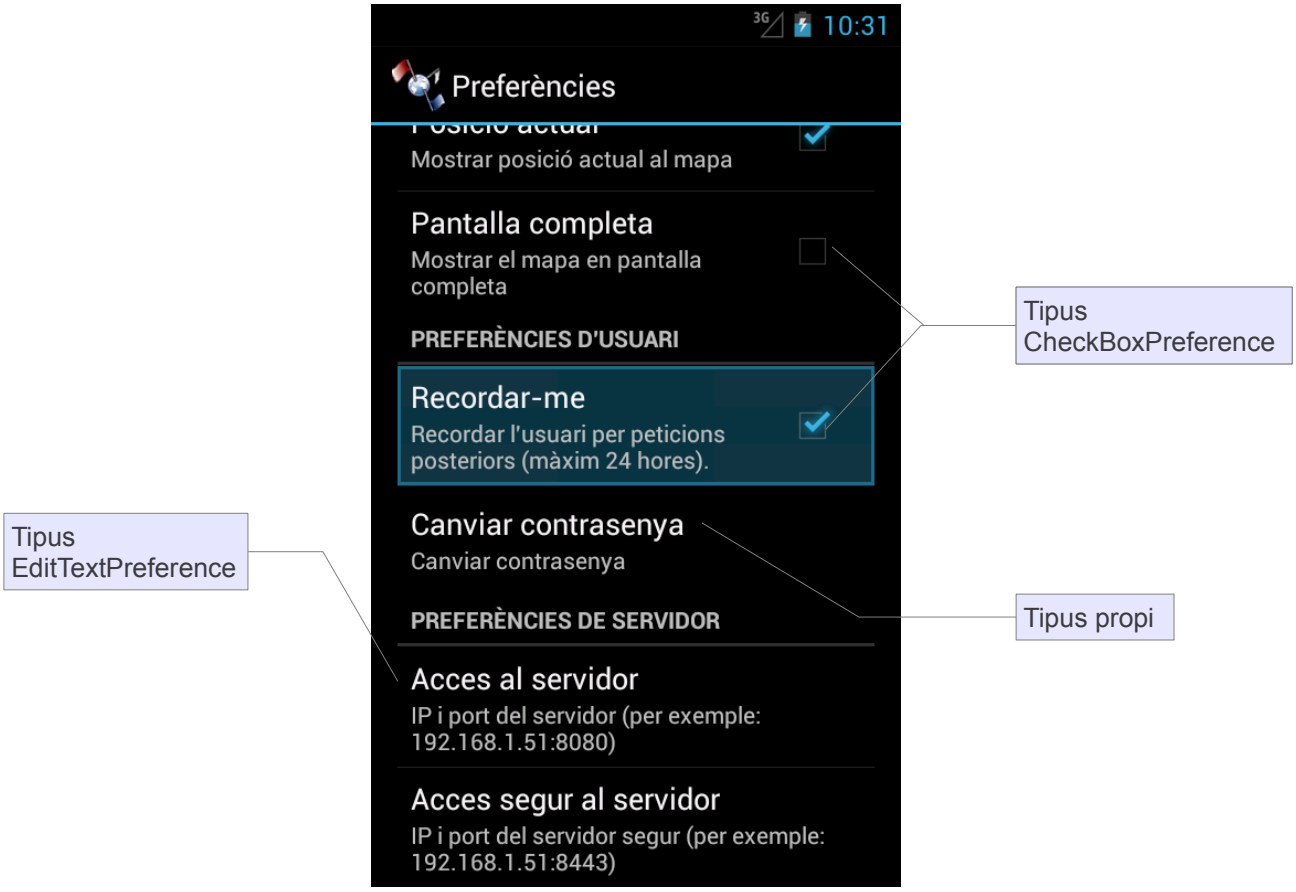
3. Es combinen els resultats dels dos punts anteriors per determinar en quant cal girar la imatge perquè apunti a la marca.

Per actualitzar la brúixola, tots aquests càlculs s'executen cada vegada que alguna de les dades d'entrada canvia. Si canvia la posició del dispositiu cal recalcular el primer punt (la velocitat d'actualització solen ser qüestions de segons). Si canvia la rotació del dispositiu cal recalcular el segon punt (si el dispositiu esta en moviment passa contínuament en qüestions de milisegons).

5.12. Preferències

Per uniformitzar el manteniment de les preferències que es poden trobar en les diferents aplicacions, Android proporciona una activitat (*PreferencesActivity*) i uns components explícits per

aquestes funcionalitats. En l'aplicació desenvolupada s'ha fet servir aquesta activitat seguint les recomanacions trobades i millorar així l'experiència d'ús del sistema. Pel mateix motiu també s'ha procurat que totes les preferències siguin canviables en viu sense la necessitat de reiniciar l'aplicació perquè tinguin efecte. Això és especialment important a l'hora de canviar el tipus de mapa, ja que significa que cal recrear l'estat de l'aplicació en el moment del canvi afegint al nou mapa totes les marques visibles anteriorment. El mateix passa amb la posició on estava treballant l'usuari (centre del mapa i nivell de zoom).



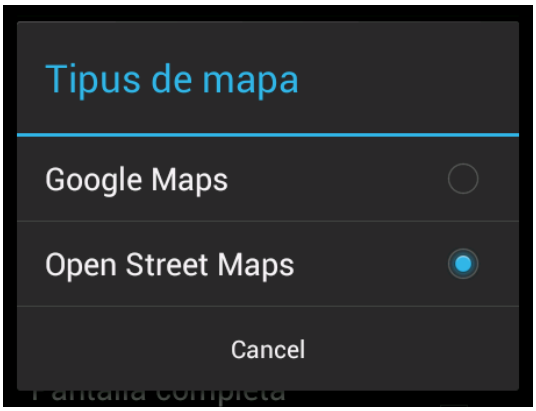
Il·lustració 54: Activitat de preferències

Com es pot veure en aquesta imatge, les preferències no tenen botó de cancel·lació ni d'acceptació. Tot i que això podria semblar estrany, és el comportament per defecte que Android implementa per aquestes activitats i, per tant, és millor no canviar aquest comportament per funcionar igual que totes les aplicacions per aquest sistema. Qualsevol canvi efectuat es guarda automàticament i per sortir de l'activitat es sol utilitzar el botó de tornar enrere del qual disposen tots els dispositius Android.

Les opcions de configuració que s'han implementat per aquesta aplicació són les següents:

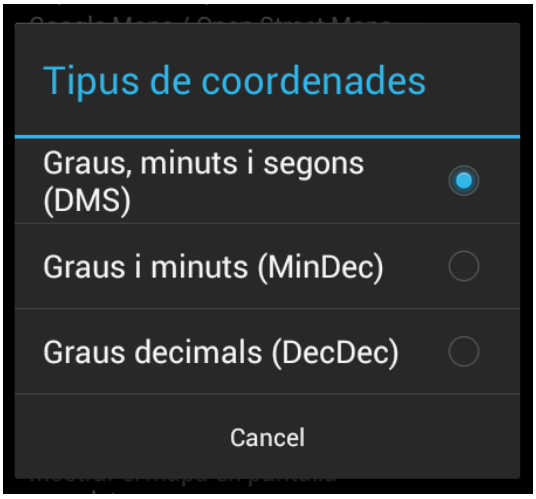
Preferències de mapa

- Tipus de mapa: és de tipus *ListPreference*, la qual obre un diàleg amb les opcions possibles. En aquest cas són els dos tipus Google Maps i Open Street Maps.



Il·lustració 55: Tipus de mapes suportats

- Tipus de coordenades: permet canviar la manera de com es mostren les coordenades en els punts corresponents de l'aplicació (el diàleg informatiu d'una marca, per exemple). És de tipus *ListPreference* i les opcions possibles són DMS (Graus, minuts i segons), MinDec (Graus i minuts) i DegDec (Graus Decimals).



