Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Ingeniería Técnica en Informática de Gestión

Aplicación Smartphone para el Ayuntamiento de Palafolls

Memoria

Manuel Alfonso Díaz Sanguino
PONENTE: Antoni Satué Villar

OTOÑO 2012

TecnoCampus Mataró-Maresme

Agradecimientos

Quiero dar las gracias a mi familia, por todas las oportunidades y apoyo que me han dado.

También agradecer a mis compañeros de carrera, en especial a Sergio Álvarez y Alex Molina. Al tutor del proyecto, Antoni Satué, así como a todos los profesores por ayudarme a ampliar mis conocimientos, en especial a Josep Mª Gabriel.

Resum

L'objectiu del projecte és el de desenvolupar una aplicació per a dispositius mòbils amb sistema operatiu Android, que serveixi com a guia de la ciutat de Palafolls. La informació que s'ofereix a l'usuari està emmagatzemada en una base de dades externa, de manera que la connexió es realitza a través d'Internet mitjançant un *script*, evitant així la publicació de contínues actualitzacions de l'aplicació. El resultat final és una aplicació mòbil disponible en Google Play.

Resumen

El objetivo del proyecto es el de desarrollar una aplicación para dispositivos móviles con sistema operativo Android, que sirva como guía para la ciudad de Palafolls. La información que se ofrece al usuario está almacenada en una base de datos externa, por lo que la conexión se realiza a través de Internet mediante un *script*, evitando así la publicación de continuas actualizaciones de la aplicación. El resultado final es una aplicación móvil disponible en Google Play.

Abstract

The goal of the project is to develop an application for mobile phones with Android operating system, which serves as a guide for the city of Palafolls. The information provided to the user is stored in an external database, so that the connection is made via the Internet with a script, thereby avoiding the publication of continual updates of the application. The end result is a mobile application available on Google Play.

Índice

Índice de figuras.

Índice de tablas.

1. Objetivos.

1.1. Propósito.

Desarrollar una aplicación para móviles con sistema operativo Android, que sirva como guía de la ciudad de Palafolls, tanto para habitantes de la misma o visitantes.

A nivel personal, aprender a desarrollar aplicaciones móviles y así poder obtener experiencia que facilite la inserción en el sector del desarrollo de aplicaciones móviles del mundo laboral.

1.2. Finalidad.

Satisfacer la necesidad del Ayuntamiento de Palafolls de tener una aplicación móvil con la cual dar a conocer la ciudad a través de las nuevas tecnologías.

1.3. Objeto.

Una aplicación móvil para Android donde el usuario puede ver la información que desea acerca de la ciudad de Palafolls, así como un manual para su mantenimiento.

1.4. Alcance.

La aplicación está disponible en el mercado de aplicaciones Google Play. La base de datos donde se almacena la información, y los *scripts* para conectarse a la base de datos están en manos del Ayuntamiento, así como el manual para el mantenimiento de la aplicación.

Destacar el aprendizaje previo a la realización del proyecto sobre el desarrollo de aplicaciones Android.

2. Introducción.

2.1. Palafollejant.

Palafollejant es una aplicación móvil para Android, que muestra información sobre la ciudad de Palafolls, y que sirve como guía de visita de la ciudad.

El contenido de la aplicación está basado en la información que se muestra en la página web del ayuntamiento de Palafolls, para que de esta manera el contenido sea dinámico sin tener que lanzar actualizaciones de la aplicación cada poco tiempo.

Toda la información que se muestra al usuario está detallada en el capítulo 6.

2.2. Diseño.

El único diseño facilitado por el cliente ha sido el del menú principal (ver Fig. 2.1). Todo el resto de interfaz gráfica ha sido diseñado e implementado teniendo en cuenta otras aplicaciones similares, de manera consensuada con el cliente. A lo largo del desarrollo de la aplicación, se han aplicado cambios para mejorar su aspecto.

Dado que el cliente solo ha facilitado el diseño del menú principal, el resto de imágenes que se han utilizado disponían de una licencia libre para su uso comercial.

Fig. 2.1. Diseño inicial del menú principal.

2.3. Planificación.

Dado que al inicio del proyecto no se disponía de conocimientos de la programación para Android, y que uno de los propósitos de este proyecto es la formación en el desarrollo de aplicaciones móviles, una de las principales tareas ha sido la iniciación y familiarización con el entorno de programación de Android. Para ello se han seguido diversos cursos *online* así como a través de un curso (ver referencia [1]), al mismo tiempo que se desarrollaba una aplicación ajena a este proyecto para facilitar el aprendizaje. Esta formación previa ha facilitado en gran medida el desarrollo de la aplicación final.

A continuación se detalla en la Tabla 2.1 las horas de trabajo por cada fase del desarrollo del proyecto.

Fase	Horas
Formación previa	80
Análisis previo (incluye estudio de mercado)	10
Codificación de la aplicación (incluye fase de pruebas)	110
Redacción documentación	65

Tabla 2.1. Horas de trabajo por fase.

3. Android.

3.1. ¿Qué es?

Android es un sistema operativo móvil de código abierto que nació a partir de una versión modificada de Linux, enfocado a ser utilizado en *smartphones*, *tablets* y *smart TVs*. La mayoría de código se encuentra bajo la licencia Apache de código abierto.

Fue desarrollado inicialmente por la compañía Android Inc., creada en 2003, posteriormente comprada por la Open Handset Alliance (OHA) en 2005, liderada por Google.

La OHA es una alianza comercial formada por 84 empresas (algunas como HTC, Samsung, Dell, Intel, Motorola, LG, T-Mobile y Nvidia), que se dedica a desarrollar estándares abiertos para dispositivos móviles.

Uno de los primeros prototipos que implementaba Android fue el Google Sooner de HTC, fabricado en 2007 y con la versión de Android de marzo del mismo año, que no llegó a salir al mercado.

No obstante, su lanzamiento al mercado fue en octubre de 2008, de manos del T-Mobile G1, casualmente de HTC también.

3.2. Distribución de versiones Android.

Uno de los grandes problemas de Android es la fragmentación de su distribución.

Debido a que es un software libre, los fabricantes de móviles pueden decidir que versión del sistema operativo llevará cada dispositivo y personalizarlo. También pueden decidir si dejan de dar soporte a algunos de sus dispositivos, es decir, no van a ofrecer al usuario la posibilidad de actualizar su versión de Android.

