5178-1: Controlador horari d'esdeveniments en dispositius amb Android

Memòria del Projecte de Final de Carrera d'Enginyeria Informàtica realitzat per **Xavier Egea Vila** i dirigit per Aura Hernàndez Sabaté

Bellaterra, **12** de **Setembre** de **2013**

La sotasignant, Aura Hernàndez Sabaté, professora del Departament de Ciències de la Computació de la Universitat Autònoma de Barcelona.

CERTIFICA:

Que la present memòria ha estat realitzada sota la seva direcció per **Xavier Egea Vila**

Bellaterra, **Setembre** de **2013**

Signat: Aura Hernàndez Sabaté

Índex

Índex de figures

Capítol 1

Introducció

Al llarg del dia són diverses les situacions i ambients en les que es posa de relleu la importància de poder planificar tasques en els nostres dispositius mòbils. Anar al cinema i que a meitat de la pel·lícula soni un mòbil, podem considerar que és un fet puntual i depèn de cada persona recordar-se o no d'apagar el so. No obstant, aquesta mateixa situació pot esdevenir-se quan l'activitat és recurrent i periòdica en el temps, com per exemple al matí durant el trajecte en cotxe fins a la feina quan activem el *Bluetooth* del mòbil per que es conecti amb el "mans lliures"o bé quan desactivem el so i conectem la *Wi-fi* a l'arribar-hi. En aquest cas ja no seria un fet puntual i podria ser planificat. Per aquest motiu, l'objectiu del projecte és el disseny i implementació d'una aplicació per a dispositius mòbils amb Sistema Operatiu Android que permeti la planificació temporal d'esdeveniments que executin diverses accions relacionades amb els serveis que proporcionen els dispositius. Per tal de compartir l'aplicació Android amb tots els usuaris i donar l'opció de descarregar-la del *Google Play* (abans anomenat *Android Market*) en primer lloc cal decidir un nom i una icona. El nom triat per a l'aplicació Android ha estat *System Scheduler*, i ens referirem a ella així al llarg del projecte. La icona triada és la que es mostra a la figura 1.1.

1.1 Estat de l'art

Al mercat d'aplicacions per a dispositius mòbils hi ha diverses aplicacions relacionades amb el projecte proposat. Les aplicacions estudiades es poden classificar en dos grans grups pel que fa a la planificació de les tasques:

- *Aplicacions que no treballen amb rangs horaris.*

  En aquest tipus d'aplicacions es planifiquen accions que s'executen un

Figura 1.1: Icona de *System Scheduler*

dia determinat de la setmana en un moment concret. D'aquesta manera, es pot planificar que a una determinada hora s'apagui el so del dispositiu. No obstant, per a poder activar el so un altre cop s'ha de fer de manualment amb la interacció de l'usuari o bé creant una altra planificació a una determinada hora que torni a activar el so. Les aplicacions que treballen sense rangs horaris intenten solucionar aquesta mancança amb la possibilitat de crear perfils que agrupen les accions. Això fa que la llista on es mostren les planificacions encara creixi més. A més, aquest tipus d'aplicacions continuen sense donar la possibilitat de definir excepcions.

Exemple: *Timerrific*

- *Aplicacions amb planificacions amb rangs horaris.*

  En aquest tipus d'aplicacions es planifiquen esdeveniments que es mantenen actius durant un període de temps. És a dir, es pot planificar, per exemple, un esdeveniment en el que en una determinada hora es desactivi el so i que es torni a activar al cap d'unes hores. En aquest cas quan ens trobem dintre del rang horari de la planificació, es mantindrà actiu un servei fins a la fi de la planificació. D'aquesta manera existeix l'opció de planificar excepcions com, per exemple, activar la vibració en cas de rebre la trucada d'algú determinat. Les aplicacions que treballen amb rangs horaris que s'han estudiat fins al moment, no permeten accions més enllà de les relacionades amb la gestió de la vibració o el

so. Exemple: *Silent Time*

L'aplicació que proposem treballava en un inici amb rangs horaris, de manera que en un mateixa planificació es podia definir l'hora d'inici i fi. No obstant, per evitar perdre la possibilitat de tenir planificacions sense fi, ja que es pot donar el cas, per exemple, de tenir una reunió amb hora d'inici però sense hora de fi, s'ha modificat el desenvolupament de l'aplicació per tal que abasti les funcionalitats dels dos tipus d'aplicacions, fent l'aplicació més flexible i augmentant el número d'usuaris potencials a les que va destinada.

1.2 Viabilitat del projecte

A l'hora d'estudiar la viabilitat del projecte hi ha diversos factors a tenir en compte que són comuns a totes les aplicacions Android:

- *Diferents dispositius sobre diferents versions del sistema operatiu*

  Existeixen una gran quantitat de dispositius de diferents fabricants que treballen amb diferents versions del sistema operatiu Android. Aquest fet s'ha tingut en compte a l'hora de compatibilitzar l'aplicació, intentant que ho sigui el màxim possible, amb les diferents versions d'Android. Per això, s'ha escollit una versió de l'SDK d'Android mínima a partir de la qual l'aplicació serà compatible.

- *Diferents resolucions de pantalla*

  També s'ha tingut en compte que la interfície d'usuari s'haurà d'adaptar a diferents resolucions de les pantalles. Tot i que l'aplicació està destinada a dispositius mòbils petits i no a tablets ni a dispositius Android TV, el fet que les finestres de l'aplicació es puguin veure tant en vertical com en horitzontal, fa que s'hagi de tenir en compte que el re-escalat de les finestres s'adapti correctament en tot moment.

Un aspecte a tenir en compte a l'hora d'estudiar la viabilitat de l'aplicació *System Scheduler* es posa de manifest al moment de crear una planificació. En aquest instant l'usuari pot seleccionar, per exemple, que una acció d'una planificació s'executi a una hora en concret del dia cada dia de la setmana. Aquest tipus de planificació seria el cas més senzill, ja que de fet es podria arribar a complicar molt tenint en compte que una planificació es pot programar per executar-se diàriament, setmanalment o mensualment. Per això, s'ha estudiat el funcionament de les classes que s'encarreguen de la planificació d'execució de tasques. Aquestes classes, que les analitzarem

més endavant, permeten tenir corrent en *background* un servei que es capaç d'executar tasques inclòs quan el dispositiu mòbil es troba en mode *stand-by*.

Per a la implementació del projecte s'han fet servir les APIs que proporciona l'*Android SDK*, les eines de desenvolupament del plugin de la IDE d'Eclipse *Android Development Tools*(ADT) i l'emulador que simula un dispositiu Android i permet testejar de forma interactiva l'aplicació dins del mateix entorn de treball sense haver-la de carregar en un dispositiu Android real. A la figura 1.2 es pot veure una captura de pantalla de l'emulador.

Figura 1.2: Emulador d'Android versió 2.3.3 (Gingerbread)

L'emulador ens permet provar l'aplicació amb qualsevol versió del sistema operatiu Android que desitgem i també amb diverses resolucions de pantalla. No obstant, durant el desenvolupament les proves només s'han fet sobre la versió mínima amb que és compatible l'aplicació. És a dir, la versió de la API 10 que correspon a la 2.3.3 (Gingerbread). Tot i que amb l'emulador no es pot provar el 100% de les funcionalitats de l'aplicació, és una eina molt útil

per agilitzar el procés de desenvolpament. No obstant, sempre es pot provar l'aplicació sobre mòbils reals i inclús debugar els errors que puguin sorgir.

1.3 Requeriments de l'aplicació

De manera general, els requeriments de l'aplicació del projecte són:

- Usabilitat i disseny de la interfície d'usuari.
- Planificació de les accions.
- Rangs horaris flexibles.
- Compatibilitat.
- Accions disponibles (Activar o Desactivar el so, la *Wi-fi*, el *Bluetooth*, la Conexió de dades, gestió de la lluminositat, mode avió, activar la sincronització, etc)

1.4 Planificació temporal del treball

La planificació temporal del treball consisteix en tres parts essencials. Una fase prèvia de posada a punt del projecte, una altra d'implementació i una final de redacció de l'informe.

1.4.1 Posada a punt

Com a fase prèvia a l'inici del projecte, s'ha estudiat a fons les aplicacions Android que existeixen al mercat relacionades amb planificacions d'esdeveniments horaris i/o amb l'execució d'accions relacionades amb els serveis del dispositiu (apagar o encendre el so, activar o desactivar la wifi, etc...).

Paral·lelament s'ha estudiat el funcionament de les llibreries de Java que existeixen per a la programació de dispositius Android, així com les tècniques utilitzades a l'hora de programar la interfície d'usuari. Tot això sense deixar de banda els temes relacionats amb l'usabilitat de l'aplicació, així com l'adaptació de l'aplicació a les possibles resolucions de pantalla que poden tenir els dispositius.

Una part important de la fase prèvia ha estat la familiarització amb l'entorn de desenvolupament (IDE d'Eclipse) i el funcionament de l'emulador i de les llibreries de l'Android SDK.

Un cop adquirit el coneixement del que s'havia de fer, es va començar a dissenyar les diferents finestres de la interfície d'usuari de l'aplicació. Tot i que aquest disseny ha anat canviant durant el desenvolupament del projecte, es van definir almenys de forma global les principals pantalles que han servit servit de guia al llarg del procés.

L'estimació en temps de la fase de posada a punt és d'un mes.

1.4.2 Implementació del projecte

Les fases del projecte que es van definir al principi del projecte van ser les següents:

- Tasca 1. Implementar la planificació d'un esdeveniment. En aquesta primera fase es crea la interfície d'usuari per permetre definir l'hora en concret en la que s'executarà una planificació.

- Tasca 2. Emmagatzemar la planificació en un sistema de base de dades intern, de manera que l'usuari la pugui editar i modificar segons la seva conveniència.

- Tasca 3. Desenvolupar la interfície d'usuari per tal que es pugui emmagatzemar el rang horari, és a dir l'hora d'inici i fi, a més del dia o dies de la setmana en que s'executarà l'acció de la planificació.

- Tasca 4. Implementació de l'execució de la planificació d'activació del so a l'inici del rang horari i desactivació del so al final.

- Tasca 5. Investigar com fer una planificació que actuï en un rang horari, creant un servei que es mantingui actiu durant un període de temps.

