Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Enginyeria tècnica en informàtica de gestió

APLICACIÓ SMARTPHONE RÀDIO PALAFOLLS

Memòria

SERGI ROJAS

PONENT: ANTONI SATUÉ VILLAR

TARDOR 2012

TecnoCampus Mataró-Maresme

Dedicatòria

Dedico aquest projecte a tots els que han intervingut en el meu procés acadèmic.

Agraïments

Tenen el més gran i etern agraïment els meus pares, Josep Rojas i Claudia Serra, que han fet possible que pugui cursar els estudis que he volgut posant al meu abast tots els seus medis.

A l'empresa BeeCubu per ajudar-me tot el temps que he necessitat en el procés d'elaboració d'aquest projecte.

A Manu Diaz, que sempre que he tingut qualsevol dubte m'ha donat suport i solucions.

Als meus amics més propers, que més que un grup d'amics, són una família.

A tots, moltes gràcies.

Resum

Ràdio Palafolls per *smartphone* és una aplicació dirigida a tots els usuaris d'aquesta sintonia posseïdors d'un dispositiu amb sistema operatiu Android.

Aquest projecte permet escoltar en directe Ràdio Palafolls mentre el mòbil està en repòs, mentre es miren les ultimes notícies de Palafolls o mentre es consulta la graella de la ràdio. També permet unir-se a les xarxes socials de Ràdio Palafolls per estar actualitzat de l'última hora del municipi.

Resumen

Radio Palafolls para *smartphone* es una aplicación dirigida a todos los usuarios de esta sintonía poseedores de un dispositivo con sistema operativo Android.

Este proyecto permite escuchar en directo Radio Palafolls mientras el móvil está en reposo, mientras se miran las ultimas noticias de Palafolls o mientras se consulta la parrilla de la radio. También permite unirse a las redes sociales de Radio Palafolls para estar actualizado de la última hora del municipio.

Abstract

Radio Palafolls is an application for smartphone aimed at all users of this dial that possess a device with Android OS.

This project allows you to listen live Radio Palafolls while the phone is idle, while watching the latest news of Palafolls or consult the grid. It also allows you to join social networks Radio Palafolls to watch the last hour of the municipality.

Índex.

Índex de figures.

1. Objectius

1.1. Propòsit

Aprendre a programar aplicacions per Android. Conèixer les llibreries de Java que s'usen per programar en Android i conèixer l'entorn de desenvolupament per aplicacions Android. Arribar a un nivell mínim de coneixement per sentir-me capaç d'encarar qualsevol aplicació en un futur.

1.2. Finalitat

Explicar detalladament el funcionament tant intern com funcional de l'aplicació de Ràdio Palafolls. Proporcionar la informació suficient com per entendre tot el codi desenvolupat i saber modificar des del servidor web el contingut que mostra l'aplicació.

1.3. Objecte

Una aplicació per Android perquè els oients de ràdio Palafolls puguin escoltar la sintonia mitjançant el seu *smartphone*. Alhora, proporcionar una eina perquè els usuaris puguin estar informats en tot moment de l'actualitat de Palafolls.

1.4. Abast

Una aplicació dinàmica i viva, que mantingui a l'usuari informat sobre l'actualitat de Palafolls des de qualsevol punt del món mitjançant una connexió a internet.

A nivell de programador, entregar una aplicació que sigui prou modulable i personalitzable (en quant a contingut) com per poder fer canvis a la programació, horaris, programes i notícies que es volen publicar sobre Palafolls.

2. Informació sobre la realització del projecte

El projecte s'ha desenvolupat com a conseqüència de l'assignació del projecte proposat per l' Escola Universitària Politècnica de Mataró, que s'ha posat d'acord amb Ràdio Palafolls per tal de desenvolupar una aplicació per *smartphone*, i de la voluntat de l'estudiant que ha utilitzat aquest projecte final de carrera per aprendre a programar aplicacions per Android.

L'objectiu del projecte és entregar una aplicació per *smartphone* fàcil de mantenir i que tot el seu contingut sigui actualitzable via web.

Les funcionalitats que ha de satisfer l'aplicació un cop finalitzada han de ser:

- Permetre escoltar Ràdio Palafolls des d'un *smartphone* Android.
- Consultar les últimes notícies de Palafolls.
- Poder escoltar els dos últims plens municipals de Palafolls.
- Poder escoltar l'últim informatiu de Ràdio Palafolls a qualsevol hora.
- Unir-se a les xarxes socials de Ràdio Palafolls.

L' aplicació de Ràdio Palafolls consta de 5 blocs molt diferenciats i independents, la qual cosa permet desenvolupar-los de forma independent si hagués estat necessari.

2.1. Recopilació d'informació, estudi i disseny de la solució

S'han descarregat diverses aplicacions de ràdio i analitzat la navegabilitat per les mateixes. Paral·lelament s'ha parlat amb la direcció de Ràdio Palafolls sobre quina opció els agradava més a l'hora de desenvolupar l'aplicació i finalment s'ha optat per a una aplicació que permeti la navegació per pestanyes per passar d'una funcionalitat a una altra.

Després d'entrevistes amb diversos programadors d'aplicacions, s'ha decidit que es programarà l'aplicació amb el IDE (*Integrated Development Environment*) *Eclipse*, ja que tots coincidien en utilitzar el mateix entorn.

3. Estudi de mercat

3.1. Ràdios al Google Play

Aquest projecte s'executa per un encàrrec de tercers, i com en molts altres aspectes, hi ha moltes especificacions que venen dictades des de Ràdio Palafolls, com les funcionalitats, la navegabilitat i el disseny. Totes aquestes especificacions són fruit d'un estudi de mercat fet per part de Ràdio Palafolls.

La direcció de Ràdio Palafolls ha estat molt clara en els exemples ha seguir que són disponibles al *Android Market*. L'aplicació de RAC1 és el model a seguir, tant en algunes de les funcionalitats com en la navegabilitat.

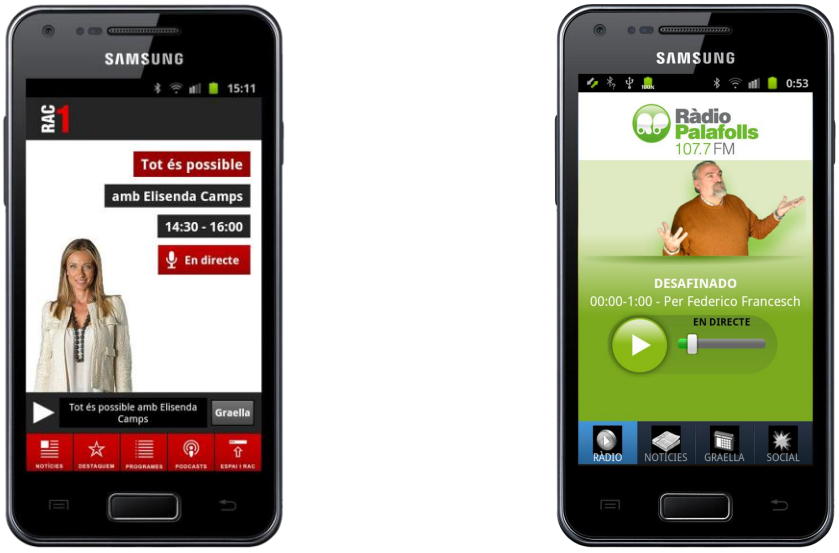

Fig. 3.1. Comparació entre RAC1 i Ràdio Palafolls

Tot i així es va decidir mirar algunes altres aplicacions per tal d'incorporar només funcionalitats a Ràdio Palafolls. Algunes de les aplicacions mirades són Catalunya ràdio, EUROPAFM i RNE entre d'altres. Tot i que no s'han incorporat en el projecte, s'han agafat idees per properes actualitzacions.

De l'aplicació Catalunya ràdio, s'ha agafat la idea de poder compartir les notícies mitjançant les xarxes socials (figura 3.2) a les quals l'usuari estigui donat d'alta.

Fig. 3.2. Detall d'una notícia de Catalunya ràdio.

De l'aplicació EUROPAFM, s'ha copiat el format de mostrar la programació d'un dia en concret (figura 3.3). Es fa amb un *listview* on cada fila de la llista és un programa.

Fig. 3.3. Detall de la programació d'un dia d' EUROPAFM

4. Metodologia i tecnologia

4.1. Entorn de desenvolupament

El projecte està desenvolupat amb els llenguatges de programació Java i XML. El primer és per a desenvolupar l'aplicació per Android, el segon és el llenguatge dels documents online dels quals l'aplicació es proveeix d'informació.

A continuació s'especifica tot el software utilitzat per a la realització del projecte:

- Sistema operatiu Mac OSX.
- Processador de text Microsoft Word for Mac 2011.
- IDE *Eclipse*.
- Emulador de mòbils Android.
- Mozilla Firefox.

Abans de començar a programar s'han especificat, juntament amb la direcció de ràdio Palafolls, els requeriments funcionals que ha de complir l'aplicació i s'han desenvolupat els conseqüents casos d'ús.

El software per desenvolupar l'aplicació ha estat *Eclipse*. Per modificar els arxius XML que estan penjats a la web, s'usa un editor de text i es pengen al servidor web mitjançant una connexió FTP.

L'aplicació s'ha programat en Java, que és el llenguatge més comú a l'hora de programar per Android. Aquest usa la programació orientada a objectes, i el patró MVC (Model Vista Controlador). Aquest patró per definició aïlla el model de dades, la interfície usuari i la lògica de control.

4.1.2. Requisits tecnològics

Realment, no és necessari res més que un editor de text per desenvolupar la totalitat de l'aplicació, un entorn de programació integrat però, facilita enormement la feina, ja que juntament amb un emulador i un visor gràfic, es pot anar testejant l'aplicació mentre es desenvolupa.

4.2. Sistema operatiu Android

Android és un sistema operatiu mòbil basat en Linux, orientat a ser utilitzat per *smartphones*, tablets, Google Tv i altres dispositius. És desenvolupat per la companyia Open Handset Alliance, que pertany a Google.

L'arquitectura d'aquest sistema operatiu (fig. 4.1), està distribuïda en diverses capes, cada una d'elles té les seves pròpies característiques i propòsits. Aquestes capes són: Aplicacions, Marc de treball de les aplicacions, biblioteques, Android runtime i el nucli de Linux.