Incluso las operadoras de telefonía móvil pueden personalizar el sistema operativo (lo hacen en menor medida que los fabricantes) e incrustar aplicaciones con escaso interés

para el usuario, las cuales no se pueden desinstalar sin tener acceso *root* (control absoluto sobre el sistema).

No obstante, Google también tiene parte de culpa al lanzar cada poco tiempo una nueva versión de Android. Sin ir más lejos, cuando se lanzó la versión *4.0 Ice Cream Sandwich*, la anterior versión, *3.0 HoneyComb* lanzada ocho meses atrás, sólo estaba instalada en un 3.3% de dispositivos Android, tal como se muestra en la Tabla. 3.1.

Versión	Nombre	Nivel API	Distribución
1.5	*Cupcake*	3	0.6%
1.6	*Donut*	4	1.1%
2.1	*Eclair*	7	8.5%
2.2	*Froyo*	8	30.4%
2.3 - 2.3.2	*Gingerbread*	9	0.6%
2.3.3 - 2.3.7		10	54.9%
3.0	*Honeycomb*	11	0.1%
3.1		12	1.5%
3.2		13	1.7%
4.0 - 4.0.2	*Ice Cream Sandwich*	14	0.3%
4.0.3		15	0.3%

Tabla 3.1. Distribución de versiones Android - Enero 2012

Por otra parte, es posible que el hardware de algunos dispositivos no esté capacitado para soportar nuevas versiones del sistema operativo y provoca se queden obsoletos en cuanto a sistema operativo. Con cada nueva versión, también se lanza una nueva interfaz de programación de aplicaciones (API), la cual ofrece nuevas funcionalidades.

Como solución a este problema, Android tiene una gran comunidad libre y después de que Google lance una nueva versión del sistema operativo, ya están preparando adaptaciones para los principales dispositivos del mercado. Gracias a esto, en muy poco tiempo, los usuarios avanzados pueden limpiar sus dispositivos de personalizaciones de fabricante y operadora, e instalar una ROM.

Las ROMs, son compilaciones de Android preparadas por los usuarios con opciones y características propias que no tienen las versiones oficiales. Se utilizan los recursos del terminal para sacarles el máximo provecho dentro de sus límites.

3.3. Dispositivos con Android.

Al ser Android un sistema operativo de código abierto, cada fabricante de móviles lo puede aplicar a los terminales que desee, cada uno con su hardware correspondiente. Y lo que representa un inconveniente para los desarrolladores de aplicaciones, al mismo tiempo es una ventaja, ya que se puede utilizar en múltiples dispositivos, al contrario que iOS, que sólo se distribuye en dispositivos de Apple.

Como se observa en la Fig. 3.1, existe una gran variedad de dispositivos con Android. Esto implica que los desarrolladores han de optimizar las aplicaciones para todos los terminales.

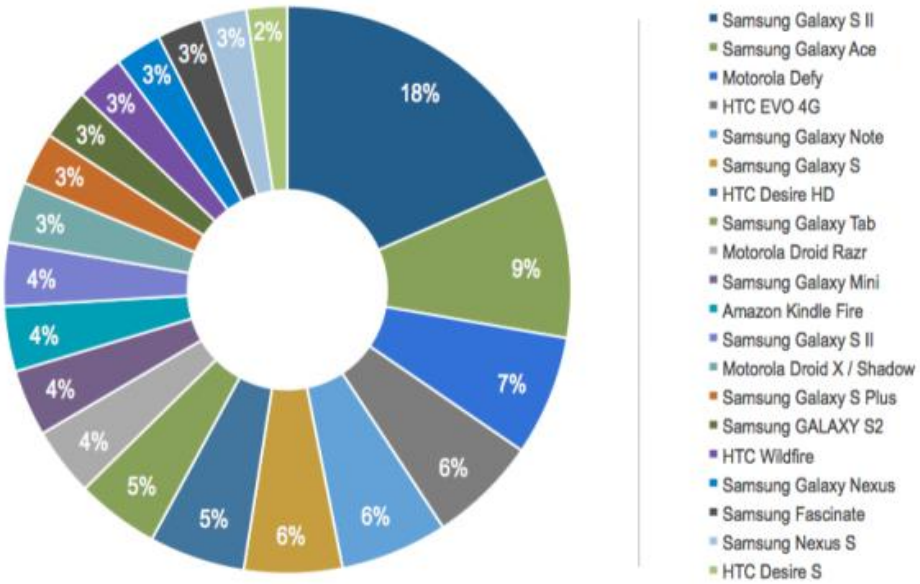

Fig. 3.1. Dispositivos con Android.
(Fuente: www.flurry.com)

Uno de los aspectos a tener en cuenta a la hora de optimizar una aplicación, es la interfaz gráfica. Para ello, hay que tener en cuenta la densidad de pantalla de cada dispositivo, es decir, el número de píxeles por pulgada física de pantalla (dpi). Y dado que hay una gran variedad de dispositivos, Android define las siguientes densidades de pantalla:

Tamaños de pantalla			Densidades de pantalla	
Small	2 - 3 pulgadas		*ldpi*	100 - 150 dpi
Normal	3 - 4,5 pulgadas		*mdpi*	150 - 200 dpi
Large	4,5 - 7 pulgadas		*hdpi*	200 - 250 dpi
Xlarge	7 - 10 pulgadas		*xhdpi*	250 - 300 dpi

Tabla 3.2. Tamaños y densidades de pantalla.

Esto implica que por cada imagen que se utilice en la aplicación, habrá que realizar cuatro copias, una para cada densidad. Para ello, como se muestra en la Fig. 3.2 habrá que realizar una conversión de las imágenes entre las diferentes densidades. De esta manera, las imágenes se verán siempre del mismo tamaño, sin distinción de dispositivos.