Il·lustració 56: Tipus de coordenades suportats

Alguns exemples de com es mostraran els valors són els següents:

  - DMS (Graus, minuts i segons):
    - 87°43'41"W
    - 79°58'36"W
  - MinDec (Graus i minuts):
    - 40°26.7717'N
    - 79°56.93172'W
  - DegDec (Graus Decimals):
    - 41.729600N
    - 02.123456S

- Brúixola: permet activar/desactivar la brúixola que apareix a la part superior esquerra del mapa principal i del mapa per posicionar una marca. Aquesta preferència és de tipus *CheckBoxPreference* el qual permet dos estats (si o no). Aquesta brúixola apunta al nord i a les següents imatges es pot veure la seva aparença per Google Maps i per Open Street Maps respectivament:

Il·lustració 57: Google Maps

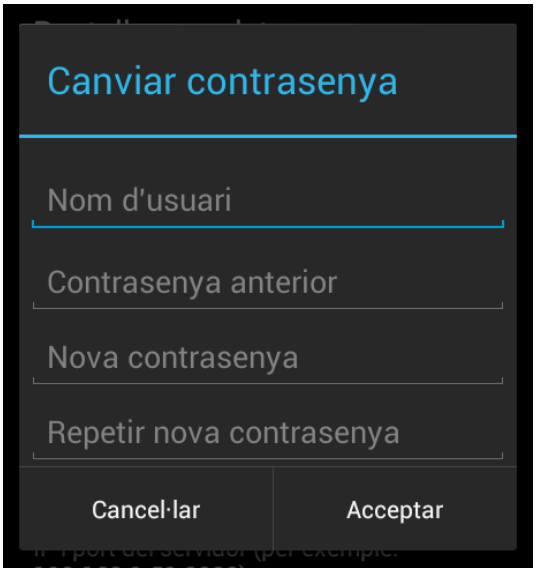
Il·lustració 58: OSM

- Posició actual: mostrar o no la posició actual del dispositiu al mapa. És de tipus *CheckBoxPreference* i en cas des desactivar-lo pot estalviar bateria del dispositiu ja que quan esta activat el sensor de GPS obtindrà la posició contínuament per actualitzar l'estat. La imatge que mostra la posició actual en el mapa és la següent:

- Pantalla completa: tot i que no es mostra el títol de l'activitat que conté el mapa, es continua mostrant la barra de tasques d'Android (que conté l'estat de connectivitat, l'estat de GPS, el rellotge, etc.). Amb aquesta opció es pot fer que el mapa ocupi tota la pantalla. És de tipus *CheckBoxPreference*.

Preferències d'usuari

- Recordar-me: permet desactivar la funcionalitat de l'*auth-token* la qual fa que l'usuari només s'hagi d'identificar una vegada en el servidor i a les peticions consecutives s'utilitza aquest *token* per identificar-lo. Desactivant aquesta opció, per totes les peticions al servidor que necessitin identificació es demanarà a l'usuari el nom i la contrasenya. És de tipus *CheckBoxPreference*.
- Canviar contrasenya: aquesta opció permet canviar la contrasenya de l'usuari en el servidor. És un cas especial ja que Android no proporciona cap component per defecte per fer-ho i ha calgut crear un de nou. Consisteix en un diàleg on cal entrar el nom de l'usuari, la contrasenya anterior i la nova contrasenya dues vegades. Un cop s'ha acceptat, es crida el Servlet corresponent per canviar la contrasenya en el servidor i es mostra un missatge indicatiu del resultat.

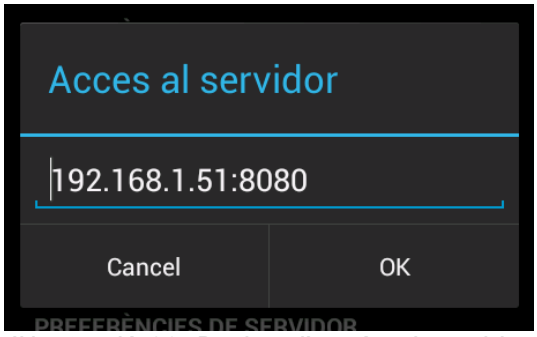


Il·lustració 59: Diàleg per canviar la contrasenya de l'usuari

Preferències de servidor

Aquestes preferències són únicament per les proves d'aquest projecte. Es podrien deixar en productiu però els valors que s'hi defineixen en principi ja estaran definits com una propietat en el fitxer de propietats ja que no haurien de ser canviats mai per l'usuari. El problema trobat és que els fitxers de propietats es troben dins del paquet APK i no es poden canviar un cop signat el paquet (tal com s'explica a l'apartat de signar APK final). De totes maneres s'han definit al fitxer de propietats i s'utilitzen en el cas de que el camp de preferència esta en blanc.

- Accés al servidor: permet especificar la IP i el port del servidor que conté els Servlets de l'aplicació. És de tipus *EditTextPreference* el qual permet entrar text.
- Accés segur al servidor: permet especificar la IP i el port per l'accés segur al servidor que conté els Servlets. També és de tipus *EditTextPreference*.



Il·lustració 60: Dades d'accés al servidor

Problemes trobats

Els principals problemes trobats durant el desenvolupament de les preferències són els següents:

- El problema explicat a l'apartat de Fragments.
- El problema de canviar els mapes en viu explicats a l'apartat de Mapes de Google Maps.

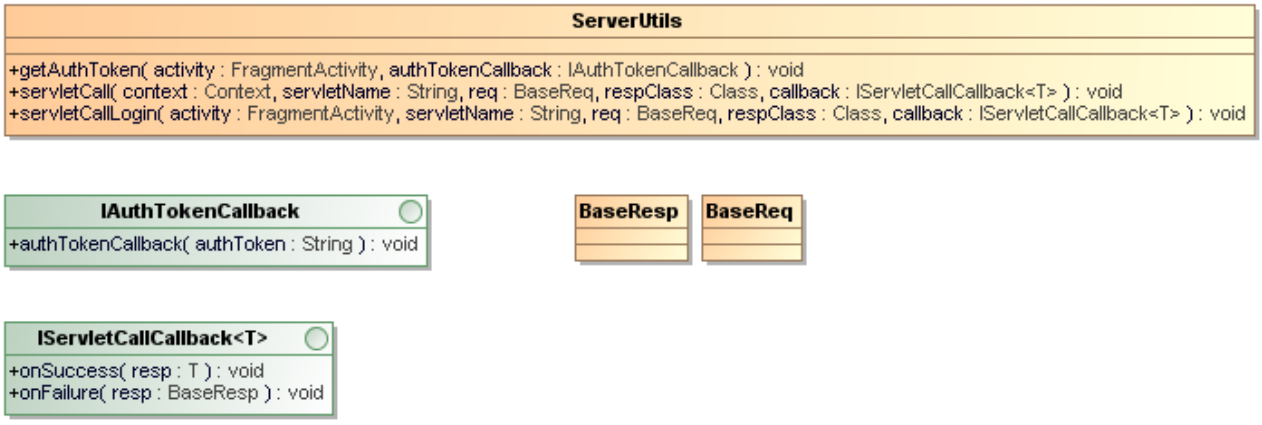
5.13. Comunicació amb el servidor

Per evitar bloquejar la interfície gràfica, totes les peticions que es fan al servidor es realitzen de forma asíncrona. Tot i que en l'aplicació que s'esta desenvolupant sempre es mostra un missatge d'espera per evitar problemes, la interfície no es bloqueja i amb això qualsevol actualització de la GUI s'acaba executant (com l'animació del missatge d'espera). Això significa que la consulta al servidor s'inicia però el client no espera la resposta per seguir sinó que segueix l'execució local de forma normal. Un cop el servidor retorna la resposta, es crida un mètode *callback* en el dispositiu client. El mateix Android ja proporciona classes i mètodes per fer-ho, l'opció que s'ha fet servir en aquest projecte és utilitzar la classe *AsyncTask* implementant els mètodes necessaris:

```
public abstract class AsyncTask<Params, Progress, Result> {
     protected abstract Result doInBackground(Params... params);
     protected void onPostExecute(Result result) {}
}
```

Aquesta classe té més mètodes però els que s'han fet servir per l'aplicació actual són aquests dos. Dins del primer mètode s'executa la petició al servidor i la mateixa classe s'ocupa d'executar-ho de forma asíncrona, o sigui, en un *thread* paral·lel a la resta del codi que s'esta executant en el mateix moment. Això mateix comporta el problema de que en aquest mètode no es poden tractar missatges destinats a l'usuari, ja que aquests únicament es poden mostrar en el *thread* de la interfície gràfica. Per tant, qualsevol missatge destinat a la GUI es perd si es processa dins del mètode *doInBackground* (si es fa, Android avisa llençant una excepció). Per aquest motiu existeix el mètode *onPostExecute*, el qual s'executa un cop ha acabat la petició i, a diferència del primer mètode, s'executa en el mateix *thread* que la GUI. En aquest mètode s'ha implementat el tractament del resultat de la petició el qual podria ser informar d'un error o mostrar el resultat obtingut.

