14mtig 2012

Màster en Tecnologies de la Informació Geogràfica, 14a. Edició

Disseny i implementació d'una aplicació per a dispositius mòbils per identificar elements geològics en el terreny

Projecte de final de màster – febrer 2013

Autor

Marc Pérez Castells

Tutors

MTIG: **José Quirós Jiménez**
IGC: **Jordi Marturià Alavedra**

Organitzador

Universitat Autònoma de Barcelona
Departament de Geografia

Institució col·laboradora

IGC Institut Geològic de Catalunya

Resum

Avui en dia els dispositius mòbils s'ha convertit en una eina comuna i de grans possibilitats. Per aprofitar les seves potencialitats, en aquests treball es presenta el procés de disseny i desenvolupament d'un prototipus d'aplicació per a mòbil que permet millorar la localització i identificació d'informació geològica sobre el terreny. En concret, l'aplicació creada s'anomena **IdAllau** i s'ha creat per ajudar a identificar les zones d'allaus en el camp.

S'ha marcat com a objectiu dotar l'aplicació de les següents funcions:

- **Emmagatzematge**: emmagatzematge al dispositiu perquè l'usuari pugui utilitzar l'aplicació quan no hi ha disponible una connexió a Internet.
- **GEO localització**: establir la posició de l'usuari i de les zones d'allaus al seu voltant.
- **Fotografia**: poder fer fotografies.
- **Representació cartogràfica**: visualització de cartografia en el dispositiu mòbil per ajudar al usuari a situar-se ell i a les zones d'allaus.
- **Realitat augmentada**: representar les zones d'allaus en realitat augmentada per millorar la identificació només mirant a través de la càmera del dispositiu.
- **Recerca per proximitat**: buscar les zones d'allaus pròximes a l'usuari.

Tenint clares les funcions, s'ha comprovat l'estat de la tecnologia i es decideix dur a terme l'aplicació per a dispositius amb sistema operatiu (SO) **Android**, degut a les seves facilitats per començar a programar, el cost zero, la gran quantitat d'informació disponible i perquè s'ha convertit en el SO que utilitzen més dispositius mòbils en el nostre país.

Amb les funcions i la tecnologia en ment, s'ha dissenyat la aplicació pensant en un funcionament fàcil i intuïtiu per a facilitar el seu ús en camp. El disseny i desenvolupament s'ha dividit en 6 bloc funcionals que duen a terme les diferents funcionalitats de l'aplicació.

En primer lloc, al iniciar l'aplicació apareix un menú per accedir a les tres principals funcions (mapa, realitat augmentada i càmera) i a la informació de l'aplicació. Al iniciar-se per primer cop, i de forma interna, es crea la base de dades en la plataforma SQLite a partir d'un arxiu XML amb la informació de totes les zones d'allau.

Per a desenvolupar la vista del mapa s'ha utilitzat la llibreria de codi lliure ***Osmdroid***. Aquesta permet dibuixar els mapes offline (carregant-los prèviament a l'aplicació), marca i seguir la posició de l'usuari, i dibuixar les zones d'allau com a punts d'interès. Al seleccionar-ne un apareix un quadre amb la seva informació bàsica. Des d'aquí es pot accedir a una vista amb més informació (i amb accés a la seva fitxa) o a la realitat augmentada (mantenint la selecció).

Per desenvolupar la realitat augmentada s'ha utilitzat la llibreria ***Appunta***. Aquí es mostren les zones d'allau superposades sobre la imatge captada per la càmera. Es pot seleccionar un radi de visualització gràcies a una barra lliscant per no sobrecarregar la vista amb massa informació. Igual que en el mapa, al

seleccionar una zona d'allau es mostra la seva informació bàsica, com un enllaç a la informació ampliada i al mapa. Des de la realitat augmentada (i des del mapa) es pot fer una recerca d'aquelles zones d'allaus que es trobin a una determinada distància. Apareix una llista amb totes elles, i al seleccionar-ne una, l'aplicació retorna a la vista des d'on l'usuari a fer la recerca.

El resultat final és una aplicació funcional per a dispositius mòbils amb Android que permetrà als tècnics tenir una nova eina per a la identificació de les zones d'allaus en el camp. L'aplicació s'ha desenvolupat amb codi lliure, amb els avantatges i inconvenients que comporta. Com primera versió d'una aplicació tecnològica, s'inclouen un conjunt de propostes de millorar o de futur, ja que qualsevol recurs tecnològic està lligat als canvis que pateix la seva pròpia tecnologia.

Paraules clau: Sistemes d'informació geogràfica, Zones d'allaus, Aplicació Android, Realitat augmentada, Osmdroid, Appunta.

Resumen

Hoy en día los dispositivos móviles se han convertido en una herramienta comuna y de grandes posibilidades. Para aprovechar sus potencialidades, en este trabajo se presenta el proceso de diseño y desarrollo de un prototipo de aplicación per a móvil que permite mejorar la localización y identificación de información geológica sobre el terreno. En concreto, la aplicación creada se llama **IdAllau** i se ha creado para ayudar a identificar las zonas de aludes en el campo.

Se han marcado como objetivos dotar la aplicación de las siguientes funciones:

- **Almacenamiento**: almacenamiento en el dispositivo para que el usuario pueda utilizar la aplicación cuando no hay disponible una conexión a Internet.
- **GEO localización**: establecer la posición del usuario y de las zonas de aludes a su alrededor.
- **Fotografía**: poder hacer fotografías.
- **Representación cartográfica**: visualización de cartografía en el dispositivo móvil para ayudar al usuario a situarse él y las zonas de aludes.
- **Realidad aumentada**: representar las zonas de aludes en realidad aumentada para mejorar la identificación solo mirando a través de la cámara del dispositivo.
- **Búsqueda por proximidad**: buscar las zonas de alud próximas al usuario.

Teniendo claras las funciones, se ha comprobado el estado de la tecnología i se ha decidido llevar a cabo la aplicación para dispositivos con el sistema operativo (SO) Android, debido a sus facilidades para empezar a programar, el coste cero, la gran cantidad de información disponible i porqué se ha convertido en el SO que utilizan más dispositivos móviles en nuestro país.

Con las funciones i la tecnología en mente, se ha diseñado la aplicación pensando en un funcionamiento fácil y intuitivo para facilitar su uso en campo. El diseño y desarrollo se ha dividido en 6 bloques funcionales que llevan a cabo las diferentes funcionalidades de la aplicación.

En primer lugar, al iniciar la aplicación aparece un menú para acceder a las tres principales (mapa, realidad aumentada y cámara) y a la información de la aplicación. Al iniciarse por primera vez, y de forma interna, se crea la base de datos en la plataforma SQLite a partir de un archivo XML con la información de todas las zonas de alud.

Para desarrollar la vista de mapa se ha utilizado la librería de código libre ***Osmdroid***. Ésta permite dibujar los mapas offline (cargando éstos previamente en la aplicación), marcar y seguir la posición del usuario, y dibujar las zonas de alud como punto de interés. Al seleccionar una aparece un cuadro con su información básica. Desde aquí se puede acceder a una vista con más información (y con acceso a su ficha) o a la realidad aumentada (manteniendo la selección).

Para desarrollar la realidad aumentada se ha utilizado la librería ***Appunta***. Aquí se muestran las zonas de alud superpuestas sobre la imagen captada por la cámara. Se puede seleccionar un radio de visualización gracias a una barra deslizante para no sobrecargar la vista con demasiada información. Igual que en el mapa, al seleccionar una zona de alud se muestra su información básica, con un enlace a la información ampliada y al mapa. De de la realidad aumentada (y des del mapa) se puede hacer una búsqueda de esas zonas de alud que se encuentren a una determinada distancia. Aparece una lista con todas ellas, y al seleccionar una, la aplicación vuelve a la vista des de donde el usuario a echo la búsqueda.

El resultado final es una aplicación funcional para dispositivos móviles con Android que permitirá a los técnicos tener una nueva herramienta para la identificación de las zonas de aludes en el campo. La aplicación se ha desarrollado con código libre, con las ventajas e inconvenientes que comporta. Como primera versión de una aplicación tecnológica, se incluye un conjunto de propuestas de mejora o de futuro, ya que cualquier recurso tecnológico está atado a los cambios que padece su propia tecnología.

Palabras clave: Sistema de información geográfica, Zonas de aludes, aplicación Android, Realidad aumentada, Osmdroid, Appunta.

Agraïments

Vull agrair José Quirós Jiménez (UAB) i Jordi Marturià Alabedra (IGC) per haver-me ajudat i orientat en el desenvolupament del projecte. Sent la primera vegada que m'enfronto al disseny i desenvolupament d'una aplicació des de zero, la seva orientació ha estat clau per poder dur a bon port aquest projecte.

Al personal del Màster en Tecnologies de la Informació Geogràfica (MTIG) i del LIGIT pels coneixements impartits durant l'any i el suport i ajut en el desenvolupament dels projecte del màster.

Al Institut Geològic de Catalunya (IGC) per confiar i donar-me l'oportunitat de desenvolupar una aplicació per a dispositius mòbils, projecte que mai s'havia dut a terme en aquesta institució. També al seu personal que m'ha acollit durant el període de pràctiques; especialment a la Neus Querol Vidal per la seva implicació en el projecte i el seu ajut.

Un fort agraïment als que van començar sent els companys de màster hi han acabat sent amics. A la 14ª promoció del MTIG, els quals ens hem ajudat i donat suport durant les llargues hores en el LIGIT com posteriorment en les pràctiques d'empresa. Especialment a en Dani Vidal per ser el nostre *delegat*.

No pot faltar un agraïment a família i amics, els quals m'han acompanyat durant el màster i el projecte, i espero que ho continuïn fent en el futur.

Índex

Índex de figures

Índex de codi

1. Introducció

En els darrers anys, els dispositius mòbils, com poden ser els *smartphones* i *tablets*, s'ha convertit en un objecte habitual, i fins i tot indispensable, per la societat moderna. La dotació d'aquests aparells amb connectivitat (per *Bluethooth*, per infrarojos i/o Internet), càmera, sistemes de posicionament (GPS o WIFI), brúixola digital, acceleròmetre, etc. han convertit els dispositius mòbils amb potents ordinadors capaços de dur a terme una gran varietat de funcions i usos.

Aprofitant tot aquest conjunt d'eines, s'ha decidit dissenyar i desenvolupar un prototipus d'aplicació per a mòbil que permeti millorar la localització i identificació d'informació geològica sobre el terreny d'una forma senzilla utilitzant totes aquestes noves funcionalitats.

1.1. Antecedents

Tot i que existeixen diversos projectes anteriors a aquest enfocats a la programació, no hi ha cap que s'hagi posat com a fita dissenyar i desenvolupar una aplicació per a dispositius mòbils. No es d'estranyar doncs el *boom* dels dispositius mòbils és recent i just ara comencen a obrir-se pel gran públic els mitjans i eines per a desenvolupar aplicacions per aquests dispositius tan versàtils.

Espero crear un precedent i que les futures promocions del màster aprofitin per portar a terme projectes de programació enfocats aquest nou important mercat que encara avui comença a obrir-se davant nostre.

1.2. Marc institucional

Aquest projecte de final del Màster en Tecnologies de la Informació Geogràfica[1] (MTIG), a la seva 14ª edició, impartit en el Laboratori d'Informació Geogràfica i de Teledetecció[2] (LIGIT), Departament de Geografia de la Universitat Autònoma de Barcelona[3] (UAB), s'ha dut a terme conjuntament amb l'Institut Geològic de Catalunya[4] (IGC) gràcies a un conveni de pràctiques d'empresa entre les dues institucions.

El MTIG està ben consolidat després de 16 any d'experiència. Ofereix una formació de qualitat sobre quatre àrees bàsiques en el desenvolupament dels sistemes d'informació geogràfica: la expressió cartogràfica i les noves formes de visualització de dades; la definició, estructuració i gestió de la informació en els sistemes d'informació (SIG i bases de dades relacionals); les funcions d'anàlisis geoespacial (geoprocés, anàlisis de terreny i de xarxes); i la programació d'aplicactius de SIG, en particular per Internet i amb tecnologia de components.

[1] Lloc web del MTIG: http://ligit0.uab.es/mtig/index.htm
[2] Lloc web del LIGIT: http://ligit0.uab.es/web/
[3] Lloc web de la UAB: http://www.uab.es/
[4] Lloc web de l'IGC: http://www.igc.cat/

El LIGIT es dedica, principalment, al tractament informàtic de la informació geogràfica (tant cartogràfica com alfanumèrica). La seva tasca consisteix en produir informació del territori i organitzar-la en sistemes d'informació geogràfica per tal de facilitar la consulta, gestió i anàlisi territorial. Bàsicament dóna suport a la recerca; realitza projectes de desenvolupament de tecnologies de la informació geogràfica mitjançant convenis i contractes amb empreses i institucions externes; i proporciona formació específica en el seu camp.

El 2005 el Parlament de Catalunya va aprovar la creació de l'IGC amb personalitat jurídica pròpia i plena capacitat d'obrar per a exercir les seves funcions; encara que l'existència d'un servei geològic institucional a nivell català ja era present en la forma de Servei Geològic de Catalunya (SGC).

Les activitats del IGC, creat l'any 2005, se centren fonamentalment en l'estudi, l'assessorament, la investigació i la prevenció del sòl i el subsòl de Catalunya; abastant, el seu camp d'actuació, totes les branques de la geologia (mineralogia, petrologia, estratigrafia, geomorfologia, geoquímica, hidrogeologia...) i també les disciplines que hi estan relacionades: les ciències del sòl i l'edafologia, la geofísica, la sismologia, l'enginyeria geològica i la geotècnia.

Les funcions principals que l'Institut exerceix són, entre moltes altres: elaborar el mapa geològic de Catalunya, a escales diverses, i portar a terme programes de cartografia, bases de dades i sistemes d'informació sobre el sòl i el subsòl; desenvolupar i mantenir la xarxa sísmica; estudiar i avaluar els riscos geològics, inclòs el sísmic i el d'allaus; elaborar i fomentar estudis, treballs i avaluacions en el camp de la geologia i de les ciències que s'hi relacionen; assessorar i prestar assistència tècnica en el camp de la geologia i de les disciplines afins al DPTOP, altres departaments i altres administracions; supervisar, si se sol·licita, estudis geotècnics de terrenys; elaborar estudis sobre el sòl i el subsòl; establir protocols a seguir en l'elaboració dels estudis geològics, geofísics i geotècnics, i facilitar informació aplegada en les bases de dades.

2. Objectius i definició de l'aplicació

2.1 Objectius

L'aplicació que es vol desenvolupar té l'objectiu primordial d'ajudar a la identificació de les zones d'allaus en camp. Partint d'aquesta objectiu, es plantegen les següents funcionalitats per a l'aplicació:

- Facilitar el posicionament de les zones d'allaus sobre cartografia o ortofotos.
- Ajudar a la identificació de les zones d'allaus amb realitat augmentada.
- Cerca de zones d'allaus per proximitat.
- Treball off-line.

2.2 Definició de l'aplicació

L'aplicació es desenvolupar per a dispositius mòbils amb sistema operatiu Android. Aquesta farà servir el sistema de posicionament per a situar-se en el territori i així poder mostrar les zones d'allaus en realitat augmentada; és a dir, sobre la imatge real captada a través de la càmera. Degut a la mala cobertura en les zones de muntanya, l'aplicació ha de poder funcionar sense cobertura 3G o WIFI. Les dades, tant de les zones d'allaus com la cartografia i ortofotos, servides en un principi pel propi IGC, s'hauran de carregar prèviament.

Tenint en compte les funcionalitats plantejades, els usuaris potencials de l'aplicació seran el personal de IGC o altres institucions que participen en el recull d'informació a les zones d'allaus i que utilitzaran l'aplicació per a facilitar la recollida d'informació en camp. Ara mateix no es planteja l'ús de l'aplicació pel personal no vinculat al IGC o a les seves activitats. Amb tot això, l'aplicació es defineix com a un element de suport pel personal del departament de *Servei d'allaus* de l'IGC.

Alguns exemples d'aplicacions que fan us d'alguna de les característiques o funcions que es volen implementar en l'aplicació es mostren a continuació:

- *Geologia CAT*[5]: serveix per descobrir el patrimoni geològic de Catalunya i les seves zones geològiques més rellevants de utilitzant la realitat augmentada.

- *iGeology 3D*[6]: és una aplicació que mostra un mapa geològic al voltant de l'usuari utilitzant la realitat augmentada.

- *GeoPaparazzi*[7]: és una eina desenvolupada per realitzar estudis qualitatius tècnics i científics. La seva força recau en la seva connexió directa amb el GIS BeeGIS, que es pot utilitzar per processar encara més les dades recollides. Encara que l'objectiu principal és el camp de la topografia, conté eines que poden ser de gran utilitat pel turisme interessat en la geologia.

[5] Geologia CAT: https://play.google.com/store/apps/details?id=com.Geology&hl=es_419

[6] iGeology: http://www.bgs.ac.uk/igeology/3D.html

[7] GeoPaparazzi: https://play.google.com/store/apps/details?id=eu.hydrologis.geopaparazzi

- *GeoCam*[8]: està dissenyada per a crear i veure fotografies GEO localitzades; és a dir que disposen d'informació addicional com les coordenades geogràfiques, l'orientació de la càmera en el moment de fer la fotografia, etc.

2.3 Pla de treball

El pla de treball es divideix en cinc fases principals. En la primera es va definir l'aplicació, tant els objectius que vol complir com les funcionalitats que ha de desenvolupar.

En la segona es va dissenyar l'aplicació. Amb un disseny previ s'aconsegueix que en el moment de començar a programar, el comportament de l'aplicació ja està definit a grans trets.

La tercera fase correspon a la programació. Un com preparat l'entorn de treball es començà treballar en la programació. Per cada funció primer es creen les vistes que requerirà i després es programen les funcions que ha de desenvolupar.

En la quarta fase es desenvolupen les proves en terminals reals, tant en un entorn controlat con en l'entorn real d'ús. Després de les proves, es torna a l'entorn de programació per corregir errades o *bugs*. Finalment, amb la fase cinc, es conclou amb la compilació de l'aplicació en el format adequat per a la distribució.

Fase I – Definició de l'Aplicació

1. Definir objectius: es defineixen els objectius que ha de complir l'aplicació.
2. Definir funcionalitats: tenint en compte l'objectiu de l'aplicació, es fa un recull de les principals funcionalitats que haurà de complir per conèixer els recursos necessaris alhora de programar l'aplicació.
3. Recull documental: es recullen documentació referent a la programació Andorid com manuals, treballs d'altres aplicacions, etc.