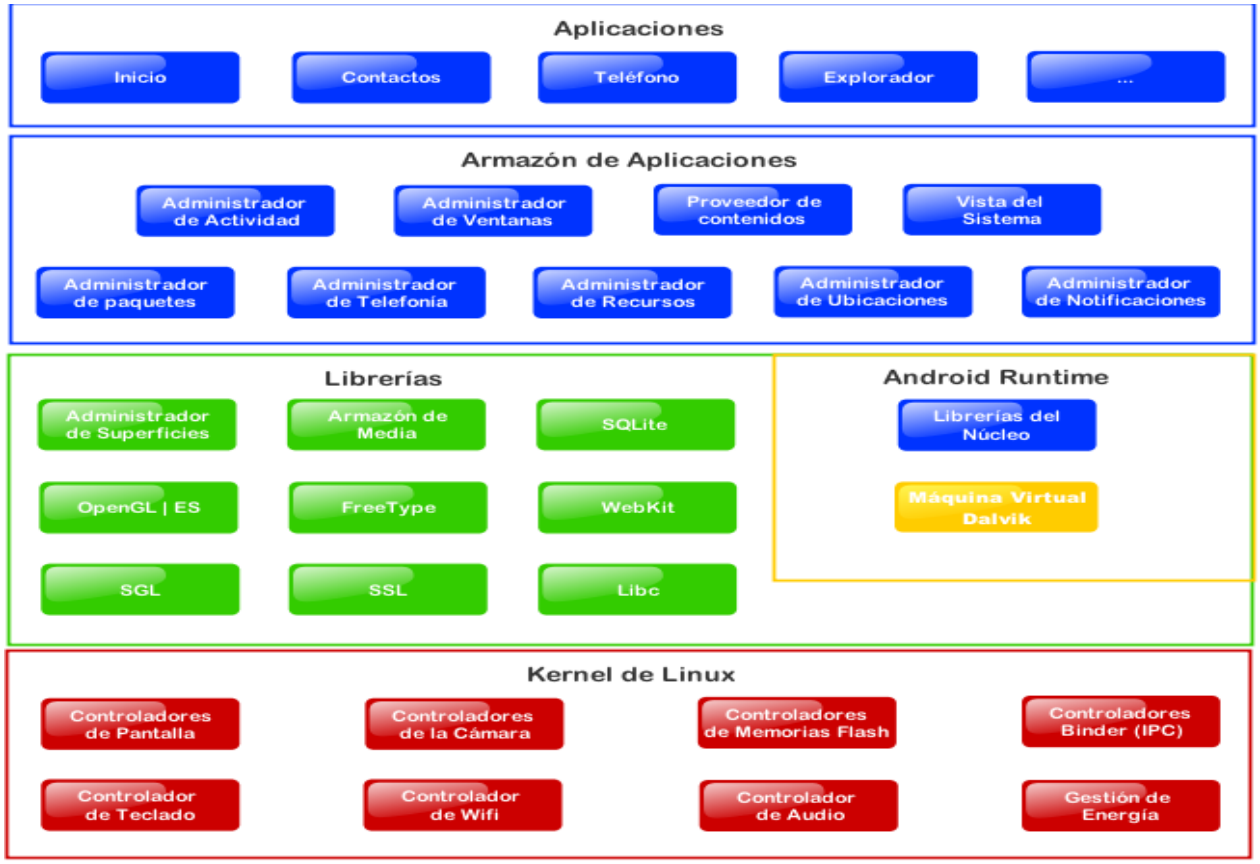

Fig. 4.1. Diagrama de les capes d'Android.

4.2.1. Kernel de Linux

Linux és un sistema d'alta seguretat provat en diversos entorns des de fa dècades. Per això, Android ha escollit Linux com a base primària pel seu sistema operatiu, encara que les aplicacions Android s'executen de manera independent a als processos Linux del sistema.

4.2.2. Framework d'aplicacions

Representa fonamentalment el conjunt d'eines de desenvolupament de qualsevol aplicació. Totes les aplicacions que es desenvolupin per Android, ja siguin del propi dispositiu, desenvolupades per Google o terceres companyies, utilitzen el mateix conjunt de API i el mateix *framework*.

És aquest conjunt d'eines a partir del qual els desenvolupadors poden explotar la seva creativitat per llençar noves funcionalitats.

També és en aquesta capa on es poden trobar diverses llibreries de Java especialment creades per Android, a més a més, es poden trobar *services* com els sensors, el *Wi-Fi*, servei telefònic i accés a la informació d'altres aplicacions entre d'altres.

4.2.3. Llibreries

La següent capa es correspon amb les llibreries utilitzades per Android. S'han escrit en codi C/C++ i proporcionen a Android la majoria de les seves capacitats més característiques.

4.2.4. Aplicacions Android

Aquest nivell conté, les aplicacions incloses per defecte d'Android com aquelles que l'usuari afegeix posteriorment, ja siguin de terceres empreses o de desenvolupament propi. Totes les aplicacions utilitzen les API, els serveis i les llibreries explicades anteriorment.

Aquestes aplicacions es desenvolupen en Java juntament amb *Android SDK*, però també és possible desenvolupar-les amb altres eines com el Kit de desenvolupament nadiu per aplicacions amb extensions en C, C++ o Google App Inventor.

Les aplicacions estan distribuïdes en *Activities*, que fan referència als casos d'ús de l'aplicació. En el cas de Ràdio Palafolls però, només s'usa una *Activity* per a tota la aplicació pràcticament. Només es criden a noves *Activities* quan s'utilitzen altres aplicacions (Facebook, Twitter o Navegador web).

Quan s'executa una *Activity*, dins seu hi ha un fil d'execució (figura 4.2), és important que el desenvolupador conegui el seu estat en tot moment. Ja que així es podran definir quins són els

mètodes que es cridaran quan s'invoqui a l'*Activity* per primera vegada, quan aquesta està en segon pla, pausada, o quan torna a la aplicació després d'estar en un segon pla.

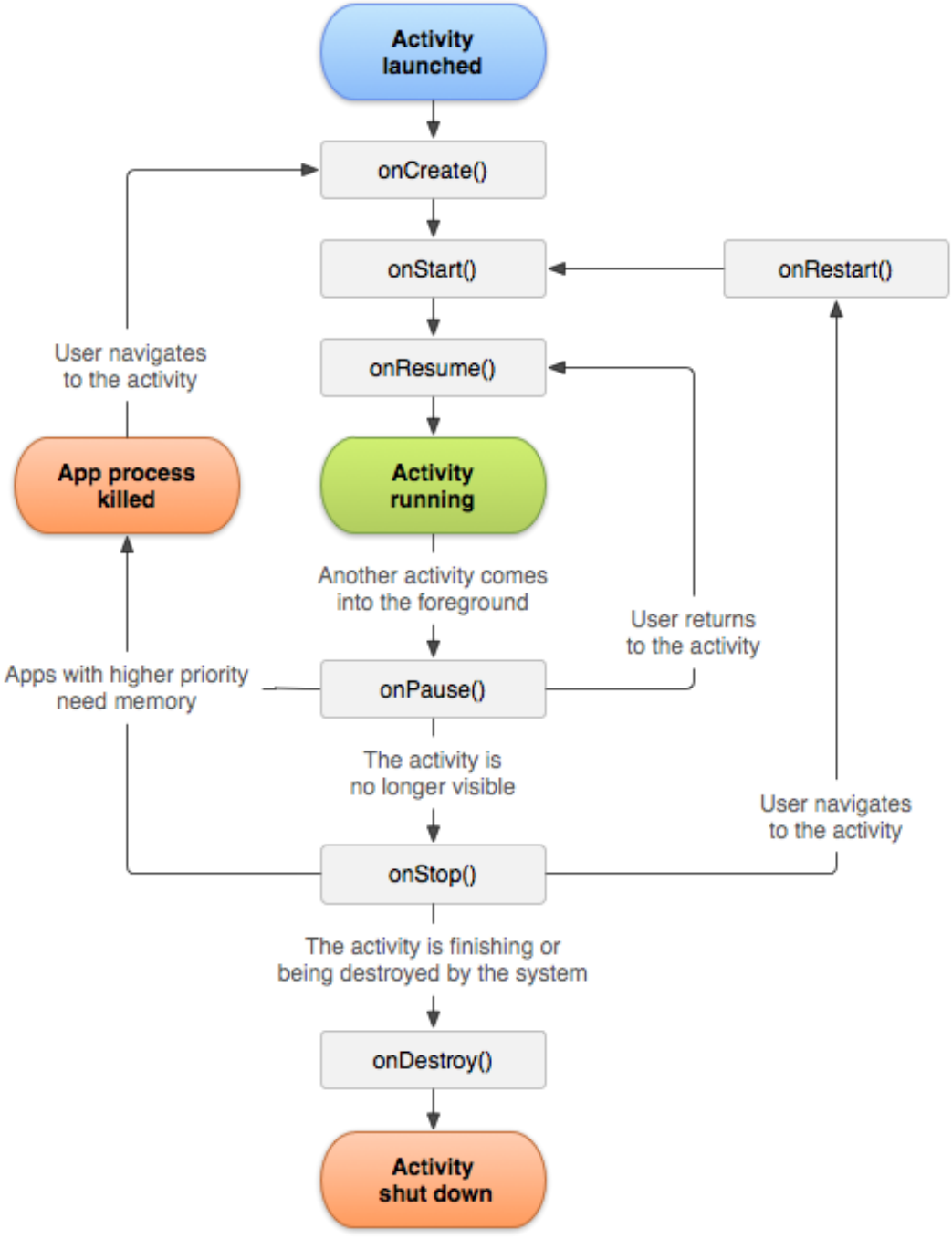

Fig. 4.2. Diagrama del cicle de una *Activity*.

4.3. Eines de programació (Eclipse)

Eclipse és una eina que integra l'editor de codi (figura 4.3) i l'editor d'interfícies gràfiques (figura 4.4) així que no cal cap eina més per desenvolupar l'aplicació. *Eclipse* també gestiona els emuladors per tal de poder provar l'aplicatiu, tot i que no el té integrat (figura 4.5).

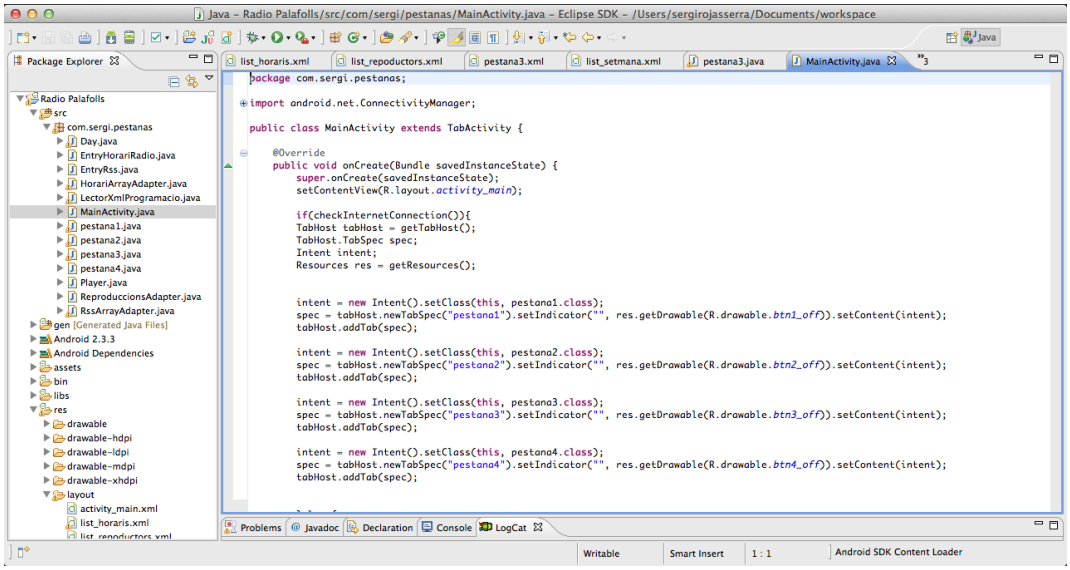

Fig. 4.3. *Eclipse* amb visualització d'entorn de codi.