- Tasca 6. Desenvolupar el servei que farà de "*handle*"de les planificacions que es produexin i que gestionarà les excepcions definides per l'usuari.

- Tasca 7. Investigar les altres possibles accions a més del so. *Wifi* encès/apagat, vibració, il·luminació, background, etc...

- Tasca 8. Implementar altres accions a més del so.

- Tasca 9. En una última fase, pujar l'aplicació a *Google Play* per tal que els usuaris puguin descarregar-la i provar-la.

Cada tasca ha estat definida com un entregable, amb les implementacions de les parts de servidor i d'interfície d'usuari (UI) necessàries. Per a cada tasca de la fase d'implementació es va fer una estimació incial de dues setmanes per tasca, pel que l'estimació temporal de la fase d'implementació seria d'uns dos mesos mig. No obstant, es reserva un mes adicional per fer proves i per resoldre errors que puguin sorgir.

1.4.3 Redacció de l'informe

Per a la redacció de l'informe l'estimació temporal és d'un mes incloses les correcccions.

1.4.4 Estimació temporal

El calendari inicial tenint en compte les tres fases que inclouen l'estudi d'altres aplicacions existents, l'estudi de les llibreries, la familiarització amb l'entorn de desenvolupament Android, la implementació i també la redacció de l'informe, seria d'uns 5 mesos i mig.

1.5 Comentaris

Com s'ha pogut veure, les possibilitats que existeixen a l'hora de configurar accions sobre un dispositiu mòbil són molt grans. No obstant, quan una aplicació és molt versàtil s'ha de fer un gran esforç en la usabilitat per tal que no esdevingui massa complicada per a l'usuari. Per tant, a mesura que l'aplicació ha anat creixent s'ha prioritzat la seva usabilitat enfront a un augment de les opcions disponibles.

Capítol 2

Funcionalitats Android

En aquest capítol s'introduïran algunes de les funcionalitats que ofereix Android que s'han utilitzat per al desenvolupament d'aquest projecte, així com les llibreries i extensions de llibreries que han facilitat la programació de l'aplicació i han estat molt útils per a la compatibilitat amb dispositius mòbils amb versions d'Android antigues.

2.1 SQLite

SQLite [1] és una base de dades de codi obert que està incorporada a Android. Es tracta d'un motor de base de dades que requereix molt poca memòria per treballar (aprox. 250 KBytes) i que permet bases de dades relacionals, transaccions, etc. Aquestes característiques la fan perfecta per tal que les aplicacions Android la puguin fer servir per consultar, afegir, eliminar i qualsevol tipus de gestió que es vulgui fer sobre les dades, sempre i quan el volum d'aquestes sigui limitat.

Els tipus que permet SQLite són Text, Integer i Real. Els altres tipus de dades s'hauran de convertir a aquests tres tipus abans de ser emmagatzemats a la base de dades.

2.1.1 Base de dades de l'aplicació

Per a l'aplicació *System Scheduler* s'ha creat una base de dades per emmagatzemar les dades de les planificacions dels usuaris. En un inici s'ha creat una taula que contindrà la informació de les planificacions de l'usuari. Els camps de la taula són els següents:

[1] http://www.sqlite.org

- *_id*. És l'identificador únic de cada planificació.
- *start_time*. Hora d'inici.
- *end_time*. Hora de finalització.
- *repeat*. Dia o dies en que es repeteix una planificació.
- *action*. Acció que s'executarà a l'hora d'inici.
- *action_end*. Acció que s'executarà a l'hora de finalització. En el moment que comença un event es guardarà en aquest camp l'estat del dispositiu depenent de l'acció de la planificació, de manera que en el moment que acaba la planificació es pugui recuperar l'estat.
- *enable*. Camp que identifica si la planificació està activa o no.

2.2 Llibreries

2.2.1 *Support Library*

Les llibreries de compatibilitat o *Support Library* són extensions de les APIs que ens premeten fer ús de noves funcionalitats que no existien per una determinada API antiga. Això facilita la programació, de manera que el programador no s'ha de preocupar de si una determinada funcionalitat no existeix per una API en concret. Només caldrà que afegeixi l'API de support i canvi els *imports* i les herències de les classes per tal que les faci servir.

Així per exemple, si una aplicació és compatible a partir de l'API 14 i es vol fer compatible amb una API anterior, com per exemple la 10, en tindrà prou amb fer servir la *Support Library*. L'objectiu és simplificar el desenvolupament permeten fer ús de noves funcionalitats a les aplicacions, sense preocupar-se per la compatibilitat amb les versions de l'API.

2.2.2 Llibreria *ActionBarSherlock*

Segons l'autor d'aquesta llibreria,

ActionBarSherlock *és una extensió de les llibreries de compatibilitat o* Support Library *dissenyada per facilitar l'ús del patró de disseny* Action Bar *en una única API a través de totes les versions d'Android. La libreria utilitzarà l'*Action Bar *natiu d'Android quan estigui disponible, i en cas contrari utilitzarà una implementació pròpia sobre les plantilles de l'aplicació d'una forma*

automàtica. Això permet desenvolupar d'una manera fàcil una aplicació amb una Action bar *per totes les versions d'Android des de la 2.x cap endavant.*

Així doncs, *ActionBarSherlock* bàsicament permet la compatibilitat cap enrere, arribant a on les llibreries de compatibilitat no arriben. Per tant, si volguéssim programar una aplicació per les últimes versions d'Android, aquesta llibreria no ens faria falta. Però com que no és el cas, i necessitem compatibilitat des de la versió 10 de l'API, aquesta llibreria resultarà molt útil.

Entre les múltiples opcions disponibles d'aquesta llibreria, a l'aplicació *System Scheduler* s'han fet servir per compatibilitzar les següents característiques:

- Títols de les finestres.
- Possibilitar la navegació cap enrere des de la icona de *System Scheduler*, tot afegint la fletxa cap enrere com indicador.
- Menú a la barra d'accions (*Action Bar*) que s'adapten a la posició vertical i horitzontal. No obstant, aquesta funcionalitat s'ha substituït per la utilització d'icones d'afegir planificació i d'esborrat.

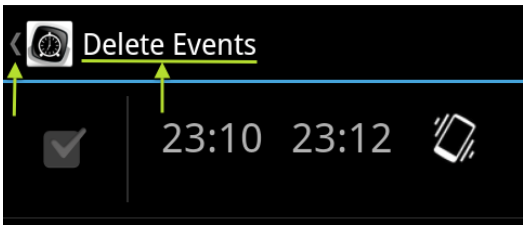

Figura 2.1: Indicador de fletxa cap enrere i títol de la finestra en Android versió 2.3.3 (*Ginger Bread*)

Totes aquestes opcions, al fer servir l'extensió de la llibreria de suport *ActionBarSherlock* estan disponibles també per mòbils amb versions antigues de l'API, tot i que en el seu moment no estaven disponibles de forma nativa. A la figura 2.1 es pot veure la barra d'accions de l'aplicació sobre la versió d'Android 2.3.3 (*Ginger Bread*). El títol de la finestra i la icona per tornar enrere no està disponible de forma nativa, no obstant la llibreria *ActionBarSherlock* fa possible aquesta funcionalitat només disponible fent servir les versions més recents de l'API d'Android.

2.3 *Fragments*

Un dels canvis principals que s'han produït al llarg de les diferents versions de l'API per Android ha estat la introducció dels *Fragments* a partir de la versió Honeycomb (versió 3.2) que aporten una major flexibilitat enfront a les *Activities* o activitats que existien fins al moment.

Tot i que es pot obtenir pràcticament el mateix resultat fent servir Activitats, l'ús de Fragments ajuda a evitar la repetició de codi ja que permeten que diverses Activitats facin servir un mateix Fragment, o dit d'una altra manera, cada Fragment pot pertànyer a una o més d'una Activitat.

Els Fragments faciliten al programador compatibilitzar les aplicacions amb pantalles de diferents tamanys i resolucions.

2.4 Orientació de la pantalla

Les diferents finestres de l'aplicació *System Scheduler* estan dissenyades per treballar tant en posició vertical com horitzontal. De manera que segons l'orientació del mòbil s'expandiran o reduiran les amplades dels llistats i formularis o bé s'activarà o desactivarà l'scroll quan convingui per tal de mostrar tota la informació de la pantalla. La figura 2.2 mostra un exemple d'aquesta adaptació, a més de la diferència de posició de les icones, de la paparera i el signe '+', fent l'aplicació que tingui un disseny adaptatiu (*Responsive Design*). Per poder percebre aquest efecte el dispositiu ha de tenir activada l'opció de canvi d'orientació.

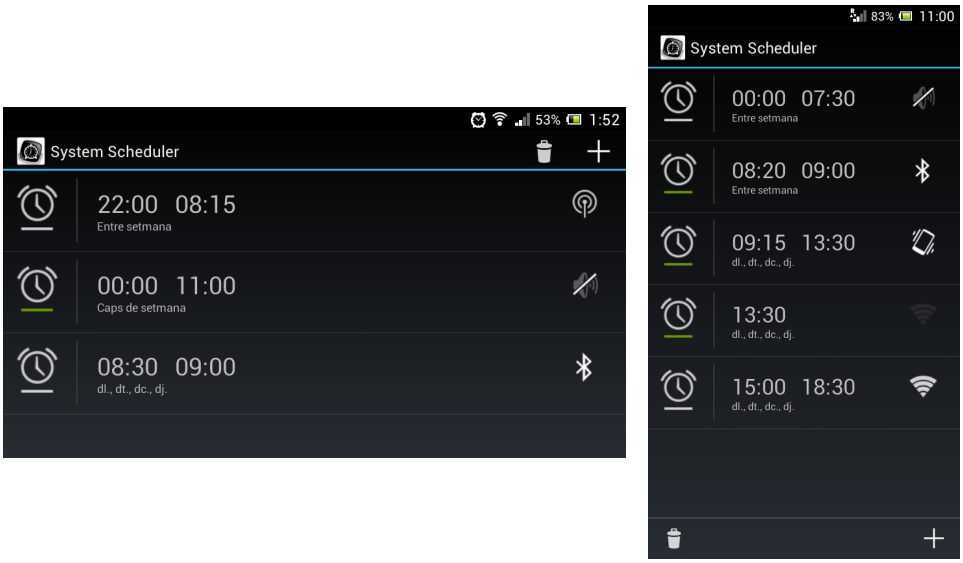

Figura 2.2: Vistes en posició horitzontal i vertical

2.5 *Adapters*

Els *Adapters* són el vincle entre la vista que s'encarrega de mostrar les dades per pantalla (*AdapterView*), i la font de dades (*DataSource*) que indica quines dades s'han de mostrar.