Per centralitzar totes les crides als Servlets, en el client s'ha implementat una classe d'utilitats relacionades amb aquestes tasques:



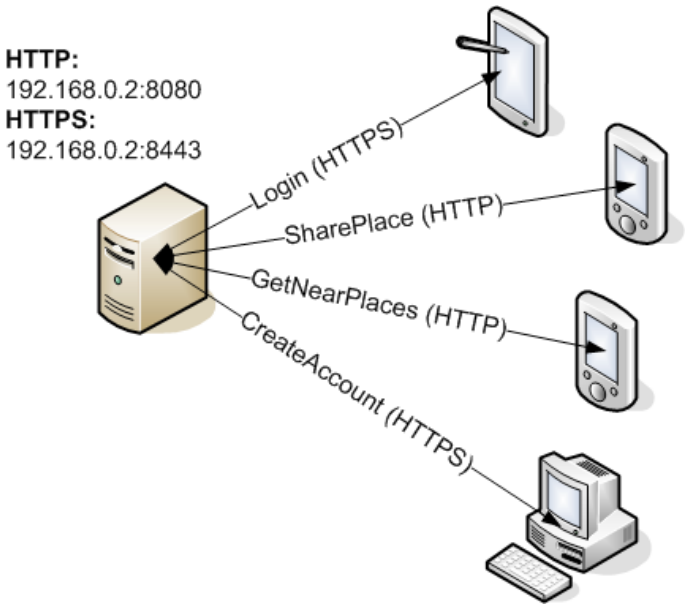
El mètode *getAuthToken* serveix per obtenir un *auth-token*. Si el dispositiu client ja en té un, no es fa cap petició i en cas contrari es demana un *token* al servidor cridant el Servlet d'identificació d'usuari. Pels motius que s'han comentat anteriorment, aquesta crida és asíncrona i, per tant, el resultat no es pot retornar amb el *return* del mètode sinó que cal fer-ho a traves d'un *callback*. En

aquest cas es fa a traves de la interfície *IAuthTokenCallback*, passada com a paràmetre d'entrada al mètode, cridant el mètode *authTokenCallback(String authToken)*.

Els mètodes *servletCall* i *servletCallLogin* fan una crida a un dels Servlets. El primer pels Servlets que no necessiten identificació per part de l'usuari (per HTTP) i el segon pels que si que en necessiten (per HTTPS) i, per tant, internament crida el mètode *getAuthToken*. Requereixen alguns paràmetres d'entrada, un dels quals és l'objecte de petició explicat a Servlets del disseny tècnic d'aquest mateix document. En aquest cas el *callback* es fa a traves de la interfície *IServletCallCallback<T>* passada com a paràmetre. Els dos mètodes a implementar d'aquesta interfície són *onSuccess(T resp)* i *onFailure(BaseResp resp)* que s'executen quan la petició ha acabat amb èxit o amb error respectivament. Si ha acabat amb èxit es retorna el resultat que és un dels objectes explicats a Servlets del disseny tècnic (segons quin Servlet s'ha cridat). Si hi ha hagut algun error es retorna un *BaseResp* que és el que tots els objectes de resposta implementen (també explicat a “Servlets” del disseny tècnic) i amb la descripció de l'error. Aquests dos mètodes s'ocupen de tot el que cal per la comunicació amb el servidor (asincronisme, serialitzar a text/JSON, deserialitzar, comprovar si cal fer *login* o ja existeix un *auth-token*, etc.).

5.14. Servlets exposats

Els Servlets que el servidor exposa per ser cridats des de l'exterior són els que s'han comentat en el disseny tècnic. Aquests Servlets són accessibles per tothom i realment es podrien cridar directament des d'un navegador tot i que, com ja s'ha comentat, és necessari proporcionar paràmetres d'entrada en format JSON. Es considera que és perfectament vàlid accedir-hi mitjançant una aplicació que no sigui la que s'ha desenvolupat per Android en aquest projecte, sempre i quan els paràmetres d'entrada són correctes (identificació / *auth-token* vàlid, dades dins dels rangs permesos, etc.). Com s'explica a l'apartat de Seguretats, la validesa d'aquestes dades d'entrada es comprova en qualsevol cas al costat del servidor i, per tant, no suposa cap risc de seguretat per l'aplicació.



Il·lustració 61: Comunicació amb els Servlets

Una qüestió important a tenir en compte en els Servlets és la concurrència. A nivell global no hauria d'haver-hi problemes ja que la filosofia de Servlets és completament compatible amb les característiques de la concurrència. Però per l'aplicació en concret podria significar un problema el fet que un usuari pot accedir als Servlets des de dispositius diferents al mateix temps. Per aquest

motiu, abans d'executar qualsevol operació, a part de validar les dades en quant a rangs i codificacions, sempre es comprova que les dades d'entrada encara siguin valides perquè podrien procedir d'un dispositiu amb dades desactualitzades (intentar actualitzar una marca inexistent per exemple).

Pel que fa el tractament d'excepcions que poden ocórrer en el servidor, s'ha decidit capturar-les i fer *log* perquè un administrador de sistemes pugui obtenir-lo per analitzar la causa de l'excepció. En cap cas s'envia l'excepció directament al client, primer de tot perquè per un usuari normal no té cap sentit la informació que mostra una excepció de Java però també perquè podria contenir informació que no es vol exposar a l'exterior (noms de taules o camps de la base de dades, llibreries utilitzades, etc.). Sempre que sigui possible s'intenta retornar un missatge informatiu del que ha passat i en cas de no ser possible es retorna un missatge genèric.

A continuació s'enumeren els Servlets exposats i les principals característiques d'implementació de cada un:

5.14.1. SharePlace

Aquest Servlet permet compartir una marca. L'operació consisteix en registrar les dades relacionades a la marca a la taula *shared_placemarks*. El que fa aquest Servlet és comprovar si la marca encara no existeix i, si és així, l'insereix. En canvi, si ja existeix actualitza les dades ja existents amb les noves dades rebudes (actualització).

Validacions que fa aquest Servlet sobre les dades d'entrada:

- L'*auth-token* és vàlid.
- Si és una modificació, comprovar que la marca sigui de l'usuari al qual pertany l'*auth-token*.
- L'identificador de la marca (UUID) esta especificat i té un format correcte.
- S'ha especificat el títol de la marca i no sobrepassa el nombre màxim de caràcters.
- El text de descripció no sobrepassa el nombre màxim de caràcters.
- Els valors de latitud i longitud estan dins dels rangs permesos.

5.14.2. UnsharePlace

Cal cridar aquest Servlet per deixar de compartir una marca i consisteix en esborrar les dades de la marca de la taula *shared_placemarks*. D'aquesta manera els altres Servlets que interroguen la base de dades (*GetMySharedPlaces* i *GetNearPlaces*) ja no trobaran la marca.

Validacions de les dades d'entrada:

- L'*auth-token* és vàlid.
- Comprovar que la marca sigui de l'usuari al qual pertany l'*auth-token*.
- L'identificador de la marca (UUID) esta especificat i té un format correcte.
- L'UUID existeix a la base de dades.