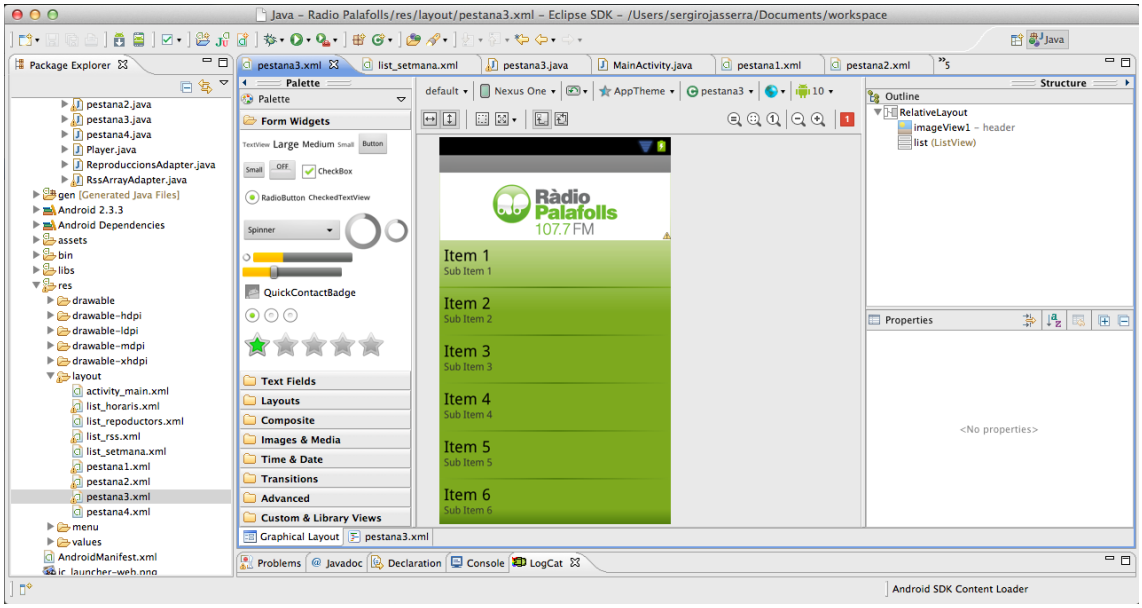

Fig. 4.4. *Eclipse* amb visualització d'entorn d'editor d'interfícies gràfiques

Fig. 4.5. Visualització d'entorn de gestor d'emuladors i l'emulador.

Consultant estadístiques oficials d'Android s'ha decidit desenvolupar l'aplicació per la versió 2.3 ja que és la més utilitzada (figura 4.6); totes les versions posteriors també poden utilitzar l'aplicació.

Version	Codename	API	Distribution
1.5	Cupcake	3	0.1%
1.6	Donut	4	0.3%
2.1	Eclair	7	2.7%
2.2	Froyo	8	10.3%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	50.6%
3.1	Honeycomb	12	0.4%
3.2		13	1.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1	Jelly Bean	16	5.9%
4.2		17	0.8%

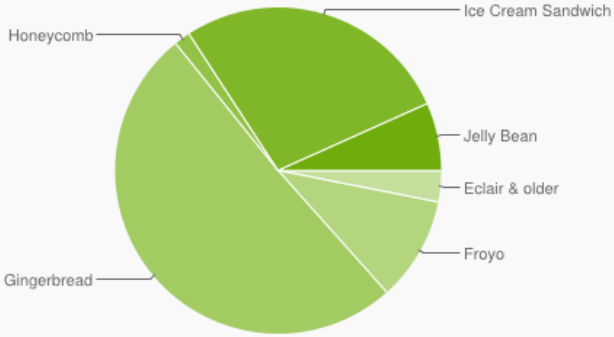

Fig. 4.6. Resum de dades d'utilització de les plataformes d'Android.

Les dades requerides per calcular els gràfics de la Figura 4.6. van ser agafats des del 19-11-2012 fins el 3-12-2012.

4.4. Llenguatges de programació

Com ja s'ha comentat en aquest projecte, s'usen bàsicament dos llenguatges de programació:

- Java
- XML

4.4.1. Java

És un llenguatge de programació originalment desenvolupat per James Gosling, de *Sun Microsystems* (que posteriorment va ser adquirida per *Oracle*) i publicat el 1995. La sintaxis de Java deriva de *C* i de *C++*, tot i que té menys facilitats a baix nivell. Les aplicacions Java són generalment compilades per *bytecode* (classe Java) que pot ser executada en qualsevol màquina virtual Java, sense tenir en compte l'arquitectura de la computadora. Java és un llenguatge de programació concurrent, orientat a objectes i basat en classes.

4.4.2. XML

És un metallenguatge extensible, d'etiquetes, desenvolupat per *World Wide Web Consortium* (W3C). És una simplificació i adaptació de SGML, i permet definir la gramàtica de llenguatges específics. Per tant, XML no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. XML es proposa com un estàndard per l'intercanvi d'informació estructurada entre diverses plataformes. Es pot utilitzar per moltes aplicacions diverses com bases de dades, fulls de càlcul, o com és el cas, en intercanvi d'informació entre plataformes de servidor Web i app mòbil.

5. Desenvolupament

Per programar l'aplicació s'ha decidit utilitzar el kit de desenvolupament software (SDK) *Eclipse*, ja que és un dels desenvolupadors més normalitzats, i això facilita l'accés a la informació.

5.1. Desenvolupament de l'aplicació

La implementació dels casos d'ús es pot realitzar de manera independent, de forma que es poden tractar com mòduls separats, però és necessari un entorn on l'usuari pugui navegar entre tots els casos d'ús.

Segons el que s'ha acordat amb el client, s'utilitza un sistema de pestanyes per navegar entre els casos d'ús. Aquestes pestanyes contenen els títols i les icones corresponents.

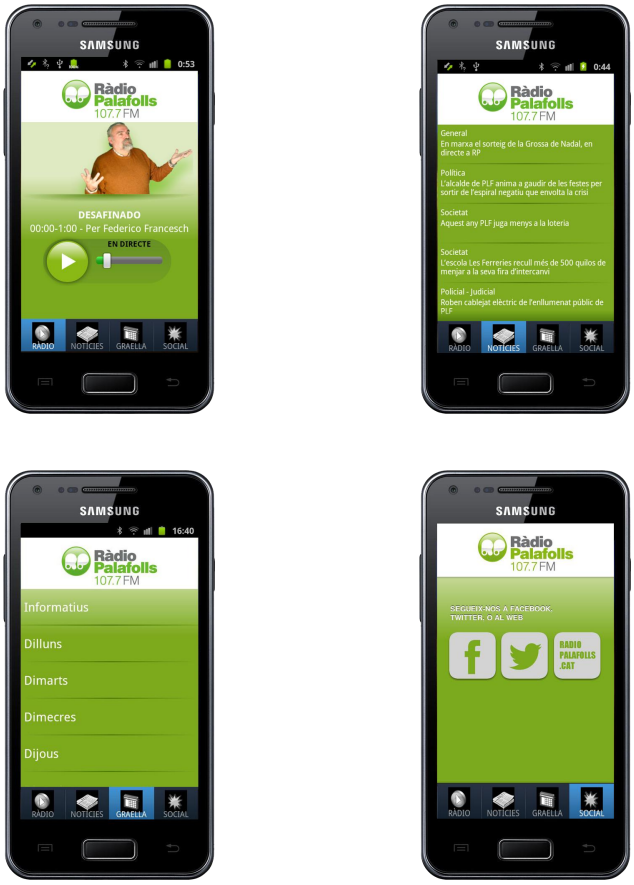

Fig. 5.1. Comparació de les pantalles principals dels blocs funcionals

Totes les imatges de l'aplicació es troben en format PNG (*Portable Network Graphics*). El desenvolupament per Android permet treballar per quatre resolucions (figura 5.2), per adaptar-se a la resolució de pantalla diferents dispositius que usen la plataforma. En l'aplicació Ràdio Palafolls totes les imatges internes (les que no es llegeixen de la web) es troben en dues resolucions, hdpi i mdpi. El client ha escollit treballar en aquestes dues densitats de pantalla ja que són dues de les més usades.

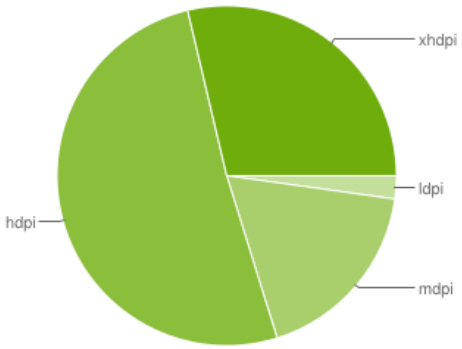

Fig. 5.2. Gràfic de la utilització de les diferents densitats de pantalla existents.

S'ha acordat amb el client que les imatges mostrades en la figura 5.3. són les úniques que es guardarien en local, ja que les imatges corresponents a cada programa de ràdio es llegeixen del servidor. Això fa que el client pugui canviar la imatge del programa de ràdio, afegir programes i treure'n sense que s'hagi de modificar el codi de l'aplicació. S'ha anteposat la modulabilitat de l'aplicació per davant la velocitat de carga.

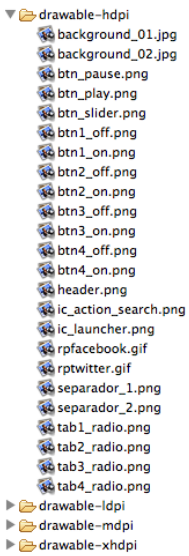

Fig. 5.3. Detall de les diferents densitats de pantalla utilitzades.