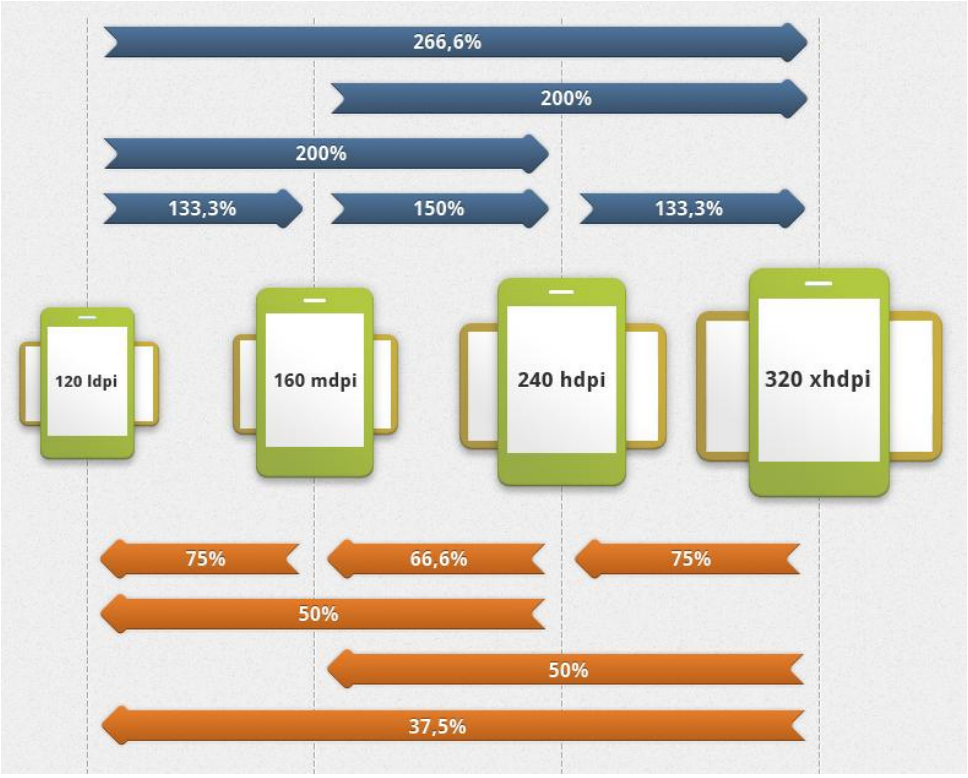

Fig. 3.2. Conversión entre densidades.

(Fuente: www.untipodigital.com)

3.4. Por qué Android y no iOS.

Se ha escogido realizar la aplicación para Android debido a que se programa en Java, y es el principal lenguaje de programación que se enseña en la Ingeniería Técnica de Informática de Gestión. Por otra parte, para programar aplicaciones en iOS se utiliza el lenguaje de programación Objective-C, lenguaje que no se enseña en la carrera.

Otro de los motivos ha sido el aspecto económico. El software necesario para programar aplicaciones para Android, explicado en el sub apartado 3.10.2, se puede descargar de forma totalmente gratuita. Sin embargo, para programar en iOS es necesario comprar la licencia de desarrollador por 99$ al año.

También se ha tenido en cuenta el estado actual del mercado de los *Smartphone*, dónde Android es el sistema operativo más usado a nivel mundial, como se puede observar en la Fig. 3.3.

Fig. 3.3. Distribución de sistemas operativos móviles.
(Fuente: www.flurry.com)

3.5. Google Play Vs App Store.

A pesar de que Android es el sistema operativo móvil más utilizado, la App Store es más apetecible para los desarrolladores. Según un estudio realizado por la consultora Flurry entre enero y mayo del año 2012 (ver referencia [2]) por cada aplicación que se ha desarrollado para la plataforma de Google, se han creado 2 para terminales de Apple, una proporción que a gran escala es transcendente.

El gran problema que los desarrolladores encuestados y los analistas achacan es la gran variedad de dispositivos Android**,** comentada en el apartado 3.3. El sistema operativo iOS juega con un sistema en el que si funciona en un dispositivo seguramente funcione en todos los demás dispositivos, pero en Android la variedad es tan enorme que es difícil que algo que requiera algunos recursos, funcione en cualquier terminal.

Muchos desarrolladores se ven obligados moralmente y no técnicamente a optimizarlo todo al máximo en Android a todo terminal posible y ello hace que los gastos aumenten y los beneficios se vean lastrados. Esto crea tal diferencia que por cada dólar de beneficio en iOS, el beneficio en Android se vea reducido a 0.24$, una gran diferencia.

3.6. Versión Android escogida.

Debido a la fragmentación en la distribución de Android antes comentada, se ha de tener en cuenta la versión para la cual se programará una aplicación.

A medida que una nueva versión es lanzada, ésta implementa más funcionalidades para los desarrolladores en su correspondiente API (interfaz de programación de aplicaciones). Sin embargo, no todos los dispositivos móviles tendrán las últimas versiones del sistema operativo.

Y como pasa en todas las versiones de software, sí que se podrán utilizar aplicaciones para versiones de Android antiguas en las nuevas versiones del sistema operativo, pero no al revés.

Por eso, con los datos que facilita Google mensualmente (ver referencia [3]), se ha escogido la versión *2.3.5 Gingerbread* al ser la versión con más presencia en dispositivos Android en el momento de iniciar el proyecto (ver Tabla 3.3.).

Versión	Nombre	Nivel API	Distribución
1.5	*Cupcake*	3	0.2%
1.6	*Donut*	4	0.5%
2.1	*Eclair*	7	4.2%
2.2	*Froyo*	8	15.5%
2.3 - 2.3.2	*Gingerbread*	9	0.3%
2.3.3 - 2.3.7		10	60.3%
3.1	*Honeycomb*	12	0.5%
3.2		13	1.8%
4.0 - 4.0.2	*Ice Cream Sandwich*	14	0.1%
4.0.3 - 4.0.4		15	15.8%
4.1	*Jelly Bean*	16	0.8%

Tabla 3.3. Distribución de versiones Android – Agosto 2012

3.7. Funcionalidades.

Debido a que Android es un software libre, abierto y disponible para todos los desarrolladores, no existe una configuración única del sistema operativo. Aún así, se expone un listado con las funcionalidades básicas:

Almacenamiento	Se utiliza el gestor de base de datos SQLite, simple pero muy funcional.
Conectividad	Soporta GSM/EDGE, IDEN, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE y WiMax.
Mensajería	Soporta SMS, MMS y Mensajes *Push* o *Cloud to Device*, es decir, mensajes del servidor a la aplicación.
Navegador Web	Basado en el motor de renderizado de código abierto WebKit.
Soporte de hardware	Cámara de fotos y video, GPS, sensor de proximidad, sensor de presión, acelerómetro, giroscopio, pantallas táctiles, sensor de luz, *gamepad*, termómetro y aceleración por GPU 2D y 3D.
Soporte multimedia	Soporta los formatos WebM, H.263, H.264, MPEG-4 MP3, MIDI, WAV, JPEG, PNG, GIF, BMP, entre otros.
Multitarea	Da soporte para aplicaciones multitarea.
Entorno de desarrollo	Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software.
Tethering	Permite al dispositivo ser usado como un punto de acceso alámbrico o inalámbrico.