Fase II – Disseny de l'Aplicació

1. Estructuració del sistema
    a. Disseny del sistema: es dissenya la infraestructura que ha de tenir l'aplicació; tenint en compte també les funcions de connexió amb els servidors de dades.
    b. Format de les dades: s'ha d'establir el format de les dades que utilitzarà l'aplicació.
2. Diagrama del sistema: es dissenya l'estructura interna de l'aplicació. Com les diferents vistes de l'aplicació estan interconnectades i les diferents funcions que desenvoluparen les accions que porti a terme l'usuari.
3. Disseny de les vistes: es dissenya l'aspecte visual de les diferents aplicacions. Els elements que s'han de mostrar i la seva distribució per fer l'aplicació agradable, fàcil i intuïtiva al usuari.

Fase III – Programació

1. Entorn de treball: s'han d'instal·lar els programes i aplicacions necessàries per dur a terme la programació.

[8] GeoCam: https://play.google.com/store/apps/details?id=ru.sitis.android.geocam&hl=ca

2. Implementació funcionalitats
    a. Lectura de les dades: implementar la lectura i anàlisi de les dades per part de l'aplicació.
    b. Emmagatzematge: implementar l'emmagatzematge de les dades en el dispositiu.
    c. Representació cartogràfica: implementar la visualització de cartografia.
    d. Posicionament: implementar la GEO localització.
    e. Realitat augmentada: implementar la realitat augmentada.
    f. Càmera: implementar la càmera.
    g. Recerca: crear els mètodes de recerca de les zones d'allaus per proximitat.
3. Depuració: és un procés continuo durant tota la fase de programació on es prova el codi per no acumular errors en aquest.

Fase IV – Proves funcionals

1. Proves del funcionament: proves de funcionament prèvies al camp real.
2. Prova en camp: prova final de l'aplicació en un entorn real d'ús.
3. Revisió del codi: depuració dels errors trobats en el camp.

Fase V – Compilació

Compilació de l'aplicació: neteja del codi i compilació en el format de distribució

En l'*Annex I* trobareu el cronograma del pla de treball.

3. Funcionalitats

Aquí es presenten aquelles funcionalitats principals que l'aplicació té. Cal tenir en compte que aquestes funcionalitats han evolucionat alhora que el desenvolupament de l'aplicació avançada, degut a l'estat de la tecnologia o els mitjans disponibles. Així mateix, en general les funcions inicial s'han mantingut (o millorat) i s'han agregat noves funcions.

3.1 Emmagatzematge

L'aplicació ha de control sobre les funcions d'emmagatzematge al dispositiu perquè l'usuari pugui utilitzar l'aplicació quan no hi ha disponible una connexió a la xarxa. L'aplicació guardarà les seves dades en un conjunt de carpetes creades expressament al instal·lar l'aplicació en el dispositiu. Indistintament de l'existència de targeta SD, aquell espai existeix en tots els dispositius.

3.2 GEO localització

L'aplicació permetrà, a través del dispositiu mòbil, establir la posició de l'usuari. Aquesta informació servirà per a dos propòsits:

a) Situar l'usuari a partir de les seves coordenades.

b) Situar el dispositiu mòbil, i utilitzant la brúixola i l'acceleròmetre, determinar cap on està dirigit i poder activar la realitat augmentada.

3.3 Fotografia

La informació visual del camp pot ajudar molt al futur anàlisi de les allaus succeïdes.

3.4 Representació cartogràfica

Avui en dia hi ha moltes aplicacions que utilitzen la GEO localització, i la majoria acostumen ha estar complementades amb la representació d'aquesta posició sobre una cartografia. L'aplicació que es desenvolupa també ha de poder representar cartografia, ja sigui des del ICC (com topogràfica) o cartografia temàtica. Per tant, l'aplicació ajudarà a situar millor a l'usuari gràcies a veure la seva posició (i moviment) sobre un mapa i podrà mostrar informació temàtica.

Com en altres aplicacions, s'ha de permetre la interacció amb la cartografia. Aquesta pots ser de molts tipus, però les més bàsiques han de ser el canvi d'escala i el moviment lliure per la cartografia.

3.5 Realitat augmentada

Observant l'entorn amb el dispositiu mòbil (gràcies a una càmera que ha d'incorporar el dispositiu mòbil) l'usuari podrà veure la realitat del seu voltant juntament amb una capa d'informació superposada. Aquesta capa virtual pot ser de molts tipus, però com més complexa és més informació "real" es perd. Gràcies als instruments que incorporen aquests dispositiu (acceleròmetre, brúixola digital, etc.), el moviment de la càmera o de l'usuari serà compensat i la informació virtual representada seguirà els moviments i es mantindrà adequadament representada.

La informació virtual estarà composta per uns punts d'interès (POI), corresponents a les zones d'allaus, i simbolitzats de forma senzilla. A més de la simbolització, els POI han de dur un identificador per fer més fàcil la seva identificació. Un cop seleccionat un POI, la informació pot ser ampliada en una finestra o en un *pop-up*.

3.6 Recerca per proximitat

La funció de recerca per proximitat ha de permetre a l'usuari buscar una zona d'allau concretat en un radi de proximitat establert i poder obtenir-ne la posició en el mapa, en la realitat augmentada o simplement per conèixer tota la seva informació.

4. Estat de la tecnologia

En aquest apartat es presenta l'estat de la tecnologia disponible per complir amb els objectius i funcions que volem desenvolupar. Com en qualsevol ambient de programació, i encara més tractant-se de dispositius mòbils, aquest recull està sotmès al moment concret en que s'ha desenvolupat l'aplicació; per tant, la tecnologia disponible pot haver variat i hi haurà més possibilitats per a resoldre els reptes de programació.

4.1 Sistema operatiu (SO)

Avui en dia el mercat de les aplicacions mòbils es reparteixen principalment entre Apple amb el seu sistema operatiu iOS[9] i Google amb Android[10]. En cas de iOS, amb tecnologies pròpies i només disponible en sistemes del propi Apple. La seva força està en una base d'usuaris amb un perfil econòmic mitja–alt, que permet als desenvolupadors fixar uns preus més atractius i que compensen l'elevada inversió inicial per entrar al *App Store* (no s'adquireixen les aplicacions o App's). Tot i existir un kit de desenvolupament, només és possible utilitzar les aplicacions en els dispositius després de pagar la quota del iPhone Developer Program.

En el cas d'Android, la seva ràpida expansió i disponibilitat en multitud de dispositius de diversos fabricants, així com unes tecnologies obertes que permeten un cicle de desenvolupament d'aplicacions amb un cost inicial nul, li han servit per ocupar el mercat de dispositius i aprofiten aquesta situació de força per atreure a més o menys desenvolupaments. El gran avantatge d'Android és que disposa d'una gran comunitat de desenvolupadors que escriuen aplicacions per estendre la funcionalitat dels dispositius Android.

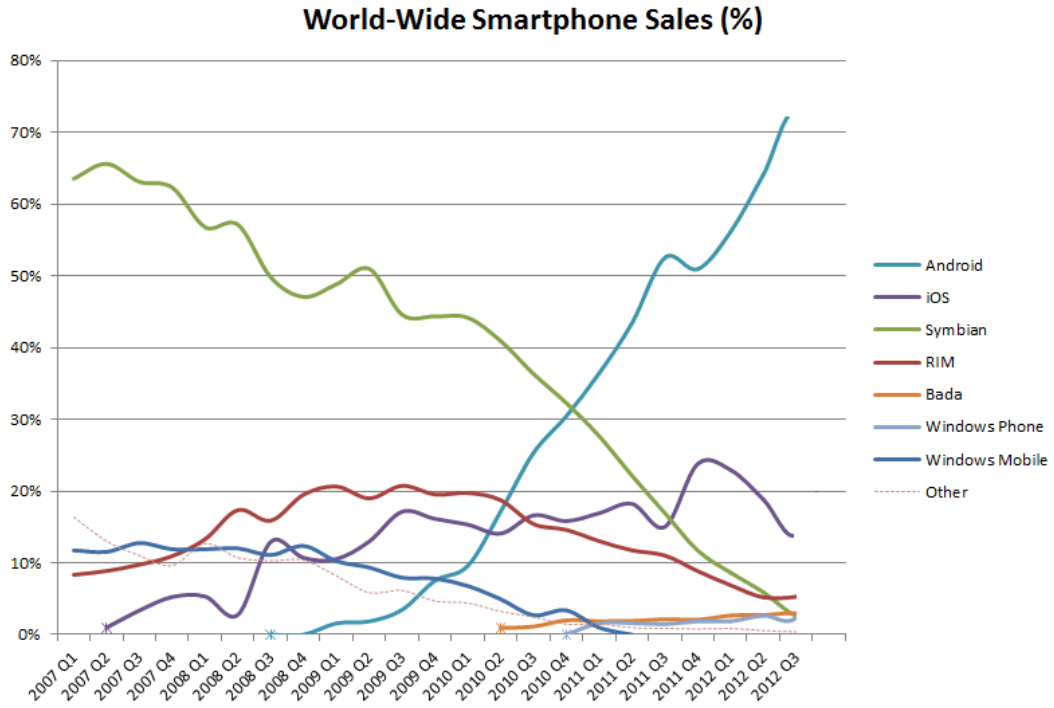


Figura 1. Evolució del percentatge de Smartphones venuts amb Q3 de 2012 pels diferents SO[11].

[9] Lloc del SO iOS: https://developer.apple.com/devcenter/ios/index.action
[10] Lloc del SO Android:
[11] Gartner Smart Phone Marketshare: http://www.gartner.com/newsroom/id/2237315

4.1.1 Solució escollida

L'aplicació es desenvolupa per a ser utilitzada en dispositius Android perquè cada dia hi ha més dispositius mòbils que utilitzen aquests SO. Els avantatges de programar amb aquest SO són molts: costos nuls per a programar (tant el codi propi d'Android com pel que fa al seu ús i l'entorn de treball), una documentació extensa (tant oficial com d'altres programadors) i de facilitat per crear un entorn de programació i fer els primers passos en la programació en JAVA. Així doncs, la resta de tecnologies esmentades en aquest apartat seran per Android.

4.2 Projecció de cartografia

Gràcies als sistemes de projecció que permeten representar un tros o àrea del planeta en un pla en 2 dimensions i la GEO codificació que assigna coordenades geogràfiques a punts del mapa, hom pot representar mapes en un dispositiu mòbil i interactuar-hi. Els elements representats són els anomenats SIG (Sistemes d'Informació Geogràfica) que no deixen de ser sistemes de mapes digitalitzats i que guarden dades geogràfiques dissenyades per poder ser capturades, emmagatzemades, editades, estudiades i analitzades de forma senzilla.

Els elements d'una capa d'un SIG es poden representar de diferents maneres, segons el seu ús, les dos més conegudes són:

a) El mètode *Raster*: es basa en definir tota la zona que avarca un mapa en una malla. Així s'obté un gran número de petites celles regulars, a les quals se lis assigna una propietat dins la capa. Poden emmagatzemar-se com a fitxers binaris (BLOB) o en fitxers d'imatge (com JPEG).

b) El mètode *Vectorial*: és el sistema més utilitzat a l'hora de representar les dades en capes. Aquest sistema intenta definir amb precisió la localització de diferents elements geogràfics. Per guardar aquesta informació s'utilitzen tres elements geomètrics: punt, línia i polígon.

Un cop representada la cartografia, s'ha de tenir en compte com i fins on es pot interactuar amb aquesta. Amb l'aparició dels dispositius tàctils, la interacció persona-dispositiu s'ha simplificat gràcies a les opcions que ofereixen els esdeveniments tàctils. Concretament els esdeveniments multi tàctils (o *multi-touch* tal i com va registrar Apple) permeten canviar d'escala (gestió d'ampliació i gestió de reducció) o moure's per la cartografia lliurement.

4.2.1 Recursos disponibles

Hi ha moltes diverses API's de mapes per a mòbils que poden utilitzar-se. Algunes d'elles són:

- *Google APIs Add-On*[12]: és una extensió de l'entorn de desenvolupament d'Android SDK que permet desenvolupar aplicacions per a dispositius que, entre altres, inclou la biblioteca externa *Maps* que li permet afegir potents capacitats de mapeig per aplicar a Android. És necessari obtenir prèviament una clau d'ús (API Key), que estarà associada el certificat amb la qual es firma digitalment l'aplicació. Amb aquesta clau es té un control d'ús ja que hi ha un límit de 25000 peticions de mapes per dia i aplicació.

[12] Lloc web de la Google APIs Add-On: https://developers.google.com/android/add-ons/google-apis/

- *Osmodroid*[13]: és una aplicació i una API de OpenStreetMap que bàsicament substitueix a la API de Google per a mapes. La diferencia principal recau en la llicencia de l'aplicació i de la API, que són lliures com OpenStreetMap. Té connexió directa als mapes servits per OpenStreetMap i pot utilitzar-se off-line.

- *MapsForge*[14]: ofereix una API de codi obert, que pot ser utilitzada desconnectada de la xarxa. Pot dibuixar mapes ràster com vectorials. L'objectiu del projecte és proporcionar eines lliures i obertes que permetin crear fàcilment noves aplicacacions basades en OpenStreetMap.

- *MGMaps Lib* SDK[15]: és una API lliure basada en Java Developer Kit Mobile Mapping que permet afegir característiques cartogràfiques al mòbil (mostra mapes, geocodificació, posicionament, POI, etc.). Aquesta plataforma està disponible per ús en ME Java, Android i BlackBerry. Per utilitzar l'API dins d'una aplicació pròpia, s'ha de generar una clau d'avaluació de llicencia per l'aplicació específica i el nom del proveïdor.

- *MapQuest Android* SDK[16]: s'ha dissenyat per a substituir l'API de mapeig nativa de Google. Pot utilitzar dades d'OpenStreetMap o el seu conjunt de dades amb llicència. No té límits preestablerts en el mapes amb la llicencia lliure Community Edition. Suporta arxius KML estàndard i arxius GeoRSS.

- *ArcGIS Runtime SDK*[17]: l'API d'ArcGIS per Android permet crear aplicacions que utilitzen les potents funcions de representació cartogràfica, geocoificació, geoprocesament i altres funcions personalitzades que proporciona ArcGIS Server utilitzant Java, i implementant-les en dispositius d'Android. L'API inclou un plug-in per l'entorn de desenvolupament integrat (IDE) d'Eclipse, que proporciona un ampli conjunt d'eines, documentació i mostres per ajudar als desenvolupadors a crear aplicacions. No és una API lliure i requereix com a mínim el registre en el Web d'Esri.

4.2.2 Solució escollida

Per al projecte, de les llibreries o API's anteriors, s'ha escollit utilitzar ***Osmdroid***. En primer lloc es va començar a utilitzar la API de Google, però en certs aspectes de simbolització i representació cartografia offline no s'acabava d'adequar a les necessitats. *Osmdroid* satisfeia les necessitats existents i a més es tractava de codi lliure (amb llicencia GNU Lesser GPL) i si fes falta es podria aplicar modificacions al codi de la API per a integrar-lo de forma adequada a l'aplicació.

4.3 GEO localització

La GEO localització permet la localització geogràfica real d'un objecte o persona, ja sigui mitjançant un dispositiu connectada a Internet, un telèfon mòbil o qualsevol altre aparell. Aquesta localització pot ser en un pla de dos dimensions (com per exemple Google Maps) o en un la de tres dimensions (GPS). En tot cas, conèixer la localització en un dispositiu mòbil ofereix una gran quantitat d'informació, actualitzada o no, al voltant del dispositiu.

[13] Lloc web de l'API Osmdroid: http://code.google.com/p/osmdroid/
[14] Lloc web de MapsForge: http://code.google.com/p/mapsforge
[15] Lloc web de MGMaps Lib SDK: http://www.nutiteq.com/mobile-map-api-sdk-guides
[16] Lloc web de MapQuest Android SDK: http://developer.mapquest.com/
[17] Lloc web de ArcGis Runtime SDK: http://www.esri.com/software/arcgis/smartphones/develop

Actualment existeixen diferents formes de localitzar un dispositiu mòbil, però l'efectivitat dependrà d'algunes variables com el mitja o la disponibilitat d'aquesta medició en el terminal. Els diferents sistemes es poden classificar en tres:

a) Basats en la xarxa: aquests sistemes utilitzen els sistemes del proveïdor de serveis per a determinar la posició d'un terminal, per la qual cosa no es necessita cap aplicació específica funcionant en el mòbil. El problema principal d'aquest sistema és que és precís estar a prop del proveïdor perquè funcioni.

b) Basats en el terminal: els dispositius utilitzen aquests sistemes disposen d'un receptor de senyals y un software per determinar la posició del terminal a través de senyals externes. El sistema més utilitzat en dispositius mòbils és el Global Positioning System (GPS).

c) Híbrids: els sistemes híbrids són una combinació de sistemes anteriors. Encara que contingui el mètode més fiable, també adquireix els problemes dels grups anteriors.

4.4 Realitat Augmentada

La realitat augmentada és una tecnologia on la visió de la realitat a través d'un dispositiu mòbil es veu ampliada amb elements virtuals que afegeixen informació digital. Aquest procés es basa en tecnologies derivades de la visualització i/o reconeixement de la posició per crear un sistema que reconegui la informació real que hi ha al voltant i creí una nova capa d'informació superposa a la visió de la realitat, ja sigui a través de gràfics en 2 dimensions o en 3. Aquesta informació es mescla amb el món real de forma que per l'usuari coexisteixen objectes virtuals i reals en el mateix espai. Aquesta és la principal diferencia de qualsevol aplicació tradicional d'un dispositiu, on tota la informació que veiem és virtual[18].

Hi ha diferents tècniques pels diferents passos que requereixin la realitat augmentada, però bàsicament les especificacions mínimes són:

- Eines de reconeixement del entorn a través d'imatges captades per una càmera, sensors o del posicionament.
- Una càmera que capti la imatge que estem enfocant perquè se li afegeixi la informació digital.
- Una unitat de procés i software especialitzat capaços de gestionar les diferents recursos necessaris per fer funcionar la realitat augmentada i per poder suportar la major potencia gràfica de la superposició de models 3D.
- Una font d'informació que el software que sàpiga relacionar amb a realitat que existeix al seu voltant i que proporcioni les dades per generar aquesta capa virtual.

Hi ha diversos mètodes visualització:

a) De cap: es basa en un casc o sistema de posicionament en el cap, i que ens pot mostrar la imatge de forma indirecta, veien la informació del món real a través d'un mirall, on a través

[18] Bover 2010.

d'una imatge afegida s'imprimeix a la vegada la informació digital; o de forma directa, veien la imatge a través d'un LCD on apareix la imatge gravada per una càmera junt amb la informació virtual.

b) De ma: dispositius portàtils que disposen d'una pantalla, en la qual es pot veure el món real i es pot superposar informació virtual.

c) Especials: dispositius anomenats SAR (Spatially Augmented Reallity) on la realitat del usuari és augmentada mostrant informació virtual en el propi entorn del usuari. Molts d'aquests sistemes encara es troben en investigació. Permet l'ús i interacció de diferents usuaris.