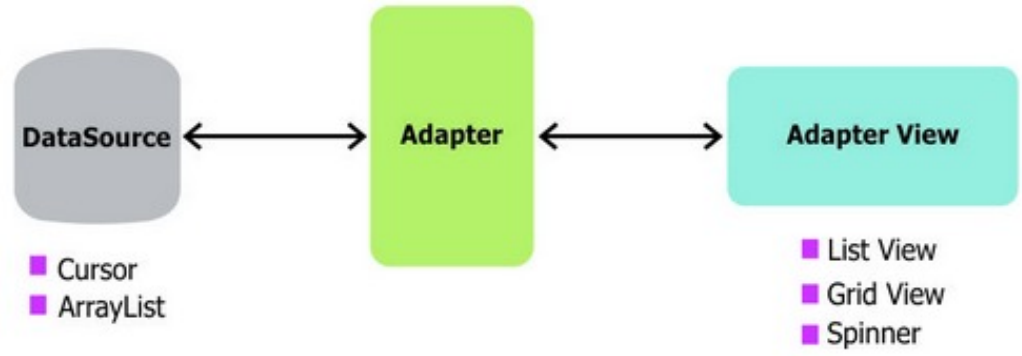

Figura 2.3: Diagrama Adapter

A la figura 2.3 es mostra el paper de l'*Adapter* que seria l'encarregada d'organitzar i dibuixar cadascun dels elements de qualsevol de les classes que hereten de l'*AdapterView* (*ListView*, *GridView*, *Spinner* i *Gallery*). Això fa que els Adapters siguin en molts casos els colls d'ampolla de les aplicacions.

Per tal de fer més eficient l'ús dels Adapters primer cal coneixer com

funcionen [2]

A la imatge 2.4 es mostra el funcionament d'una finestra quan es fa scroll. Es pot veure que quan es fa scroll cap avall la vista de la part de dalt deixa de mostrar-se i n'apareix una altra de nova a la part de baix. Internament aquest procés es basa en fer una crida a la funció *getView()* per mostrar la nova vista que apareixerà a la part de sota de la pantalla, i guardar la vista que ha desaparegut de la part de dalt a la *Android Recycler*, que és un buffer on es guarden totes les vistes candidates a ser esborrades. Per això, si recuperem la vista que havíem perdut abans fent scroll cap amunt, ja no ens caldrà buscar-la a la font de dades, sinó que la podrem recuperar de l'*Android Recycler*, millorant així l'eficiència.

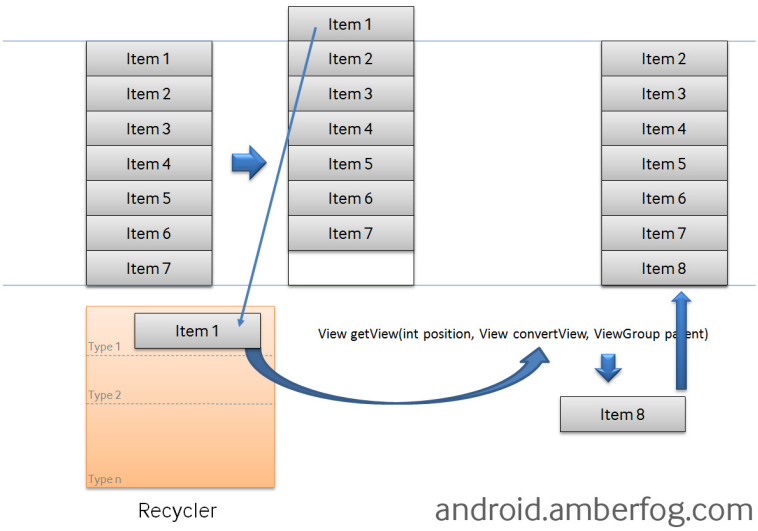

Figura 2.4: Android Recycler

La comanda *findViewById* que s'utilitza per referenciar els elements de la vista és un filtre de cerca bastant pesat, és per això que si la vista a mostrar l'estem reciclant de l'*Android Recycler* és aconsellable fer servir el "*holder pattern*". Aquest patró es basa en crear una classe estàtica, on les propietats seran els elements que volem fer servir a la vista. Quan es cridi a *getView()* i es llegeixi una vista per primer cop, es crearà una instància d'aquesta classe estàtica i s'hi guardarà la informació de la *View* per després guardar l'objecte sencer mitjançant la comanda setTag() de la *View*.

A l'exemple de codi que es mostra a la figura 2.5 es pot veure més clar:

[2]http://www.google.com/events/io/2009/sessions/TurboChargeUiAndroidFast.html. Video-presentació feta per Roman Guy @ GoogleIO 2009 que mostra com funcionen els Adapters i com fer-ne un ús correcte a les aplicacions per millorar l'eficiciència.

```
static class ViewHolder{
    TextView text;
    ImageView icon;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;

    if (convertView==null) {

        convertView = layoutInflater.inflate(R.layout.activity_main, parent, false);

        holder = new ViewHolder();
        holder.text = (TextView) convertView.findViewById(R.id.text);
        holder.icon = (ImageView) convertView.findViewById(R.id.icon);

        convertView.setTag(holder);
    }
    else {
        holder = (ViewHolder) convertView.getTag();
    }

    holder.text.setText(DATA[position]);
    holder.icon.setImageBitmap((position & 1) == 1 ? micon1 : micon2);

    return convertView;
}
```

Figura 2.5: Exemple de funcionament del getView() d'un Adapter

2.6 *Loaders*

La classe *Loader* es va introduir a partir de la versió 3.0 d'Android (*Honeycomb*) i permet a més de la càrrega asíncrona de dades, la monitorització de la font de les dades, de manera que quan es produeix algun canvi en aquesta, actualitzarà les vistes. Els *Loaders* milloren l'eficiència al guardar l'estat, així en cas que es faci una recàrrega per qualsevol canvi de posició del mòbil, d'horitzontal a vertical per exemple, no necessitaran tornar a carregar les dades.

Les finestres de l'aplicació *System Scheduler* que contenen llistats, finestra de llistat de planificacions i finestra d'esborrat de planificacions, estan implementats amb *Loaders*.

2.7 Notificacions

Les notificacions són un mecanisme per avisar l'usuari d'algun esdeveniment d'importància. N'hi ha de dos tipus:

- Missatge per pantalla. Es mostra una finestra de diàleg amb informació

per a l'usuari que desapareix al cap de pocs segons.

- Notificacions en *background*. Es fa servir principalment des d'elements purament background: *Services*, *Receivers*, *Alarms*, ... L'objectiu és informar d'un esdeveniment detectat o produït per un dels anteriors. Es poden exhibir de tres formes diferents:
  - Una icona a la barra de notificacions
  - Una *View* a la barra de notificacions estesa
  - Efectes audio-visuals com ara vibracions, sons o leds

A l'aplicació *System Scheduler* es mostraran diverses notificacions amb missatges per pantalla. També es mostraran notificacions a la barra superior quan una planificació activa s'inicia.

2.8 *BroadcastReceiver* i *AlarmManager*

Per tal de poder implementar la programació de les planificacions i que permetin rebre les notificacions per poder llançar les accions com activar/desactivar el so, la *Wi-fi*, el *Bluetooth* o la conexió de dades, seran necessaris diversos components d'Android i seguir el següent procés.

El component *BroadcastReceiver* permet registrar esdeveniments de sistema o d'aplicacions. Per poder fer-lo fer servir només cal crear una classe que hereti de la *BroadcastReceiver* i implementar el mètode *onReceive()*. Cada cop que es produeixi un esdeveniment dels que estiguin registrats seran notificats per l'Android runtime i el mètode *onReceive()* serà l'encarregat de gestionar aquestes notificacions. Es necessitarà també crear un *PendingIntent*. Un *Intent* és una estructura de dades que es passa per paràmetre entre les activitats. Per posar un exemple, quan fem click a sobre d'una planificació per tal d'obrir la finestra de modificació, es crearà un *Intent* i se li afegirà el paràmetre identificador de la planificació. D'aquesta manera quan s'obri la finestra de modificació es recuperarà l'*Intent* i el paràmetre identificador per tal de poder cercar les dades de la planificació a la base de dades. Un cop s'acabi la modificació al apretar el botó de guardar, s'actualitzarà l'*Intent* amb la informació modificada. Per la seva banda un *PendingIntent* s'utilitza per passar per paràmetre un *Intent* que s'executarà amb els mateixos permisos de l'aplicació, inclús si aquesta ja s'ha apagat. En el nostre cas el *PendingIntent* el passarem a la classe *BroadcastReceiver* creada. A la classe *AlarmManager* es defineix la planificació passant el *PendingItent*. Quan es rebi un notificació es farà servir la classe *PowerManager* per despertar el mòbil i poder executar les operacions necessàries.

Un dels punts que cal tenir en compte relacionada amb la planificació dels esdeveniments succeeix quan s'apaga el mòbil. En aquest moment totes les planificacions que s'han fet fins al moment es perden. És per això que s'ha crear un *BroadcastReceiver* que rebi l'acció d'arrencada del dispositiu. Un cop capturada l'acció, es fa una cerca a la base de dades local del mòbil per cercar totes les planificacions actives creades per l'usuari per tornar a programar-los un altre cop.

2.9 Pujar l'aplicació a *Google Play*

Mitjançant la consola de *Google Play Developer* és possible publicar una aplicació de manera que els usuaris de dispostiu mòbils Android la puguin descarregar. És per això que el primer pas consistirà en crear un compte de desenvolupador de *Google Play*. Un cop fet això, s'han de fer un seguit de modicacions al fitxer ".apk"que genera la compilació per tal que el paquet compleixi un seguit de criteris d'acceptació. En primer lloc l'arxiu "SystemScheduler.apk"que genera l'*Eclipse* s'haurà de firmar amb un fitxer de claus o *keystore* que generarem. I en segon lloc s'haurà d'executar sobre el fitxer resultant un alineament fent servir l'aplicació *zipalign*. Amb aquesta aplicació s'assegura que les dades no comprimides, com per exemple les imatges, comencin a partir d'un cert alineament comptant a partir de l'inici del fitxer, de manera que totes les parts puguin ser accedides de manera directa, reduint així l'ús de la memòria RAM consumida quan l'aplicació estigui en funcionament.