5.14.3. GetMySharedPlaces

Aquest Servlet interroga la base de dades obtenint totes les marques de la taula *shared_placemarks* que siguin de l'usuari que ha fet la consulta. En aquesta consulta no s'aplica cap límit per filtrar la quantitat de dades a retornar, ja que s'ha de permetre a l'usuari obtenir totes les seves marques i la informació relacionada (localització, títol i descripció). Tot i que és poc probable que un usuari tingui centenars o milers de marques definides (com a mínim dins del propòsit d'aquest projecte) però si és el cas s'obtindran i es retornaran totes.

Validacions de les dades d'entrada:

- L'*auth-token* és vàlid.

Optimització fora de l'abast

Si en un projecte real s'arriba a l'extrem de que un usuari té centenars o milers de marques s'hauria de veure si és necessari afegir alguna millora. No només per la quantitat de dades que s'envien entre client i servidor, sinó també pel rendiment de les llistes del dispositiu client en cas d'haver-hi grans quantitats de dades.

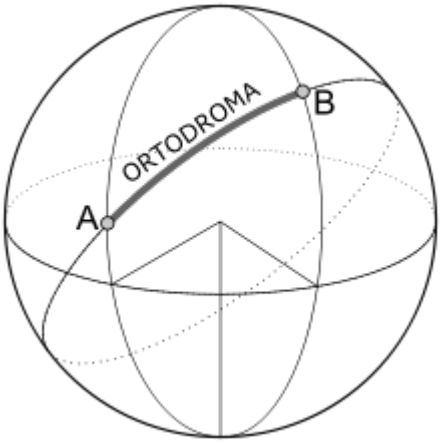
5.14.4. GetNearPlaces

A partir d'una coordenada i un radi com a dades d'entrada, aquest Servlet retorna totes les marques que es troben dins de l'àrea indicada. L'operació consisteix en interrogar la base de dades del servidor obtenint totes les marques de la taula *shared_placemarks* que compleixen la condició. A part d'aquests dos paràmetres d'entrada, aquest Servlet n'accepta un tercer mitjançant el qual és possible indicar si es volen excloure o no les marques del mateix l'usuari. Aquest paràmetre és opcional i, tot i que podria semblar lògic excloure les marques pròpies de l'usuari en qualsevol cas, s'ha afegit perquè té l'avantatge que aquest Servlet no requereix identificació per part de l'usuari, si aquest decideix cercar entre totes les marques del servidor (les pròpies incloses). D'aquesta manera, usuaris no registrats al sistema poden utilitzar-lo per consultar marques compartides. En cas contrari, si vol excloure les pròpies, és obligatori la identificació per saber de quin usuari es tracta.

Per saber si un punt es troba dins de l'àrea a cercar és necessari calcular la distància entre el centre de l'àrea a cercar i cada un dels punts continguts a la base de dades. Si aquesta distància és més petita que el radi de l'àrea a cercar, el punt entra a la selecció. La línia que descriu la distància mínima entre dos punts sobre una esfera, que en aquest cas representa la terra, és la línia ortodròmica. La fórmula per calcular aquesta distància és la següent:

$$dist = \arccos(\sin(lat_1) \cdot \sin(lat_2) + \cos(lat_1) \cdot \cos(lat_2) \cdot \cos(lon_1 - lon_2)) \cdot R$$

on *lat1* és la latitud del centre, *long1* la longitud del centre, *lat2* la latitud de la marca, *long2* la longitud de la marca i *R* el radi de la terra (6.371 Km).



Il·lustració 62: Línia ortodròmica

La base de dades del servidor utilitzada en aquest projecte (MySQL) proporciona totes les funcions necessàries per aquests càlculs i, per tant, la condició es pot afegir directament a la consulta SQL. La consulta resultant que s'executa per seleccionar els registres corresponents de la bases de dades és la següent:

```
SELECT * FROM shared_places WHERE R * acos(sin(radians(latReq)) *
sin(radians(latDB)) + cos(radians(latReq)) * cos(radians(latDB)) *
cos(radians(longDB) - radians(longReq))) <= radius
```

on *latReq* és la latitud del request, *longReq* la longitud del request, *latDB* la latitud de la marca de la base de dades, *longDB* la longitud de la marca de la base de dades, *radius* el radi de l'àrea a cercar i *R* el radi de la terra (6.371 Km).

A la consulta anterior s'afegeix la condició opcional d'excloure les marques pròpies de l'usuari, el qual consisteix en una condició simple per comprovar el propietari de la marca de la base de dades. També cal tenir en compte que la informació de totes les marques trobades es retornarà al client per mostrar-les a l'usuari. Per evitar retornar un nombre excessiu de dades del servidor al client, s'ha decidit limitar els resultats obtinguts. Aquest límit s'ha fixat a 100 registres i es comprova a la mateixa consulta, ja que d'aquesta manera des del principi es limiten les dades obtingudes i el cost que comporta. El MySQL permet fer-ho amb la paraula reservada següent que s'afegeix a la consulta anterior:

```
LIMIT 0,100
```

Validacions de les dades d'entrada del Servlet:

- Si es volen excloure les marques pròpies es comprova la validesa de l'*auth-token*.
- El valor del radi esta dins dels rangs permesos (100m – 50Km).
- Els valors de latitud i longitud estan dins dels rangs possibles.

Optimitzacions fora de l'abast

Aquesta operació, de moment i pel propòsit d'aquest projecte, funciona sense problemes però podria arribar a ser molt costosa ja que la base de dades no pot executar la consulta utilitzant índexs. Això és degut a que la condició de filtre (la clàusula WHERE) fa operacions de càlcul complexes. Si en un futur hi haguessin molts usuaris accedint al sistema, podria ser útil optimitzar

aquesta consulta mitjançant aquestes dues solucions possibles que es podrien afegir sense gaires complicacions:

- Afegir un camp a la base de dades amb la latitud i la longitud directament en radians, d'aquesta manera la consulta no hauria de calcular-ho cada cop per cada registre.
- Fer una primera consulta d'una àrea quadrada ja que així la base de dades si que podria fer servir els índexs i posteriorment fer una segona consulta de l'àrea rodona però només sobre els registres seleccionats a la primera consulta.

5.14.5. CreateAccount

Crear una compte nova significa inserir les dades que proporciona l'usuari (nom d'usuari i contrasenya) a la taula *users*. Per estalviar un pas a l'usuari i una petició al servidor, en cas de que totes les dades són correctes, aquest mateix Servlet crida el codi corresponent per generar un *auth-token* i el retorna amb la mateixa resposta. D'aquesta manera un usuari que s'acaba de registrar, automàticament estarà identificat i podrà accedir al servidor sense necessitat de tornar-se a identificar fins que caduca l'*auth-token*. El codi que utilitza aquest Servlet per fer el *login* de l'usuari és compartit amb el Servlet Login.

Cal notar que la contrasenya que s'insereix a la base de dades durant l'execució d'aquest Servlet és encriptada abans de guardar-la tal com s'explica a l'apartat de Seguretats.

Validacions de les dades d'entrada:

- Longitud mínima i màxima del nom d'usuari.
- Longitud mínima i màxima de la contrasenya.
- Que el nom d'usuari no existeixi a la base de dades.

5.14.6. Login

El Servlet de Login comprova a la taula *users* si el nom d'usuari existeix i si la contrasenya que l'usuari ha introduït és correcte. Si és així, crida el codi corresponent per generar un *auth-token*, el qual permet identificar l'usuari fins que el *token* caduca, i el retorna al dispositiu client.

Validacions de les dades d'entrada:

- S'ha especificat el nom d'usuari.
- S'ha especificat una contrasenya.
- La contrasenya és correcte.

5.14.7. ChangePassword

Aquest Servlet selecciona les dades actuals de l'usuari de la taula *users* i valida si la contrasenya anterior és correcte comparant-la amb la contrasenya enviada per l'usuari. Si és correcte canvia la contrasenya anterior encriptada per la nova contrasenya encriptada actualitzant la taula.