Tabla 3.4. Funcionalidades de Android.

3.8. Arquitectura.

El sistema operativo Android se puede subdividir en cuatro capas, como se observa en la Fig.3.4.

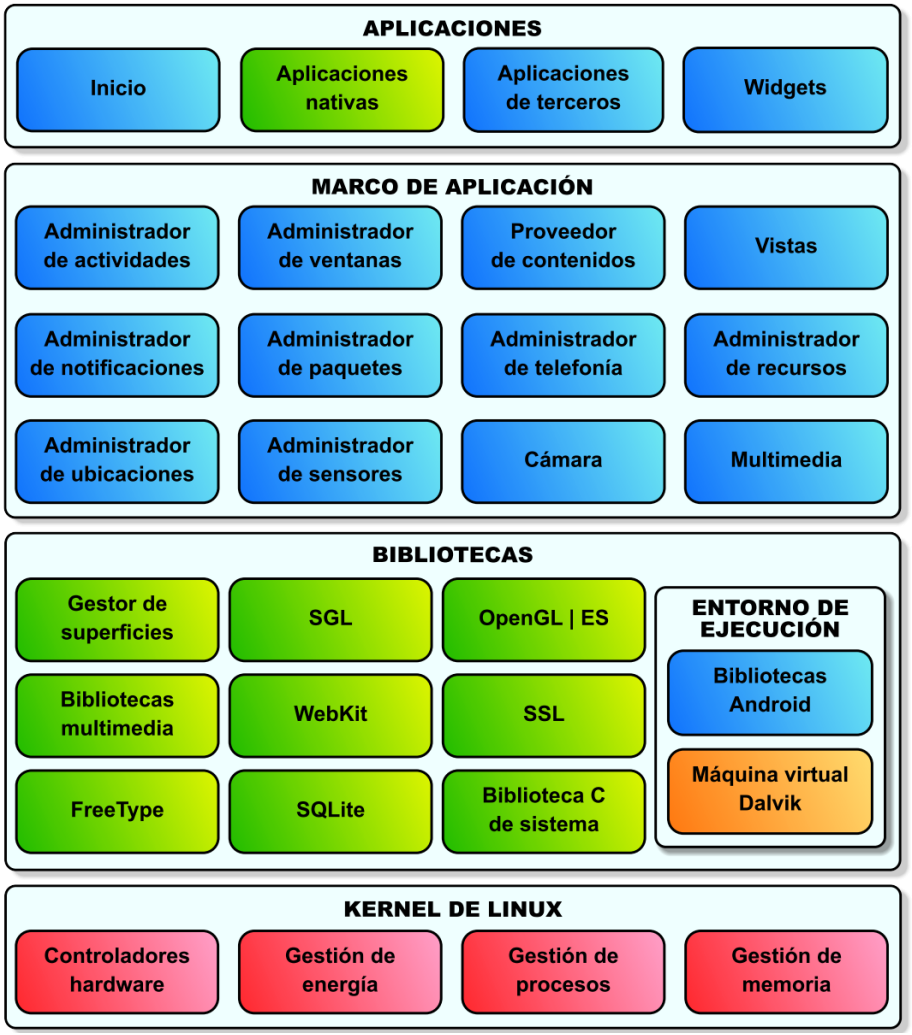

Fig. 3.4. Estructura de Android.

Estas cuatro capas forman sus cinco secciones principales:

- ***Kernel* Linux:** contiene los controladores de dispositivos de bajo nivel para los distintos componentes del hardware de un dispositivo Android. Se incluyen los servicios de seguridad, gestión de la memoria, gestión del procesador, etc. Está formado a partir del sistema operativo Linux versión 2.6.
- **Bibliotecas:** contienen todo el código que suministran las principales funcionalidades de Android. Las principales bibliotecas son:
    - **SQLite:** sistema de gestión de base de datos relacionales.
    - **WebKit:** motor de navegación web que utiliza Android. Es la misma biblioteca que utiliza Google Chrome y Safari.

  - **Freetype:** fuentes en mapas de bits y renderizado vectorial.
  - **System C:** basada en BSD de C.
  - **SGL:** motor de gráficos 2D
  - **3S:** bibliotecas basadas en OpenGL.
  - **SSL:** servicios de cifrado *Secure Socket Layer*.
- ***Runtime*:** en la misma capa que las bibliotecas, se encuentra el tiempo de ejecución (*runtime*), que contiene las bibliotecas del núcleo que permiten que los desarrolladores generen sus aplicaciones utilizando Java. También contiene la máquina virtual Dalvik, específicamente diseñada para Android. Con una instancia de la máquina virtual Dalvik, cada aplicación se puede ejecutar en un proceso independiente.
- ***Framework* de aplicaciones:** esta capa contiene los gestores de las funcionalidades que utilizan los desarrolladores de las aplicaciones.
- **Aplicaciones:** es la capa superior de la pila, se encuentran las aplicaciones que se integran en el dispositivo Android, así como las aplicaciones de terceros que se incorporan mediante descarga o las creadas por desarrolladores.

3.9. Elementos básicos.

3.9.1. *Activity*.

Las actividades son clases derivadas de la clase *Activity*, que representan el componente principal de la interfaz gráfica de una aplicación. Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual. Su objetivo principal es interactuar con el usuario. Ha de contener la función "onCreate()", que es la qué se lanza cuando se inicia la actividad. Entre otros aspectos, se define el archivo XML de *layout* de la pantalla adecuada.

Desde el momento que una actividad aparece en pantalla hasta que se oculta, pasa por varias etapas, que se pueden observar en la Fig. 3.5.