El registre d'objectes virtuals en la realitat augmentada es basa en descobrir la informació de l'entorn per saber on es troba el dispositiu, i per tant, on posicionar i com orientar els objectes virtuals. Aquesta és la part que requereix més esforç de la realitat augmentada, i depenent de la tècnica es requereix d'un tipus de hardware o d'un altre.

a) Basat en marcadors: utilitza imatges de l'entorn com referències anomenades "marcadors", els quals són reconeguts pel dispositiu. El dispositiu reconeix aquests marcadors en la imatge rebudes per una càmera, i amb això aconsegueix la informació de la posició del dispositiu per col·locar les imatges de realitat augmentada. El càlcul de posició es fa a través del anàlisi de la distància del marcador, segons la mida i l'angle en que es troba respecte a la càmera.

   La principal avantatge d'aquest sistema és la capacitat d'aconseguir conèixer la posició del dispositiu utilitzat tan sols amb la càmera. També té un ús computacional que necessita és més baix que el reconeixement per objectes. Per contra, el problema del sistema és la necessitat d'intervenir en el entorn posant un marcador.

b) Basat en reconeixement d'objectes: és el sistema més difícil d'implementar i el més car a nivell de cost computacional. A través de la càmera, reconeix objectes en particular i els compara amb una base de dades d'objectes segons la seva forma per descobrir de quin es tracta. Aquest sistema només requereix una càmera en el dispositiu. Per un altre banda, aquesta tecnologia requereix identificar uns patrons difícils de definir per diferenciar objectes.

c) Basats en la posició i l'orientació: és utilitzat en la majoria de les aplicacions actuals de realitat augmentada i no funciona en el reconeixement en les imatges de la càmera. Requereix d'un sistema de localització, com per exemple el GPS, i de sistemes que reconeguin l'orientació del dispositiu, com per exemple brúixoles digitals, acceleròmetres, etc. Marcant una sèrie de punts de referència en unes coordenades, el dispositiu aproxima si l'objecte està en el seu angle de visió i a quina distància.

4.3.1 Recursos disponibles

Hi ha diverses aplicacions i/o API's per treballar amb realitat augmentada. Algunes d'aquestes són:

- *Wikitude SDK*[19]: combina la potencia de realitat augmentada en una biblioteca. L'ús d'eines estàndard de la indústria de desenvolupament (HTML, Jacascript i CSS) fa que sigui fàcil

[19] Lloc web de Wikitude: http://www.wikitude.com/developer

d'integrar-la en un aplicació. Pot utilitzar-se en tant iOS com Android. Wikitude SDK està disponible de franc quan s'utilitza en projectes no comercials.

- *Layar*[20]: és una bilbioteca estàtica que implementa la funcionalitat bàsica de Layar en una aplicació, com la càrrega d'una capa i la presentació de la visió de realitat augmentada. Amb la plataforma mòbil Layar de realitat augmentada, els desenvolupadors de tercers poden crear diversos tipus d'experiències atractives.

- *Mixare*[21]: és un visor de realitat augmentada de codi obert, que es publica sota la llicència GPLv3. Funciona com una aplicació completament autònoma i està disponible també per al desenvolupament de les pròpies aplicacions.

- *Appunta*[22]: és una API que permet no només mostrar tota la informació geopositional per a l'usuari, també por crear noves formes de mostrar aquesta informació. És una API amb codi OS amb llicencia Apache 2.0, per tant codi obert.

- *Junaio*[23]: ofereix un servei gratuït que li permet utilitzar els seus coneixements i habilitat existents en el idiomes d'Internet per crear emocionats experiències de realitat augmentada.

4.3.2 Solució escollida

En un principi es volia prescindir de programar una solució pròpia per a la realitat augmentada i utilitzar una aplicació externa que hauria de ser instal·lada alhora que la aplicació desenvolupada. Això significava, en el cas de les app més comercials, tenir que pujar les dades als servidors (propis o externs) de les aplicacions.

Així doncs es va decidir crear una solució pròpia utilitzant una llibreria. Tot i que en primer lloc es va començar amb l'API de *Mixare*, degut al seu complicat i extens codi, es va decidir prescindir-ne i utilitzar una altre llibreria de codi lliure, però més simple. Així, es va optar per *Appunta*. És una API molt senzilla amb un codi molt simplificat. El seu ús es molt intuïtiu i compleix els objectius establerts. Si més no, al tractar-se d'una API de codi lliure (no comercial), hi ha mancances amb comparació amb altres de comercials, que requereixen permisos especials o la compra d'una llicencia.

[20] Lloc web de Layar: http://www.layar.com/
[21] Lloc web de Mixare: http://www.mixare.org/
[22] Lloc web d'Appunta: http://appunta.com/. Lloc web del codi: http://code.google.com/p/appunta/
[23] Lloc web de Junaio: http://www.junaio.com/

5. Disseny de l'aplicació

5.1 Android com a sistema operatiu

Com s'ha dit en l'apartat d'estat de la tecnologia, l'aplicació es dissenya per a dispositius mòbils amb sistema operatiu Android; però per quines versions d'aquest software? Cal tenir molt en compte aquest factor, doncs utilitzar l'aplicació en versions per la qual no està optimitzada por provocar el seu mal funcionament.

Les dades que aporta Android del seu ús indiquen quines són les versions més utilitzades actualment[24]. Cal pensar que les versions es van renovant i que el mercat de dispositius mòbils s'actualitza amb versions més noves d'Android. Tot i així s'ha de tenir en compte les versions més antigues que encara estan en un ampli ús perquè les versions més noves acostumen a tenir components o funcions no suportades per les versions més antigues d'Android.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.4%
2.2	Froyo	8	9.0%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	47.4%
3.1	Honeycomb	12	0.4%
3.2		13	1.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.1%
4.1	Jelly Bean	16	9.0%
4.2		17	1.2%

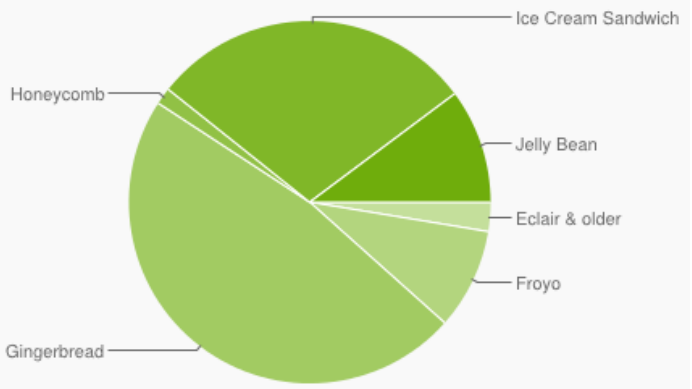


Figura 2. Gràfic i taula basats en el nombre de dispositius Android que han accedit a Google Play en un termini de 14 dies que finalitza el 13 de gener de 2013[24].

Tenint en compte les dades presentades, es decideix que l'aplicació serà compatible entre les versions *2.2 Froyo* (API 8) i la més nova *4.2 Jelly Bean* (API 17). Amb aquest rang de compatibilitat s'arriba al 97.4% de dispositius del mercat.

Cal esmentar que al agafar versions tan antigues implica que algunes funcions no poden ser del tot desenvolupades i que hi ha elements en la programació que han de ser diferents per versions majors o menors d'Android. En tot cas, es prioritza que l'aplicació pugui se utilitzada pel major número de dispositius possible.

[24] Informació extreta del lloc de desenvolupadors d'Android: http://developer.android.com/about/dashboards/index.html

5.2 Estructura bàsica d'una aplicació Android

La estructura d'una aplicació d'Android en el seu disseny és en forma de carpetes. Cada una d'aquestes conté els diferents elements que componen el projecte. Les bàsiques i més importants són:

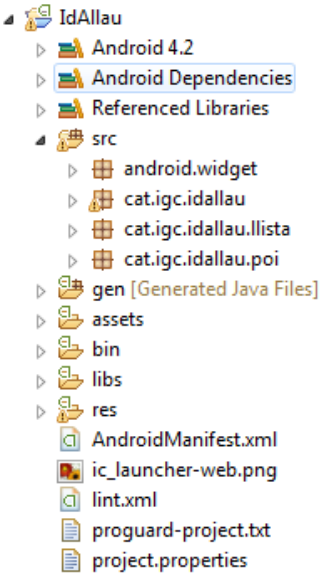


Figura 3. Estructura de IdAllau en l'entorn de programació.

- La carpeta ***src*** que conté tot el codi font de l'aplicació, el codi de la interfície gràfica, les classes auxiliars, etc.
- La carpeta ***libs*** que conté les llibreries auxiliars necessaries.
- La carpeta ***res*** que conté tots els fitxers de recursos necessaris pel projecte: imatges, vídeos, cadenes de text, estils, colors, etc. Al mateix temps, els diferents tipus de recursos es distribueixen en carpetes. Per un mateix recurs poden existir diferents carpetes si es volen distingir el seu ús depenent de la resolució de la pantalla (per imatges), idioma, orientació etc.
- El ***AndroidManifest*** conté la definició en un arxiu XML dels aspectes principals de l'aplicació.

Des del punt de vista d'un usuari una aplicació està formada per un conjunt de pantalles que permeten la interacció. En Android aquestes pantalles reben el nom de ***activities*** i són objectes que s'estenen a la classe **`Activity`**. Les *activities* tenen un elevat grau d'autonomia, i cadascuna s'encarrega dels components que conté.

Els components que es visualitzen en una *activity* s'estenen de la classe **`View`**. Alguns exemples de views són els `Buttons` i els `ViewText`. I aquells components que permeten agrupar a altres s'estenen a **`ViewGroup`**, com els `LinearLayout` o `RelativeLayout`. Els ***layouts*** són els elements que permeten distribuir els components amb una disposició determinada.

Al desenvolupar per Android s'anima a declarar les *activities*, amb els seus corresponents elements, en fitxers XML independents del codi. D'aquesta manera es guanya en flexibilitat i claritat.

Per comunicar-se entre elles les *activiteies* utilitzen els ***intents***. Bàsicament un *intent* ens permet indicar a Android que volem fer a través d'una altre *activity*. S'utilitzen per enviar missatges asíncrones dins d'una aplicació o entre varies aplicacions. Així, els *intents* permeten enviar i rebre informació entre *activities* o serveis.

Finalment, les aplicacions són empaquetades per a la seva distribució i instal·lació. El format per empaquetar tots els components és el **.apk**. El format és una variant del format **JAR** de Java; però bàsicament és un arxiu comprimit ZIP amb una extensió concreta per a Android. ***Android Play Store***[25], o simplement ***Google Play***, és la principal tenda de software en línia per distribuir les aplicacions (entre altres coses). Està desenvolupada per Google i aquells desenvolupadors que volen utilitzar-la per distribuir les seves aplicacions han de fer un únic pagament inicial per a registrar-se com a desenvolupadors (Google Play es queda una comissió si les aplicacions són de pagament).

[25] Android Play Store: https://play.google.com/store

5.3 Estructura de l'aplicació

L'aplicació s'anomena **IdAllau** i s'estructura en 6 blocs que duen a terme funcions concretes. Potser que aquests blocs estiguin formats per més d'una *activity*. Els diferents blocs (és a dir, les seves *activities*) es comunicaran a través dels *intents*. Tot seguit es presenten aquest sis blocs i les seves característiques.

5.3.1 Vista inicial

És la vista inicial de l'aplicació. Des d'aquí es pot accedir a gran part de les funcions de l'aplicació al pressionar els botons (Imatge 5.3). En cas que el GPS del dispositiu no estigui activat, es llança un avís per informar d'aquest impediment pel bon funcionament de l'aplicació (imatge 5.3) i si es vol activar obre les opcions de l'aplicació per activar-lo.[26] En cas que no s'activi o que encara no s'hagi posicionat el GPS, no es permet la entrada a la realitat augmentada perquè sense la posició de l'usuari aquesta no pot funcionar.

Figura 4. Captures del menú inicial i de l'avís sobre l'estat del GPS.[26]

Al iniciar-se l'aplicació per primer cop, i de forma interna, es crea la carpeta on són guardades les dades de l'aplicació dins del *External Storage* del dispositiu. També es llegeix l'arxiu amb les dades, s'analitza i es crea la base de dades (BD).

El format del arxiu que inclou les dades és un XML. S'ha escollit aquest format perquè es pot servir directament des de la BD geogràfiques del IGC, servides amb GeoServer[27]. Realment Geoserver serveix un GML (XML amb dades geogràfiques), però Android no està preparat per a llegir aquest format. En el anàlisi del fitxer es deixa de banda la informació geogràfica del XML i s'agafen les dades de posicionament dels atributs de cada registre.

La BD es crear amb la plataforma **SQLite** que Android incorpora. SQLite es caracteritza per ser de poca mida, no necessitar servidor, precisar poca configuració, ser transaccional i ser de codi lliure. En cas que no existeixi cap fitxer per crear la base de dades, s'avisa de que l'aplicació no pot funcionar sense les dades corresponents a les zones d'allaus.

[26] Captures fetes en un terminal Samsung Galaxy Mini. Veure Annex de dispositius per més informació.
[27] Lloc web de GeoServer: http://geoserver.org/display/GEOS/Welcome

En el cas que si existeixi aquest arxiu, i degut a la gran quantitat de dades que ha de processar internament per crear la base de dades, l'aplicació es mantingui inactiva, apareix un avís que indica que s'està treballant per crear la base de dades. Quan acaba, continua corrent l'aplicació de forma normal.

5.3.2 Mapa

Aquí l'usuari pot veure en el mapa la seva posició (sempre i quan hi hagi posicionament per part del GPS) i la localització de les diferents zones d'allaus.[28] Si en selecciona una, el símbol que l'identifica canvia de color i apareix un quadre translúcid a la part inferior de la pantalla amb la informació bàsica de la zona d'allaus. Des d'aquest quadre informatiu també es pot anar a la realitat augmentada (mantenint la zona d'allaus seleccionada) o anar a una vista ampliada de la seva informació.

Figura 5. Captura del mapa amb una zona d'allaus seleccionada.[28]

Els mapes dibuixats han d'estar situats en la carpeta (creada per l'aplicació al iniciar-se per primer cop) */osmdroid/*. És el lloc on la llibreria utilitzada va a buscar les fonts de dades cartogràfiques. En concret han d'estar empaquetats en un fitxer ZIP que inclou les *tiles* o retalls de la cartografia. Aquests *tiles* estan agrupats en una carpeta principal que necessariament ha de dir-se *Mapnik* i subcarpetes segons en el nivell de zoom i zona. Per construir aquets paquets cartogràfics s'ha utilitzat el programa MOBAC[29], servint-li cartografia pròpia des de GeoServer.

Depenent de la versió d'Android que tingui el dispositiu mòbil, les opcions del menú es presentaren de dos maneres diferents:

a) Versions inferiors a Android 3.0 (API 11): el menú es presenta al seleccionar el botó de menú del dispositiu i es mostren les tres opcions en un menú emergent a la part inferior de la pantalla.

b) Versions a partir d'Android 3.0 (API 11): les opcions apareixen en la part superior de la pantalla, a l'anomenada *action bar*, en forma d'icona.

Hi ha tres opcions disponibles en el menú:

1. "Navegar on sóc": opció pensada per ajudar a l'usuari a centrar la seva posició un altre cop en el centra del mapa si ha estat navegant per aquest.

[28] Captura d'un Samsung Tab 2 10.1. Veure Annex de dispositius per més informació.
[29] Mobil Atlas Creator. Lloc web: http://mobac.sourceforge.net/

2. "On sóc?": mostra en un diàleg les coordenades on es troba l'usuari

3. "Buscar POI's proxims": opció que serveix per a buscar les zones d'allaus (o punts d'interès, anomenats de forma abreviada POI's) pròximes a la posició de l'usuari. Primer es mostra un diàleg per indicar el radi de recerca. Després s'obre l'*activiy* amb la llista dels POI's.

En qualsevol de les tres opcions, si el GPS no està activat o no està establerta la posició, es mostra un avís informant de la manca de posicionament necessària per accedir a aquetes opcions..

5.3.3 Realitat augmentada

En aquesta *activity* l'usuari percep la imatge real del seu entorn a traves de la càmera del dispositiu i l'aplicació el que fa es disposar sobre aquesta imatge real les diferents zones d'allaus identificades per punts. Com passava en el mapa, al seleccionar-ne una el punt que la identifica canvia de color i apareix el quadre translúcid amb la seva informació i dos opcions. Igual que al mapa, des d'aquest quadre informatiu es pot accedir a més informació sobre la zona d'allau seleccionada o al mapa (mantenint aquesta zona seleccionada).

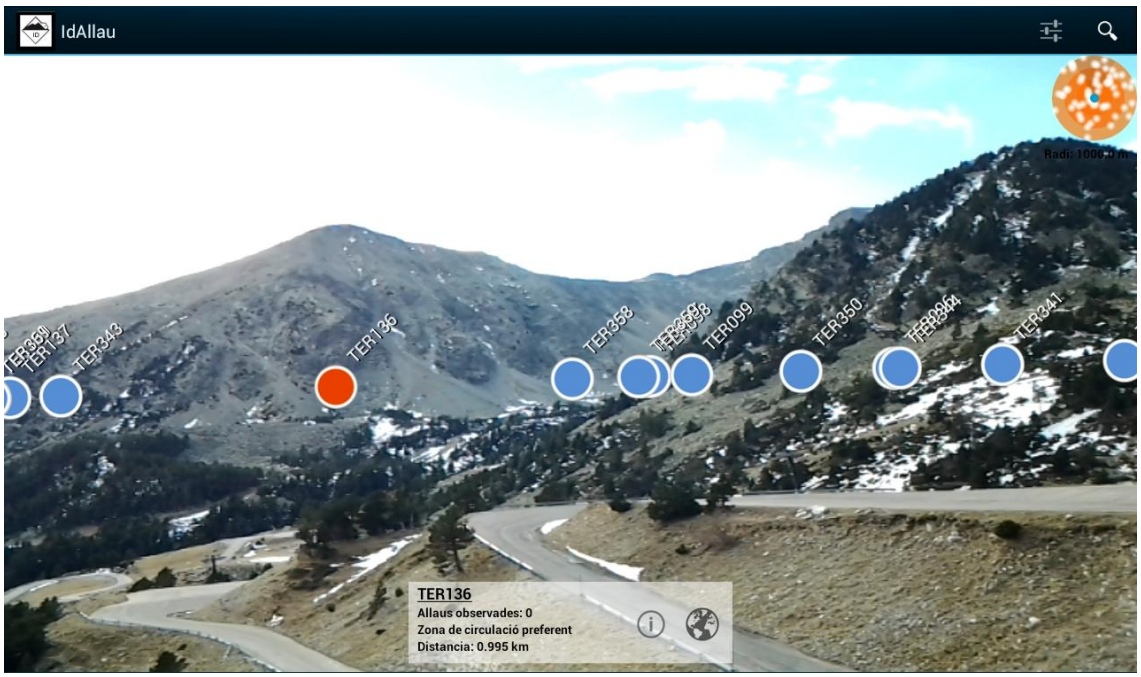


Figura 6. Captura de la vista de realitat augmentada amb una zona d'allaus seleccionada.[30]

El menú d'aquesta *activiy* consta de dos opcions[31]:

1. "Distància de visualització": en prémer aquesta opció apareix una barra lliscant que permet canviar el radi màxim fins el que l'aplicació mostra les diferents zones d'allaus. Si es prem una segona vegada (quan la barra està visible) la barra desapareix perquè no molesti en la visualització.