L'arxiu de claus també haurà de complir uns criteris. Entre ells és obligatori que la data d'expiració sigui més enllà del 22 d'octubre del 2033, així que es recomana posar un període de 25 anys pel qual la clau serà vàlida. Per això el paràmetre "-validity"que contè els dies de validesa de la clau serà de 10.000. La comanda serà la següent:

```
$ keytool -genkey -v -keystore claus.android -alias aliasname
   -keyalg RSA -keysize 2048 -validity 10000
```

Un cop tinguem l'arxiu de claus que serveix per firmar l'aplicació, haurem de seguir els següents passos:

Des de l'entorn de desenvolupament *Eclipse* farem botó dret a sobre del projecte. A continuació seleccionem *Android Tools* → *Export Unsigned Application Package*. Això ens crearà un arxiu "SystemScheduler.apk"al directori que haguem seleccionat. A més un missatge de l'*Eclipse* per pantalla ens avisarà del següents passos que haurem de realitzar. Tal com es pot veure a la figura 2.6 aquests passos consiteixen en firmar el paquet i a con-

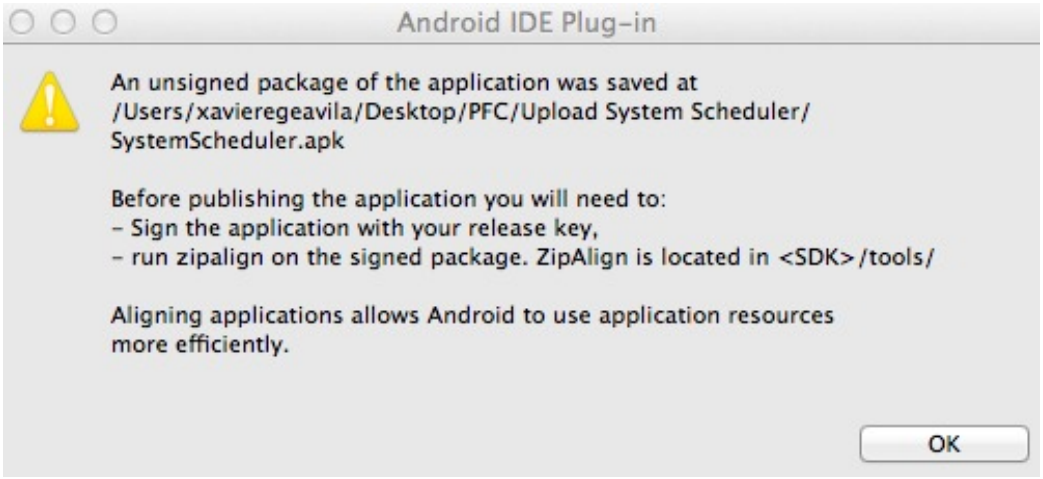

Figura 2.6: Missatge d'avís de l'IDE d'*Eclipse*

tinuació alinear-lo amb l'aplicació *zipalign*. Així doncs obrirem un terminal i ens col·locarem al directori on hem emmagatzemat l'arxiu "SystemScheduler.apk". Per facilitat copiarem l'arxiu de claus al mateix directori i escriurem la següent comanda:

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
    claus.android "SystemScheduler.apk" systemscheduler
```

Intrduïrem la paraula clau de l'arxiu de claus i el paquet "SystemScheduler.apk"quedarà firmat.

També podem utilitzar la comanda jarsigner apuntant als directoris on estiguin emmagatzemats tant l'arxiu "claus.android"com l'arxiu "SystemScheduler.apk".

Opcionalment, per verificar la firma podem executar la següent comanda:

```
$ jarsigner -verify SystemScheduler.apk
```

Hauria de sortir el missatge "*SystemScheduler.apk jar verified.*"

A continuació canviem el nom de l'arxiu ja firmat "SystemScheduler.apk"per "SystemScheduler-signed.apk".

Per la següent comanda necessitem l'aplicació "zipalign"que es troba a la ruta <SDK>/tools/

Executem la comanda següent:

```
$ ./zipalign -f -v 4 "SystemScheduler-signed.apk"
    "SystemScheduler.apk"
```

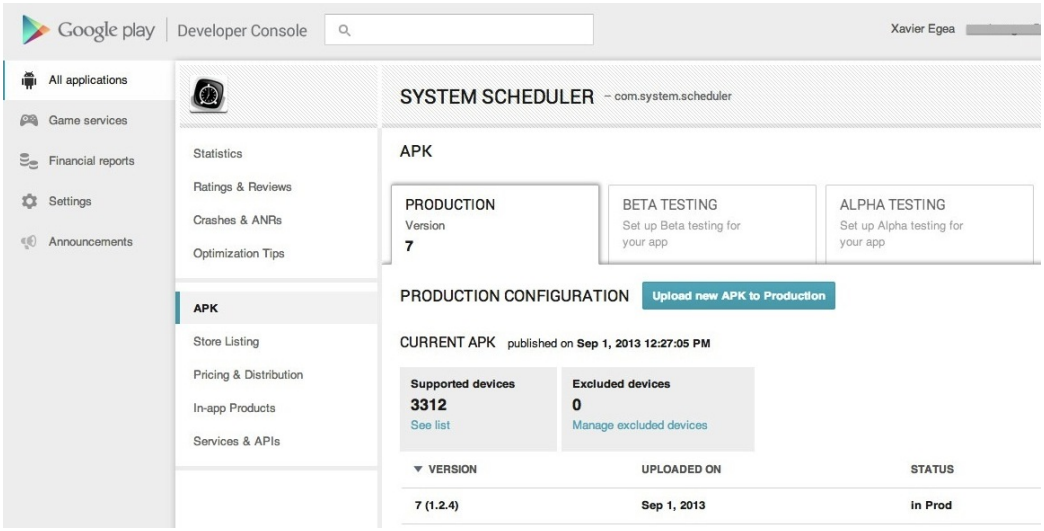

Figura 2.7: Consola de desenvolupador de *Google Play*

Amb això ja tindrem el paquet preparat per pujar. A partir d'aquí només cal obrir la consola de desenvolupador: *https://play.google.com/apps/publish/* i prémer el botó de pujar l'APK. A la figura 2.7 es mostra una finestra de la web de la consola de desenvolupador de *Google Play*.

Quan necessitem fer canvis a l'aplicació i pujar noves versions haurem de canviar la versió del codi definida dintre de l'arxiu "manifest.xml". Aquest fitxer conté les configuracions globals que afecten al projecte, com la versió mínima a partir de la qual l'aplicació serà compatible, els permisos que necessita l'aplicació i que hauran de ser acceptats per l'usuari, així com les diferents *Activities* o *Receivers* de l'aplicació. Un cop modificada la versió seguirem els passos que s'han fet servir per pujar per primera vegada l'aplicació.

Capítol 3

Funcional

System Scheduler és una aplicació per a la planificació diària i setmanal de planificacions que permeten gestionar funcionalitats del mòbil com poden ser el so, les conexions Wi-fi o Bluetooth o la conexió de dades.

En aquest capítol es veurà com descarregar i instal·lar *System Scheduler* des de *Google Play*, així com el funcionament de les diferents finestres de l'aplicació enacarregades de crear, modificar i eliminar planificacions. A més es comentaran els tipus de notificacions que s'envien a l'usuari.

3.1 Instal·lació

L'aplicació *System Scheduler* està publicada a *Google Play*, per tant per fer-la servir només caldrà descarregar-la i instal·lar-la com qualsevol aplicació Android. Per tal de trobar-la a *Google Play* es pot cercar "*System Scheduler*"dintre de la categoria d'aplicacions i fer scroll fins a trobar l'aplicació, o bé cercant directament "*com.system.scheduler*"que és l'identifiacor únic del paquet *System Scheduler* (Figura 3.1).

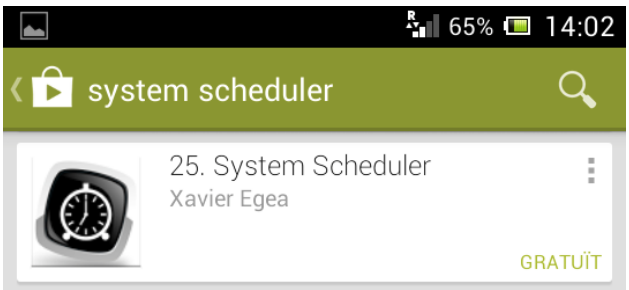

Figura 3.1: *System Scheduler* a Google Play

Abans de poder fer la descàrrega serà necessari acceptar els permisos que requereix l'aplicació. Aquests permisos alerten a l'usuari de les operacions que es podran fer des de l'aplicació que afecten a elements susceptibles de la seguretat. En el cas de *System Scheduler*, aquests permisos afecten al control de la vibració i a la gestió de la Wi-fi i Bluetooth que són intrínsecs a l'aplicació i per tant és lògic acceptar-los.

Al fer click sobre l'aplicació del llistat s'obrirà la finestra des d'on podrem descarregar i instal·lar l'aplicació, tal com es pot veure a la figura 3.2

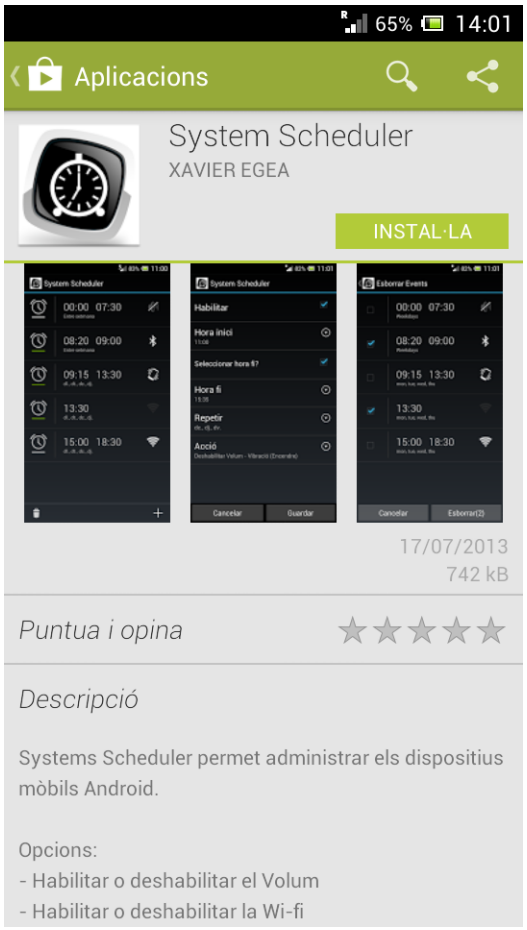

Figura 3.2: Finestra d'instal·lació

Un cop instal·lada l'aplicació apareixerà la nova icona al dispositiu i ja es podrà obrir.

3.2 Llistat de planificacions

La finestra de llistat de planificacions és la primera finestra que trobem quan iniciem l'aplicació. Aquesta, recull les diverses planificacions creades per l'usuari i des d'aquí es poden gestionar (crear, modificar, esborrar) totes les planificacions. En un primer moment apareixerà buida amb les icones de la paperera per esborrar planificacions i la icona '+' per afegir-ne. Les planificacions afegides es mostraran en forma de llista on cada línia mostra la informació rellevant de cada planificació per tal que l'usuari pugui identificar-la: Hora d'inici i fi de la planificació, dies en que s'executarà aquesta i una icona que indica el tipus d'acció que s'efectuarà. A la figura 3.3 es mostra una captura d'exemple. La barra de progrés que es mostra a sota del rellotge a l'esquerra de cada línia, indica que aquesta està activada (de color verd) o desactivada (de color gris).

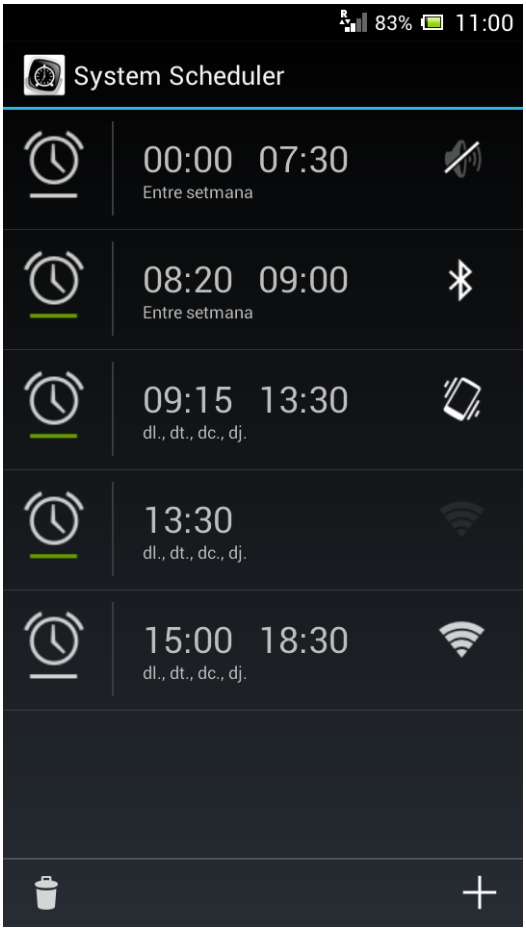

Figura 3.3: Llistat de planificacions

Des d'aquesta finestra principal es poden fer les seguents accions:

- **Canviar l'estat d'una planificació**. Per tal d'activar o desactivar una planificació es pot fer des del propi llistat fent click sobre el rellotge. Al fer-ho, en cas que es tracti de l'activació, es mostrarà una notificació en forma de missatge de text informant que la planificació s'ha activat.
- **Crear una planificació**. Per afegir una planificació, fent click a sobre la icona '+' s'obrirà la finestra del detall de la planificació amb els camps buits llestos per poder ser omplerts.
- **Edició d'una planificació**. Fent click sobre una planificació determinada s'obrirà la finestra de detall de la planificació per a poder-la modificar.
- **Esborrar planificacions**. Fent click sobre la icona de la paperera s'obrirà una nova finestra amb el llistat de les planificacions per tal de poder-les seleccionar per ser esborrades.

3.3 Detall de la planificació (Creació i edició)

La finestra de detall de la planificació és un formulari que permet modificar una planificació existent o afegir-ne una de nova. Els camps necessaris per poder crear una planificació seran els dies de la setmana en que s'executarà aquesta, l'acció que s'executarà en el moment d'inici de la planificació, l'hora d'inici i per últim l'hora de finalització. Aquest últim camp serà opcional i per activar-lo haurem de marcar la casella de *CheckBox* amb el text "*Seleccionar hora fi?*". Si la casella està marcada podrem afegir el camp d'hora de finalització. En cas contrari aquest camp quedarà buit i actuarà com una planificació que comença a una determinada hora però que no acaba mai. A la figura (Figura 3.4) es mostra un exemple de la finestra d'edició d'una planificació.

Les accions que es poden seleccionar en el moment d'activació de la planificació seran les següents:

- Activar o desactivar el volum. En cas de desactivar el volum hi haurà la possibilitat d'activar la vibració.
- Activar o desactivar la connexió *Wi-fi*
- Activar o desactivar la connexió *Bluetooth*
- Activar o desactivar la connexió de dades

Figura 3.4: Edició d'una planificació

Al final de l'execució d'una planificació es retornarà a l'estat previ en que es trobava l'element hardware sobre el que es va actuar (Volum, Wi-fi, Bluetooth, Conexió de dades) en el moment de l'inici de l'event.

En el cas que no hi hagi hora de finalització només s'executarà l'acció d'inici i no es retornarà a l'estat previ.

Per tal que una planificació pugui ser creada o modificada ha de complir els següents requisits:

- L'hora d'inici és obligatòria.
- L'hora d'inici i fi no poden ser iguals.
- S'ha de seleccionar almenys un dia de la setmana en que s'executarà la planificació.

- S'ha de seleccionar una acció obligatòriament.

En cas que un d'aquests requisits no es compleixi al clickar al botó *Guardar*, es mostrarà un missatge per pantalla indicant l'error i no deixarà continuar fins que tots els errors s'hagin corregit.

3.4 Esborrat de planificacions

Per esborrar planificacions es farà click a la icona d'escombraries de la finestra del llistat de planificacions. Això obrirà una nova finestra on es mostrarà el mateix llistat que teníem a la finestra anterior, però en comptes de la icona del rellotge que ens permetia activar o desactivar una planificació, es mostrarà un *checkbox* per cada línia per poder seleccionar les planificacions que desitgem eliminar. Tal com es mostra a la figura 3.5, al fer click sobre la paperera a la finestra del llistat de planificacions, ens portarà a la finestra d'esborrat