5.2. Classes

El codi està distribuït en diferents classes:

- Day.java
- EntryHorariRadio.java
- EntryRss.java
- HorariArrayAdapter.java
- LectorXmlProgramacio.java
- MainActivity.java
- Pestana1.java
- Pestana2.java
- Pestana3.java
- Pestana4.java
- Player.java
- ReproduccionsArrayAdapter.java
- RssArrayAdapter.java

5.2.1. Day.java

És la classe delegada de la gestió de tot el relacionat amb el temps. És el mediador per aconseguir l'hora real (hora que marca el dispositiu mòbil)

Quan l'aplicació necessita saber el dia, hora i minut real, és aquesta classe l'encarregada de proporcionar aquesta informació a les altres classes. Per tal de mostrar la informació del programa de ràdio que sona en cada moment, es crida a aquesta classe. També es crida quan s'han de ensenyar els horaris ja que es mostren per dies i s'ha de saber quin dia s'ha de imprimir per poder donar el resultat correcte.

És una classe usada tant per la classe *Pestana1* com per *LectorXmlProgramació*, aquesta última l'usa en el seu mètode per comparar l'hora del dispositiu amb els horaris de la graella i escollir el programa que s'emet al moment.

Fig. 5.4. Detall de l'ús aplicat de la classe *Day*

5.2.2. LectorXmlProgramacio.java

És una classe creada per llegir els arxius .xml de la programació de la ràdio.

Tota la informació que utilitza l'aplicació i no és interna d'aquesta, es llegeix d'Internet mitjançant documents .xml. Aquesta classe consta d' uns mètodes que es criden recursivament per tal d' analitzar i guardar en una taula d'objectes *EntryHorariRadio* tots els programes de la ràdio, el seu locutor, el seu horari i la URL on s'allotja la imatge de cada programa.

Aquesta classe també és l'encarregada de donar la URL on hi ha l'arxiu .xml del dia corresponent.

5.2.3. ActivitatPrincipal.java

Classe principal i la primera que s'obre al executar l'aplicació. És la classe contenidora de les pestanyes que permeten la navegabilitat per tota l'aplicatiu.

Comprova si hi ha connexió a internet, si n'hi ha, engega l'aplicació amb normalitat. Si no n'hi ha, llença un avís per pantalla dient que és necessària la connexió a internet per tal d'executar l'aplicació i tanca la mateixa.

Fig. 5.5. Detall de una de les funcions de la classe *ActivitatPrincipal*

5.2.4. Pestana1.java

És una extensió de la classe Activity ja que és una activitat.

Visualment, és la primera pantalla que apareix. Gestiona el reproductor de la ràdio consultant al patró Singletó del reproductor, des d' aquesta pestanya s'encén i s'apaga la ràdio.

També consulta les classes Day.java i LectorXmlProgramacio.java per poder mostrar en pantalla la informació del programa que s'està emetent al moment.

5.2.5. Pestana2.java

És una extensió de la classe *ListActivity*.

En aquesta classe es llegeixen les notícies RSS de ràdio Palafolls. Si es decideix llegir una notícia llença el navegador web per la pàgina de la notícia corresponent.

Les notícies RSS arriben en un arxiu .xml, per tal de poder llegir les notícies d'una manera visualment correcta. S'ha d'analitzar aquest arxiu, per això pestana2.java té una classe interna encarregada d'analitzar aquest document .xml i guardar-lo en una taula d'objectes *EntryRSS*.

Pestana2.java llegeix tota la informació d'Internet, el codi que s'encarrega de fer-ho és mostrat a la figura 5.6.

```
String str = "";
String imprimeix = "";
LlegirRss lector = new LlegirRss();
InputStream fin;
OutputStream out = null;

try {
    URL url = new URL("http://www.radiopalafolls.cat/feed/");
    BufferedReader in = new BufferedReader(new InputStreamReader(
            url.openStream()));
    while ((str = in.readLine()) != null) {
        imprimeix = imprimeix + str;
        str = null;
    }
    in.close();
    OutputStream fout = openFileOutput("rssOnline.xml",
            MODE_WORLD_READABLE);
    fout.write(imprimeix.getBytes());
    fout.close();
    fin = openFileInput("rssOnline.xml");

    llegit = lector.parse(fin);
    llegit.remove(llegit.size() - 1);
```

Fig. 5.6. Codi encarregat de passar el arxiu .xml extern a un arxiu local

Com es pot observar en la figura 5.6, amb l'ús de les classes *URL*, *BufferedReader* i *String*, es passa de tenir un arxiu allotjat en una direcció web a tenir-ho tot en un sol *String*, mitjançant la classe BufferedReader. Él següent és crear un directori local, amb la classe *OutputStream*, on poder bolcar tot el *String* "imprimeix" en un document per poder ser llegit posteriorment per la classe *InputStream*, que és la classe que es passa per paràmetre a un mètode de la classe privada *LlegirRss*, que s'encarrega de analitzar i guardar les diferents notícies en una taula d'objectes *EntryRss*.

Tot aquest procediment es produeix dins d'una classe extensora de *AsyncTask*, ja que la velocitat de l'aplicació depèn de la velocitat de connexió a la xarxa, i així la descàrrega es produeix en un segon pla.

5.2.6. Pestana3.java

És una extensió de la classe *ListActivity*.

Aquesta classe té dues funcions, la primera, gestionar la impressió per pantalla de la graella d'horaris i programes de Ràdio Palafolls, i la segona, gestionar l'escolta de plens municipals i de l'últim informatiu d'actualitat.

Consulta al patró Singletó de la classe *Player* i l'utilitza per tal d'escoltar els plens i informatius usant una instància d'aquesta classe. Gestiona totes les incoherències i possibles errors que puguin haver amb la instància de la classe *Player* usada en altres classes del projecte.

La graella d'horaris que mostra no està guardada en local, així que s'ha de llegir d'internet amb un mètode molt similar al de la classe *Pestana2.java* explicat en el punt 5.6.5. Al igual que en la classe *Pestana2.java*, *pestana3.java* també gaudeix d'una classe privada extensora de *AsyncTask* per tal de treballar en un segon pla, ja que ha de descarregar les imatges de cada programa i això fa que baixi la velocitat de l'aplicació.

5.2.7. Pestana4.java

És una extensió de la classe *Activity*.

Aquesta classe gestiona l'adhesió del usuari a xarxes socials. Per pantalla es mostren dos botons, un de Twitter i un altre de Facebook, ambdós obren les respectives xarxes socials per la pàgina de Ràdio Palafolls. Al apretar el botó, tant per Twitter com per Facebook, l'aplicació intenta obrir les aplicacions oficials de les respectives xarxes socials, si l'usuari no les té instal·lades en el dispositiu mòbil, des de l'aplicació, s'obre una pàgina de navegació web per la pàgina de Ràdio Palafolls.

5.2.8. Player.java

És una extensió de la classe *MediaPlayer* i implementa la *MediaPlayer.OnPreparedListener*.

Conté la implementació del patró Singletó, que s'utilitza per tenir només una instància de la classe Player en tota la aplicació, així s'eviten problemes de solapament de so. També gestiona el *Listener* que s'activa quan la càrrega del *Buffer* del *Player* ja s'ha realitzat.

Com l'activitat principal de la classe *Player* és escoltar Ràdio Palafolls, s'ha implementat un mètode dins d'aquesta classe per tal de poder carregar el *buffer* des de qualsevol altre classe del projecte i agilitzar el procés.

5.2.9. ArrayAdapters

Les classes extensores de *ArrayAdapter* són *HorariArrayAdapter*, *ReproduccionsArrayAdapter*, *DiesArrayAdapter* i *RssArrayAdapter*.

Els *ArrayAdapters* són classes que reben una taula d'objectes i l'adapten per poder ser mostrades en una llista per pantalla.

Totes tres tenen la mateixa estructura, implementen un constructor i el mètode *getView(...)*. El constructor inicialitza el Context en el qual es mostrarà la informació i la taula d'objectes que s'usarà per omplir la llista que es mostrarà per pantalla. El mètode *getView(...)* adapta per cada objecte de la *List* una fila de la llista que es mostrarà en pantalla, col·locant els paràmetres dels objectes de la *List* d'una manera personalitzada en cada fila de la llista.

Fig. 5.7. Detall de quatre resultats de la funció *getView(...)*

5.2.10. Classes *Entry*

Les classes *Entry* són *EntryRss* i *EntryHorariRadio*.

Aquestes classes han estat creades per poder tractar com a objectes els *tag* dels arxius .xml.

En el cas de la figura 5.8. els *tag* que interessa tractar com un objecte és el "track". Cada "track" és un programa de ràdio, així doncs, un objecte *EntryHorariRadio* conté la mateixa informació que conté un "track". Per tant, una taula d'objectes *EntryHorariRadio* conté la mateixa informació que tot el document .xml.

```xml
<?xml version="1.0" encoding="utf-8"?>
    <jornada>
    <track>
        <titol>CAP DE SETMANA</titol>
        <locutor>Selecció</locutor>
        <horari>1:00-6:00</horari>
        <imatge>http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/seleccio-musical.png</imatge>
    </track>


    <!-- programació dia -->

    <track>
        <titol>BON DIA</titol>
        <locutor>Selecció musical matinal</locutor>
        <horari>6:00-8:00</horari>
        <imatge>http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/bon-dia.png</imatge>
    </track>

    <track>
        <titol>TIC TAC MÚSICA</titol>
        <locutor>...</locutor>
        <horari>8:00-9:00</horari>
        <imatge>http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/helena-carles.png</imatge>
    </track>
```

Fig. 5.8. Detall del codi d'un arxiu .xml

Un cop es disposa de la informació del *tag* en una classe *Entry*, es pot accedir més fàcilment a la informació continguda amb mètodes *get*.

El mateix passa en el cas de la classe *EntryRss*, on "item" és el *tag* que interessa tractar com a objecte.

5.3. Blocs Funcionals

L'aplicació de Ràdio Palafolls per *smartphone* es divideix en quatre grans blocs funcionals. Aquests són independents entre sí.

Tots els blocs estan units per la classe *ActivitatPrincipal*, ja que aquesta és el contenidor del menú de pestanyes que s'usa per navegar per tota l'aplicació. Igualment tots els blocs funcionals tenen implementada la funció *onBackPressed()* per tal de apagar el reproductor de la ràdio, ja que al poder reproduir-se en segon pla, es pot estar reproduint la Ràdio Palafolls des de qualsevol bloc de l'aplicació

5.3.1. Ràdio

El primer gran bloc funcional, i el que dóna nom a l'aplicació. Consta d'una sola pantalla. Permet escoltar ràdio Palafolls en directe. Com es veu a la figura 5.9 també té la funcionalitat afegida de poder controlar el volum del mòbil des de la *seekBar* que es veu per pantalla.

Fig. 5.9. Detall de la *seekBar*

Aquest bloc, mostra en pantalla tota la informació del programa (imatge, nom, horari i locutor). Tota la informació que mostra de cada programa, es llegeix de un document online que no pertany a l'aplicació, per tant, modificant aquest document XML, es pot modificar el contingut de l'aplicació.