[30] Captura d'un Samsung Tab 2 10.1. Veure Annex de dispositius per més informació.
[31] Com s'ha explicat en l'apartat del mapa, aquí el comportament del menú també es diferencia depenent de la API d'Android.

2. "Buscar POI's proxims": com en el cas del mapa, també serveix per a buscar les zones pròximes a la posició de l'usuari. En aquest cas l'aplicació no pregunta al usuari el radi perquè agafa aquest valor de la distància de visualització seleccionada.

5.3.4 Informació del POI

En aquesta *activity* es mostra tota la informació disponible de la zona d'allau seleccionada. A més, hi ha un botó per poder visitar la fitxa de la zona d'allau ubicada en els servidors del IGC. Aquesta opció només estarà habilitada quan hi hagi connexió a Internet.[32]

5.3.5 Llista de POI's pròxims

Aquí es mostren en forma de llista totes les zones d'allaus que es troben en el radi especificat (tan venint del mapa com de la realitat augmentada). Al seleccionar-ne un, l'aplicació obre l'*activity* des d'on s'ha accedit al llistat (mapa o realitat augmentada).[32]

5.3.6 Càmera

L'aplicació obre l'aplicació per defecte per fer fotografies o mostra una llista amb totes les aplicacions que poden fer-ne. Un cop feta la fotografia, aquesta es guarda en la carpeta */photo/* de l'aplicació.

5.3.7 Informació de l'aplicació

En aquesta secció es mostren en format de llista diverses opcions:

- "Informació sobre l'aplicació": breu informació sobre l'aplicació; com la versió, desenvolupador, institucions participants, etc.

- "Actualització de la base de dades d'allaus": opció que serveix per actualitzar la BD.

- "Ajuda sobre l'aplicació": un recull d'informació que pot ajudar als usuaris a orientar-se i utilitzar l'aplicació. Està separada en les diferents activitats principals de l'aplicació.

- "Accés al web del IGC": en cas que hi hagi connexió a Internet disponible, l'aplicació obre un navegador web i navega al lloc del IGC.

[32] Per veure captures de pantalla d'aquestes vistes anar a l'Annex II.

5.4 Diagrames de funcionament

Els següents diagrames que es presenten són una forma més visual i intuïtiva d'entendre el funcionament de l'aplicació; les seves opcions i com es relacionen totes les *activities*.

5.4.1 Diagrama general

En aquest diagrama es mostra el funcionament global de l'aplicació. Cada bloc (distingits per colors) representen les principals funcions que porta a terme l'aplicació.

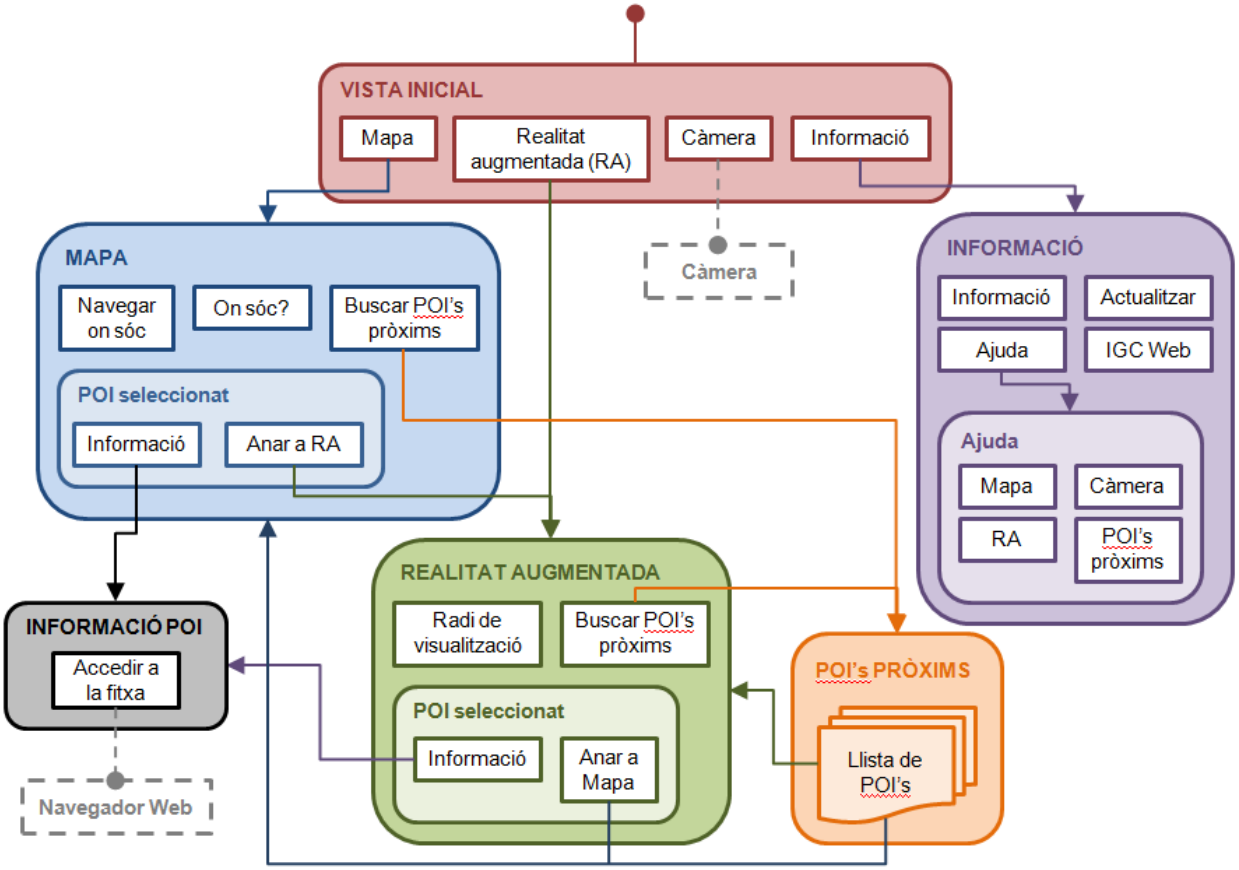


Figura 7. Diagrama general del funcionament de l'aplicació.

Esmentar que les línies discontinues mostren quan es surt de l'aplicació per accedir a altres aplicacions o serveis del dispositiu. A continuació es mostra amb més detall el funcionament de cada bloc o *activity*.

5.4.2 Vista inicial

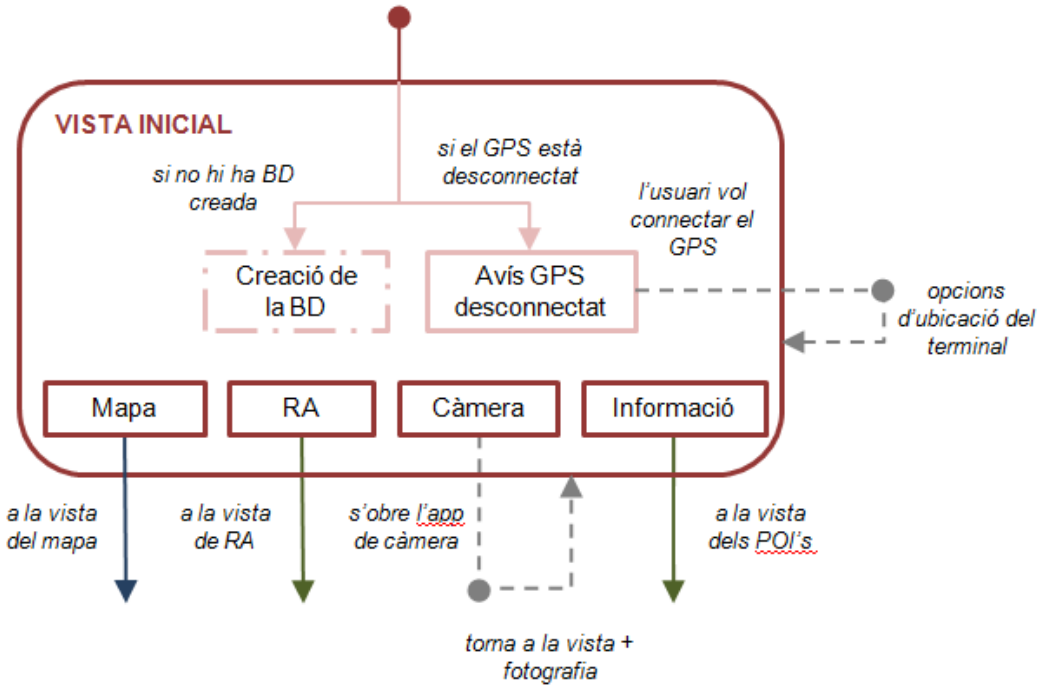


Figura 8. Diagrama del funcionament de la vista inicial.

Al iniciar-se l'aplicació en *background* (no a la vista de l'usuari) es crea la base de dades, si no està ja creada, i la carpeta on guardar les dades. Si el procés s'allarga molt, apareix un avís informant que l'aplicació està treballant en la creació de la BD.

Si el GPS no està activat surt l'avís; i si l'usuari vol activar-lo l'aplicació obre les opcions d''ubicació del dispositiu mòbil. Cal recordar que si el posicionament per GPS no està activat o no ha aconseguit una senyal no es pot accedir a la realitat augmentada (RA).

Els diferents botons duen a les seves respectives *activities* a excepció de la càmera. En aquest cas, el dispositiu obre l'aplicació que tingui per defecte per fer fotografies (o la escollida per l'usuari), fa la fotografia i l'aplicació la guardar en el directori corresponent.

5.4.3 Mapa

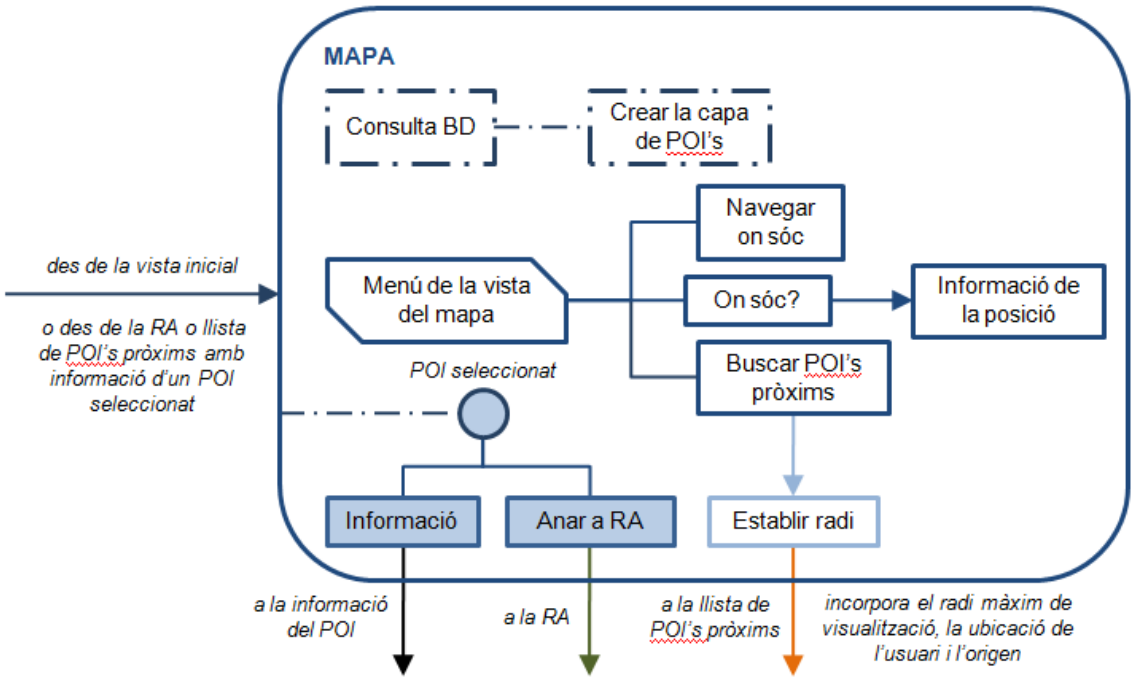


Figura 9. Diagrama del funcionament de la vista del mapa.

Al iniciar-se, el primer que fa l'*activity* és crear els POI's a partir de la base de dades de les zones d'allaus. Després dibuixar els POI sobre el mapa. A més, en el moment de dibuixar el mapa, l'aplicació busca en les seves carpetes els arxius per poder la cartografia i treballar off-line.

A aquesta vista si pot arribar de 3 llocs diferents: des de la vista inicial, des de la realitat augmentada o des de la llista de POI's pròxims (sempre que a aquesta última s'hi hagi arribat des del mateix mapa). En els dos últims casos, a la *activity* li arriba informació sobre un POI seleccionat, per tant aquest és seleccionant.

Quan es prem un POI, aquest queda seleccionat i es mostra el quadre amb la seva informació bàsica alhora que dues icones/botons. El primer porta a la vista d'informació del POI, i la segona a la realitat augmentada mantenint el POI seleccionat.

Les opcions del menú de la vista de mapa són tres. La primera, "Navegar on són", posiciona a l'usuari en el centre del mapa; la segona, "On sóc?", mostrà les coordenades de la posició de l'usuari. Finalment, la tercera, "Buscar POI's pròxims", obre la llista de POI's pròxims després de preguntar al usuari pel radi que en que vol buscar aquests.

5.4.4 Realitat augmentada

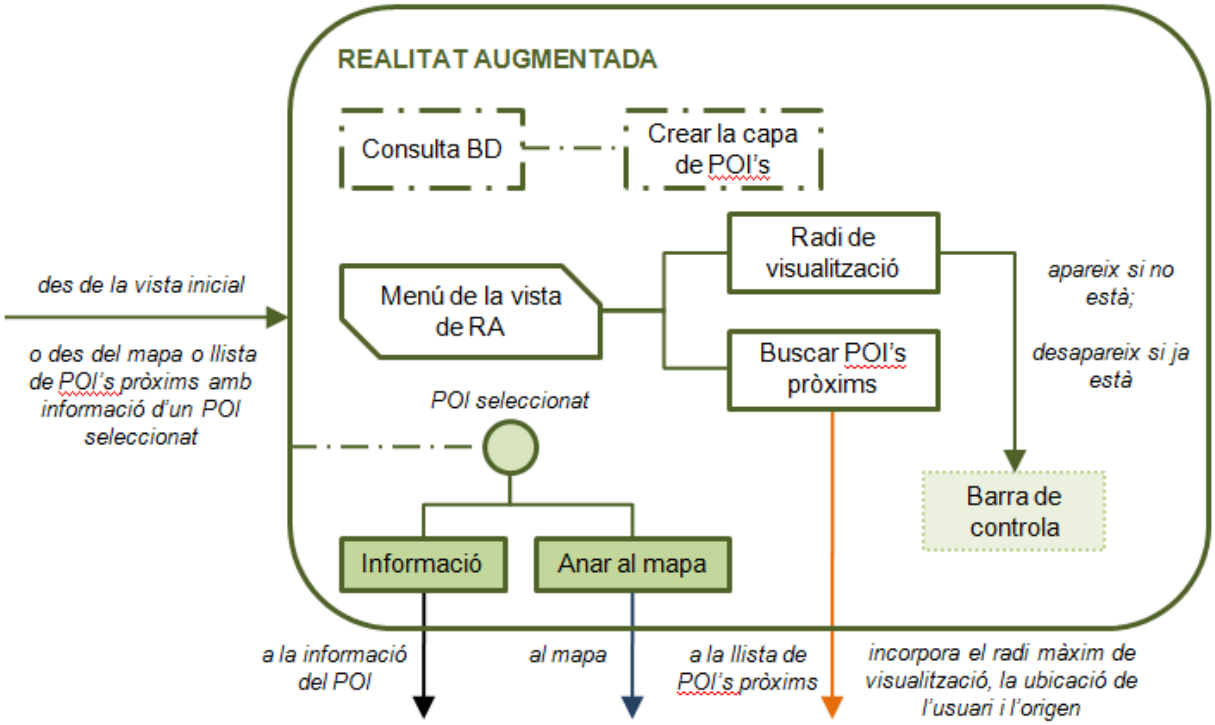


Figura 10. Diagrama del funcionament de la vista de la realitat augmentada.

Al iniciar-se també crea la capa de POI's a partir de la base de dades de les zones d'allaus. Després dibuixa els POI's en la realitat augmentada. Aquests POI's també es mostren en un radar que hi ha a la part superior dreta. Així l'usuari pot veure quines zones d'allau té al voltant.

Com en el cas del mapa, a aquesta *activity* s'hi pot arribar de 3 llocs diferents: des de la vista inicial, des del mapa o des de la llista de POI's pròxims (sempre que a aquesta última s'hi hagi arribat des de la mateixa realitat augmentada). Un altre cop, en el dos últims casos a la *activity* li arriba informació sobre un POI seleccionat, per tant és selecciona.

Aquesta activitat té un funcionament similar al mapa pel que fa a les opcions que té l'usuari. Quan es prem un POI, aquest queda seleccionat i es mostra el quadre amb la seva informació bàsica alhora que dues icones/botons. El primer porta a la vista d'informació del POI, i la segona al mapa mantenint el POI seleccionat.

Aquí el menú té dos opcions: la primera, "Radi de visualització" fa aparèixer (o desaparèixer si ja està visible) la barra lliscant que controla el radi màxim de visualització. La segona, "Buscar POI's pròxims", obre la llista de POI's pròxims agafant el valor actual de radi màxim de visualització.