Validacions de les dades d'entrada:

- Longitud mínima i màxima del nom d'usuari.

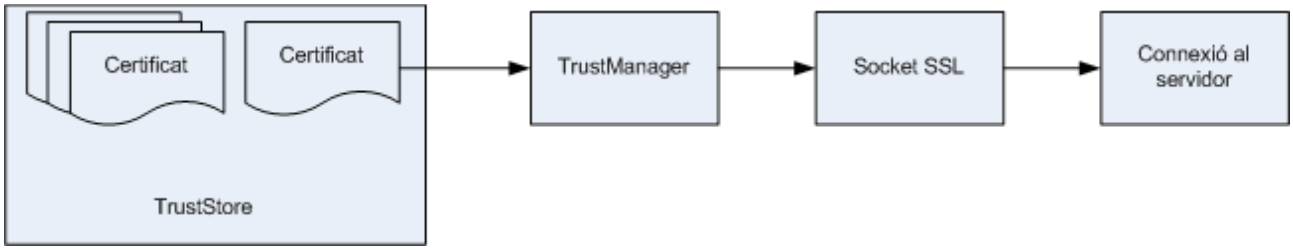
- Longitud mínima i màxima de la contrasenya.
- La contrasenya anterior és correcte.

5.15. Seguretats

En aquest apartat es descriuen els aspectes més importants pel que fa la seguretat que s'han tingut en compte durant el desenvolupament de l'aplicació.

5.15.1. Connexió segura

Un dels paràmetres d'entrada dels tres Servlets CreateAccout, Login i ChangePassword és la contrasenya de l'usuari. Per assegurar que ningú pugui capturar la contrasenya (mitjançant atacs tipus *man-in-the-middle*), aquestes dades confidencials es transmeten fent servir una connexió segura HTTPS. Per fer-ho és necessari un certificat, que en aquest cas és un certificat autofirmat i, per tant, no forma part de cap autoritat de confiança (CA). Això significa que ni l'usuari ni Android poden confiar-hi però l'aplicació desenvolupada, en canvi, si que pot confiar-hi ja que pel propòsit d'aquesta és vàlid. Per tant, el mateix Android no podrà classificar el certificat com a vàlid i ha estat necessari implementar la validació a nivell de l'aplicació creant un *TrustManager* independent que si que hi confia.



A partir d'un dels certificats del *TrustStore*, el *TrustManager* crea un *Socket SSL* per demanar l'establiment de la connexió segura amb el servidor. Un cop acabat aquest procés comença el *handshake* entre client i servidor per establir definitivament la connexió segura o no, si alguna de les entitats ho considera inviable per qualsevol motiu (no confia en el certificat de l'altre, per exemple).

L'inconvenient de les connexions segures és que són més costoses tant pel que fa els càlculs relacionats amb el certificat com a l'hora d'establir la connexió entre el servidor i el client. Per aquest motiu per totes les altres operacions entre client i servidor es fan servir connexions normals (HTTP).

5.15.2. Contrasenyes encriptades

Les contrasenyes només les haurien de conèixer els mateixos propietaris. Això significa que no es poden guardar directament a la base de dades perquè altres persones com l'administrador podrien interrogar-la directament. Per evitar-ho s'ha optat a aplicar una funció de *hash* a les contrasenyes i guardar el resultat a la base de dades. Per fer-ho s'ha fet servir la llibreria *jasypt-1.9.0.jar*, tant per encriptar les contrasenyes com per comprovar-les en el moment de l'identificació. Tot i que els algorismes necessaris ja els porta el mateix Java (la llibreria no implementa cap algorisme), mitjançant l'ús d'aquesta llibreria es guanya facilitat i seguretat a l'hora de l'implementació.

De les diferents classes que proporciona el *jasypt* per encriptar dades, s'ha escollit la classe *BasicPasswordEncryptor* que encripta les contrasenyes amb les següents característiques:

- Algoritme: MD5.
- Mida del salt que s'afegeix per aleatoritzar: 8 bytes.
- Iterations: 1000.

Aplicant aquest algorisme a una contrasenya d'exemple “1234” s'obté la següent contrasenya encriptada:

```
3oo4EjWRxoLVLxVrO18DtBDaQhLq2RVb
```

5.15.3. Auth-token

A part de l'avantatge de facilitar la identificació per part de l'usuari, l'*auth-token* també millora lleugerament la seguretat. Un mateix usuari pot tenir més d'un *token* vàlid al mateix moment (si té varis dispositius, per exemple) però un *auth-token* només pot pertànyer a un usuari en concret i, per tant, identifica a l'usuari de forma única. D'aquesta manera en cap moment és necessari transmetre l'identificador real de l'usuari entre client i servidor el qual ajuda a amagar les dades davant de tercers. Per saber quin usuari esta fent la consulta sempre es fa mitjançant l'*auth-token,* a partir del qual el servidor pot saber a quin usuari pertany consultant la taula *auth_tokens*.

També per millorar la seguretat s'ha especificat que qualsevol *token* caduca al cap de 24 hores. Un cop passat aquest temps, l'usuari s'ha d'identificar novament amb el seu nom d'usuari i la contrasenya. Per poder comprovar posteriorment si un *auth-token* ha caducat o no, en el moment de crear-lo també es guarden la data i l'hora de creació d'aquest:

```
+---------------+----------+------+-----+---------+-------+
| Field         | Type     | Null | Key | Default | Extra |
+---------------+----------+------+-----+---------+-------+
| auth_token    | char(36) | NO   | PRI | NULL    |       |
| user_id       | int(11)  | NO   |     | NULL    |       |
| creation_date | char(8)  | NO   |     | NULL    |       |
| creation_time | char(6)  | NO   |     | NULL    |       |
+---------------+----------+------+-----+---------+-------+
```

5.15.4. Prepared Statements

Per interrogar la base de dades no s'ha fet servir cap ORM (Object-relational mapping) com Hibernate ja que l'aplicació desenvolupada no fa consultes o modificacions complexes a la base de dades i s'ha decidit que es pot fer perfectament amb consultes natives d'SQL. El que si que s'ha fet és utilitzar els *Prepared Statements* de Java que, a part de millorar l'eficiència de les consultes, ajuden a millorar la seguretat de l'aplicació. En una consulta mitjançant *Prepared Statements* no es concatenen els valors de les variables amb la resta de la consulta sinó que s'insereixen en els punts indicats explícitament (indicats per un interrogant). D'aquesta manera s'eviten descuits a l'hora de programar, els quals podrien permetre a usuaris malintencionats aconseguir modificar la consulta de tal manera que el resultat sigui diferent al que hauria de ser, contenint dades que no es volen proporcionar a l'usuari durant el funcionament normal.

Un exemple de *Prepared Statement* seria la següent consulta que obté una marca apartir de l'identificador únic universal:

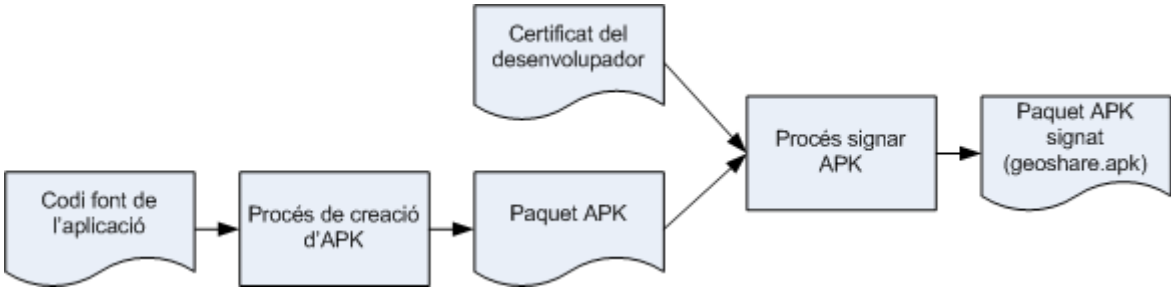
```
SELECT * FROM shared_places WHERE placemark_uuid=?
```

5.15.5. Comprovar dades d'entrada al servidor

El servidor no pot suposar res de les dades que li arriben des del client perquè, com ja s'ha comentat, és possible accedir als Servlets mitjançant qualsevol aplicació externa. Per aquest motiu, per qualsevol dels paràmetres d'entrada dels Servlets, abans de començar l'operació, es comprova que el valor sigui vàlid. En part, aquest també és el motiu pel qual no s'intercanvia l'identificador d'usuari entre client i servidor ja que el servidor mai no pot estar segur de que l'identificador proporcionat és el correcte o es tracta d'un usuari malintencionat que ha aconseguit injectar un identificador d'una altra persona. L'usuari sempre s'identifica mitjançant un *auth-token* i tot i que aquest també podria ser un de falsificat, la probabilitat d'encertar un *auth-token* que realment existeixi en aquell moment és molt i molt baixa (és un UUID) i més tenint en compte que els *auth-tokens* expiren cada 24 hores.