Aquesta funcionalitat necessita connexió a internet per tal de poder fer qualsevol de les seves funcions.

L'aplicació permet escoltar la ràdio mentre es navega per altres aplicacions del dispositiu Android, ja que quan es tanca l'aplicacio amb el botó *home* (fig. 5.10) no es deixa de reproduir l'àudio de Ràdio Palafolls, ja que el botó *home* invoca el mètode *onPause()* i en aquest cas l'aplicació no l'implementa. No és el cas del botó *back* (fig. 5.10) que sí fa que deixi de reproduir-se Ràdio Palafolls, ja que invoca el mètode *onBackPressed()* i aquest sí està implementat per tal d'apagar la reproducció al sortir de l'aplicació.

Una de les funcions és el fet de reproduir en *streaming* ràdio Palafolls. Per això s'utilitza una instància de la classe *Player*, per configurar la URL de la qual s'escolta la ràdio, amb aquesta finalitat s'usa la funcio *setDataSource(...)* de la classe *Player*, en aquesta funció es passa per paràmetre la direcció web, aleshores el objecte *Player*, passa a l'estat *initializated* com es pot veure a la figura 5.11.

Fig. 5.10. Detall dels botons *menú*, *home* i *back* respectivament

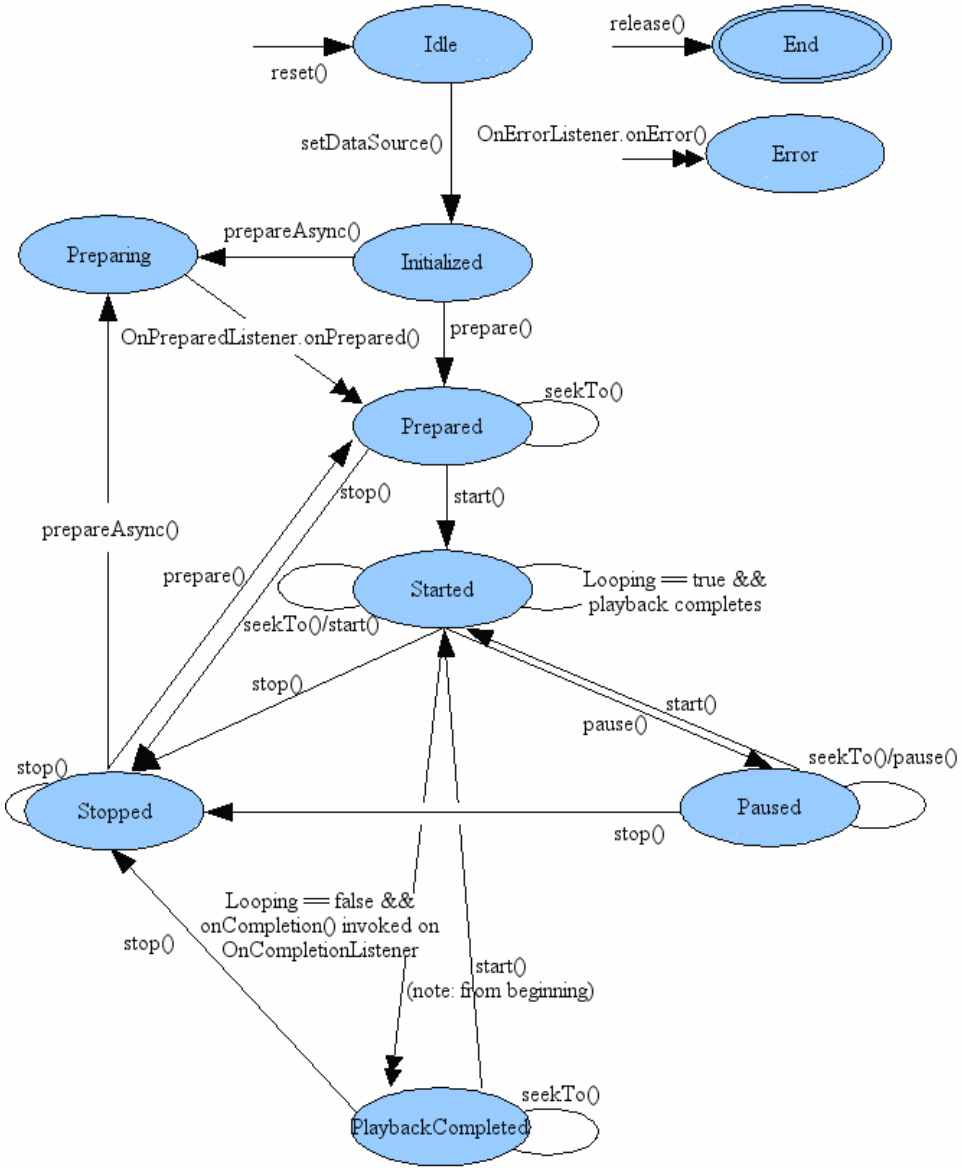

Fig. 5.11. Diagrama d'estats del objecte *Player*.

Un cop l'objecte *Player* està en estat inicialitzat, en aquesta aplicació s'usa el mètode *preareAsync()* que permet preparar el *Player* en un segon pla i així poder carregar els altres objectes de la classe. Un cop està preparat (estat *prepared*) es crida automàticament a un mètode *listener*, que modifica el paràmetre *alpha* del botó play/stop (aquest botó està directament lligat al estat del *Player*), el fa menys transparent, donant sensació que ja és pot prémer i a l'hora el torna clicable.

Quant el *Player* s'està reproduint (estat *started*) s'ha de prémer el botó play/stop, per tal de parar la reproducció de la ràdio i passar el *Player* al estat *stopped* mitjançant el mètode *stop()*. En el mateix mètode que es crida al prémer el botó play/stop per passar al estat *stopped*, juga

amb les característiques del botó, canviant la transparència i deshabilitant la possibilitat de clicar (figura 5.12), per donar la sensació de que l'aplicació està treballant en un segon pla. Aquest mateix mètode, crida al mètode *prepareAsync()* per donar l'opció a tornar a escoltar la ràdio un cop s'ha clicat "stop", i aquest *prepareAsync()* funciona completament igual (a nivell de jugar amb les característiques del botó play/stop i els estats del *Player*) que s'ha explicat anteriorment.

Fig. 5.12. Detall del botó play/stop quan està preparat i quan està carregant.

El botó play/stop, és sempre el mateix, encara que la imatge que es mostra per pantalla canvia, en el mètode *onClick(...)* cridat per el mètode *setOnClickListener(...)* de la classe *ImageButton*, és on a partir del codi, es juga tant amb els paràmetres del propi botó, com en els estats del *Player*.

Aquest mateix bloc funcional, fa més feina que la gestió del reproductor i del botó play/stop; també imprimeix per pantalla la informació del programa que s'escolta en aquell moment. Tot i que sembla simple, no ho és. Es podria haver desenvolupat l'aplicació de tal manera que tenint tota la informació en local (guardada dins la pròpia aplicació) mostrés una imatge o una altra en funció de la hora, però no s'ha fet així. Des de ràdio Palafolls es va demanar una aplicació modulable, i per tal de fer-ho així s'ha decidit llegir la informació dels programes d'un arxiu allotjat en un servidor web.

El primer que s'ha de fer per tal de mostrar la informació correcte, és saber en quin dia s'està, i això es soluciona creant un objecte *Day()*, que és una classe pròpia que crea un objecte amb el dia (monday, tuesday,...) hora i minut (format 23:45). L'objecte *Day()* es passa per paràmetre al mètode *getUrlDia(...)* i en la implementació d'aquest crida al mètode *getDiaXml()* per configurar a quina URL hem d'anar a buscar el arxiu XML del dia (hi ha un arxiu XML per cada dia de la setmana). Un cop es té la URL del dia pertinent, amb un bucle i els objectes *BufferedReader*, *InputStream* i *String*, es guarda tot l'arxiu XML que hi ha a la URL en una cadena de text (*String*), que a la vegada, es guarda en un objecte *OutputStream*. Aquest

OutputStream crea un document .xml en local, que és obert per un *InputStream*, objecte que s'utilitza per referenciar el document .xml. Tot aquest procés és per passar de tenir un document .xml en el servidor web a local.

Un cop es té el document .xml en local i referenciat per una variable de la classe, es passa per paràmetre al mètode *parse(...)* de la classe *LectorXmlProgramacio*. Aquest mètode crida recursivament altres mètodes per tal d'analitzar el document .xml i crear tants objectes *EntryHoraryRadio* com etiquetes "track" tingui el document i es guarden en una *Array* d'objectes *EntryHorariRadio*. S'utilitza aquesta *Array*, per passar-la per paràmetre al mètode *getPrograma(...)* de la classe *LectorXmlProgramacio*, per tal de guardar en una variable *EntryHoraryRadio* el programa que fan al moment. Així doncs, el mètode *getPrograma(...)* és l'encarregat de saber quina és l'hora actual, comparar-la amb l'atribut *horari* de les classes *EntryHorariRadio* que hi ha a la Taula i per retornar l'objecte *EntryHorariRadio* (que és la informació d'un programa de ràdio) que s'està fent al moment. Un cop es té l'objecte *EntryHorariRadio* "actual" es crida als mètodes *get* per obtenir el locutor, horari, nom del programa i URL de la imatge del programa. El locutor, horari i nom del programa són objectes de la classe *String* que es mostren directament per pantalla- La imatge del programa es descarrega de la URL i es guarda en un objecte *Drawable*, que és el que es mostra per pantalla.

Com es pot veure a la figura 4.2, quan s'entra a la pestanya de ràdio, venint d'una altra pestanya, es crida el mètode *onResume()* que deixa sonar la ràdio en cas de que estigui sonant, i la prepara com en el mètode *onCreate()* quan no està sonant.

5.3.2. Notícies

És el segon dels blocs funcionals (per ordre d'aparició en el menú de pestanyes). Aquest bloc es proveeix d'informació penjada al servidor web per tal d'oferir notícies actualitzades sobre l'actualitat de Palafolls.

Aquest bloc, consta de dos tipus de pantalla, la interna de l'aplicació i la que es llança en una *Activity* que obre el navegador nadiu del mòbil.

Com tota la informació que rep aquest bloc funcional prové d'internet, s'executa tot en un segon pla. Mentre, s'avisa a l'usuari que s'està carregant el contingut. Això s'aconsegueix creant una classe privada que estengui a *AsyncTask*. Dins aquesta classe hi ha tres mètodes

onPreExecute(), *doInBackground(...)* i *onPostExecute()* que executant el nom-de-la-classe.*execute()*, s'executen per aquest ordre. En el *onPreExecute()*, es mostra el quadre que indica que s'està carregant (figura 5.13), en el *doInBackground()*, es fa tot el procés de lectura al servidor web i en el *onPostExecute()* es treu el quadre "carregant..." i s'imprimeix el resultat del procés fet per el *doInBackground()* per pantalla.