5.4.5 Informació del POI

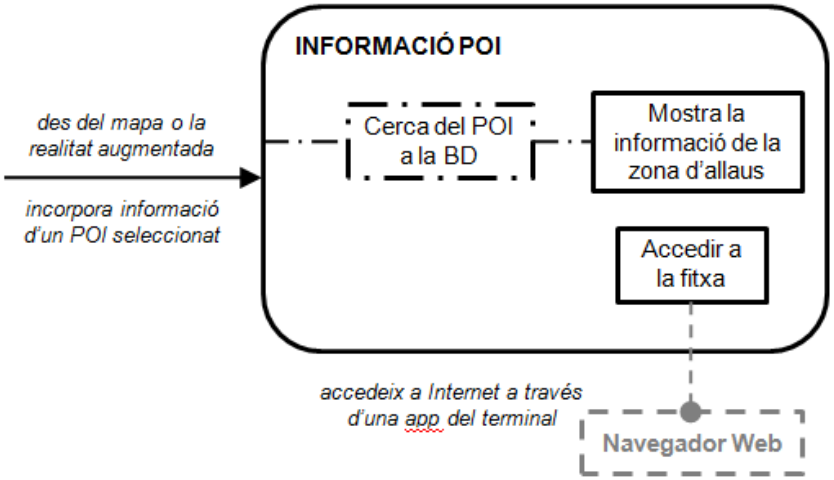


Figura 11. Diagrama de funcionament de la vista d'informació de la zona d'allaus seleccionada.

Es pot accedir a la vista d'informació de una POI seleccionat des del mapa o des de la realitat augmentada. En tots dos casos, l'activitat rebrà la informació del POI a buscar en la base de dades gracies a un *intent*. Després mostra tota la informació de la zona d'allaus. Si l'usuari té disponible una connexió a Internet, pot accedir a la fitxa que hi ha en els servidors del IGC. L'aplicació crida al navegador per defecte (o surt una llista dels disponibles) del dispositiu i accedeix a la fitxa.

5.4.6 Llista de POI's pròxims

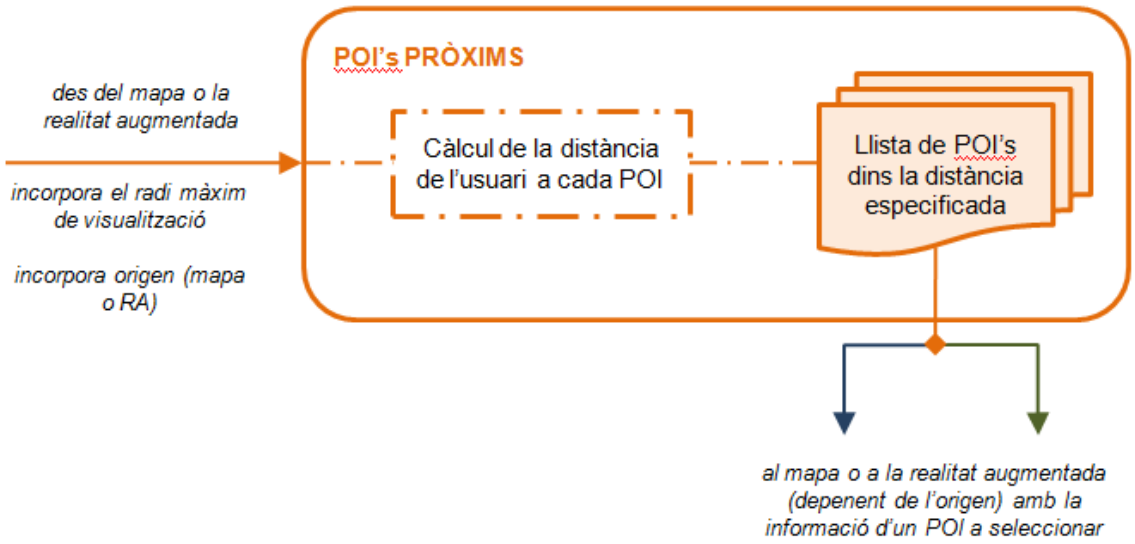


Figura 12. Diagrama de funcionament de la llista de POI's pròxims.

Quan s'accedeix a aquesta *activity*, tan sigui del mapa com de la realitat augmentada, és guarda l'origen per després retornar a l'*activity* d'origen. També es rep el radi màxim de visualització. Amb aquesta informació es calcula la distància entre l'usuari i cada una de les zones d'allaus en la base de dades. Si la distància es menor a la distància establerta s'incorpora a la llista. Un cop finalitzats els càlculs es mostra la llista amb una breu informació (que inclou la distància exacte).

Al seleccionar una zona d'allaus l'aplicació tornarà a aquella *activity* des d'on l'usuari a accedit a la llista. Tan si va al mapa com a la realitat augmentada, el POI estarà seleccionat.

5.4.7 Informació de l'aplicació

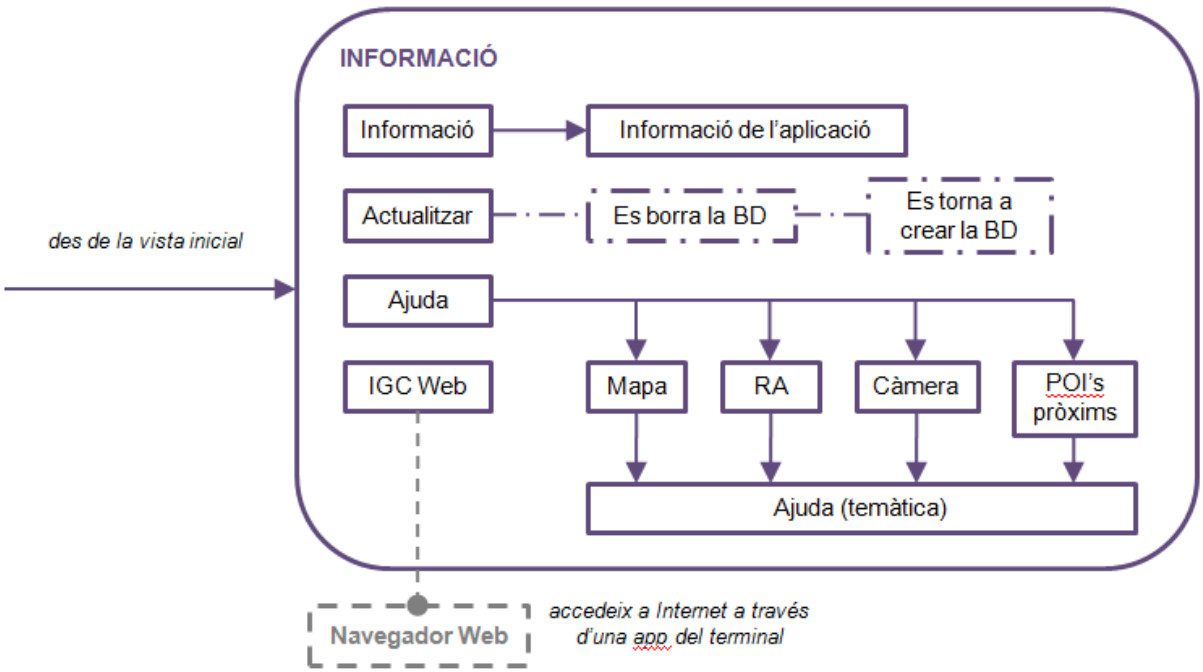


Figura 13. Diagrama de funcionament de la secció d'informació de l'aplicació.

S'accedeix des de la vista inicial i en format llista es mostren les següents opcions:

- "Informació sobre l'aplicació": breu informació sobre l'aplicació; com la versió, desenvolupador, institucions participants, etc.

- "Actualització de la base de dades d'allaus": borra la informació de la base de dades i torna a crear-la a partir d'un document XML existent en la carpeta de l'aplicació. Aquesta eina està pensada per actualitzar la base de dades. Per fer-ho caldrà carregar el nou fitxer XML a la carpeta de l'aplicació.

- "Ajuda sobre l'aplicació": un recull d'informació que pot ajudar als usuaris a orientar-se i utilitzar l'aplicació. En una nova llista es mostren les opcions d'ajuda per a cada una de les funcions principals de l'aplicació: mapa, realitat augmentada, càmera i llista de POI's pròxims. Al seleccionar un apareix l'ajuda sobre la funció i l'*activity* que la implementa.

- "Accés al web del IGC": en cas que hi hagi connexió a Internet disponible, l'aplicació obre el navegador web per defecte (o mostra una llista dels disponibles) del dispositiu i navega al lloc del IGC.

6. Programació

6.1 Entorn de desenvolupament

Totes les aplicacions per Android es programen en llenguatge Java i són executades en una maquina virtual especialment dissenyat per aquesta plataforma anomenada *Dalvik*. El nucli d'Android està basat en Linux 2.6.

Són necessàries les següents aplicacions i eines per a crear un entorn adequat per a desenvolupar aplicacions en Android:

- JDK de Java: es tracta d'un conjunt d'eines (programes i llibreries) que permet desenvolupar programes en llenguatge Java (compilar, executar, generar documentació, etc).[33]

- Eclipse: és un entorn de desenvolupament integrat de codi obert i lliure multi plataforma. Es extensible mitjançant mòduls i *plug-ins*. Està orientada inicialment a la creació d'aplicacions Java mitjançant les seves funcions com entorn de desenvolupament integrat (IDE en anglès).

  Actualment és una plataforma àmpliament utilitzada tant per desenvolupadors independents com en entorns empresarials, degut a que la majoria d'eines, mòduls i p*plug-ins* existens per aquest són lliures i gratuïts. Per tot això, els propis creadors d'Android recomanen com a IDE per a desenvolupar en el seu SO a Eclipse. De fet, és el únic que disposa oficialment d'un *plug-in* que facilita tot el procés de desenvolupament, el ADT.[34]

- SDK Android: es un kit de desenvolupament de software que conté el conjunt d'eines bàsiques que permet compilar i depurar aplicacions escites pel sistema operatiu Android; així com empaquetar i firmar l'aplicació al finalitzar-la.[35]

- ADT: de les seves inicial *Android Development Tools*, és un *plug-in* dissenyat especificament per a Eclipse. Aquesta eina incorpora a Eclipse els menús i opcions que faciliten les feines habituals en el desenvolupament d'aplicacions. Es descarrega mitjançant les opcions de Eclipse com a nou software.[36]

Si és cert que existeix la maquina virtual *Dalvik* que permet provar i depurar les aplicacions; de manera que es poden provar en diferents dispositius virtuals, utilitzar-lo requereix uns ordenadors d'alt rendiment i no sempre acaba de funcionar del tot bé si es vol comprovar el funcionament de elements específics com pot ser el posicionament o l'ús de la càmera. Per això, la majoria de desenvolupadors acostumen a fer la depuració directament en dispositius reals. És important al programar amb Android provar les aplicacions en el màxim de dispositius mòbils diferents (i amb diferents versions d'Android); per això, a l'**Annex IV** hi ha el recull de dispositius amb els quals s'ha fet la depuració i s'ha comprovat el funcionament i rendiment de l'aplicació.

[33] Lloc de JDK Java: http://www.oracle.com/technetwork/java/javase/downloads/index.html
[34] Lloc de Eclipse: http://www.eclipse.org
[35] Lloc del SDK d'Android: http://developer.android.com/sdk/index.html
[36] Lloc de descarrega del ADT: https://dl-ssl.google.com/android/eclipse/

6.2 AndroidManifest

En aquest arxiu es defineix els aspectes principals de l'aplicació. En primer lloc es determina el nom de l'aplicació (**IdAllau**) i la seva versió (en aquesta cas 1.0) per controlar les diferents actualitzacions. També s'ha de determinar la compatibilitat amb les versions Android. Es determina la versió mínima amb la qual es vol que l'aplicació funcioni i la versió màxima. És important determinar aquest rang de versions, doncs al programar pot haver-hi restriccions degut a la incompatibilitat amb alguna de les versions.

```
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17"/>
```

Codi 1. Definició de la versió mínima i màxima compatibles amb l'aplicació.

Si es vol determinar un estil al disseny de l'aplicació, aquí és on s'ha d'especificar. Si no Android utilitzarà la que tingui per defecte (variable depenent de la versió). S'han de determinar els permisos que l'aplicacó utilitzarà. Aquesta és la manera que té l'aplicació de tenir l'acceptació (en el moment de la instal·lació) per part de l'usuari de dur a terme determinades accions, com connectar-se a Internet o utilitzar el GPS.

Els permisos necessaris per l'aplicació són: accés a Internet, a la càmera (tant al harware quan es vol fer una fotografia com per part de l'aplicació al utilitzar la realitat augmentada), accés a la localització, permís per escriure al *external storage* i accés al estat de les comunicacions amb Internet.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Codi 2. Permisos de l'aplicació.

Per altre banda han de ser declarades totes les *activities* i elements que componen l'aplicació. En el moment de declarar cada una es pot determinar la seva orientació: horitzontal (*landscape*), vertical (*portrait*) o sense especificar (s'adaptarà a la posició del dispositiu en tot moment). De la mateixa manera que amb l'estil de l'aplicació, es pot determinar un estil concret a cada *ativty*. S'indica també a l'aplicació quina és la *activity* que serà cridada al iniciar l'aplicació.

6.3 Vista inicial

Al iniciar-se l'aplicació, la primera vista que apareix és aquesta. Només la forma una *activity* anomenada **`MenuInicial.java`**. En primer lloc l'*activity* comprova si la localització per GPS està activada. Si no és així mostra un avís que suggereix al usuari que l'activi. Si accepta, l'*activity* envia al usuari a través d'un *intent* al menú de configuració/ajustaments d'ubicació del terminal.

```
Intent actividad = new Intent(
        Settings.ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(actividad);
```

Codi .3 *Intent* que crida al menú de configuració o ajustaments d'ubicació.

Si és la primera vegada que s'executa l'aplicació es creen les carpetes que contindran la informació de l'aplicació. A més, si no està creada BD, l'*activity* inicia el procés de lectura i anàlisi dels arxius XML, anomenats **poi_zallau#.xml** (on # és el número d'arxiu, començant per 0), els quals contenen la informació de les zones d'allaus, i es crea la base dades. En cas que el procés de creació s'allargui molt, aquest passarà a fer-se en *background* i apareix un avís que indica que s'està creant la BD.

```
if (!BD.exists()){
	RelativeLayout layoutCarga = (RelativeLayout)findViewById(id.cargant);
	layoutCarga.setVisibility(View.VISIBLE);

	Animation anim = AnimationUtils.loadAnimation(MainActivity.this, R.anim.rotate);

	View loading = findViewById(R.id.loading);
	loading.startAnimation(anim);

	Task t = new Task();
	t.execute();
```

Codi 4. Si no existeix la BD apareix l'avís informant del procés de creació de la BD i s'avisa al mètode `Task`.

Inicialment l'avís apareix si les taules a la BD no està creada i s'executa el mètode **Task**[37] que indica el procés en *background* que s'ha de dur a terme i el que ha de succeir al finalitzar (codi 6.4). Per evitar problemes amb la memòria del dispositiu al analitzar els arxius XML, s'ha optat per partir-los. Cada fitxer no ha de tenir més de 15.000 registres.

```
class Task extends AsyncTask<Void, Void, Void> {
	String resposta = null;

	protected Void doInBackground(Void... arg0) {
		File sdcard = Environment.getExternalStorageDirectory();
		boolean existeix;

		//Per cada fitxer XML existent...
		for (int i = 0; i<99; i++){
			File arxiu = new File(sdcard,"IdAllau/poi_pilot" + i + ".xml");
			existeix = arxiu.exists();

			if (existeix) {
				CrearBDAllaus(arxiu);
				Log.i("DB", "Arxiu carregat " + i );
			} else if (!existeix){
				break;
			}
		}
		return null;
	}

	protected void onPostExecute(Void result) {
		super.onPostExecute(result);

		RelativeLayout layoutCarga = (RelativeLayout)findViewById(id.cargant);
		layoutCarga.setVisibility(View.GONE);

		View loading = MainActivity.this.findViewById(R.id.loading);
		loading.clearAnimation();
	}
}
```

Codi 5. Mètode que crea la BD en *backgraound*. Quan finalitza, fa desaparèixer l'avís.

[37] Més informació sobre el mètode `Task`: http://developer.android.com/guide/components/tasks-and-back-stack.html

Al finalitzar el procés de creació de la BD només cal fer desaparèixer l'avís perquè es mostri el menú normal.

L'anàlisi del arxiu és específic pel format que té. S'han de determinar els nodes on es troba la informació a capturar i es va creant una llista d'objectes **poi** (punts d'interès). Aquest objecte s'ha definit i se li han donat una conjunt de mètodes en el fitxer **poi.java**[38] (ubicat en el paquet `cat.igc.idallau.poi`).

```
void CrearBDAllaus(File arxiu){
	try {
		DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
		DocumentBuilder builder = factory.newDocumentBuilder();
		Document dom = builder.parse(arxiu);

		 //Ens posicionem sobre el node principal de l'arbre (<rss>)
		 Element root = dom.getDocumentElement();

		 //Localizem tots el elements <item>
		 NodeList items = root.getElementsByTagName("allaus: za_ct_wgs84");

		 //Recorrem la lista de POI's
		 for (int i = 0; i<items.getLength(); i++) {
		      //Obtenemos el POI actual
		      Poi poi = new Poi();
		     Node item = items.item(i);

		     //Obtenim la llista de dades del POI actual
		     NodeList dadesPOI = item.getChildNodes();

		     //Procesem cada dada del POI
		     for (int j=0; j<dadesPOI.getLength(); j++) {

		         Node attrib = dadesPOI.item(j);
		         String etiqueta = attrib.getNodeName();

		         if (etiqueta.equals("allaus:the_geom")) { //Saltem la geometria
		         } else if (etiqueta.equals("allaus:za_dis_COD")) {
		              poi.setCodi(attrib.getFirstChild().getNodeValue());
		         } else if (etiqueta.equals("allaus:za_dis_TIP")) {
		              poi.setTipuFoto(attrib.getFirstChild().getNodeValue());
		         } else if (etiqueta.equals("allaus:obs_cnt_Cn")) {
		              poi.setObs(attrib.getFirstChild().getNodeValue());
		         } else if (etiqueta.equals("allaus:X")) {
		              poi.setX(attrib.getFirstChild().getNodeValue());
		         } else if (etiqueta.equals("allaus:Y")) {
		              poi.setY(attrib.getFirstChild().getNodeValue());
		         } else if (etiqueta.equals("allaus:Z")) {
		              poi.setZ(attrib.getFirstChild().getNodeValue());
		         }
		     }
		     pois.add(poi);
		 }
		 BDSQLite();
	} catch (Exception ex){
		throw new RuntimeException(ex);
	}
}
```

Codi 6. Funció d'anàlisi de l'arxiu XML amb les dades de les zones d'allau.

Després, la informació es transmesa a la BD registre a registre mitjançant llenguatge SQL.