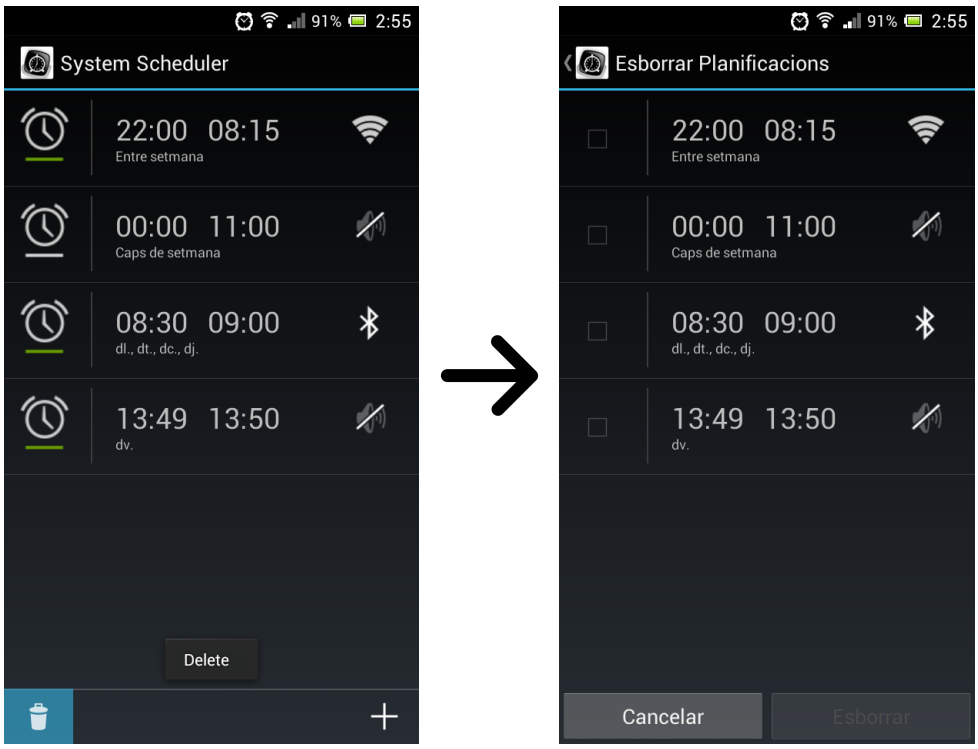

Figura 3.5: Llistat de planificacions - Esborrat de planificacions

A l'obrir-se la finestra d'esborrar planificacions, el botó *Esborrar* apareixerà desactivat ja que no hi haurà cap *checkbox* seleccionat, però a mesura que anem seleccionant les planificacions, aquest botó s'activarà i informarà

entre parèntesis del número de planificacions que s'han seleccionat fins al moment. Un exemple d'aquest comportament es pot veure a la figura 3.6.

Un cop hàgim seleccionat les planificacions que desitgem esborrar i hàgim

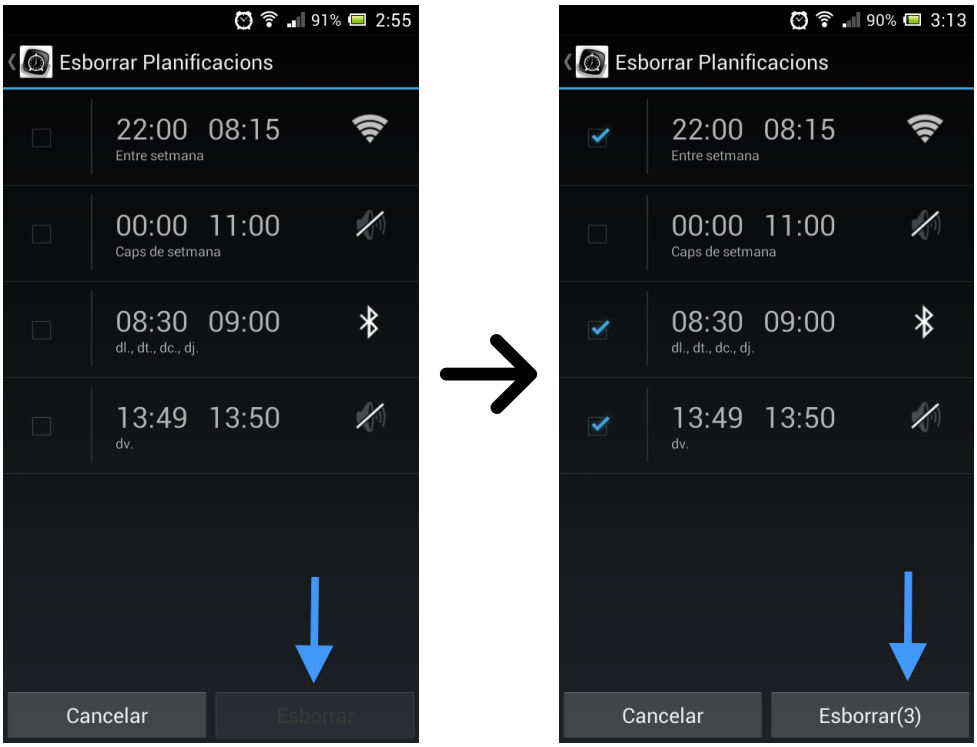

Figura 3.6: Esborrat de planificacions

fet click al botó *Esborrar* apareixerà una finestra de diàleg de confirmació. A continuació, un cop hàgim confirmat, s'obrirà la finestra del llistat de planificacions sense les esborrades recentment.

3.5 Notificacions

Per tal que l'usuari s'assabenti de que s'està executant una planificació, o dit d'una altra manera, que el moment actual es troba dintre del rang horari d'una planificació activa, es mostrarà una notificació a la barra superior del dispositiu mòbil. Els tipus de notificacions que es mostraran són:

- Planificacions amb rang horari (hora d'inici i fi). En aquest cas les notificacions avisaran de l'acció que s'ha produït i no es podran esborrar

de forma manual fins a l'hora de finalització, o bé si l'usuari desactiva o esborra la planificació de forma manual entrant a l'aplicació.

- Planificacions amb hora d'inici (però sense hora de fi). En aquest cas es mostrarà un missatge de "Planificació llançada" i podrà ser esborrada per l'usuari quan ho desitgi.

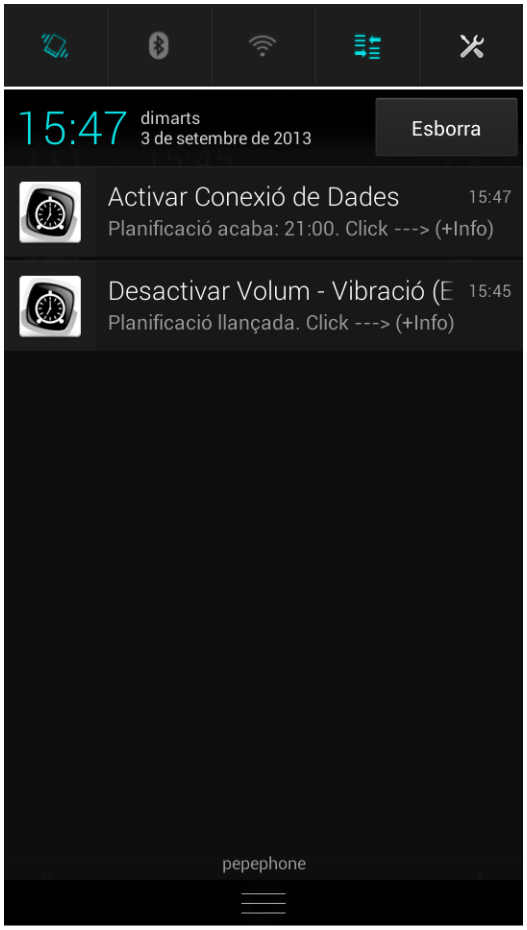

Figura 3.7: Exemples de notificacions

A la figura 3.7 es pot veure un exemple dels dos tipus de notificacions possibles. Fent clic a les notificacions s'obrirà l'aplicació a la finestra de la llista de planificacions.

Capítol 4

Decisions d'implementació

Durant el desenvolupament del projecte s'han hagut de prendre un seguit de decisions. Algunes d'elles eren intrínseques a qualsevol desenvolupament d'una aplicació Android, com pot ser la versió mínima a partir de la qual l'aplicació serà compatible o l'idioma per defecte de l'aplicació. D'altres són conseqüència de problemes sorgits relacionats amb els requeriments funcionals de l'aplicació.

4.1 Compatibilitat de l'aplicació

Idealment les aplicacions haurien de ser compatibles amb tots els dispositius mòbils Android que existeixen. Per dur a terme això, s'hauria de compatibilitzar l'aplicació des de la primera versió de l'API. No obstant, aquest fet comporta un seguit de problemes:

Pèrdua de funcionalitats : Un dels principals desavantatges, és la pèrdua de funcionalitats noves. És a dir, no podrem treballar amb les últimes característiques afegides darrerament, moltes d'elles enfocades a millorar el rendiment i la usabilitat de les aplicacions.

Major esforç de desenvolupament : Un altre inconvenient és el major esforç de desenvolupament que suposa el fer compatibles les aplicacions amb un nombre cada cop més reduït de mòbils que funcionen amb les APIs més antigues.

Impossibilitat de fer proves : Tot i que l'emulador ens permet fer proves del funcionament de l'aplicació amb totes les versions de l'API que desitgem, hi ha funcionalitats que no es poden provar amb l'emulador. A això s'hi afegeix el fet que un emulador no reprodueix al cent per

cent el comportament d'un dispositiu real. Per tant, un usuari amb un dispositu mòbil que faci servir una API antiga pot trobar-se errors a l'aplicació que no s'han detectat en la fase de proves perque no s'han pogut provar en un entorn real.

D'altra banda, s'ha d'intentar que l'aplicació sigui el més compatible possible amb els diferents dispositius del mercat, de manera que el nombre d'usuaris que puguin arribar a fer servir l'aplicació sigui el màxim possible.

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.4%
2.2	Froyo	8	3.1%
2.3.3 - 2.3.7	Gingerbread	10	34.1%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	23.3%
4.1.x	Jelly Bean	16	32.3%
4.2.x		17	5.6%

Figura 4.1: Percentatge de dispositius mòbils que utilitzen cadascuna de les diferents versions de l'API d'Android (Juliol 2013)

La figura 4.1 ens mostra una llista de les diferents versions de sistema operatiu Android corresponent al juliol del 2013. A la fila *Codename* és mostren els àlies d'una versió i API determinada. Tal com es pot veure aquests àlies estan ordenats de forma alfabètica i des d'un inici han tingut noms de pastissos. L'últim camp (*Distribution*) indica el percentatge de mòbils del mercat que utilitzen cadascuna de les versions. Així podem veure com la versió de la API 10, tot i ser una API força antiga, és la que utilitzen més dispositius mòbils. Si sumem els percentatges de les tres primeres versions (*Donut*, *Eclair*, *Froyo*) observem que només un 4.6% dels dispositius del mercat fan servir aquestes versions. D'aquesta manera, triant fer compatible l'aplicació a partir de la versió 10 de l'API (*Gingerbread*), es cobreix més del 95% dels dispositius. A més, al llarg del temps cada vegada hi haurà menys dispositius amb versions antigues de l'API, i el manteniment de l'aplicació es centrarà

en compatibilitzar amb les versions noves de l'API que vagin sorgint. Això queda palès en el llistat de versions de la figura 4.2 corresponent al setembre de 2013 on l'API 10 ja no és la més utilitzada, cedint el primer lloc a la versió 4.1.x corresponent a l'API 16.

Version	Codename	API	Distribution
2.2	Froyo	8	2.4%
2.3.3 - 2.3.7	Gingerbread	10	30.7%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	21.7%
4.1.x	Jelly Bean	16	36.6%
4.2.x		17	8.5%

Figura 4.2: Percentatge de dispositius mòbils que utilitzen cadascuna de les diferents versions de l'API d'Android (Setembre 2013)

Així doncs, s'ha près la decisió de fer compatibe l'aplicació amb els dispositius Android a partir de la versió 10 de l'API (versió 2.3.3 - GingerBread) fins a l'API més nova que existeix actualment, la versió 18 de l'API (versió 4.3 - JELLY_BEAN_MR2).

4.2 Ordre dels botons

La posició dels botons que existeixen al llarg dels diferents formularis de l'aplicació han de mantenir una coherència. És a dir, el botó de *Cancelar* no pot estar en algunes finestres a l'esquerra i en d'altres a la dreta. A més, l'usuari ha de trobar de la forma més intuïtiva possible la posició dels botons.

A la informació consultada a Internet, tot i que hi ha divergències, la manera estàndard de situar els botons en Android, si més no en les últimes versions, és el botó de *Cancelar* a l'esquerra i el botó d'acció (*Acceptar*, *Guardar*, *Esborrar*, etc...) a la dreta. Així doncs s'han implementat totes les finestres seguint aquest criteri.

4.3 Planificació d'esdeveniments

A l'hora d'implementar la planificació del llançament dels esdeveniments, sorgeix el problema que aquests es poden planificar un, dos o més dies de la setmana sense seguir un patró recurrent constant. És a dir, l'interval entre cada llançament d'una determinada planificació no té perquè ésser el mateix. Així, en una planificació que es llenci dilluns, dimarts i divendres no podriem establir un interval constant.