Fig. 5.13. Detall del *Dialog* carregant.

La pantalla interna de l'aplicació és una llista. El contingut íntegre d'aquesta pantalla és llegit a Internet, per tant tot el procés per mostrar-ho en pantalla es produeix en el mètode *doInBackground()*. Aquest mètode utilitza exactament el mateix mètode que s'ha explicat en el punt 5.3.1. per tal de descarregar un .xml i guardar-lo en local. Un cop analitzat el document xml en local, amb una classe privada molt semblant (gairebé idèntica) a *LectorXmlProgramacio*, s'obté una taula d'objectes *EntryRss*.

Per poder imprimir per pantalla tots els objectes de taula contenidora de *EntryRss* en un *ListView*, en Android, per poder imprimir per pantalla una llista modificada, s'ha de crear una

classe extensora de *ArrayAdapter*, que fa que la llista tingui tants elements com objectes conté la taula que es passa per paràmetre, i modificar el constructor i el mètode *getView()*. El constructor inicialitza el context on s'ha d'imprimir indica quin *layout* s'utilitza en aquest adaptador i la llista de la qual obté les dades per omplir el *ListView*. El mètode *getView()* es crida tants cops com files tingui el *ListView* i cada cop el respectiu element de la taula.

El mètode *getView()* infla el xml personalitzat de cada fila (per modificar una llista, s'han de modificar tots els elements de la mateixa) i agafant els elements d'aquest .xml, els edita fent coincidir la línia del *ListView* amb la posició de la taula, així s'obté una llista amb el mateix ordre que té la taula de la qual s'obté aquesta llista.

La segona pantalla que pot mostrar aquest bloc funcional és el navegador nadiu del dispositiu. Quan es clica sobre l'element de la llista, el mètode *listener* del clic, accedeix al lloc de la taula de *EntryRss* per tal d'obtenir l'objecte pertinent, i utilitzant el mètode *getLink()* s'obté el enllaç adequat per navegar a la notícia (element de la llista) seleccionada. Aquest enllaç, es passa per paràmetre en un *Intent* juntament amb Intent.ACTION_VIEW. Un cop es té el *Intent* configurat, es llança amb una *Activity* i s'obre el navegador del dispositiu mòbil per accedir a l'element de la llista seleccionat.

5.3.3. Graella

Aquest és el bloc funcional que més diversitat de funcions ofereix, ja que s'encarrega de mostrar els horaris, siguin del dia que siguin, per pantalla. També ha de permetre escoltar els dos últims plens municipals i l'últim informatiu emès per Ràdio Palafolls.

Aquest bloc s'integra per un seguit de *ListView*s navegables entre sí. Totes les funcions d'aquest bloc, es desenvolupen en l'entorn gràfic de la pròpia aplicació, i sempre dins la pestanya numero tres.

Al accedir a la pestanya numero tres, s'invoca automàticament el mètode *onCreate()*, que aquest invoca al mètode *omplirDies()*, que el que fa és imprimir per pantalla totes les opcions que permet aquest bloc funcional, imprimeix "Informatius", que s'explica més endavant en aquest mateix punt la seva funcionalitat, i imprimeix també tots els dies de la setmana, que tots tenen la mateixa funcionalitat.

La funcionalitat que s'ofereix al prémer qualsevol dia de la setmana, és la de mostrar en pantalla l'horari del dia seleccionat. Això es fa des de el mètode *onListItemClick(...)*, que és el *listener* encarregat d'actuar quan s'apreta un element de la llista, que executa la classe privada extensora de *AsyncTask*. Aquesta classe té un funcionament estructural idèntic al explicat en el punt 5.3.2. quan es cridava a una classe extensora de *AsyncTask* també. L'única diferència és el codi intern del mètode *doInBackground(...)*. En aquest cas, *doInBackground(...)* el que fa és descarregar del servidor web el document .xml del dia seleccionat a la llista i guardar-lo en local (igual que s'explica en el punt 5.3.1). En aquest cas però, no és necessari cap altre mètode per saber l'hora actual i comparar-la amb els horaris dels programes de ràdio, ja que al tractar-se de la graella de programació de la ràdio, s'han de mostrar tots els programes del dia.

Per poder mostrar una llista personalitzada dels programes (un programa per línia de la llista). S'ha de utilitzar un procediment semblant al explicat en el punt 5.3.2, cridant a una classe pròpia, extensora de *ArrayAdapter*, per sobreescriure el constructor i el mètode *getView(...)*.

Al ser un bloc funcional que tracta bàsicament d' una navegació per llistes per tal de satisfer les necessitats funcionals, s'utilitzen fins a tres classes extensores de *ArrayAdapter* diferents. Una és *DiesArrayAdapter*, utilitzada tant per imprimir els dies com per omplir les llistes omplir informatius i omplir informatiu, una altre és *ReproduccionsArrayAdapter*, que s'usa per omplir la llista per la qual és poden escoltar els dos últims plens (més endavant en aquest mateix punt s'explicarà aquesta funcionalitat) i la tercera és *HorariArrayAdapter*, adaptador que s'encarrega de personalitzar la llista d'horaris de programació.

La funcionalitat que ofereix l'opció "Informatius" no és altre que la de permetre escoltar l'últim informatiu emès per Ràdio Palafolls i els dos últims plens municipals de Palafolls. Per escoltar qualsevol dels tres àudios, l'usuari ha d'actuar de la mateixa manera, navegar per les llistes (com es pot veure en la figura 5.14) i apretar sobre l'àudio preferit. Al prémer qualsevol dels tres elements que permeten escoltar un àudio, s'invoca al mètode *reproduirInformatiu()* (en el cas de voler escoltar l'últim informatiu) o al mètode *reproduirMp3(...)*, que rep una cadena de caràcters (*String*) per paràmetre per determinar quin dels dos plens municipals disponible es vol escoltar. Per fer possible la reproducció, es fa ús de la variable de classe *mediaPlayer* que està assignada a una instància de la classe *Player*, d'aquesta manera, es criden els mètodes *reset()*, *setDataSource(...)*, *prepare()* i *start()* de a classe *Player* per tal de reproduir el arxiu desitjat. L'única diferència entre el mètode *reproduirInformatiu()* i

reproduirMp3(...) és el *String* que es passa per paràmetre al mètode *setDataSource(...)* per carregar des de un directori web o un altre.

Fig. 5.14. Diagrama de Navegació de la graella

Per saber a quin nivell del diagrama de navegació d'aquest bloc està en cada moment l'aplicació, s'usa un comptador *(int)* que s'incrementa o decrementa cada cop que es puja o es baixa de nivell respectivament.

Com aquest bloc funcional també utilitza la instància de la classe *Player*, s'ha de tenir en compte quan es permet reproduir un àudio d'aquest bloc i quan no. S'ha decidit que només es

pot escoltar un àudio d'aquest bloc en la pantalla en la que s'inicia l'audició. Quan es canvia de pestanya, es tira enrere en el diagrama de navegabilitat d'aquest bloc o es tanca l'aplicació, ja sigui amb el botó *home* (invoca el mètode *onPause()*) com amb el botó *back* (invoca el mètode *onBackPressed()*), l'àudio d'aquest bloc deixa de reproduir-se.

5.3.4. Social

És el bloc funcional més senzill. Consta de dos imatges que a l'hora són botons (*ImageButton*) per unir-se al Facebook i al Twitter de Ràdio Palafolls.

Els dos botons tenen un *listener* que al ser llençat, a través d'un *Intent* passat com a paràmetre al mètode *stratActivity(...)* proven d'obrir el perfil de Ràdio Palafolls a les respectives xarxes socials, si això dona error (aquesta *startActivity()* està dins una estructura *try-catch*) el *catch* obre el navegador nadiu del dispositiu per la pàgina de la Ràdio Palafolls de la xarxa social pertinent.

Fig 5.15. Captura de pantalla de la pestanya social.

6. Casos d'ús.

6.1. Diagrama de casos d'ús

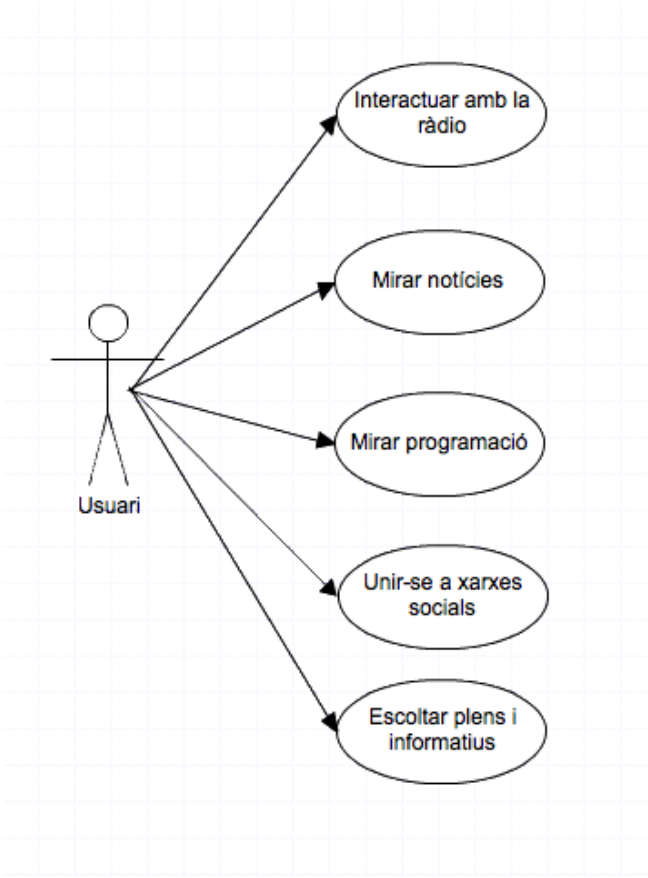

Fig. 6.1. Diagrama de casos d'ús.

6.2 Casos d'ús

6.2.1. Cas d'ús : Interactuar amb la ràdio

Actor involucrat: Oient

Pre-condició

L'oient ha d'estar a la pestanya "RÀDIO" de l'aplicació

Flux normal

1. L'usuari prem el botó "*play*" i la ràdio comença a sonar la ràdio.
2. L'usuari prem el botó "*pause*" i la ràdio deixa de sonar.

Flux Alternatiu

2.b) L'usuari surt de l'aplicació amb el botó enrere (botó del mòbil de no la aplicació) i la ràdio deixa de sonar.