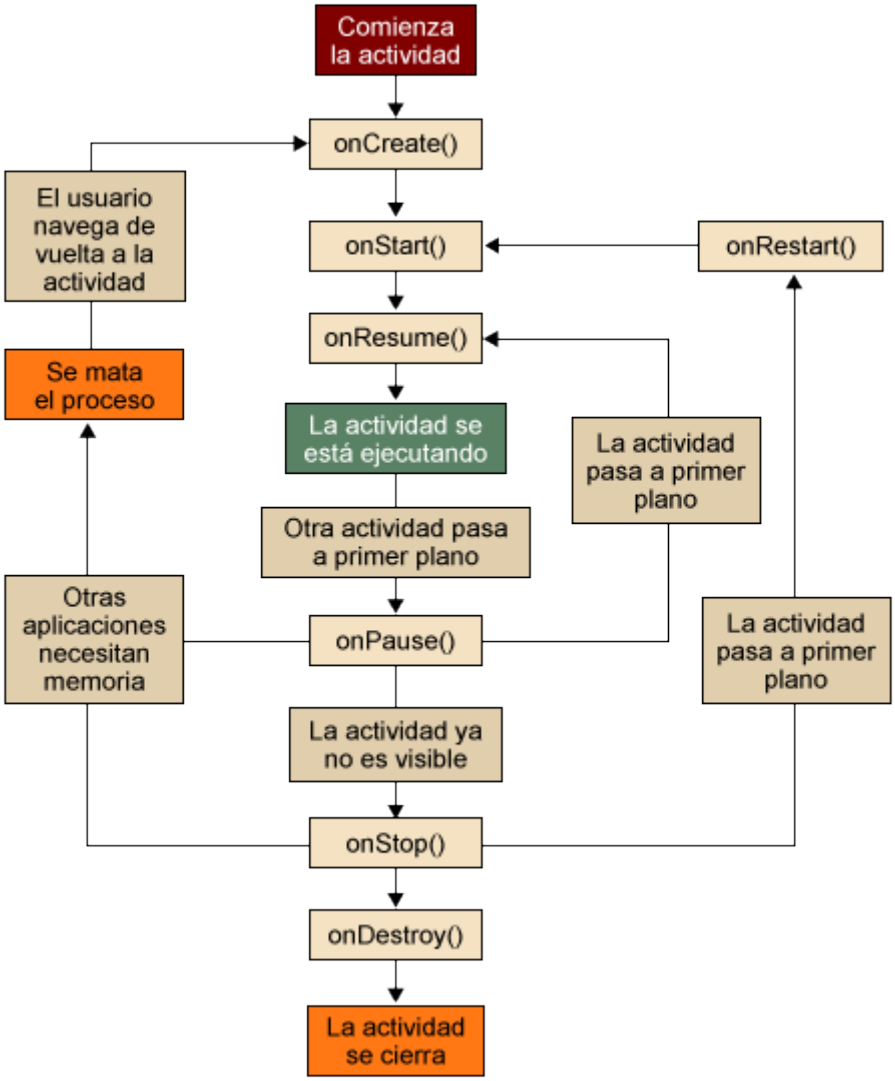

Fig. 3.5. Ciclo de vida de una actividad.

3.9.2. *View*.

Las vistas (*view*) son los componentes de la interfaz de usuario. Las vistas se definen en los archivos de *layout* o de diseño, que son archivos XML en los que se define el contenido visual de la interfaz.

3.9.3. *Intent* y *Bundle*.

Un *Intent* es un elemento básico de comunicación entre las distintas actividades de una aplicación. Son mensajes o peticiones que permiten invocar otras actividades o iniciar otras funciones, como por ejemplo, realizar una llamada telefónica, abrir una dirección web o enviar un mensaje SMS.

Para que esta comunicación sea más eficaz, se utilizan los *Bundles* que son asignaciones clave – valor, que permiten cualquier tipo de información primitiva y se añaden al *Intent*.

Como se observa en la Fig. 3.6, se crea un *Intent* al cual se indica la actividad en la se ejecuta y la actividad que se quiere iniciar. A este *Intent* se le añade un *Bundle*, el cual contiene una serie de asignaciones clave – valor. Y finalmente se ejecuta el *Intent*.

```
Intent intent = new Intent(FitxaComerc.this,
        FitxaComercContacte.class);
Bundle bundle = new Bundle();
bundle.putString("titol", titol);
bundle.putString("descripcio", descripcio);
bundle.putString("adreca", adreca);
bundle.putString("telf", telf);
bundle.putString("fax", fax);
bundle.putString("email", email);
bundle.putString("web", web);

intent.putExtras(bundle);

startActivity(intent);
```

Fig. 3.6. Ejemplo de comunicación entre actividades.

3.9.4. *Service*.

Los servicios (*service)* son componentes sin interfaz gráfica que se ejecutan en segundo plano y sin interacción con el usuario. Desarrollan tareas importantes para el resto de las aplicaciones o para el sistema.

3.9.5. *Context.*

La clase *Context* deriva directamente de la clase *Object* y actúa como interfaz a la información global del entorno de una aplicación. Es decir, mediante la clase *Context* se puede acceder a toda la información de la aplicación que hace referencia a recursos y clases, así como ejecuciones para las operaciones a nivel de Aplicaciones, como lanzamiento de actividades, difusión y recepción de *intents*, etc.

3.10. Entorno de desarrollo.

3.10.1. Lenguajes de programación.

El lenguaje de programación nativo para desarrollar aplicaciones para Android es Java, el que se ha utilizado para crear la aplicación de este proyecto.

No obstante, también se pueden desarrollar aplicaciones con otros lenguajes de programación a través de diversas plataformas:

- ***App Inventor*:** impulsada por Google con el fin de que más personas se unieran a la familia de Android, esta plataforma de desarrollo está basada en un lenguaje de desarrollo gráfico en donde no hace falta escribir ninguna línea de código, tan solo hace falta arrastrar bloques identificados con la acción que se desee. Esta herramienta utiliza el navegador web como centro principal de trabajo, y almacena todo en servidores que están disponibles cada vez que se entra a Internet.
- ***HTML5*:** también se pueden crear aplicaciones con el famoso lenguaje de programación enfocado a páginas web. Una de las plataformas con mayor éxito es PhoneGap. La ventaja de utilizar HTML5 es que la aplicación desarrollada se puede utilizar en todos los sistemas operativos.

  Sin embargo, con el uso de Java (o lenguajes de programación nativos de cada sistema operativo móvil) se obtiene mayor rendimiento del hardware y rapidez de ejecución. Sin ir más lejos Mark Zuckerberg, creador de la red social Facebook, comentó en una entrevista que haber creado la aplicación para Android usando HTML5 fue un gran error (ver referencia [4]).
- ***Basic4Android*:** es una plataforma de programación cuyo lenguaje base de programación es VisualBasic, lenguaje que está orientado a aquellas personas que empiezan en el mundo de la programación de una manera más gráfica y no tan abstracta.