Així doncs, una possible sol·lució a aquest problema consistiria en que cada cop que s'executi una acció planificada es recalculi la següent. Això seria correcte si no fos perque existeix la possibilitat que en el moment en que s'hagués d'executar una planificació, el dispositiu mòbil podria estar apagat. Això faria que no es recalculés la següent execució i per tant aquella planificació quedaria anul·lada.

Un cop vist que l'anterior sol·lució no era correcte, es va optar per mantenir una execució diària a l'hora d'inici de la planificació. D'aquesta manera cada dia a l'hora indicada pel camp d'hora d'inici de la planificació es comprovarà si s'ha d'executar l'acció en concret o no.

Per veure-ho amb un exemple, si una planificació està programada per executar-se a les 10h el dilluns i el dijous, cada dia de la setmana a les 10h es farà una comprovació de si el dia actual és dilluns o dijous. En cas afirmatiu s'executarà l'acció i en cas contari no farà res.

4.4 Finalització d'accions

Una de les desicions més complicades que s'han prés al llarg del projecte ha estat com actuar quan una planificació finalitza. Les aplicacions que no treballen amb rangs horaris no tenen aquest problema, ja que executen accions en un determinat moment i prou, ja que no tenen data de finalització. No obstant, les aplicacions que treballen amb rangs horaris com *Silent Time*, quan finalitza la planificació, executen l'acció contrària. És a dir, si s'ha creat una planificació que desactiva el so, quan finalitza l'esdeveniment el torna a activar, independentment de si a l'inici de la planificació estava activat o no. No obstant, podria ser que l'usuari vulgui que el seu dispositiu mòbil segueixi sense volum. Per tant, això pot ser una acció inesperada i/o no desitjada per a l'usuari que faci servir aquesta aplicació, ja que s'estan prenent decisions per part de l'usuari sense que aquest ho hagi acceptat ni en tingui coneixement.

Per evitar aquest comportament a l'aplicació *System Scheduler* quan una planificació finalitza retorna a l'estat incial. És a dir, retorna a l'estat en que es trobava l'element hardware sobre el que s'està actuant (Volum, *Wi-fi*,

Bluetooth, Conexió de dades) en el moment de l'inici de la planificació. Per exemple, si la planificació activava el volum, però el volum ja estava activat, el deixarà tal com estava i no el desactivarà. Si pel contrari el volum estava desactivat a l'inici de la planificació, el deixarà desactivat al finalitzar.

4.5 Esborrat de planificacions

Quan es realitzen accions crítiques, com ara l'esborrat de dades, existeixen dues possibles solucions per tal de minimitzar les possibilitats que l'usuari cometi un error:

- Mostrar un missatge de confirmació. En aquest cas es mostra una finestra modal amb un missatge avisant de l'acció que s'està a punt de produir i de si l'usuari realment desitja continuar o en cas contrari, vol cancelar l'acció.
- Desfer (Undo). En aquest cas l'usuari efectua l'acció, per exemple eliminar l'element o els elements seleccionats de la llista, i disposa d'uns quants segons per desfer l'acció. Un exemple d'aquesta implementació la podem trobar a l'aplicació de Gmail mostrada a la figura 4.3.

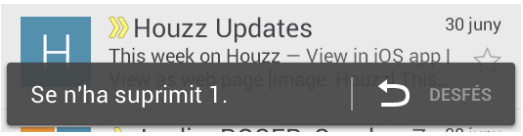

Figura 4.3: Acció de desfer de Gmail

A l'aplicació *System Scheduler* s'ha optat per la primera opció perquè ja fa la funció que ha de fer, és a dir compleix els requeriments, i per la seva facilitat d'implementació. Això ens ha permès dedicar el temps a altres funcionalitats.

4.6 Idiomes

L'idioma de l'aplicació per defecte serà l'anglès. No obstant s'han introduït els textos en català i castellà, així que si l'usuari té configurats algun d'aquests idiomes, l'aplicació apareixerà traduïda.

Capítol 5

Problemes

5.1 *TimePicker*

El component *TimePicker* és una vista que permet seleccionar una hora del dia en concret (hora, minuts, AM/PM). A l'aplicació *System Scheduler* es fa servir a la finestra d'edició de planificacions per seleccionar l'hora d'inici i fi. En principi, que aquest component ja existeixi de forma nativa al SDK hauria de facilitar la programació, no obstant ha estat una de les funcionalitats amb les que més problemes han sorgit.

El principal inconvenient ha estat gestionar com es comporta la classe *TimePicker* segons la versió de l'API que estem corrent. A continuació es detallen el comportament que s'ha detectat en les diferents versions, per tractar de resumir i aclarir els problemes sorgits.

- **Versió 2.x i 3.x**. Per a les versions més antigues, el problema que s'ha trobat és que la posició dels botons no era coherent amb l'estàndar, i es mostrava el botó *Cancelar* a la dreta i l'*Acceptar* a l'esquerra. La classe *TimePicker* no implementa la possibilitat de canviar l'ordre dels botons de forma nativa. Per corregir això s'ha modificat el text dels dos botons per tal que apareguin en l'ordre correcte, i a continuació s'ha modificat el comportament dels dos botons modificant el *listener* que captura el click, de manera que al clickar al nou botó de cancelar, s'anularà l'acció de guardar que feia fins al moment.

- **Versió 4.0.x**. Aquesta versió corregeix l'ordre dels botons i col·loca el botó *Cancelar* a l'esquerra. Per tant en aquesta versió no s'ha de fer cap modificació.

- **Versió 4.1.x**. Aquesta versió elimina l'opció de cancelar de la classe *TimePicker*. És a dir, no només elimina el botó *Cancelar*, sinó que si

s'afegeix el botó de cancelar mitjançant programació, aquest es comporta com si fos el botó d'acceptar. Mentre existeixi aquest problema, els dispositius mòbils que funcionen amb la versió 4.1 o superior tindran només un botó d'acceptar. D'altra banda, es mirarà d'implementar un *TimePicker* propi en pròximes versions de l'aplicació per evitar dependre dels diferents comportaments d'aquest component en les verions d'Android que hem vist. [1]

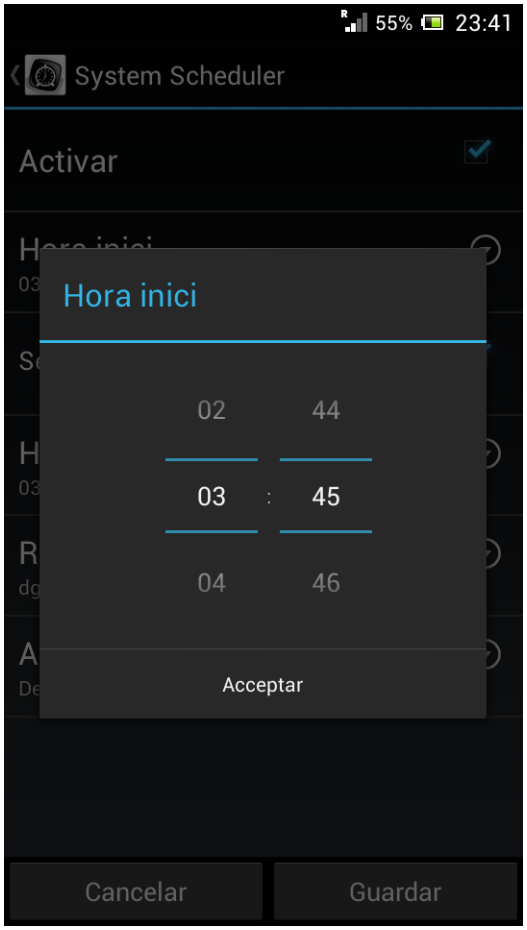

Figura 5.1: *TimePicker* d'Android versió 4.1.2, on s'ha suprimit el botó de cancelar

[1]Per més informació sobre aquest bug que afecta a les classes *DatePicker* i *TimePicker* consultar el següent enllaç: `http://stackoverflow.com/questions/11444238/jelly-bean-datepickerdialog-is-there-a-way-to-cancel`

5.2 Posició Vertical/Horitzontal

L'aplicació *System Scheduler* s'ha desenvolupat per tal que les diferents finestres i diàlegs de què consta es puguin visualitzar tant si el dispositiu mòbil està en posició vertical com en horitzontal. Això no obstant, ha portat un seguit de problemes:

- **Activar scrolls**. Quan es posa el dispositiu mòbil en posició horitzontal, es perd espai en els formularis, és a dir, es mostren menys elements en pantalla i generalemnt cal activar l'scroll. Cal tenir en compte que l'scroll es fa sobre els elements del formulari, exceptuant els botons de *Guardar* i *Cancelar* que han de restar immòbils. Per aquest motiu, la plantilla que s'ha fet servir ha estat la de la figura 5.2 on la part entre els dos tags *<ScrollView>* i *</ScrollView>* és la part del formulari on es podrà fer scroll, i la part entre *<LinearLayout>* i *</LinearLayout>* quedarà inmòbil. S'ha de tenir en compte que *<ScrollView>* té com a limitació que només pot tenir un fill, en el nostre cas *<RelativeLayout>*, que contindrà agrupat tot el contingut que es pot fer scroll.