6.2.2. Cas d'ús : Mirar notícies

Actor involucrat: Oient

Pre-condició

L'oient ha d'estar a la pestanya "Notícies" de l'aplicació

Flux normal

1. L'usuari escull una de les notícies que es mostren en pantalla.
2. L'aplicació mostra la notícia en una pantalla de navegació web.
3. L'usuari tanca la noticia amb el botó enrere del telèfon.
4. L'aplicació mostra el llistat de notícies.

Flux Alternatiu

2.b) L'usuari navega per la pantalla web.

2.b.2) L'usuari haurà de tirar tants cops enrere com enllaços hagi apretat per arribar al punt 3.

6.2.3. Cas d'ús : Mirar programació

Actor involucrat: Oient

Pre-condició

L'oient ha d'estar a la pestanya "GRAELLA" de l'aplicació.

Flux normal

1. L'usuari escull un dels dies de la setmana mostrats en pantalla.
2. L'aplicació mostra la programació del dia escollit en el punt 1.
3. L'usuari prem el botó enrere i torna al punt 1.

6.2.4. Cas d'ús : Unir-se a les xarxes socials

Actor involucrat: Oient

Pre-condició

L'oient ha d'estar a la pestanya "SOCIAL" de l'aplicació

Flux normal

1. L'usuari prem el botó de la xarxa social a la que es vol unir.
2. L'aplicació obre la app de la xarxa social escollida.
3. L'usuari s'uneix des de l'aplicació de la xarxa social.
4. L'usuari prem el botó enrere i torna a la pantalla del punt 1.

Flux Alternatiu

2.b) L'aplicació obre la pàgina de Ràdio Palafolls en la xarxa social escollida.

6.2.5. Cas d'ús : Escoltar plens i informatius

Actor involucrat: Oient

Pre-condició

L'oient ha d'estar a la pestanya "GRAELLA" de l'aplicació

Flux normal

1. L'usuari escull l'opció "Informatiu" de les diferents opcions mostrades en pantalla.
2. L'aplicació mostra dos opcions, "Plens" i "Informatius".
3. L'usuari escull "Plens".
4. L'aplicació deixa triar a l'usuari entre els tres últims plens municipals.
5. L'usuari selecciona el ple.

Flux Alternatiu

3.b) L'usuari escull "Informatius".

3.b.2) L'aplicació mostra l'últim informatiu.

3.b.3) L'usuari selecciona l'últim informatiu.

7. Proves i tests

Per tal de provar l'aplicació s'han anat fent proves durant el desenvolupament de la mateixa. Han anat sorgint errors com els que s'explicaran en aquest punt.

S'ha experimentat en el transcurs del desenvolupament d'aquesta aplicació, que al programar per Android, hi ha moltes maneres diferents de programar per tal d'aconseguir un objectiu. Això té la seva part bona, fa que sigui un llenguatge flexible i amb moltes llibreries amb molts mètodes per defecte, i la seva part dolenta, fa que sigui més difícil aprendre el llenguatge, ja que a l'hora de buscar informació no es troba la mateixa a dos llocs. Aquesta aplicació ha estat testejada en diferents dispositius amb diferents versions d'Android. Són aquests:

- Samsung Galaxy S Advance (Versió d'Android 2.3.6)
- Samsung Galaxy Note 2 (Versió d' Android 4.0.4)
- Samsung Galaxy (Versio d'Android 4.1.2)
- Tablet PC (Versió d'Android 4.0.3)
- Samsung Galaxy S Plus (Versió d'Android 2.3.5)
- Google Nexus (Versió d'Android 4.2)
- emulador (qualsevol versió i qualsevol resolució)

Respecte la diferència entre dispositius, ha estat útil testejar amb diferents dispositius, per comprovar com queden els *layouts* en pantalles de diferents tamanys. Pel que fa a la versió d'Android, Ràdio Palafolls ha estat programada per funcionar a partir de la versió 2.3.3. Gràcies a testejar l'aplicació amb altres versions d'Android, s'ha comprovat que tot el que funciona per la versió 2.3.3, no ho fa per versions 4.x.x. Això ha fet que s'hagués de canviar codi ja escrit i que es donava per bo.

7.1. El *MediaPlayer*

En un principi s'iniciava la ràdio en el mètode *onCreate()* de la classe, sense tenir en compte si sonava algun altre objecte *MediaPlayer* o no, més endavant, quant és va començar a navegar per les diferents pestanyes, es podia navegar per la aplicació amb la ràdio sonant, però que a l'anar a la pestanya corresponent a la funcionalitat de la ràdio per tal d' apagar-la,

no s'apagava sinó que creava un altre objecte *MediaPlayer* i també es reproduïa, sobreposant-se a l'altre i sonant els dos a l'hora. Això es va solucionar usant el patró Singletó a la classe *Player* (extensora de *MediaPlayer*), que només deixa crear un objecte d'aquesta classe.

Més endavant, al implementar el cas d'ús "escoltar plens i informatius" que també usa la classe *Player* donava incompatibilitats de estat en el *Player* (veure taula 5.11), per això es va haver d'implementar el mètode *onResume()* a la classe *Pestana1* (que és la que cobreix gran part de la funcionalitat de la ràdio) .

7.2. Llegir documents XML

Un dels reptes més importants en aquest projecte era aconseguir llegir arxius XML de la web des de l'aplicació, ja que és el mètode per el qual es passa gairebé tota la informació externa de l'aplicació. Hi havia dos dificultats que sortien per sobre les demés, analitzar un arxiu XML i llegir-lo d'Internet.

Per simplificar l'aprenentatge primer es va aprendre a llegir un XML en local, i després a descarregar-lo del servidor.

Després de diferents intents i gràcies a recerques per Internet i a la pàgina oficial de Android, és va trobar un mètode que, analitzant el codi, és va creure que podia ser aplicable a la nostra aplicació. A mida d'anar analitzant els errors que donava l'aplicació, es va anar modificant el mètode, el que fa bàsicament aquest mètode són crides recursives a diferents mètodes per analitzar els diversos nivells d'un document XML. Això va fer que s'arribés a la conclusió que s'havia de modificar l'arxiu XML de Ràdio Palafolls, ja que no hi havia un *tag* que englobés tot el contingut XML i això feia que no funcionés el mètode.

Un cop funcionava el local, s'havia de fer alguna cosa per tal de baixar el document XML del servidor web per guardar-lo en local (s'explica com en el punt 5.2.5).

Un cop finalitzat el lector d'arxius XML que llegeix la programació de la ràdio, es va agafar aquest mateix lector i es van fer petites modificacions per tal d'adaptar-lo al XML que genera el servidor de Ràdio Palafolls per obtenir les últimes noticies de Palafolls.

Així com el lector de XML de programació és una classe pública, el lector de RSS és una classe privada dins la classe Pestana2 ja que és la única que usa aquesta classe. A diferència

del Lector de XML que apart de analitzar els documents, té dos mètodes més usats per diferents classes.

7.3. Altres dificultats

Al ser la primera aplicació que s'ha desenvolupat per part del desenvolupador, s'han hagut de fer més proves del previst, ja que no es sabia si el que fallava era el programador o la aplicació. Un cas evident d'això és el temps de connexió a Internet, al no haver un mètode estàndard per connectar amb el servidor, no es sabia si el problema era de codi, o de connexió.

Per testejar qué és el que fallava, s'han fet proves de velocitat connectant amb diferents servidors. Els resultats són clars, al connectar amb el servidor de Ràdio Palafolls, el temps de càrrega de la ràdio és molt més lent, la solució de la qual queda fora de l'abast d'aquest projecte.

Un altre punt a tenir en compte és la càrrega d'imatges al mostrar la graella de programació d'un dia concret, ja que es descarreguen les imatges de tots els programes i això alenteix molt la navegabilitat per la graella. Es va proposar a la direcció de Ràdio Palafolls l'opció de guardar les imatges en local per tal d'augmentar considerablement el temps de carrega de la graella, però es va decidir que no, que s'han de llegir del servidor, ja que al no ser així, es perdria el fet de poder canviar tot el contingut de l'aplicació des d'un servidor extern a l'aplicació.

8. Exportar aplicació

Per tal de poder penjar al market d'Android una aplicació, s'ha d'exportar en un executable .apk. Aquest arxiu s'ha de firmar per tal de fer-lo personal. L'eina que té Android per fer això s'anomena *Keystore* que és una signatura digital, que es crea amb una pantalla com la de la figura 8.1.

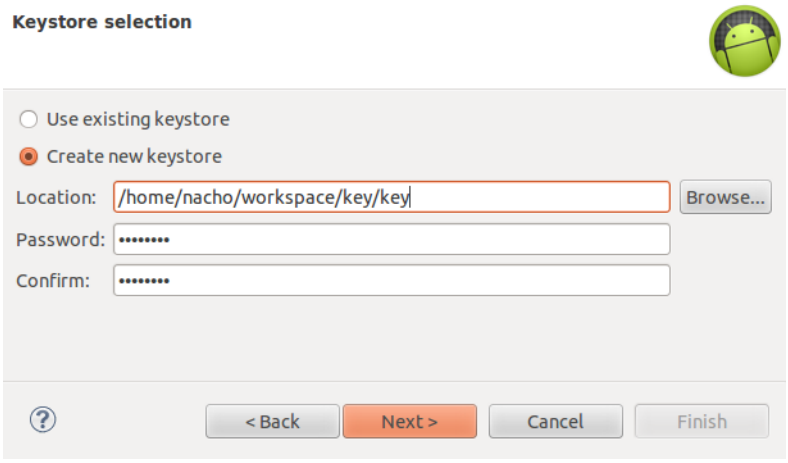

Fig. 8.1. Detall de la creació de una *Keystore*

El *Keystore* serveix per vincular el desenvolupador a l'aplicació, per lligar l'aplicació al seu propietari (que és possible que no sigui el seu desenvolupador), s'utilitza una eina anomenada *Key* (fig 8.2), que com el seu nom indica és la clau per accedir a l'aplicació en qüestió.

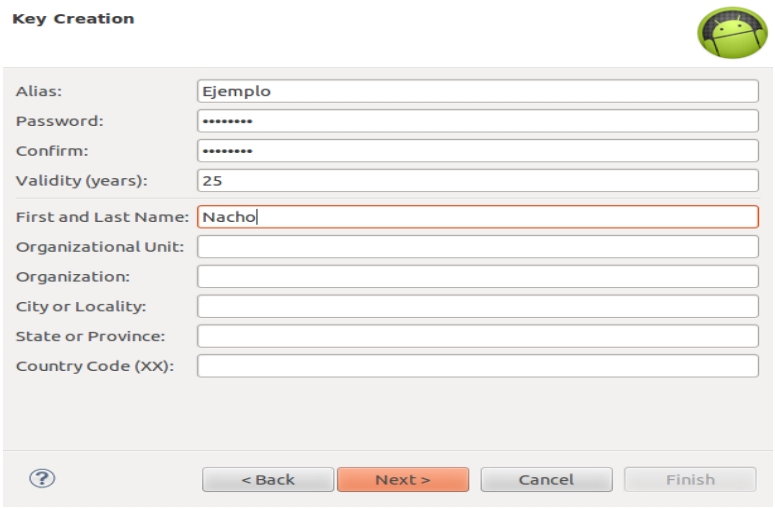

Fig 8.2. Detall de la creació d'una *key*

9. Conclusions

La realització d'aquest projecte m'ha suposat moltes hores dedicades tant a mi com a gent del meu voltant amb coneixements d'Android. Vaig escollir aquest projecte perquè tenia com a repte personal aprendre a programar per dispositius mòbils, així que aquest treball m'ha fet comprometre'm a aprendre un nou llenguatge de programació totalment desconegut per mi.