[38] Veure Annex III per veure el codi.

```
void BDSQLite (){
	final String  DATABASE_FILE_PATH = "/sdcard/IdAllau";
	final String  DATABASE_NAME = "ZonaAllau";

	//Creem la Base de Dades
	SQLiteDatabase db;
	db = SQLiteDatabase.openOrCreateDatabase(DATABASE_FILE_PATH + File.separator +
DATABASE_NAME, null);
	db.setVersion(1);

	//Creem la taula DBPoi
	try {
	    final String Crear_DBPoi =
	     "CREATE TABLE POIs (codi TEXT, tipus INTEGER, allaus_obs INTEGER, latitud
	       DOUBLE, longitud DOUBLE, altitud DOUBLE, id INTEGER)";
	    db.execSQL(Crear_DBPoi);
	} catch (SQLiteException e) {
		Log.i("BD", "la taula ja està creada", e);
	}

	//Llegim el XML
	ParseXML();

	for (int i = 0; i < pois.size(); i++) {
		Double lat = Double.valueOf(pois.get(i).CENTRE_Y);
		Double lon = Double.valueOf(pois.get(i).CENTRE_X);
		Double alt = Double.valueOf(pois.get(i).cota_z);
		Integer type = Integer.valueOf(pois.get(i).TIPUFOTO);

		Integer cod = i;

		db.execSQL("INSERT INTO POIs VALUES ('" + pois.get(i).CODI + "', " +
				type + ", " +
				pois.get(i).allaus_ob + ", " +
				lat + ", " + lon + ", " + alt + ", " +
				cod + ")");
		Log.i("BD", "BD Correctament Omplerta");
	}
	db.close();
}
```

Codi 7. Es crea la taula amb les dades de zones d'allaus a la base de dades.

La resta de codi inclou els mètodes per anar a les diferents *activities* al prémer els diferents botons i la construcció de la vista que veu l'usuari. Només esmentar la funció de fotografia que envia un *intent* a Android per obrir qualsevol aplicació que faci fotografies (s'obre directament si només hi ha una). Un cop feta la fotografia, aquesta es retornada a l'aplicació i es guarda a la carpeta de fotografies de l'aplicació (*/photo/*). El nom de la fotografia correspondrà a la data en que s'ha fet.

```
public void ObreCamera(View view){
	//Definim el nom de la fotografia
	String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
	Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
	File file = new File(Environment.getExternalStorageDirectory(),
			"IdAllau/photo/" + timeStamp + ".jpg");

	outputFileUri = Uri.fromFile(file);
	intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
	startActivityForResult(intent, TAKE_PICTURE);
}
```

Codi 8. Mètode per cridar a una aplicació que pugui fer fotografies. Un cop feta, la fotografia es guarda a la carpeta `/photo/` del directori de l'aplicació. El seu nom és l'hora en que s'ha fet la fotografia.

6.4 Visor del mapa

Per desenvolupar el visor de mapa s'ha utilitzat la llibreria (o API) anomenada **osmdroid**[39]. Aquesta llibreria proporciona el conjunt d'eines per a visualitzar i controlar la projecció cartogràfica i els POI's. Amb la corresponent llibreria integrada a l'aplicació es pot accedir a aquestes eines important-les a l'*activity*.

Al iniciar l'*activity*, anomenada **`VisorMapa.java`** especifiquem les característiques del mapa. En primer lloc definim quin és l'objecte on es dibuixarà. Aquest objecte és un *MapView*, objecte de *osmdorid*. Algunes de les característiques definides per aquest objecte són aquestes:

```
myOpenMapView = (MapView) findViewById(R.id.openmapview); //Definim l'objecte MapView

myOpenMapView.setBuiltInZoomControls(false); //Botons de Zoom desactivats

myMapController = myOpenMapView.getController(); //Definim controlador del MapView

myMapController.setZoom(16); //Determinem nivell de zoom inicial
myOpenMapView.setUseDataConnection(false); //Es desactiva la connexió a Internet com a
repositori de mapes

myOpenMapView.setMultiTouchControls(true); //Activem el MultiTouch
myOverlayLocation.setDrawAccuracyEnabled(true); //Dibuixem la pressició del GPS

myOverlayLocation.enableFollowLocation(); //Activem el seguiment de la nostra posició
```

Codi 9. Algunes opcions utilitzades per crear la vista de mapa.

Cal destacar que s'ha decidit no incloure l'escala (**`ScaleBarOverlay`** com objecte a *osmdroid*) perquè en versions d'Android superiors o iguals a 4 (API 14) no funciona. En alguns dispositius simplement aquest objecte no es dibuixa, però en el cas de Samsung Tab 2 10.1 amb Android 4.1 fa que l'aplicació deixi de funcionar.

L'*activity* implementa el control **`LocationListener`**. D'aquesta manera, es subscriu als possibles canvis que hi hagi en la ubicació del usuari i determinem cada quan volem que s'actualitzi; indicant-li el proveïdor (en aquest cas s'utilitzarà el GPS ja en muntanya la localització per xarxa Wifi és poc precisa o inexistent), el temps d'actualització i la distància.

```
myLocManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

myLocManager.requestLocationUpdates(provider, TEMPS_ACTUALITZAR, DIST_ACTUALITZAR, this);
```

Codi 10. Determinem els paràmetres d'actualització de la ubicació de l'usuari.

Per tant, quan succeeix aquest algun canvi en l'estat del servei podrem controlar-ho. En concret, si no està activat el GPS apareix un símbol a la part superior dreta per indicar-ho. En el cas que l'usuari canvi de posició, es redibuixarà el mapa per centrar la posició del usuari al seu centre cridant a la funció pròpia de *osmdroid*.

[39] Veure secció *4.2 Projecció cartogràfica* per més informació i enllaç a la llibreria i al codi.

```
@Override
public void onLocationChanged(Location location) {
	myOpenMapView.getOverlays().add(myOverlayLocation);
}
@Override
public void onProviderDisabled(String provider) {
	//La imatge que indica el GPS està desactiva és visible
	ImageView img=(ImageView)findViewById(R.id.gps_status);
	img.setVisibility(View.VISIBLE);
}
@Override
public void onProviderEnabled(String provider) {
	//La imatge que indica el GPS està desactiva no és visible
	ImageView img=(ImageView)findViewById(R.id.gps_status);
	img.setVisibility(View.INVISIBLE);
}
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
	// TODO Auto-generated method stub
}
```

Codi 11. Controlem els successos relacionats amb la disponibilitat del GPS.

Per simbolitzar zones d'allau en forma de punts amb els que es pugui interaccionar, l'*activity* crea un objecte propi de *osmodroid* anomenat **ItemizedIconOverlay** que actua com una capa més a dibuixar sobre el mapa. S'obre la BD i registre a registre s'afegeixen com objectes POI a una llista anomenada **OberlayItems**, objecte propi de *osmodroid*. Un cop afegits tots els POI, s'afegeixen a l'objecte ItemizedIconOverlay i aquest passa a ser dibuixat en el mapa.

```
POIOverlayItemArray = new ArrayList<OverlayItem>();

final String  DATABASE_FILE_PATH = "/sdcard/IdAllau";
final String  DATABASE_NAME = "ZonaAllau";
SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(DATABASE_FILE_PATH +   File.separator
+ DATABASE_NAME, null);

Cursor c = db.rawQuery("SELECT codi, tipus, allaus_obs, latitud, longitud, altitud FROM
POIs", null);

if (c.moveToFirst()) {
	do {
		//Creem un nou OverlayItem per afegir a la Array
		OverlayItem poi = new OverlayItem (c.getString(1), c.getString(0),
				c.getString(2), new GeoPoint(c.getDouble(3),
				c.getDouble(4), c.getDouble(5)));

		//Busquem el recurs de dibuix i l'apliquem al poi
		Drawable newMarker =
		this.getResources().getDrawable(R.drawable.poi_blau);
		poi.setMarker(newMarker);

		POIOverlayItemArray.add(poi);

	} while(c.moveToNext());

	ItemizedIconOverlay<OverlayItem> POIItemizedIconOverlay
		= new ItemizedIconOverlay<OverlayItem>(
			this, POIOverlayItemArray, myOnItemGestureListener);
	POIItemizedIconOverlay.getFocus();
	myOpenMapView.getOverlays().add(POIItemizedIconOverlay);
}
```

Codi 12. Codi per crear, a partir de la BD, una capa de punts que indiquen la posició de les zones d'allau.

En el cas que vinguem d'una altre *activity* i sigui necessari seleccionar de bon començament, s'aprofita que la BD està oberta per consultar de quina zona d'allaus es tracta, treure el seu identificador i seleccionar-lo a la capa que hem creat anteriorment. El següent tros de codi mostra com es gestiona les pulsacions sobre un POI (és a dir, quan és seleccionat):

```
OnItemGestureListener<OverlayItem> myOnItemGestureListener = new
OnItemGestureListener<OverlayItem>() {

	public boolean onItemLongPress(int arg0, OverlayItem arg1) {
		// TODO Auto-generated method stub
		return false;
	}

	 /** Informació que es mostra al fer un clic simple sobre el POI */
	public boolean onItemSingleTapUp(int index, OverlayItem item) {

		LinearLayout Info = (LinearLayout)findViewById(R.id.linearLayoutInfo);
		Info.setVisibility(View.VISIBLE);

		if (!(indexPoi == -1)) {
			PoiNoSelect(); //Deseleccionem el POI
		}

		indexPoi = index;
		item.setMarker(getResources().getDrawable(R.drawable.poi_ver));
		myOpenMapView.invalidate();

		TextView mapCodi = (TextView)findViewById(R.id.map_Codi);
		SpannableString content = new SpannableString(item.mTitle);
		content.setSpan(new UnderlineSpan(), 0, content.length(), 0);
		mapCodi.setText(content);

		TextView mapObs = (TextView)findViewById(R.id.map_Obs);
		mapObs.setText("Allaus observades: " + item.mDescription);

		TextView mapTipus = (TextView)findViewById(R.id.map_Tipus);
		Integer tipus = Integer.valueOf(item.mUid);

		if ( tipus == 1){
		      mapTipus.setText("Zona de circulació preferent");

		} else if (tipus == 2){
		      mapTipus.setText("Zona de difícil individualització");
		}

		ID = item.mTitle;

	return true;
	}
};
```

Codi 13. Mètode que gestiona les pulsacions sobre un POI.

En aquest procés el símbol del POI seleccionat canvia de color. En cas que hi hagués prèviament un altre seleccionat es desselecciona i torna al color original. També es fa visible el quadre d'informació del POI on apareix la seva informació i s'omplen els objectes *TextView* amb les dades.

En aquest quadre es pot fer desapareixia prement-hi a sobre. El quadre inclou també dos icones que al ser pressionats obren la vista d'informació del POI o es va a la realitat augmentada. En els dos casos, el *intent* que obre les *activities* du la informació sobre la zona d'allaus que està seleccionada.

```
public void ObreVisorInfoPoi (View view){
        Intent i = new Intent(view.getContext(), InfoPoi.class);
        i.putExtra("poiID", ID);
        view.getContext().startActivity(i);
}
```

Codi 14. Mètode que obre l'*activiy* d'informació sobre una zona d'allau seleccioanda. L'identificador d'aquesta zona és el **ID**, i s'envia a l'altre *activity*.

Les opcions en el menú d'aquesta vista són tres:

- "Navegar un sóc": simplement s'activa un altre com el **`enableFollowLocation()`** que es desactiva al navegar pel mapa i és navega amb el mètode **`animateTo`** a la posició actual de l'usuari.

- "On sóc?": s'obre un diàleg on s'especifica la posició actual de l'usuari.

- "Buscar POI's pròxims": també apareix un diàleg, en el que es pregunta el radi de cerca que es vol. Un cop especificat s'envia amb un *intent* a l'*activity* corresponent a la llista de POI's pròxims.

Aquestes opcions només funcionaran en el ben entès que hi hagi dades de la posició de l'usuari.

6.5 Realitat augmentada

Per a desenvolupar la realitat augmentada s'ha utilitzat la llibreria (API) d'**Appunta**[40]. Com en el cas del mapa, aquesta llibreria proporciona les eines per poder desenvolupar la realitat augmentada. Per fer-ho caldrà importar algunes eines de la llibreria i crar *activities* complementaries.

Per començar l'*activity* anomenada **`RealitatAugmentada.java`** ha d'implementar el control del canvi d'orientació, el control sobre els la selecció dels POI's i la barra lliscant que controla el radi de visualització.

Al crear l'*activity* primer definim la barra lliscant i el seu controlador.

```
SeekBar seekBarDist = (SeekBar)findViewById(R.id.seekBarDist);
seekBarDist.setOnSeekBarChangeListener(this);
```

Codi 15. Definició de la barra lliscant i del seu controlador.

Aquest tipus de barres per defecte Android només les permet posar horitzontals. Perquè aquesta barra no molestes als altres elements de la vista, s'ha decidit posar-la vertical; i per tant s'ha hagut de crear una barra lliscant pròpia anomenada **`VerticalSeekBar.java`**[41] [42]. Està definida en el paquet `android.widget` de l'aplicació i es crida en el *layout* de l'*activity*. Com a valor màxim que pot adquirir se li dóna 2000 (metres) i com a valor inicial al crear-se 500 (metres).

[40] Veure secció *4.4 Realitat augmentada* per més informació i enllaç a la llibreria i al codi.
[41] Veure codi al Annex III.
[42] Codi extret de la web: http://www.edumobile.org/android/android-apps/seek-bar-example/

```
<android.widget.VerticalSeekBar
        android:id="@+id/seekBarDist"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:maxWidth="20dp"
        android:max="2000"
        android:progress="500"
        android:visibility="gone"/>
```

Codi 16. Definició de la barra lliscant en l'arxiu **`ar.xml`**, que corresponent al *layout* de la realitat augmentada.

La resta de codi responsable de crear a la realitat augmentada és el que s'ha d'utilitzar segons les especificacions de l'**Appunta**.

```
private double MAX_DISTANCE = 0.5;

private EyeView ar;
private RadarView cv;
private CameraView camera;
private FrameLayout cameraFrame;
private OrientationManager compass;
private LocationManager locManager;

.
.
.

compass = new OrientationManager(this);
compass.setAxisMode(OrientationManager.MODE_AR);
compass.setOnOrientationChangeListener(this);
compass.startSensor(this);

ar = (EyeView) findViewById(R.id.augmentedView1);
cv = (RadarView) findViewById(R.id.radarView1);

ar.setMaxDistance(MAX_DISTANCE);
cv.setMaxDistance(MAX_DISTANCE);

ar.setOnPointPressedListener(this);
cv.setOnPointPressedListener(this);

PointRenderer arRenderer = new EyeViewRenderer(getResources(),
       R.drawable.taronja_50,R.drawable.blau_50);

points = PointsModel.getPoints(arRenderer);
cpoints = PointsModel.getPoints(new SimplePointRenderer());


locManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
            locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 60000,
50, this);

Location loc =
           locManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

lat = loc.getLatitude();
log = loc.getLongitude();
alt = loc.getAltitude();

ar.setPoints(points);

ar.setPosition(LocationFactory.createLocation(lat,log,alt));
ar.setOnPointPressedListener(this);
```

```
cv.setPoints(cpoints);
cv.setPosition(LocationFactory.createLocation(lat,log,alt));
cv.setRotableBackground(R.drawable.arrow);

cameraFrame = (FrameLayout) findViewById(R.id.cameraFrame);
camera = new CameraView(this);
cameraFrame.addView(camera);
```

Codi 17. Codi del OnCreate de l'*activity* de realitat augmentada.

El `OrientationManager` és el controlador de l'orientació del dispositiu i serveix per saber cap a quina direcció s'orienta el dispositiu i així saber quins elements virtuals ha de mostrar. El `EyeView` és l'objecte on després passarem els POI's perquè es visualitzin sobre el que capta la càmera. El `RadarView` és un objecte que funciona com un radar i permet veure els POI's qui hi ha al voltant. Als dos se'ls indica on s'han de dibuixar (en el *layout*); quina és la distància màxima de visualització actual; quin és el controlador de selecció de POI's, quina és la posició actual del dispositiu i els POI's a dibuixar.

Aquest últims és construeixen mitjançant una classe apart que s'exporta a l'*activity*. Aquesta classe s'anomena **`PointsModel.java`**[43] i es trova en el paquet `cat.igc.idallau8.poi`. Aquesta classe s'encarrega entra a la base de dades i crear un objecte amb tots les zones d'allaus i la seva informació. Després només cal passar aquest objecte al `EyeView` i al `RadarView`.

El control del GPS és igual que en l'*activity* del mapa. De la mateixa manera, el procés de selecció i els *intents* per anar a les altes *activies* també és igual.

Les opcions en el menú d'aquesta vista són dos:

- "Radi de visualització": fa visible o no la barra lliscant que controla el radi de visualització. Quan aquesta es mou canvia el valor vist de visualització pel al `EyeView` i al `RadarView` (variable amb el nom `MAX_DISTANCE` en el *Codi 6.14*)

- "Buscar POI's pròxims": actua igual que en el mapa, però en lloc de preguntar al usuari el radi, agafa el valor de barra lliscant.

6.6 Informació de les zones d'allaus

Es tracta d'una única *activity* (**`PoiInformacio.java`**) molt senzilla on es mostra la informació d'un POI, del qual arriba la ID a través del *intent* que la crida. Com en altres casos de selecció de un POI, s'entrà a la BD i es busca la zona d'allaus gràcies a una sentencia SQL. Amb els valors retornats s'omplen els objectes `TextView` amb la informació corresponent.

També té una funció que només està habilitada quan hi ha connexió a Internet. És un botó que al pressionar-lo crida alguna aplicació del dispositiu que obri direccions web (si només hi ha una, l'obrirà; si hi ha més d'una mostra una llista perquè l'usuari esculli una) per obrir la fitxa de la zona d'allau. Si no hi ha accés a Internet, apareix un petit avís temporal, anomenat `Toast`, avisant d'aquest eventualitat al iniciar-se l'*activity* i el botó queda deshabilitat.

[43] Veure el codi a l'Annex III.

```
btnUrl = (Button) findViewById(R.id.btnUrl);

if (!isOnline()){

       btnUrl.setEnabled(false);
       Toast.makeText(this, "Sense connexió a Internet no es pot accedir a la fitxa.",
                             Toast.LENGTH_SHORT).show();
} else {

       btnUrl.setOnClickListener(new OnClickListener(){
                public void onClick(View v) {

                     Intent browserIntent = new Intent(
                             Intent.ACTION_VIEW,
       Uri.parse("http://172.22.0.1:8080/siapc/submitZAllauListFormQCUD.do?reportZA=" + ID));
                       startActivity(browserIntent);
                }
         });
}
```

Codi 18. Mètode que controla el botó; tant la seva accessibilitat com la seva resposta.