3.10.2. Herramientas de desarrollo.

Todas las herramientas para el desarrollo de aplicaciones nativas son gratuitas y se pueden descargar fácilmente desde Internet. A continuación se dan a conocer las herramientas necesarias:

- **Java JDK (*Java SE Development KIT*):** es un kit de desarrollo de Java, el cual es necesario tener instalado en el ordenador a utilizar. Se puede obtener desde: ver referencia [5].
- **Eclipse:** es un entorno de desarrollo integrado muy poderoso, con el que se pueden desarrollar aplicaciones con diferentes lenguajes. Se puede descargar desde: ver referencia [6].
- **Android SDK (*Software Development Kit)*:** es el kit de desarrollo de Android, que contiene bibliotecas, documentación, ejemplos, un depurador para probar las aplicaciones, a demás de un emulador por si no se dispone de un dispositivo Android real aunque su rendimiento es bastante inferior a un dispositivo real. Se puede obtener desde: ver referencia [7].
- **ADT (*Android Development Tools*):** es un conjunto de herramientas de desarrollo de Android, que suministra:
    - Un asistente de proyectos que genera todos los archivos necesarios.
    - Editores de recursos específicos de Android.
    - Gestor de dispositivos virtuales, perspectiva DDMS (*Dalvik Debug Monitor Server*) de Eclipse para controlar y depurar aplicaciones.
    - Integración con la utilidad *LogCat* de Android para la creación de trazas de ejecución.
    - Integración con la utilidad AHV (*Android Hierarchy Viewer)*.
    - Compilación y ejecución de aplicaciones para su ejecución en dispositivos y emuladores.
    - Herramientas de firma de código con certificados digitales y empaquetado (APK, *Android Packages Kit*) para la distribución de aplicaciones.

  Para instalarlo, hay que abrir el elemento de menú Ayuda/Instalar nuevo software… del menú de tareas de Eclipse. En la ventana de instalación que surge, se debe introducir la dirección: ver referencia [8].

3.10.3. Ejecución y depuración de aplicaciones.

Para poder ejecutar las aplicaciones creadas, primero se ha de configurar el proyecto de la aplicación para que pueda ser depurado. Mediante el ADT, se puede configurar el proyecto fácilmente. Para ello, hay que seguir tres pasos:

- Crear y configurar un dispositivo virtual, si no se dispone de un dispositivo real. Se ha de tener en cuenta la versión Android en la que está programada la aplicación a la hora de escoger la versión del dispositivo virtual.
- Crear una configuración de depuración para el proyecto. Si se desea probar la aplicación en un dispositivo real, hay que activar el modo "Depuración de USB". Para ello, hay que dirigirse a: Ajustes → Aplicaciones → Desarrollo → Marcar la opción "Depuración de USB".
- Ejecutar la aplicación.

3.10.4. Proyectos Android.

Un proyecto Android está formado por diversos directorios y archivos, de los cuales hay que destacar:

- **Carpeta *src*:** contiene los archivos con el código fuente de la aplicación.
- **Android 2.3.3 (versión escogida)**: esta carpeta contiene el archivo *android.jar*, que almacena todas las bibliotecas de clases necesarias para una aplicación.
- **Carpeta *gen*:** contiene el archivo *R.java*, generado por el compilador y que hace referencia a todos los recursos encontrados en el proyecto. Mediante este archivo, se podrá acceder a todos los recursos mediante la referencia correspondiente. No se debe editar manualmente este archivo.
- **Carpeta *assets*:** en esta carpeta se encuentran los recursos no compilados e integrados en la aplicación, como archivos de texto, base de datos, etc. Estos recursos no se encuentran reflejados en el archivo *R.java*.
- **Carpeta *res*:** contiene los recursos utilizados por la aplicación: animaciones, imágenes, cadenas de texto, estilos, etc. Cabe destacar las siguientes carpetas que incluye:

  - ***Res/drawable:*** se encuentran las imágenes de la aplicación. Teniendo en cuenta las diferentes densidades de pantallas de los dispositivos con Android, también se encuentran las carpetas *drawable-ldpi, drawable-mdpi, drawable-hdpi* y *drawable-xhdpi.*
  - ***Res/layout:*** contiene los diferentes archivos XML donde se define el contenido de las pantallas de la interfaz gráfica.
  - ***Res/menu*****:** archivos XML con los menús de la aplicación.
  - ***Res/values*****:** contiene recursos de la aplicación, como cadenas de texto, estilos, colores o dimensiones. Para desarrollar la aplicación, se han utilizado estilos y cadenas de texto, para evitar la repetición de código.
  - ***Res/xml*****:** contiene archivos XML para otros usos.
- ***AndroidManifest.xml*****:** es un archivo XML con el manifiesto de la aplicación, lo que incluye información sobre el paquete, la versión mínima SDK, las actividades utilizadas y los permisos que la aplicación debe tener (por ejemplo, acceder a la función de llamar por teléfono, abrir otra aplicación, acceder al estado de la batería, etc.) para poder ejecutarse. Es decir, contiene la configuración de la aplicación.

4. Descripción del proyecto.

4.1. Especificaciones del proyecto.

En el momento de iniciar el proyecto, se realizó una reunión con un representante de la empresa o cliente que proponía el proyecto, en este caso con la persona al cargo del Área de Nuevas Tecnologías del Ayuntamiento de Palafolls, Emília García.

En esta primera reunión, se definieron los siguientes aspectos:

- La información que tendría que mostrar la aplicación.
- La aplicación tendría que ser dinámica, es decir, obtener la información de Internet. De esta manera, se evita tener que ir actualizando la aplicación cada vez que se desea añadir, borrar o modificar información.
- La aplicación que el cliente había tomado como referencia era *GironaIn*, del Ayuntamiento de Girona.
- Creación de un *widget* de escritorio que mostrase el tiempo de la ciudad, las últimas noticias y últimas publicaciones del perfil de Facebook.

4.1.1. Estudio de mercado.

Una vez conocidos los requerimientos y las necesidades del cliente, se realizó un estudio de mercado, pese a que el cliente ya tenía una aplicación de referencia. Se analizaron las siguientes aplicaciones móviles:

- ***GironaIn* (Ayto. de Girona).** Cuenta con un agradable y sencillo diseño para el usuario. Presenta la información siempre con el mismo formato, de manera que hace el uso de la aplicación más fácil e intuitivo. Se accede a Internet para obtener los datos, si bien los tiempos de espera no superan los tres segundos, un tiempo que el usuario puede esperar sin exaltarse.