Aquest projecte ha estat molt més complicat del que em pensava, ja que en un principi volia desenvolupar-lo tant per Android com per iPhone, i així complir també el meu objectiu personal d'aprendre a programar per les dos grans plataformes de dispositius mòbils que hi ha actualment, i al final no ha pogut ser.

Tot i que el llenguatge de programació és Java, llenguatge amb el qual jo ja havia treballat, desconeixia tant l'entorn de desenvolupament de l'aplicació, com les llibreries, com els patrons i models a seguir. Tant és així, que el nivell de dificultat molts cops residia més en com escriure la solució, que la solució en sí.

Personalment, estic orgullós d'haver aconseguit quelcom que per mi era impensable fa quatre mesos enrere, he complert tots els objectius principals que ens hem proposat juntament amb ràdio Palafolls.

D'altra banda, m'he trobat amb limitacions temporals, ja que el que a la conclusió d'aquest projecte podria fer en cinc minuts, la feina que al iniciar-lo em podría portar tota una tarda.

Hi ha diferents aspectes a millorar de l'aplicació, un d'ells és el *background* de les imatges del *TabHost*, que com es pot veure en la figura 9.1 són d'un color diferent i m'hagués agradat que fos transparent.

Fig 9.1 Detall del *TabHost*.

D'altre banda, no he aconseguit quadrar la pantalla d'inici a tots els tipus de pantalles. Per pantalles molt petites i per tauletes, els elements de la pantalla corresponents a la funcionalitat ràdio, queden desquadrats. He demanat ajuda a amics i programadors, però cap m'ha sabut donar una resposta clara. Més que res m'ha faltat la disponibilitat de diferents mòbils per fer proves, ja que és una feina de prova i assaig constant, i per falta de recursos no hi he pogut dedicar el temps que m'hagués agradat.

10. Referències

[1] http://upcommons.upc.edu/pfc/bitstream/2099.1/16037/1/PFC%20Jacobo%20J.%20Pedrosa%20Mar%C3%ADn.pdf

Informació teòrica sobre la plataforma Android. (20/12/2012)

[2] https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

Informació teòrica sobre la plataforma Android. (20/12/2012)

[3] http://www.linuxhispano.net/2012/06/26/exportar-a-apk-una-aplicacion-android-en-eclipse/

Com entregar una aplicació al client. (08/01/2013)

[4] http://stackoverflow.com

Foro sobre programació. (13/01/2013)

[5] http://androideity.com

Tutorial de navegabilitat per pestanyes. (10/11/2012)

[6] https://groups.google.com

Foro sobre programció. (13/01/2013)

ANNEX 1 : Instruccions d'ús.

1. Introducció

L' aplicació Ràdio Palafolls s'alimenta gairebé en la seva totalitat de la informació que llegeix de la web de ràdio Palafolls. Per poder actualitzar l'aplicació de forma correcte, s'han de seguir uns patrons i unes normes. En aquestes instruccions s'explicarà com canviar tot el contingut de l' aplicació des de la web sense que deixi de funcionar l'aplicació.

Aquestes instruccions estan pensades per l'encarregat de modificar la web des d'on es llegeix la informació i per tant, canviar el contingut de l' aplicació, per facilitar la feina a aquesta persona, farem un recorregut per les 4 pestanyes (figura AN1.1) de l'aplicació explicant com es canviaria tot el contingut a mida que l'anem veient.

Figura AN1.1. Menú de navegabilitat

2. Pestanya 1: Ràdio

Figura AN2.1. Captura de pantalla de la pestanya 1

Aquesta pantalla està composta de tres *TextViews* i un *ImageView*, els *TextViews* mostren el nom del programa, el locutor i la franja horària del programa, el *ImageView* imprimeix la foto del programa.

La lectura del tota la informació del programa és llegeix d' un arxiu ".xml" penjat a la web ("http://www.radiopalafolls.cat//wp-content/themes/rp/images/bg/programacio-xml/app/dimecres.xml") que segueix un patró com aquest:

```
▼<jornada>
  ▼<track>
     <titol>DESAFINADO</titol>
     <locutor>Per Federico Francesch</locutor>
     <horari>00:00-1:00</horari>
    ▼<imatge>
       http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/federico.png
     </imatge>
   </track>
  ▼<track>
     <titol>SELECCIÓ NIT</titol>
     <locutor>ZZZZzzzz...</locutor>
     <horari>1:00-6:00</horari>
    ▼<imatge>
       http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/relax.png
     </imatge>
   </track>
   <!--  programació dia  -->
  ▼<track>
     <titol>BON DIA</titol>
     <locutor>Selecció musical matinal</locutor>
     <horari>6:00-8:00</horari>
    ▼<imatge>
       http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/bon-dia.png
     </imatge>
   </track>
  ▼<track>
     <titol>TIC TAC RIIIINGGG</titol>
     <locutor>Helena Carles</locutor>
     <horari>8:00-6:00</horari>
    ▼<imatge>
       http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/helena-carles.png
     </imatge>
   </track>
```

Figura AN2.2. Detall del document original XML dels RSS

El *tag* "jornada" és el dia, que engloba tots els programes, i el *tag* "track" és un programa, dins seu té tota la informació de cada programa, mentre es respecti totes les etiquetes, es poden afegir o treure programes, canviar el títol, locutor, horari i/o imatge de cada programa. Compte amb les modificacions del *tag* "horari" ha de seguir el format "hh:mm-hh:mm" perquè l'aplicació pugui llegir les hores correctament, també s'ha de tenir en compte que ha de cobrir les 24h del dia sense solapacions horaries entre programes perquè funcioni correctament.

3. Pestanya 2: Notícies

Figura AN3.1. Captura pantalla pestanya dos

Aquesta pantalla llegeix íntegrament la seva informació d' un arxiu ".xml" que és generat pels administradors de la web "radiopalafolls.cat". S'han de respectar totes les etiquetes i estructures del ".xml".

4. Pestanya 3: Graella

Figura AN4.1. Captura de pantalla de la pestanya 3

En aquesta pestanya hi han dos aspectes modificables, el contingut que hi ha dins els dies (tots segueixen el mateix patró) i el contingut que hi ha dins l'element "programació especial".

4.1. Informació dins dels dies

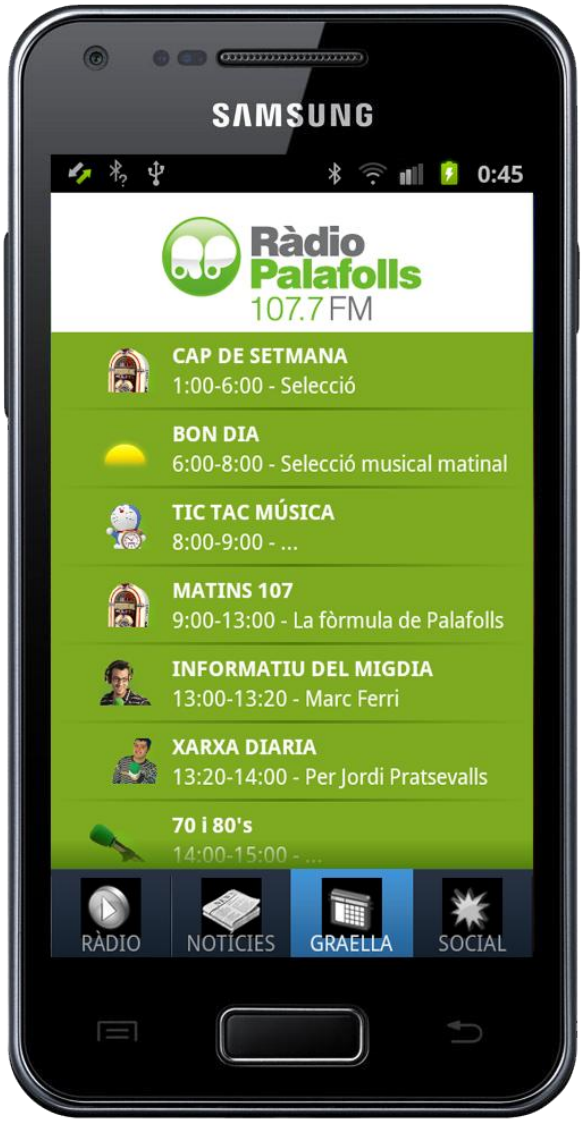

Figura AN4.2. Captura de pantalla del horari d'un dia.

Tota la informació que es mostra al clickar sobre un dia es llegeix dels mateixos arxius en els quals es llegeix el contingut de la pestanya 1. S'han de seguir les mateixes instruccions. L'únic afegit que hi ha es l'ordre de mostrar els programes, l'ordre que es segueix és el mateix ordre en què estan escrits els "track" que hi ha en l'arxiu ".xml" (figura AN2.2). Si es vol canviar la col·locació dels programes, s'ha d'agafar tota l'etiqueta "track" amb tot el seu contingut intern (desde <track> fins a </track> inclosos) i posar-los en un altre lloc del arxiu ".xml" sempre dins del *tag* "jornada" i fora de qualsevol altre *tag*.

4.2 Informació dins programació especial

Figura AN4.3. Captura pantalla de la pantalla on s'escolten els plens municipals.

En aquesta pantalla el que es pot canviar no és res que tingui incidència visual sinó l'arxiu ".mp3" que llegeix al clicar una de les 2 opcions que apareixen. Aquests arxius sempre han d'estar localitzats dins de la URL "http://www.radiopalafolls.cat/wp-content/themes/rp/images/bg/programacio-xml/app/ultim.mp3" i sempre han de tenir el mateix nom ("ultim.mp3" en aquest cas) del arxiu que substitueixen.

5. Pestanya 4 (social) i altres

D'aquesta pestanya (figura 5.1) no es pot modificar res. Qualsevol altre canvi que no estigui explicat en aquestes instruccions requereix un canvi de codi de l'aplicació i no pot ser canviat modificant el contingut web del qual es llegeix la informació.

Fig AN5.1. Captura de pantalla de la pestanya 4.