5.15.6. Signar APK final

Aquesta seguretat la imposa el mateix Android i significa que qualsevol aplicació que es vulgui instal·lar en un dispositiu amb aquest sistema operatiu ha d'estar signada amb un certificat del desenvolupador. L'entorn de desenvolupament ho fa de forma automàtica utilitzant una signatura especial per proves i evitar, així, que s'hagi de seguir el procés de signar cada vegada que es vol provar l'aplicació. Per l'aplicació final, en canvi, s'ha de fer explícitament amb un certificat real, tot i que pot ser un certificat autofirmat perfectament. El fet de signar el paquet garanteix que ningú no ha modificat el seu contingut posteriorment a la seva creació.



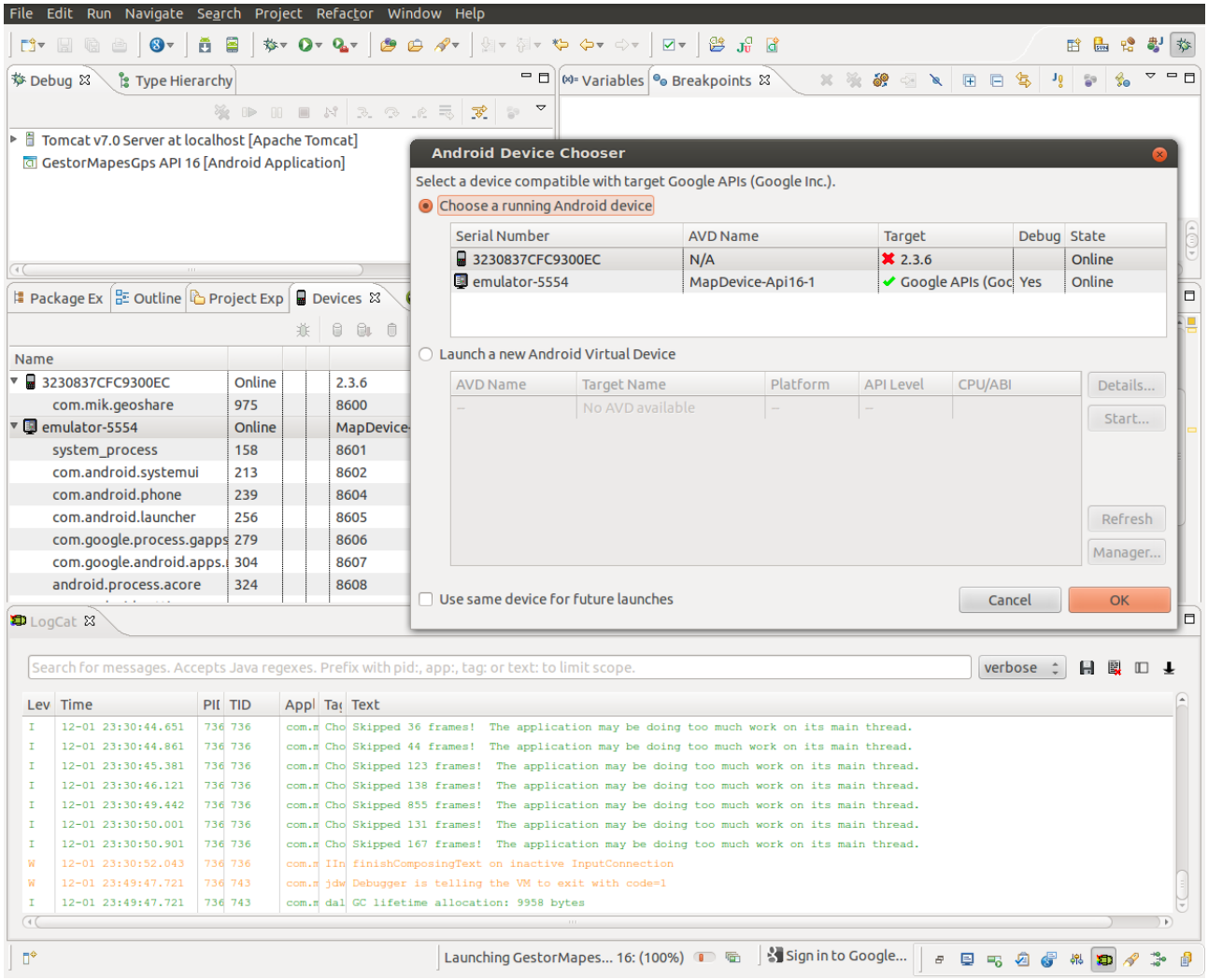
Fent la prova d'instal·lar un paquet modificat després de signar-lo (en el fons és un paquet zip que es pot modificar amb qualsevol gestor de paquets zip), resulta en el següent error:

```
adb -s 3230837CFC9300EC install geoshare.apk
3746 KB/s (897290 bytes in 0.233s)
     pkg: /data/local/tmp/geoshare.apk
Failure [INSTALL_PARSE_FAILED_NO_CERTIFICATES]
```

6. Fases de proves

Les proves s'han realitzat en tres etapes diferents, les primeres dues a traves de l'entorn de desenvolupament i la tercera en un dispositiu real desconnectat de l'ordinador de desenvolupament.

A la següent imatge es pot veure l'entorn de desenvolupament amb el servidor de proves (Tomcat) en marxa i l'aplicació d'Android a punt d'executar-se en un dels dos dispositius disponibles. El primer és un dispositiu real i el segon un dispositiu emulat per l'Android SDK. A més a més es pot veure el *Logcat,* en el qual es mostren dades informatives i errors del dispositiu Android (a la imatge es veuen els missatges de l'última execució) independentment del dispositiu el qual pot ser un de real o un d'emulat:



Il·lustració 63: Entorn de desenvolupament i de proves

Com ja s'ha comentat, el servidor de proves funciona amb un Tomcat versió 7 però el sistema pot funcionar sobre qualsevol altre servidor que suporti tot el necessari (que pugui contenir Servlets). Per realitzar les proves, tant durant el desenvolupament com les proves finals, la configuració del servidor Tomcat utilitzat és la que hi ha establerta per defecte. L'únic tema que s'ha tingut d'ajustar, tant pel servidor contingut a l'entorn de desenvolupament eclipse com pel servidor independent de les proves finals, ha estat la configuració per la connexió segura. Per fer-ho s'ha modificat el fitxer

server.xml del servidor afegint el següent:

```
<Connector
      protocol="HTTP/1.1"
      port="8443" maxThreads="200"
      scheme="https" secure="true" SSLEnabled="true"
      keystoreFile="path/al/keystore" keystorePass="contrasenya"
      clientAuth="false" sslProtocol="TLS"/>
```