  En el proyecto realizado se ha optado por almacenar los datos en Internet y presentarlos con el mismo formato en la medida de lo posible.
- **"Barcelona al mòbil" (Ayto. de Barcelona).** Consta de un menú con pestañas o TabHost (ver sub apartado 4.2.2), con cinco secciones. La aplicación en sí

misma no muestra información, sino que ofrece un listado por cada sección. Al hacer clic en un elemento del listado, se abre una página web con la información a mostrar. Los tiempos de espera superan en ocasiones los tres segundos, lo que puede incomodar al usuario.

En el proyecto realizado se ha optado a implementar los menús con pestañas al ser un modo fácil de navegación. Por el contrario, al hacer clic en cualquier listado o contenido de la aplicación, sólo se abre una página web en aquellos casos estrictamente necesarios.

- **Ayto Huesca (Ayto. de Huesca).** Cuenta con un buen diseño. Sin embargo, no ofrece suficiente información para que el usuario sepa cómo utilizar la aplicación. Ofrece información innecesaria o carente de utilidad para el usuario. Ha resultado imposible determinar los tiempos de acceso, al desconocer el funcionamiento de la aplicación.

  La aplicación realizada sólo ofrece información necesaria para el usuario, así como se ha aplicado una interfaz gráfica intuitiva para que el usuario no tenga problemas a la hora de utilizar la aplicación.
- **Ayto Cartagena (Ayto. de Cartagena).** Tiene un diseño no muy cuidado. La distribución de la información es un poco caótica. Obtiene la información de Internet pero sólo cuando se abre la aplicación por primera vez o bien si el usuario desea actualizarla. Ésta obtención de la información puede resultar incómoda por el hecho de que tarda entre 10 y 20 segundos, a la vez que puede ser perjudicial para el usuario si no se acuerda de actualizarla.

  Para el desarrollo del proyecto, se ha estudiado detenidamente la distribución de la información para que se muestre de una forma estructurada. No se ha optado por delegar en el usuario la actualización del contenido de la aplicación.

4.2. Casos de uso.

En la Fig. 4.1 se muestra el diagrama con los casos de uso más representativos de la aplicación.

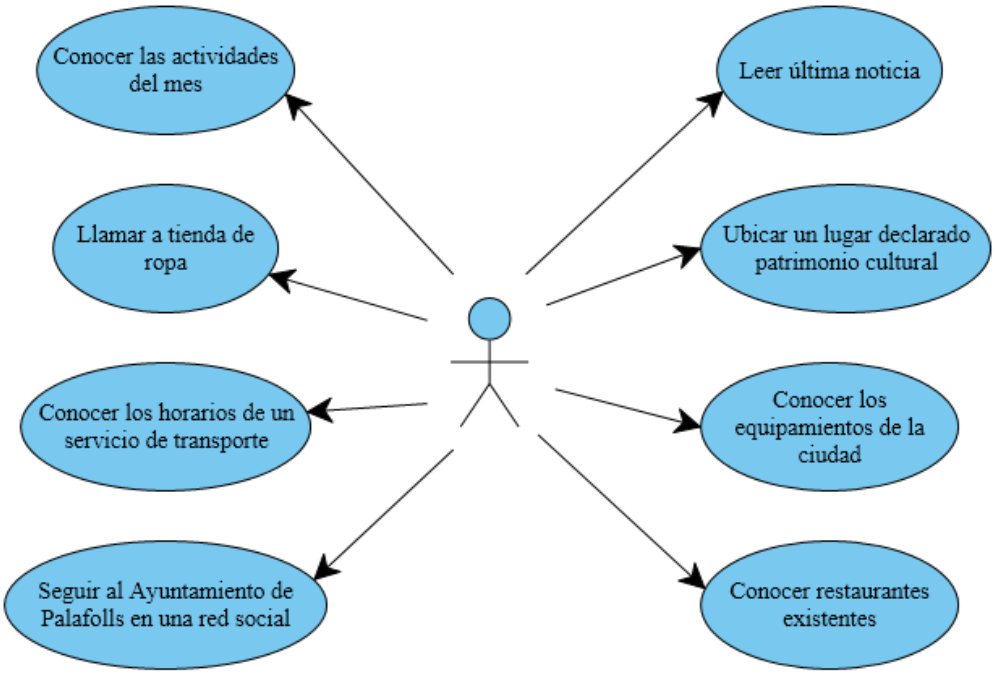

Fig. 4.1. Diagrama de casos de uso.

A continuación se describen los casos de uso:

Caso de uso: conocer las actividades del mes.	
Actor: usuario.	
Descripción: el usuario desea conocer todas las actividades que se realizan en el mes actual.	
Flujo principal	
Usuario	**Sistema**
El usuario accede al menú "Agenda" del menú principal.	
	El sistema carga un menú de pestañas con cuatro categorías de actividades. La que se muestra inicialmente es la de actividades mensuales.
El usuario se descarga un archivo PDF con las actividades del mes	

Tabla 4.1. Caso de uso: conocer las actividades del mes.

Caso de uso: llamar a tienda de ropa.	
Actor: usuario.	
Descripción: el usuario desea ponerse en contacto por teléfono con una tienda de ropa.	
Flujo principal	
Usuario	**Sistema**
Accede al menú "Comerç" del menú principal.	
	Muestra un listado con las categorías de comercios.
Selecciona la categoría "Moda i complements".	
	Muestra un listado con todos los locales de la categoría "Moda i complements".
Selecciona la tienda que desea.	
	Muestra una descripción del local y un botón para ver la información de contacto.
Aprieta el botón "Contacte".	
	Expone los datos de contacto del local.
Oprime el número de teléfono de la tienda.	
	Se abre la función de marcación de número del dispositivo.
Llama al número marcado por el sistema.	

Tabla 4.2. Caso de uso: llamar a tienda de ropa.

Caso de uso: conocer los horarios de un servicio de transporte.
Actor: usuario.
Descripción: el usuario desea conocer los horarios de un servicio de transporte
Flujo principal

Usuario	Sistema
Accede al menú "Transports" del menú principal.	
	Muestra un listado con los diferentes servicios de transporte.
Selecciona el servicio que desea.	
	Muestra la información del servicio así como un botón para descargar un tríptico informativo.
El usuario se descarga el tríptico del servicio.	

Tabla 4.3. Caso de uso: conocer los horarios de un servicio de transporte.