6.7 Llista de POI's pròxims

L'*activity* (**`LlistaProximitat.java`**) rep amb l'*intent* 3 informacions: distància de recerca, posició (longitud i latitud) i l'origen del *intent*. Amb les dos primeres informacions, l'*activity* entra a la BD i calcula la distància entre la ubicació de l'usuari i la de cada registre i comprova si és o no menor a la recerca que es du a terme. Si entra dins d'aquests radi la zona d'allau s'afegeix a la llista que es mostra.

Per crear una llista personalitzada (amb diversos objectes com són imatges o línies de text) s'ha de crear un model de dades (**`PoiItem.java`**[44]) i un adaptador de llista propis (**`PoiListAdapter.java`**[44]). El primer és una classe que conté les variables necessàries per emmagatzemar els elements que més tard es volen introduir a la llista. El segon és una classe derivada d'un `BaseAdapter`, en el qual sobreescrivim els mètodes que content la classe abstracta, per passar-li després al `ListView` perquè la pugui construir.

Finalment queda el control de al prémer sobre un element de la llista. Segons l'origen de la consulta, es crida l'*activity* del mapa o de la realitat augmentada passant-li el codi de la zona d'allau que s'ha premut.

6.8 Informació de l'aplicació

Aquest bloc està format per 3 *activities* diferents. Dos són llista (**`InformacioApp.java`** i **`AjudaActivity.java`**) i la tercera només té la funció de mostra text (**`TextActivity.java`**). Com són llistes, requereixen model de dades (**`ItemOpcions.java`**[45]) i un adaptador de llista propis (**`ItemListAdapter.java`**[45]).

S'ha reutilitzat codi i per les dues llites només s'utilitza el mateix adaptador i model de dades. En aquest cas els elements de la llista es donen manualment un a un.

[44] Estan en el paquet `cat.igc.idallau8.llista`. Per a consultar-los mirar Annex III.
[45] Codi semblant al `PoiItem.java` i al `PoiListAdapter.java`. Mirar Annex III.

La primera llista (`InformacióApp.java`) mostra les opcions següents:

- Informació sobre l'aplicació: crida al `TextActivity.java` per mostra la informació de l'aplicació (breu explicació del context de creació, versió, autor i tutors).

- Actualitzar la base de dades: en el procés d'actualitzar la BD, l'aplicació borra la taula existent a la BD SQLite que s'ha creat en el `MenuInicial.java`. Llavors torna a analitzar els fitxers XML i crea de nou les taules amb els registres. Així doncs, per ara és necessari fer el canvi dels fitxers XML manualment dins la carpeta de l'aplicació.

- Ajuda: obre l'altre llista. En la llista `AjudaActivity.java` es mostren les funcions principals de l'aplicació. Al prémer una, l'*activity* condueix al `TextActivity.java` on es mostra la informació d'ajuda per a la funció seleccionada.

- Accedir al web del IGC: crida, com en l'*activity* d'informació del POI, a una aplicació del dispositiu per accedir al lloc web. Si no hi ha connexió a Internet l'aplicació avisa a l'usuari amb un `Toast`.

7. Resultats

L'aplicació que s'ha dissenyat i desenvolupat és totalment funcional; i en les proves de camp ha complert amb les expectatives plantejades en el inici del projecte. Com que l'aplicació serà per ús intern de l'IGC i els seus col·laboradors, no es planteja publicar-la al Google Play. Serà distribuïda internament l personal que la pugui necessitar. Alhora, s'oferiran els paquets de mapes dividits en zones (paquets d'un pes suficientment manejables) segons el personal ho requereixi, ja que carregar tota la cartografia a un nivell de detall important es podria esgotar la memòria del dispositiu.

Tot seguit hi ha una sèrie d'imatges que mostren el funcionament de l'aplicació. Algunes de les imatges corresponen a captures o fotografies de les versions de proba de l'aplicació, i per tant l'aspecte o funcions poden variar lleugerament de la versió final en el camp en els alguns dels dispositius[46] on s'ha provat.

Figura 14. Dispositius utilitzats en les proves de camp.

[46] Per conèixer les característiques dels dispositius utilitzats consulteu l'Annex IV.

Figura 15. Utilitzant l'aplicació en una Samsung Tab 2 10.1.

Figura 16. Tots els dispositius en la vista del mapa.

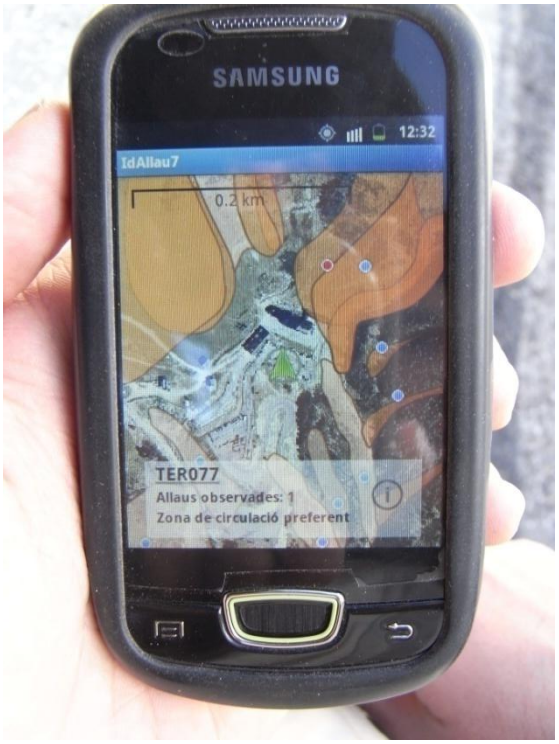


Figura 17. Fotografia del Samsung Galaxy Mini en la vista del mapa (versió no final de l'aplicació).

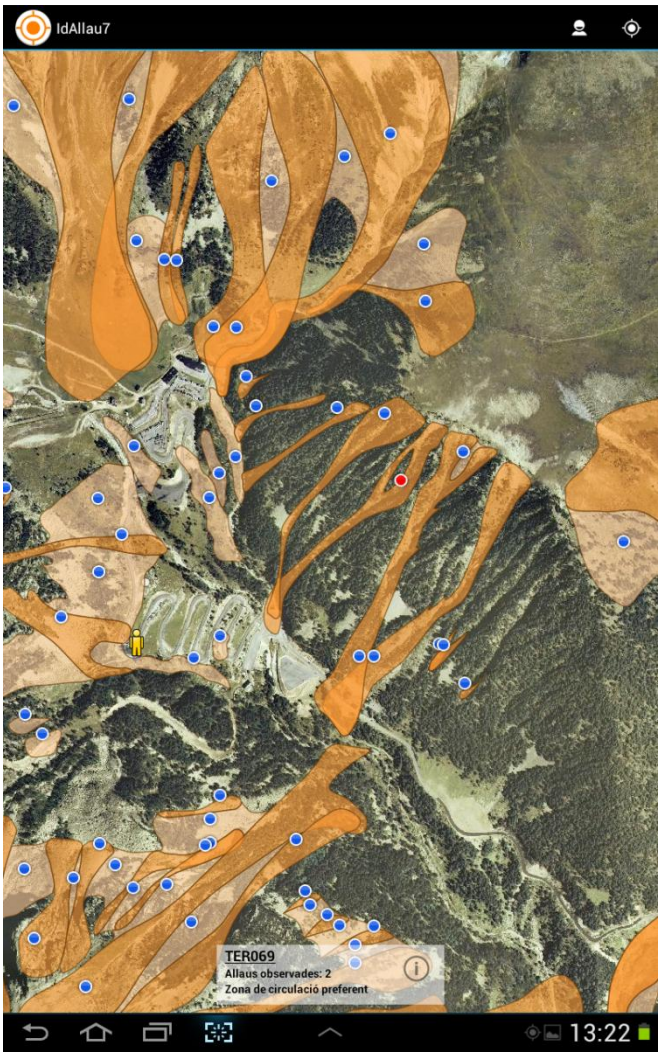


Figura 18. Captura de la vista de mapa amb una zona d'allaus seleccionada.
Captura feta amb una Samsung Tab 2 10.1.

Figura 19. Utilitzant la realitat augmentada en un Samsung S III mini (versió no final de l'aplicació).

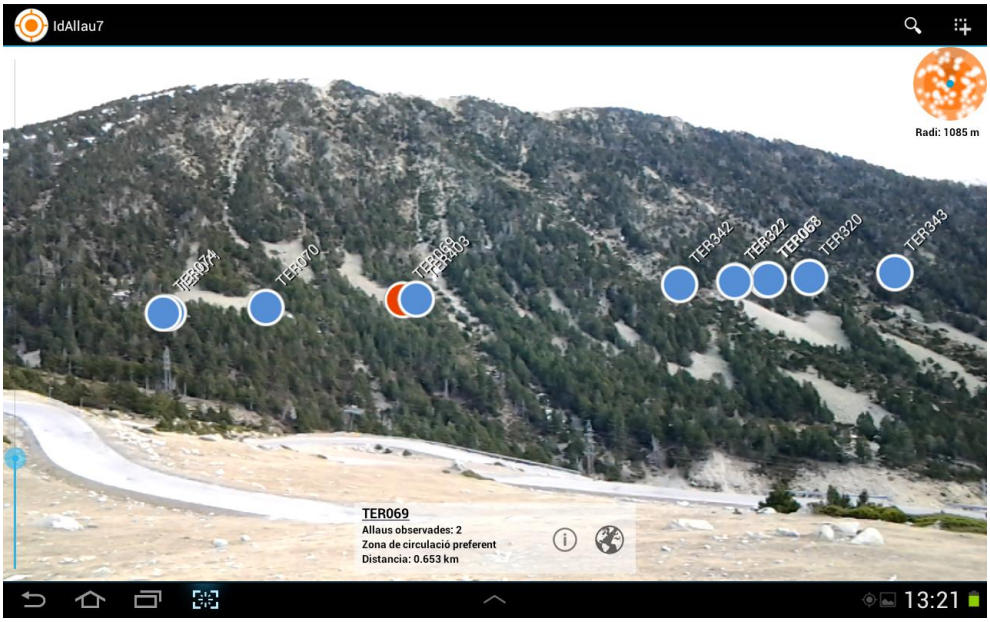


Figura 20. Captura de la vista de realitat augmentada amb una zona d'allau seleccionada i amb la barra de lliscament pel control del radi de visualització visible. Captura feta amb una Samsung Tab 2 10.1 (versió no final de l'aplicació).

8. Conclusions

Si en un principi es partia d'un coneixement relatiu en programació (principis bàsics en JavaScript i Visual Basic), la programació en Java s'ha fet assumible principalment gràcies a dos factors:

- Per una banda, a les facilitats que proporciona Android per a programar en el seu sistema operatiu; tant l'assistència en l'entorn de treball per part del SDK com per la informació que s'ofereix en el lloc oficial de desenvolupadors d'Android.

- Al treballar amb codi lliure, la informació a la xarxa, per part d'altres desenvolupadors més o menys avançats, és molt extensa. Hi ha nombrosos manuals d'iniciació en programació com fòrums de programadors.

Ha estat encertat escollir desenvolupar l'aplicació en Android i en llibreries de codi lliure. Si més no, les dificultats sorgides en el procés no han estat poques. Desenvolupar les funcions inicials plantejades i les que han anat apareixen durant el desenvolupament han implicat una revisió constant de bibliografia i de plantejament de l'estructura i funcionament de l'aplicació. I tot i ser molts els beneficis d'utilitzar software lliure, també implica que no hi ha assistència ni garanties. Les llibreries utilitzades, tot i portar unes mínimes instruccions d'ús, s'han tingut que adaptar a les funcions i necessitats requerides per introduir-les dins del codi de l'aplicació. Amb tot, l'elecció de desenvolupar l'aplicació integrament en codi lliure (quan hi havia altres opcions que no necessitaven de la integració en l'aplicació, si no l'ús extern d'altres aplicacions) ha estat un encert.

Les funcions bàsiques plantejades al inici del desenvolupament s'han assolit; i fins i tot s'han pogut millor. Per exemple, en un inici es volia utilitzar una aplicació externa per a la realitat augmentada, però al final s'ha pogut incorporar dins l'aplicació gràcies a una llibreria. En un primer moment tampoc era clar que es pogués desenvolupar el visor de mapes offline, doncs la majoria de llibreries utilitzen principalment les fonts d'Internet per dibuixar els mapes i el cache que queda en la memòria del dispositiu.

Altres funcions secundaries o i aquelles que han anat apareixen durant el disseny i programació també s'han pogut integrar en aquesta primera versió de l'aplicació **IdAllau**. Tot i així, el funcionament òptim de l'aplicació sempre estarà lligat al software (versió d'Android) i al hardware (característiques de processament, sensors, etc.) del dispositiu mòbil on s'utilitzi. En aquests aspecte, s'ha vist que en dispositius antics, la realitat augmentada no funciona degut les especificacions de hardware que requereix l'ús de l'API d'**Appunta**.

Des d'un primer moment l'aplicació s'ha plantejat com un prototip per part del IGC i del màster, doncs fins ara ningun dels dos havia dut a terme un projecte d'aquest tipus per a dispositius mòbils. Per part dels dos ha estat una experiència positiva, doncs la tecnologia i la societat estan decantant-se cada cops més cap aquest mercat. De moment l'aplicació serà utilitzada només per tècnics i personal del IGC, però de segur que en el futur s'obren nous projectes per aprofitar les grans capacitats que ofereixen els dispositius mòbils i els softwares que els acompanyen, que estan en constant evolució.

Com en qualsevol projecte tecnològic, finalitzar una primera versió d'aquesta aplicació només és un pas més en el seu desenvolupament. El constant canvi en els dispositius mòbils, les actualitzacions de software i llibreries o API's i l'aparició de nous recursos fa que s'hagi d'estat en constant revisió i actualització per no quedar-se enrere i obsoleta. Per tot això, i per millorar encara més les funcions de l'aplicació, de seguit són plantejades unes quantes propostes de millora:

- Solucionar els problemes que té el funcionament de la realitat augmentada en alguns dispositius; ja sigui fent canvis en l'API d'Appunta o canviat la llibreria.

- Millorar la selecció de les zones d'allau en la realitat augmentada. Per una banda *Appunta* no impedeix seleccionar POI's que no estan dins del rang de visualització. Per un altre, si hi ha POI's superposats, hauria d'aparèixer una llista amb tots ells per facilitar-ne la selecció.

- Programar una càmera pròpia per poder fer fotografies georeferenciades. Amb aquesta funció s'aconseguiria tenir fotografies de camp amb informació de la localització i la direcció de com s'han pres. Així, la interpretació de les mateixes seria més fàcil.

- Tenir varies capes de cartografia per mostrar en el mapa (offline).

- El procés d'actualització de la base de dades s'ha de completar, fent que a través d'Internet l'usuari pugui descarregar els arxius amb les dades (degut a la gran quantitat de dades que s'han d'analitzar, es descarta l'anàlisi directa dels fitxers a través d'Internet).

Alguns d'aquetes millores impliquen modificacions internes de les llibreries; aspecte gens fàcil alhora de programar. Si més no, al treballar amb llibreries i sistema operatiu amb llicencies de codi lliure, les possibilitats de aplicar canvis o de treballar amb altres desenvolupadors per a millorar les seves funcions obre moltes oportunitats en el futur.

8. Bibliografia

8.1 Referències bibliogràfiques

ARANAZ TUDELA, Jaime. "Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma Android de Google". Tutora: Celeste Campo Vázquez. Proyecto de fin de carrera. Universidad Carlos III de Madrid, Escuela Politécnica Superior, 2009.

BOVER ARGELAGA, Alan. "Aplicación de gestión de información geolocalizada en Android". Director: Manuel Medina Llinàs. Projecte de final de carrera. Universitat Politècnica de Catalunya, Facultat d'Informatica de Barcelona, 2010.

de la CALLE ALONSO, M. i PULIDO GALÁN, F. "Ralidad aumentada con servicios OGC implementada con librerías de fuentes abiertas". En: V Jornadas de SIG Libre. Universitat de Girona, Servei de Sistemes d'Informació Geogràfica i Teledetecció.

GESSLER, Fernando. "TFC: GeoTools-Android. Herramientas geográficas para Android." Universitat Oberta de Catalunya, 2012

GONZALO SOTO ABOAL, Luis. "Aplicación para dispositivos móviles Android: Guía de los edificios de la Universidad Politécnica de Cartagena". Director: Juan Ángel Franco. Proyecto de final de carrera. Universidad Politécnica de Cartagena, Escuela Técnica Superior de Ingeniería de Telecomunicación, 2011.

"JAVAJEFF" FRIESEN, Jeff. *Java para desarollo Android*. Traductor: Dans Álvarez, Pedro. 1a ed. Madrid: Ediciones Anaya, 2011. 671p.

LÓPEZ POMBO, Héctor. "Análisis y desarollo de sistemas de realidad augmentada". Director: Antonio Navaroo Martín. Proyecto de fin de máster. Universidad Complutense de Madrid, Facultad de Informática, Máster en Investigación en Informática, 2010.

PEDRIZA REBOLLO, A. y CITORES FERNÁNDEZ, M. "Geoserver y realidad augmentada. Extensión para la publicación de los repositorios cartográficos en los navegadores de RA". En: VI Jornadas de SIG Libre. Universitat de Girona, Servei de Sistemes d'Informació Geogràfica i Teledetecció.