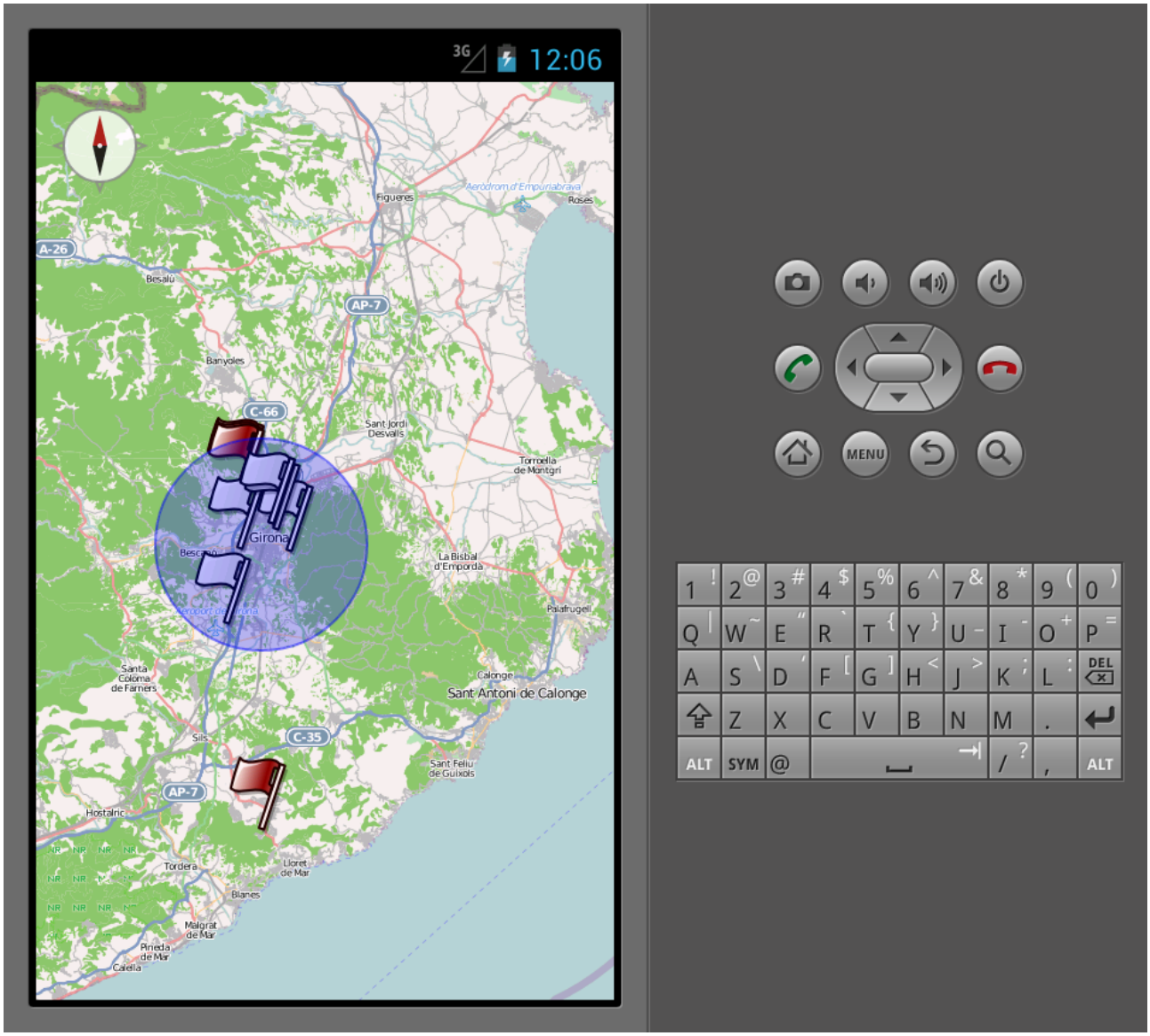
on "*path/al/keystore*" és la ruta on es troba el *keystore* que conté el certificat i "*contrasenya*" és la contrasenya del *keystore*. Cal tenir en compte que aquest certificat ha de ser el mateix que el de l'aplicació Android ja que en cas contrari no es validarà com a certificat vàlid i no funcionarà la connexió segura entre client i servidor. Entre d'altres, també es defineix que el port per les connexions segures és el 8443 i que el protocol a utilitzar és TLS.

6.1.1. Depurant amb l'emulador Android SDK

Les primeres proves durant el desenvolupament s'han realitzat mitjançant l'emulador de l'Android SDK. Té l'avantatge que no és necessari cap dispositiu real i que el dispositiu emulat es troba en el mateix ordinador i així es pot controlar tot des d'allà mateix. Té el desavantatge que és relativament lent, tant iniciant-lo per primera vegada (el qual és solucionable utilitzant l'opció de guardar *Snapshot*) com a l'hora de carregar l'aplicació i a l'hora de provar-la.

Al ser aquesta la primera aplicació desenvolupada per Android, personalment donava per suposat que l'emulador permet provar totes (o gairebé totes) les funcionalitats que ofereix un dispositiu mòbil però estava equivocat. Afortunadament és pot provar el GPS (amb dades simulades) ja que és molt important per aquesta aplicació però lamentablement els sensors (com el magnetòmetre o l'acceleròmetre) no es poden provar si no és amb un dispositiu real. Hi ha alguns projectes de tercers que intenten implementar alternatives per solventar aquestes mancances i durant el desenvolupament d'aquest projecte s'ha intentat fer servir una eina anomenada *SensorSimulator* (http://code.google.com/p/openintents/wiki/SensorSimulator) però sense èxit. Tots els passos explicats a la documentació d'aquesta eina funcionen però en el moment de provar l'aplicació, simplement no arribaven els events simulats (segurament per diferències de versions d'Android). Finalment s'ha optat a utilitzar un dispositiu real per provar aquestes parts que eren la brúixola orientativa i funcions com el *pinch-zoom* (és impossible clicar en dos punts diferents al mateix temps dins de l'emulador).

A la il·lustració següent es pot veure l'emulador de l'Android SDK executant l'aplicació:



Il·lustració 64: Emulador de l'Android SDK

La creació d'un dispositiu virtual emulat és relativament simple i el mateix plugin integrat a l'entorn de desenvolupament eclipse proporciona eines per configurar-lo. També a l'hora d'executar-lo i carregar-hi l'aplicació que es vol provar, el mateix entorn de desenvolupament s'encarrega de la majoria de les tasques i ofereix moltes opcions al desenvolupador per ajustar-ho si cal.

6.1.2. Depurant amb un dispositiu real

És sorprenentment fàcil executar l'aplicació en un dispositiu real i el funcionament és el mateix que amb l'emulador de l'Android SDK. Fins i tot és possible depurar el codi i visualitzar possibles excepcions des de l'IDE de la mateixa manera que amb l'emulador. Tot el que cal fer per possibilitar-ho és activar una opció en el dispositiu (per permetre la depuració) la qual és fàcilment accessible a traves de les configuracions del sistema Android. En aquest cas en concret també ha calgut modificar les seguretats del sistema operatiu des del qual es desenvolupa (el sistema base és Linux) pel port USB al qual es connecta el dispositiu real. De la resta, igual que en el cas anterior, s'ocupa el mateix entorn de desenvolupament, únicament s'ha d'escollir el dispositiu en

comptes de l'emulador.

Té l'avantatge que l'aplicació s'executa en un dispositiu real, el qual permet provar totes les funcionalitats de les que disposa el dispositiu i trobar errors que potser només ocorren en un dispositiu real. La velocitat de resposta a l'hora de fer servir l'aplicació també és molt superior (és velocitat real) ja que no es tracta d'un entorn emulat.

6.1.3. Proves reals

Tot i que el punt anterior ja és gairebé real, encara hi ha un tercer pas en el qual s'ha utilitzat un servidor independent de l'entorn de desenvolupament i un dispositiu real. En el servidor s'ha desplegat el mòdul WAR obtingut del projecte SharePlaces explicat anteriorment i al dispositiu Android s'ha instal·lat l'aplicació a partir del paquet APK creat a partir del projecte Geoshare. L'entorn de desenvolupament proporciona eines per crear el mòdul WAR i també per crear i signar el paquet APK.