```
<RelativeLayout>
    <ScrollView>
        <RelativeLayout>
            <TextView ...
            <ImageView ...
        </RelativeLayout>
    </ScrollView>
    <LinearLayout
        android:layout_alignParentBottom="true"
        android:weightSum="2">
        <Button
            android:layout_weight="1"/>
        <Button
            android:layout_weight="1"/>
    </LinearLayout>
</RelativeLayout>
```

Elements mòbils

Botons immòbils

Figura 5.2: Plantilla per a la finestra d'edició de planificacions

- **Pèrdua de la informació**. Un dels efectes més inesperats al programar amb Android ha estat la recàrrega d'informació de la finestra on estem treballant quan es fa un canvi d'orientació del mòbil. Per posar un exemple, si estem omplint un formulari amb un seguit de dades, i a mig procés canviem l'orientació del dispositiu mòbil, totes les dades que havíem omplert fins al moment desapareixeran i haurem de tornar

a omplir el formulari des del principi. Això és degut a que qualsevol canvi d'orientació provoca una crida als mètodes de creació de l'activitat o fragment que contenen la finestra, de manera que tot es carrega de nou.

Per solucionar això, disposem d'un mètode que es crida just després del canvi d'orientació del mòbil, l'*onSaveInstanceState(Bundle savedInstanceState)*. Aquest mètode permet emmagatzemar valors de l'estat a la variable *savedInstanceState* que rep per paràmetre. És a dir, s'emmagatzemarà tota la informació que l'usuari ha omplert fins al moment a la finestra del formulari. Aquesta variable la rebrà el mètode de creació de la nova vista *onCreate(Bundle savedInstanceState)* i es podrà, per tant, actualitzar el formulari amb les dades que l'usuari ja havia insertat.

5.3 Alineament dels RadioButtons

Els *RadioGroups* són elements de la llibreria d'Android que hereten de *LinearLayout*. Al no heretar de *RelativeLayout*, no es poden col·locar dos *RadioButtons* a la mateixa línia que s'adaptin automàticament en un canvi de posició (vertical/horitzontal) del mòbil. Fixant-nos en la figura 5.3 si els dos *RadioButtons* estan dintre d'un *RadioGroup* i per tant hereten de *LinearLayout* el botó de la dreta només podrà referenciar-se al marge esquerra i podrà, per exemple, definir una separació del marge esquerra de la finestra de 20px. Això farà que quan es canviï l'orientació de la pantalla a horitzontal el botó dret continuarà a 20px del marge esquerra i quedarà tota la resta de la línia buida. És a dir, la finestra no s'adapta correctament al nou tamany.

No obstant, el que volem aconseguir, és que el botó dret pugui estar referenciat al marge dret, de manera que si es deixa un marge de 5px entre el botó i el marge dret, al canviar l'orientació a horitzontal el botó dret pugui seguir estant a 5px del marge dret, amb el que l'efecte serà que s'adaptarà a la finestra.

Aquest efecte serà útil també per adaptar-nos a qualsevol tamany de pantalla que tinguin els dispositius mòbils en que funcioni l'aplicació.

Figura 5.3: Alineament dels *RadioButtons*

Una possible solució passa per canviar la configuració nativa dels *RadioButtons* per tal que heretin de *RelativeLayout* i no de *LinearLayout*. De totes maneres, per evitar possibles problemes que podria produir aquest canvi, s'ha optat per eliminar els *RadioGroups* i només utlitzar *RadioButtons* en el disseny de la plantilla. Això elimina el vincle físic entre els *RadioButtons*, de manera que es podria seleccionar més d'un *RadioButton* dintre del mateix grup. Aquest fet s'ha corregit mitjançant la programació que afegeix el vincle lògic entre els *RadioButtons* del mateix grup.

5.4 Pèrdua de les planificacions a l'apagar el mòbil

Un tema que no s'havia tingut en compte fins després d'haver pujat la primera versió de l'aplicació al *Google Play* ha estat la pèrdua de tota la informació de les planificacions un cop l'usuari apaga el mòbil. Això provocava que al tornar a encendre el dispositiu les planificacions actives de *System Scheduler* no es llancin. Per tal de corregir aquest problema, es va haver d'afegir la nostra aplicació al grup d'aplicacions que s'executen a l'encendre el mòbil i definir la classe i mètode de l'aplicació que es cridarà en cada encesa. Un cop definit això, només ha fet falta implementar el mètode de manera que llegís de la base de dades local les planificacions actives i les arrenqués un altre cop.

Capítol 6

Conclusions i treball futur

En aquest capítol es repassaran els objectius del projecte i la planificació inicial. A continuació es comentaran les implementacions que no s'han assolit i que s'implementaran en futures versions o les que s'han descartat.

També es farà esment d'alguna de les possibles aplicacions que podria tenir el projecte en un futur, més enllà de les accions o millores que es podrien afegir a l'aplicació.

6.1 Avaluació dels objectius i planificació

Els requeriments de l'aplicació que es van definir en un principi van ser els següents:

- Usabilitat i disseny de la interfície d'usuari.
- Planificació de les accions.
- Rangs horaris flexibles.
- Compatibilitat.
- Accions disponibles (Activar o Desactivar el so, la *Wi-fi*, el *Bluetooth*, la Conexió de dades, gestió de la lluminositat, mode avió, activar la sincronització, etc)

Tot i que no s'han arribat a implementar totes les accions que s'havien proposat en un inici, la resta d'objectius que es van plantejar s'han assolit. Algunes de les accions que no s'han implementat han estat per no ser factibles i d'altres per manca de temps degut a que es va prioritzat la millora de la interfície d'usuari i la usabilitat de l'aplicació. No obstant, les altres accions es podran afegir en fases posteriors al projecte.

Pel que fa a la planificació del projecte que es va fer en un inici, al llarg del procés s'ha anat modificant, augmentant el temps necessari. Els principals motius d'aquests canvis han estat deguts a que, tot i que es tenia experiència en programació Orientada a Objectes i en Java, no s'havia treballat mai amb l'SDK d'Android ni s'havia programat anteriorment sobre dispositius mòbils i es desconeixien els problemes intrínsecs a aquest tipus de desenvolpament.

Android és una tecnologia en canvi continu i que avança molt ràpidament. Aquest aspecte ha estat un problema a l'hora d'aprendre el llenguatge, ja que els llibres consultats en molts casos havien quedat desfassats, i moltes de les coses apreses no s'han fet servir a l'aplicació perquè havien estat substituïdes. És per això que la principal font d'informació durant el projecte ha estat la comunitat *StackOverflow* [1].

Avui en dia, el disseny de les aplicacions és un aspecte molt important, tant si aquestes van destinades a la web com a dispositius mòbils. Una bona aplicació amb un mal disseny la pot fer fracassar, i una aplicació no tant bona però amb un disseny molt acurat pot esdevenir un éxit. Així que s'ha emprat força temps a aquest aspecte, per tal que l'aplicació fos presentable per al públic.

Un dels apartats que es pensava en un primer moment que s'hauria de dedicar molt de temps per la seva complexitat, era la implementació de la planificació. No obstant, és un tema que Android té força resolt amb llibreries expressament dissenyades per als requeriments funcionals necessaris de l'aplicació. Així doncs, la part relativa a la planificació d'esdeveniments s'ha implementat de forma més ràpida i amb menys problemes del que es va pensar en un principi.

6.2 Implementacions a la pròxima versió

Les implementacions que han quedat pendents per manca de temps i que s'implementaran a curt termini dintre de les pròximes versions són les següents:

- **Desfer (undo) a l'esborrar planificacions**. Substituir el missatge de confirmació en el moment d'esborrar planificacions pel la implementació de desfer l'esborrat (exemple: Gmail).

- ***TimePicker* propi**. Implementar un *TimePicker* propi mentre no es solucioni el bug d'Android que no permet cancelar mentre s'està seleccionant una data.

[1] `http://www.stackoverflow.com`

- ***RingTonePicker***. A les accions de les planificacions, en comptes d'oferir l'opció d'activar o desactivar el so, s'oferirà una finestra modal amb un *RingTonePicker* per poder seleccionar el nivell de so que desitgem planificar, fent més flexible aquesta opció.

- **Planificacions amb més d'una acció**. Possibilitat de guardar més d'una acció dintre d'una mateixa planificació. Així, si quan arribem a la feina desitgem desactivar el so i encendre la *Wi-fi* ho poguem fer en una mateixa planificació i no un acció per cada planificació com passa ara.

- **Planificacions amb àlies**. Possibilitat de donar un nom o àlies a les planificacions per tal que l'usuari les pugui identificar i distingir millor.

- **Excepcions**. Es podran definir excepcions sempre que la planificació treballi en un rang horari, és a dir tingui hora d'inici i final.

- **Configuració**. Fer possible que l'usuari pugui configurar-se alguns aspectes de l'aplicació, com ara si vol rebre notificacions quan s'executi una planificació o no.

Pel que fa a les accions que es van valorar en un principi, algunes s'han descartat i d'altres s'afegiran més endavant:

- Gestió de la brillantor o il·luminació: Es podrà definir un percentatge d'il·luminació que s'aplicarà al dispositiu.

- Gestió del volum per percentatge. Es podrà definir un percentatge de so que s'aplicarà al dispositiu.

- Mode avió (activar/desactivar). Aquesta opció està bloquejada a partir de la versió 4.2.x per gestionar-la des d'aplicacions. Només es possible gestionar aquesta opció de forma manual.

- Gestió del fons de pantalla.

6.3 Més enllà de *System Scheduler*

System Scheduler ha estat pensada com una aplicació que permet planificar esdeveniments del dispostiu en la qual es poden introduir cada cop més accions per fer-al més versàtil. No obstant, pensant més enllà, *System Scheduler* podria ser un mòdul d'entre altres dintre d'una aplicació més gran que servís

per a la gestió i configuració a distància de no només un mòbil, sinó del conjunt de mòbils d'una empresa o entitat. *System Scheduler* funcionaria com un mòdul encarregat de les planificacions d'esdeveniments sobre el dispositius mòbils vinculats al compte de l'usuari. Les planificacions es podrien fer des del mòbil o des de la interfície web de forma remota. Aquesta aplicació més gran seria una aplicació web on un usuari faria login a través del seu compte de Google (o Gmail) i podria veure un llistat dels seus dispositius mòbils als que els podria enviar accions com les que s'han implementat al projecte (activar/desactivar so, *Wi-fi* o *Bluetooth*, connexió de dades, etc...), instal·lar aplicacions de *Google play* de forma remota, o bé veure l'estat o les característiques dels mòbils.

Entre algunes de les funcionalitats possibles estarien el control remot, per tal de veure en directe la pantalla del mòbil i interactuar-hi, o bé la localització del mòbil en un mapa via GPS o fer un fotografia de la càmara com a funcionalitats anti-robatori.

Bibliografia

- Head First Android Development.

  Autor: Jonathan Simon. Editorial: O'Reilly. Primera Edició: Octubre 2011

- StackOverflow. `http://www.stackoverflow.com`

- Curs d'Android de Cristian Tanàs. `http://www.deic.uab.cat/docencia/viewprog.php?codias=99002-0`

- Video-presentació feta per Roman Guy @ GoogleIO 2009 que mostra com funcionen els Adapters i com fer-ne un ús correcte a les aplicacions per millorar l'eficiciència. `http://www.google.com/events/io/2009/sessions/TurboChargeUiAndroidFast.html`.

- Vogella. `http://www.vogella.com/android.html`

- Android Design Patterns. `http://www.androiddesignpatterns.com/2012/07/loaders-and-loadermanager-background.html`

- ActionBarSherlock. `http://actionbarsherlock.com`