8.2 Referències Web

Android: http://www.android.com/

Android Developer: http://developer.android.com/index.html

Android Enthusiasts: http://android.stackexchange.com/

Appunta: http://appunta.com/

Gartner Smart Phone Marketshare: http://www.gartner.com/newsroom/id/2237315

Google Code: code.google.com/

Institut Geològic de Catalunya: http://www.igc.cat/web/ca/index.php

Institut Cartogràfic de Catalunya: http://www.icc.cat/

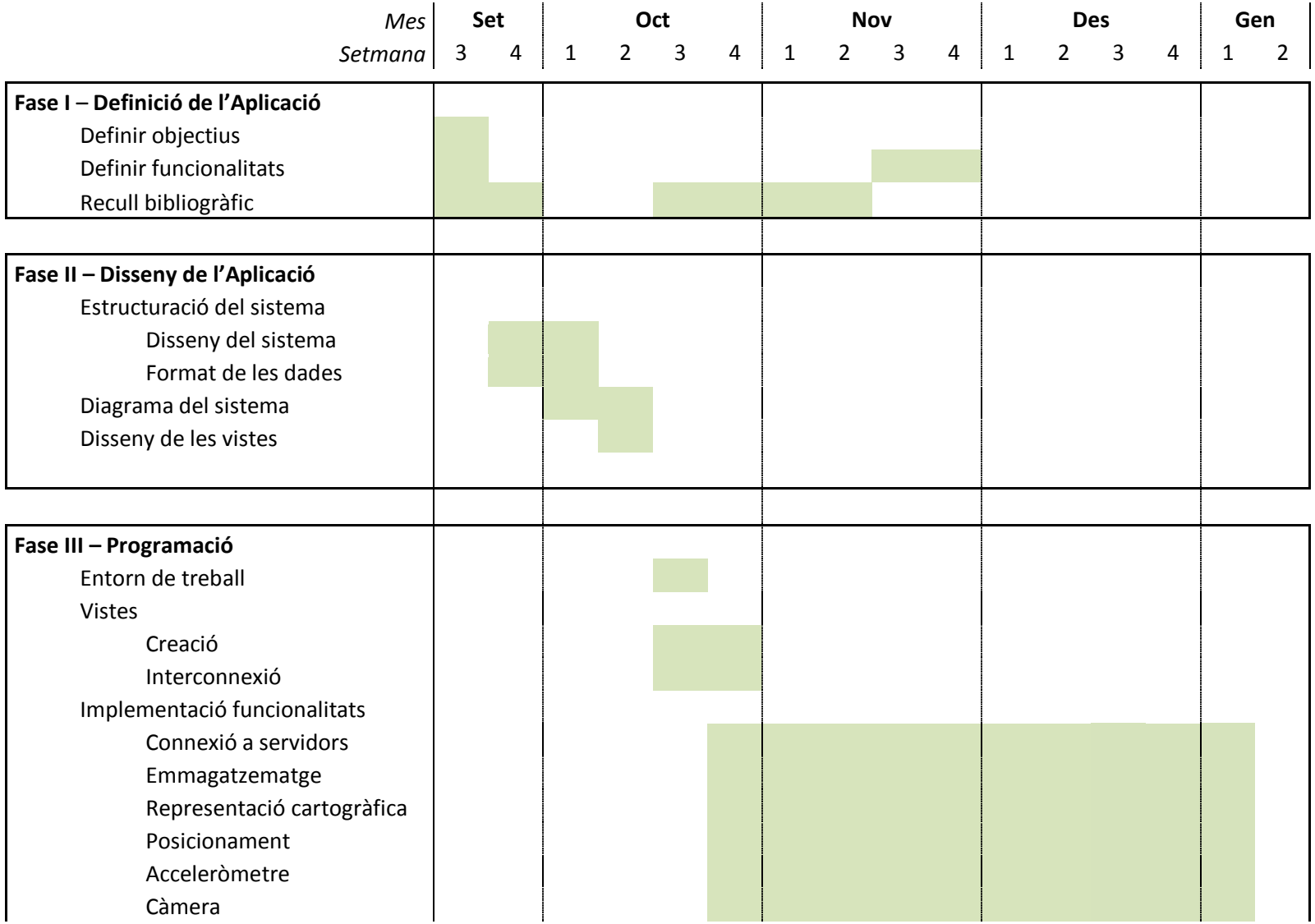
Manuals d'Android:

http://www.sgoliver.net/blog/
http://www.nosinmiubuntu.com/
http://www.edumobile.org/android/
http://www.androidcurso.com/
http://www.javaya.com.ar/androidya/index.php?inicio
http://androcode.es/la-androteca/

Stackoverflow: http://stackoverflow.com/

Wikipedia: http://www.wikipedia.org/

ANNEX I - Pla de treball



Realitat augmentada					
Fase IV – Proves funcionals					
Proves del funcionament					
Prova en camp					
Fase V – Publicació					
Publicació de l'aplicació					

ANNEX II - Imatges i captures

Totes les captures que hi ha a continuació estan fetes amb un Samsung Galaxy Mini amb Android 2.3.5.

Figura II.A Captura del menú d'informació.

Figura II.B Captura de la vista informació de l'aplicació.

Figura II.C Captura del menú d'ajuda.

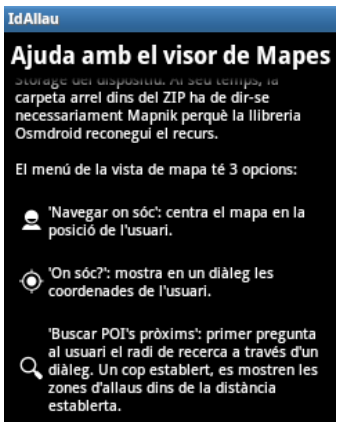


Figura II.D Captura de la vista d'ajuda del visor de mapes.

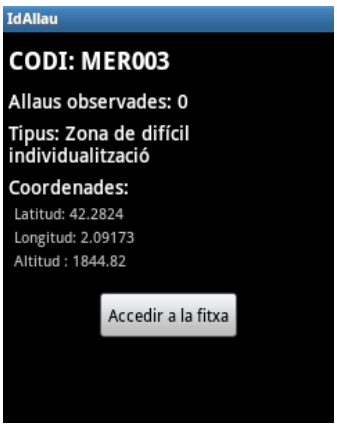


Figura II.E Captura de la informació d'una zona d'allaus seleccionada.

Figura II.F Captura on és veu el resultat de la cerca de POI's.

ANNEX III - Codi

III.A PoiItem.java

```
public class PoiItem {
        protected long id;
        protected String codi, tipus;
        protected Integer obser;
        protected Double dist;

        public PoiItem() {
            this.codi = "";
            this.obser = null;
            this.tipus = "";
            this.dist = null;
        }

        public PoiItem(long id, String codi, int observades, String tipus, Double distancia) {
            this.id = id;
            this.codi = codi;
            this.obser = observades;
            this.tipus = tipus;
            this.dist = distancia;
        }

        public long getId() {
            return id;
        }

        public void setId(long id) {
            this.id = id;
        }

        public String getCodi() {
            return codi;
        }

        public void setCodi(String codi) {
            this.codi = codi;
        }

        public Integer getObservacions() {
            return obser;
        }

        public void setObservacions (int observacions) {
            this.obser = observacions;
        }

        public String getTipus() {
            return tipus;
        }

        public void setTipus (String tipus) {
            this.tipus = tipus;
        }

        public Double getDistancia() {
            return dist;
        }

        public void setDistancia (Double distancia) {
            this.dist = distancia;
        }
}
```

III.B VerticalSeekBar.java

```
import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.MotionEvent;

public class VerticalSeekBar extends SeekBar {

    public VerticalSeekBar(Context context) {
        super(context);
    }

    public VerticalSeekBar(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public VerticalSeekBar(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(h, w, oldh, oldw);
    }

    @Override
    protected synchronized void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(heightMeasureSpec, widthMeasureSpec);
        setMeasuredDimension(getMeasuredHeight(), getMeasuredWidth());
    }

    protected void onDraw(Canvas c) {
        c.rotate(-90);
        c.translate(-getHeight(), 0);

        super.onDraw(c);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (!isEnabled()) {
            return false;
        }

        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
            case MotionEvent.ACTION_MOVE:
            case MotionEvent.ACTION_UP:
                setProgress(getMax() - (int) (getMax() * event.getY() / getHeight()));
                onSizeChanged(getWidth(), getHeight(), 0, 0);
                break;

            case MotionEvent.ACTION_CANCEL:
                break;
        }
        return true;
    }
}
```

III.C PointsModel.java

```
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import com.appunta.android.location.LocationFactory;
import com.appunta.android.point.Point;
import com.appunta.android.point.impl.SimplePoint;
import com.appunta.android.point.renderer.PointRenderer;

public class PointsModel {

	/** Crear els Poi's de a Realitat Augmentada **/
	public static List<Point> getPoints(PointRenderer renderer) {
		List<Point> points = new ArrayList<Point>();

		//Obrim la BASE DE DADES ZONAALLAU
		final String  DATABASE_FILE_PATH = "/sdcard/IdAllau";
		final String  DATABASE_NAME = "ZonaAllau";
		SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(DATABASE_FILE_PATH +
File.separator + DATABASE_NAME, null);
		Cursor c = db.rawQuery("SELECT codi, latitud, longitud, altitud FROM POIs",
null);
		//Integer num_rows = c.getCount();
		//Mirem que hi ha almenys un registre
		Integer i = 0;
		if (c.moveToFirst()) {
		     //Recorrem el cursor fins que no hi ha més registres
		     do {
		        points.add(new
SimplePoint(i,LocationFactory.createLocation(c.getDouble(1),c.getDouble(2),c.getDouble(3)),ren
derer,c.getString(0)));
		        i = i + 1;
		     } while(c.moveToNext());
		}
		c.close();
		db.close();
		return points;
	}
}
```

III.D PoiItem.java

```
public class PoiItem {
	protected long id;
	protected String codi, tipus;
	protected Integer obser;
	protected Double dist;

	public PoiItem() {
	    this.codi = "";
	    this.obser = null;
	    this.tipus = "";
	    this.dist = null;
	}

	public PoiItem(long id, String codi, int observades, String tipus, Double distancia) {
	    this.id = id;
	    this.codi = codi;
	    this.obser = observades;
	    this.tipus = tipus;
	    this.dist = distancia;
	}

	public long getId() {
	    return id;
```

```
        }

        public void setId(long id) {
            this.id = id;
        }

        public String getCodi() {
            return codi;
        }

        public void setCodi(String codi) {
            this.codi = codi;
        }

        public Integer getObservacions() {
            return obser;
        }

        public void setObservacions (int observacions) {
            this.obser = observacions;
        }

        public String getTipus() {
            return tipus;
        }

        public void setTipus (String tipus) {
            this.tipus = tipus;
        }

        public Double getDistancia() {
            return dist;
        }

        public void setDistancia (Double distancia) {
            this.dist = distancia;
        }
}
```

III.E PoiListAdapter.java

```
import java.util.ArrayList;
import cat.igc.idallau9.R;
import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

public class PoiListAdapter extends BaseAdapter {

        protected Activity activity;
        protected ArrayList<PoiItem> pois;

        public PoiListAdapter(Activity activity, ArrayList<PoiItem> pois) {
            this.activity = activity;
            this.pois = pois;
        }

        public int getCount() {
               return pois.size();
        }

        public Object getItem(int position) {
                return pois.get(position);
        }
```

```
	public long getItemId(int position) {
		return pois.get(position).getId();
	}

	public View getView(int position, View convertView, ViewGroup parent) {
		View vi=convertView;

		if(convertView == null) {
			LayoutInflater inflater = (LayoutInflater)
activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
			vi = inflater.inflate(R.layout.proxim_poi, null);
		}

		PoiItem item = pois.get(position);

		TextView nom = (TextView) vi.findViewById(R.id.codi);
		nom.setText(item.getCodi());

		TextView observa = (TextView) vi.findViewById(R.id.obs);
		observa.setText("Allaus observades: " + String.valueOf(item.getObservacions()));

		TextView tipus = (TextView) vi.findViewById(R.id.tipus);
		tipus.setText(item.getTipus());

		TextView dist = (TextView) vi.findViewById(R.id.distan);
		dist.setText("Distància: " + String.valueOf(item.getDistancia()).substring(0, 5)
+ " km");

		return vi;
	}
}
```

ANNEX IV - Dispositius

IV.A Smartphones

Samsung Galaxy Mini (GT-S5570)

Sistema operatiu:	Android 2.3.5 (Gingerbread)
Resolució pantalla:	240 x 320 (QVGA)
Mida pantalla:	3,14"
Memòria interna:	160 MB emmagatzematge; 384 MB RAM
Càmera:	3 MP

Samsung S III Mini (GT-I8190)

Sistema operatiu:	Android 4.1.1 (Jelly Bean)
Resolució pantalla:	480 x 800 (Super AMOLED WVGA)
Mida pantalla:	4"
Memòria interna:	8 GB emmagatzematge; 1GB RAM
Càmera:	5 MP

IV.B Tauletes:

Galaxy Tab 10.1 (GT-P7500)

Sistema operatiu:	Android 3.2 (Honeycomb)
Resolució pantalla:	1280 x 800 (TFT LCD)
Mida pantalla:	7"
Memòria interna:	32 GB emmagatzematge; 1GB RAM
Càmera:	1,2 MP (només càmera frontal)

Galaxy Tab 10.1 (GT-P7500)

Sistema operatiu:	Android 3.2 (Honeycomb)
Resolució pantalla:	1280 x 800 (IPS)
Mida pantalla:	10,1"
Memòria interna:	16 GB emmagatzematge; 1GB RAM
Càmera:	3 MP

Galaxy Tab 10.1 (GT-P5110)

Sistema operatiu:	Android 4.0.3 (Ice Cream Sandwich)
CPU:	Dual Core; 1 GHz
Resolució pantalla:	1280 x 800 (TFT LCD)
Mida pantalla:	10,1"
Memòria interna:	16 GB emmagatzematge; 1GB RAM
Càmera:	3 MP

ANNEX V - Manual d'usuari de IdAllau

V.A Instal·lació

Per instal·lar l'aplicació en dispositius mòbils només cal col·locar l'arxiu **IdAllau.apk** a la SD card (ques serà el *External Storage* del dispositiu) i executar com si es tractes d'un instal·lador .exe per Windows. Abans s'ha de donar permís al dispositiu per instal·lar aplicacions d'origen desconegut (és a dir, de qualsevol lloc que no sigui el *Google Play*). Per fer-ho s'ha de buscar en els ajustos o menú de configuració l'opció "orígens desconeguts" (depenent de la versió d'Android aquesta opció pot està situada en aplicacions, *ubicació i seguretat,* opcions de *desenvolupador*, etc., i amb un nom una lleugerament diferent).

Al començar la instal·lació apareixerà un avís amb els permisos que necessita l'aplicació per executar-se. Al confirmar i donar el permís d'instal·lació l'aplicació s'encarrega de crear les seves carpetes i posar-hi els arxius XML amb la informació de les zones d'allaus.

Per acabar, només col·locar els arxius de mapes (arxiu ZIP) en la carpeta *osmdroid* que s'haurà creat (veure apartat B.a Mapa per més informació).

V.B Mapa

La capa que ha de dibuixar l'aplicació ha d'estar emmagatzemada en format ".zip" en la carpeta "***osmdroid***", a l'arrel del *External Storage* del dispositiu. Al seu temps, la carpeta arrel dins del ZIP ha de dir-se necessariament "Mapnik" perquè la llibreria *Osmdroid* reconegui el recurs.

El menú de la vista de mapa té 3 opcions:

a. Navegar on sóc ( ): centra el mapa en la posició de l'usuari.
b. On sóc? ( ): mostra en un diàleg les coordenades de l'usuari.
c. Buscar POI's pròxims ( ): primer pregunta al usuari el radi de recerca a través d'un diàleg. Un cop establert, es mostren les zones d'allaus dins de la distància establerta.

Per descomptat, cap de les opcions anteriors funciona si no hi ha posicionament per part del GPS.

Al seleccionar un POI's, apareix un quadre amb la seva informació bàsica i dos opcions:

a. Serveix per accedir a la informació completa de la zona d\'allau seleccionada ( ).
b. L'aplicació obre la realitat augmentada mantenint la selecció ( ).

B.A Crear nous recursos de mapes
Per dur a nous recursos de mapes, es pot utilitzar el programa *Mobil Atlas Creator* (MOBAC). A partir d'un dels recursos WMS ja existents en el programa com a un altre nou, el MOBAC crear un conjunt de *tiles* (o imatges) dels talls que hi ha en l'àrea seleccionada per l'usuari. S'exporta com a ZIP i es modifica el nom de la carpeta interna d'aquest ZIP a "Mapnik". Es torna a comprimir i només cal posar l'arxiu a la carpeta esmentada anteriorment.

V.C Realitat Augmentada

La realitat augmentada permet mostrar les zones d'allaus superposades sobre la imatge captada per la càmera, per tant, si el dispositiu no té càmera posterior, l'aplicació no deixarà accedir a la realitat augmentada.

El menú de la realitat augmentada té 2 opcions:

a. Distància de visualització ( ): fa aparèixer al costat esquerra de la pantalla una barra lliscant que permet establir el radi màxim de visualització de les zones d'allaus. Al prémer per segon cop, s'oculta un altre cop la barra.
b. Buscar POI's pròxims( ): amb el radi igual al valor de visualització establert mitjançant la barra lliscant, es mostren les zones d\'allaus dins de la distància establerta.

Al seleccionar un POI's, apareix un quadre amb la seva informació bàsica i dos opcions:

a. Serveix per accedir a la informació completa de la zona d\'allau seleccionada( ).
b. L'aplicació obre el mapa mantenint la selecció( ).

V.D Càmera

L'aplicació no té un recurs propi per fer fotografies; per tant utilitza un recurs ja existent en el dispositiu mòbil. Si només hi ha un recurs s'obrirà directament; si n'existeixen diversos, apareix una llista perquè l'usuari seleccioni el que vol utilitzar.

Un cop feta la fotografia, aquesta es guardada per l'aplicació en la carpeta \photo\, dins de la carpeta de l'aplicació. Adquireix com a nom l'hora en que s'ha fet la fotografia.

V.E Cerca de POI's pròxims

L'aplicació accedeix a la base de dades per comprovar quines són aquelles zones d\'allaus que es troben dins del radi especificat. Al seleccionar-ne una, l\'aplicació torna a la vista des d\'on s\'ha fet la recerca; és a dir, al mapa o a la realitat augmentada.

V.F Actualitzar la base de dades

Avanç d'actualitzar la base de dades mitjançant l'opció que hi ha al menú d'informació cal substituir els arxius XML (amb les dades de les zones d'allaus) amb els nous. Aquests arxius es troben a la carpeta IdAllau de la SD Card. Un cop substituïts, al seleccionar l'opció de actualitzar del menú, l'aplicació s'encarrega de actualitzar les dades de la base de dades a partir dels nous fitxers.

Els fitxers ha d'anomenar-se "poi_zallau#.xml", on # és el número d'arxiu (començant per 0) ja que si hi ha més de 15000 registres (aprox.), Android es queda sense memòria per fer l'anàlisi dels fitxers XML.

V.G Avisos i errors possibles

- Si no hi ha posicionament per part del GPS apareixeran avisos quan a les funcions a les que es vol accedir requereixin la posició de l'usuari. En cas que el GPS estigui desconnectat, en la vista de mapa apareix a la part superior dreta un símbol avisant de que s'està treballa sense dades de posició.

- Si el dispositiu mòbil no té càmera posterior, la realitat augmentada no podrà ser utilitzada. Apareixerà l'avís al intentar accedir a la realitat augmentada.

- Si al intentat accedir a la realitat augmentada l'aplicació deixa de funcionar, vol dir que degut a problemes al el hardware l'aplicació no pot funcionar (en concret l'API Appunta no es pot executar per falta de requeriments de hardware).