La instal·lació al dispositiu es pot fer mitjançant la següent comanda que porta el kit de desenvolupament d'Android:

```
adb -s 3230837CFC9300EC install geoshare.apk
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
3434 KB/s (895411 bytes in 0.254s)
     pkg: /data/local/tmp/geoshare.apk
Success
```

on el paràmetre “-s” indica l'identificador del dispositius i serà l'identificador del dispositiu.

A diferència de les proves amb l'emulador i les depurant en el dispositiu real, aquest mètode ja no s'executa en mode de depuració sinó en mode real. El sistema Android diferencia aquests dos modes i hi ha petits canvis interns en funció d'això que, teòricament, no haurien d'afectar l'execució. El que si que cal tenir en compte en aquest cas en concret és que pel sistema de mapes de Google Maps és necessària l'*api-key* per l'entorn productiu com s'ha comentat anteriorment. Un cop l'aplicació passa les proves d'aquesta fase, es pot considerar estable ja que, tant instal·lació com l'utilització, són tal com ho farien els usuaris finals.

6.2. Millores i ampliacions

L'aplicació desenvolupada tal com s'ha presentat en aquest document esta preparada per funcionar en productiu. Tot i així, hi ha moltes funcionalitats útils que es podrien afegir, les més importants que en un projecte real s'haurien d'afegir el més aviat possible serien les següents:

- Recuperar contrasenya: per facilitar als usuaris la creació d'una compte en el sistema a traves d'un dispositiu mòbil, en aquesta aplicació només es demanen les dades estrictament necessàries (nom d'usuari i contrasenya). Això representa un problema a l'hora de recuperar la contrasenya en cas de que l'usuari l'ha oblidat, ja que no tenim cap manera d'enviar-li una de nova. Una possible solució seria un sistema alternatiu per aconseguir el mateix. Es podria fer, per exemple, que a partir del nom d'usuari es mostren una serie de marques en el mapa i l'usuari hauria d'escollir un nombre mínim de marques definides per ell.
- Pàgina web: l'aplicació esta pensada per poder funcionar sense cap pagina web complementaria però, en un futur, es podria afegir com a centre de control més complet ja

que una aplicació d'un dispositiu mòbil sempre intenta ser el mes senzilla possible per no sobrecarregar la pantalla.

- Versions: com s'ha comentat anteriorment, el botó de menú i els menús són obsolets en els dispositius nous. En l'aplicació desenvolupada s'han fet servir menús per ser compatible amb versions anteriors però es podrien implementar dues versions diferents. Una per les versions anteriors fent ús dels menús i una per les versions més noves fent ús dels *ActionsBars*. A part d'aquest punt també hi ha altres aspectes que es podrien adaptar en funció de la versió del dispositiu.
- Netejar *tokens* caducats: cada vegada que es comprova la validesa d'un *auth-token* també es comprova si hi ha altres auth-tokens caducats del mateix usuari per eliminar-los. Per un millor funcionament i sobretot si hi ha molts usuaris utilitzant el sistema, la millor opció seria comprovar els *auth-tokens* caducats temporalment mitjançant algun procés tipus *Cron*.

7. Conclusions

Al llarg del projecte s'han anat implementant amb èxit les diferents parts, tant les del client seguint les regles que imposa l'Android com les del servidor mitjançant tecnologia web en forma de Servlets. Per la part del client ha calgut un esforç inicial per entrar al món d'aquest sistema operatiu i entendre el seu funcionament, les possibilitats que proporciona i els límits que imposa. Aquesta tasca d'autoaprenentatge s'ha mantingut en certa manera durant tot el projecte per aconseguir una aplicació que segueix, sempre que és possible, les bones pràctiques exposades pels mateixos desenvolupadors del sistema com per altres desenvolupadors amb experiència.

S'ha vist bastant a fons tot el que s'ha aplicat durant aquest projecte, qüestionant el perquè de les pràctiques explicades i buscant sempre la millor manera d'implementar una funcionalitat o de dur a terme una determinada acció. Però cal notar que per cada tasca tancada s'han obert varies de noves que amb més temps es podrien investigar per aconseguir, possiblement, una aplicació més completa encara. Afortunadament, tot i que es tracta d'un sistema operatiu relativament jove, s'ha trobat molta informació sobre qualsevol dels punts investigats i aquest fet és un punt molt positiu pel sistema i ajuda a assegurar el seu futur.

Pel que s'ha vist durant aquest projecte, Android és un sistema bastant complet que dona moltes llibertats als desenvolupadors però també millorable en alguns aspectes. Així, algunes parts del sistema són una mica delicades i no acaben de funcionar si no es segueix una manera determinada la qual en alguns casos no esta documentada. També la manera de funcionar del sistema operatiu, i amb això la manera que els desenvolupadors han de seguir a l'hora d'implementar certes funcionalitats, sembla que es complica i limita amb cada versió nova que es publica. Tot i que en aquest projecte no s'ha pogut veure a fons, sembla que el sistema tendeix a complicar-se amb el temps i que per suportar tots els dispositius, tecnologies i versions caldrà implementar versions lleugerament diferents per cada cas. Aquest fet podria complicar-se molt en un futur i, com actualment ja esta passant en el mon dels navegadors web, el manteniment de codi d'aquest tipus és bastant caòtic. Aquest fet segurament és inevitable ja que els dispositius realment són diferents i els desenvolupadors d'Android actualment ja estan intentant d'organitzar-ho d'alguna manera.

A nivell de projecte s'han vist totes les parts, des de la planificació del projecte, passant per l'anàlisi, el disseny i l'implementació per realitzar, finalment, les proves finals en un entorn simulat i mitjançant un servidor i un dispositiu Android real, instal·lant l'aplicació de la mateixa manera com ho faria l'usuari final. S'ha intentat donar importància a diferents aspectes de l'aplicació (usabilitat, seguretat, mantenibilitat, etc.) i aconseguir una aplicació utilitzable tal com es presenta en aquest projecte. Tot i així, com passa amb gairebé totes les aplicacions, un cop dissenyada i implementada no s'acaba el procés i es podrien afegir moltes altres opcions i millorar algunes de les existents.

A nivell personal, l'experiència amb aquest nou sistema ha estat bona. S'ha vist bastant a fons moltes parts del sistema operatiu i el seu funcionament, tot i que amb el temps donat per realitzar el projecte és impossible veure-les totes. Tot i que considero que alguns aspectes del funcionament de l'Android són qüestionables, he trobat molt interessant el desenvolupament amb aquest sistema i el projecte en general i he arribat a la conclusió que val la pena aprofundir-hi més.

8. Fonts d'informació

Durant el desenvolupament d'aquest projecte s'ha consultat una gran quantitat de fonts d'Internet. Les principals són les següents:

- Pàgina oficial de desenvolupament Android:
  - http://developer.android.com/index.html
  - http://developer.android.com/training/index.html
  - http://developer.android.com/training/advanced.html

- Pàgines oficials de Google Maps i Open Street Maps per Android:
  - https://developers.google.com/maps/documentation/android/v1/
    (el 3 de desembre va sortir la versió v2 però pel projecte actual s'ha fet servir la v1)
  - http://code.google.com/p/osmdroid/

- Càlculs geogràfics
  - http://www.movable-type.co.uk/scripts/latlong.html
  - http://janmatuschek.de/LatitudeLongitudeBoundingCoordinates
  - http://en.wikipedia.org/wiki/Geographic_coordinate_conversion

- Bases de dades del client i del servidor:
  - http://www.sqlite.org/
  - http://www.mysql.com/
  - http://www.vogella.com/articles/AndroidSQLite/article.html

- Comunicació entre client i servidor:
  - http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html
  - http://code.google.com/p/google-gson/
  - http://wiebe-elsinga.com/blog/?p=405
  - http://nelenkov.blogspot.com.es/2011/12/using-custom-certificate-trust-store-on.html

- Brúixola orientativa
  - http://www.vogella.com/articles/AndroidSensor/article.html
  - http://developer.android.com/reference/android/hardware/SensorManager.html