Caso de uso: seguir las novedades del Ayuntamiento de Palafolls en una red social.	
Actor: usuario.	
Descripción: el usuario desea seguir las novedades del Ayuntamiento de Palafolls en una red social.	
Flujo principal	
Usuario	**Sistema**
Accede al menú "Xarxes socials" del menú principal.	
	Muestra las diferentes redes sociales en las que el Ayuntamiento está presente.
Selecciona la red social que desea.	
Sigue al Ayuntamiento en la red social escogida.	

Tabla 4.4. Caso de uso: seguir las novedades del Ayuntamiento de Palafolls en una red social.

Caso de uso: leer última noticia.	
Actor: usuario.	
Descripción: el usuario desea leer la última noticia publicada.	
Flujo principal	
Usuario	**Sistema**
El usuario accede al menú "RSS" del menú principal.	
	El sistema accede a Internet y muestra el título de las últimas noticias en un listado.
El usuario selecciona la primera noticia.	
	El sistema abre la página web dónde está publicada la noticia.

Tabla 4.5. Caso de uso: leer última noticia.

Caso de uso: ubicar un lugar declarado patrimonio cultural.	
Actor: usuario.	
Descripción: el usuario desea saber la ubicación en el mapa de un lugar declarado patrimonio cultural.	
Flujo principal	
Usuario	**Sistema**
Accede al menú "Patrimoni" del menú principal.	
	Muestra un listado con todos los lugares declarados patrimonio cultural.
Selecciona un lugar.	
	Ofrece toda la información acerca del lugar seleccionado, así como un botón para ver la video guía del lugar y otro distinto para ver su localización.

Aprieta el botón “Localització”.	
	Abre un mapa indicando la ubicación del lugar.

Tabla 4.6. Caso de uso: ubicar un lugar declarado patrimonio cultural.

Caso de uso: conocer los equipamientos de la ciudad.	
Actor: usuario.	
Descripción: el usuario desea conocer los equipamientos de los que dispone la ciudad.	
Flujo principal	
Usuario	**Sistema**
Accede al menú “Informació” del menú principal.	
	El sistema carga un menú de pestañas con dos categorías: “Serveis” y “Equipaments”. La pestaña seleccionada inicialmente es “Serveis”.
Hace clic sobre la pestaña “Equipaments”.	
	Muestra un listado con las diferentes categorías en las que están organizados los equipamientos.

Tabla 4.7. Caso de uso: conocer los equipamientos de la ciudad.

Caso de uso: conocer restaurantes existentes.	
Actor: usuario.	
Descripción: el usuario desea conocer todos los restaurantes que existen en la ciudad.	
Flujo principal	
Usuario	**Sistema**
Accede al menú “Restauració” del menú principal.	

	El sistema carga un menú de pestañas con dos categorías: “Oci” y “Restauració”. La pestaña seleccionada inicialmente es “Oci”.
Selecciona la pestaña “Restauració”.	
	Lista todos los restaurantes que existen en la ciudad.

Tabla 4.8. Caso de uso: conocer restaurantes existentes.

4.3. Estructura del proyecto.

4.3.1. Arquitectura.

La arquitectura del proyecto es la que se puede observar en la Fig. 4.2.

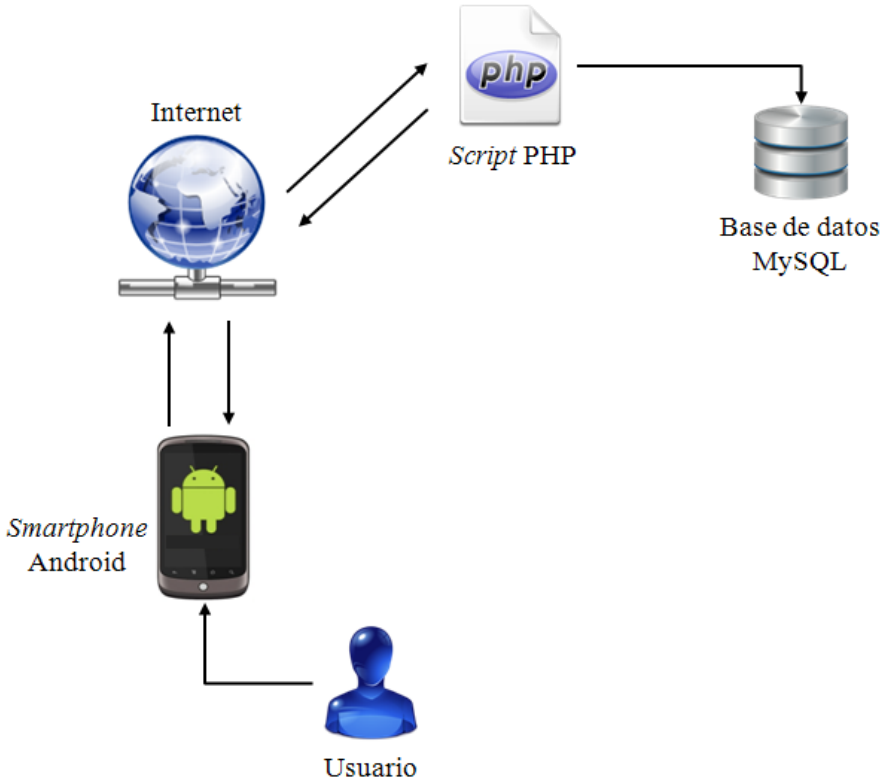

Fig. 4.2. Estructura del proyecto.

Toda la información se almacena en una base de datos MySQL. Para acceder a ella, es necesario conectarse mediante un *script* PHP en el cual se especifica que información se obtendrá y se mostrará al usuario.

La base de datos y el servidor FTP donde se almacenan los *scripts* son propiedad del Ayuntamiento de Palafolls.

4.3.2. Clases principales.

Para el correcto funcionamiento de la aplicación, las clases esenciales son las siguientes:

- **ConexionInternet.java:** esta clase sirve para comprobar que el dispositivo está conectado a Internet. A través de un método, se hace saber si existe la conexión o no. En caso negativo, se muestra un *ProgressDialog* (ver sub apartado 4.2.3) advirtiendo de que es necesario disponer de conexión.
- **InterfaceLoading.java:** es una interfaz con tres métodos abstractos, que serán definidos por las clases que lo implementen.
- **Item.java:** una clase donde se almacena la información a mostrar. Dispone de diversos atributos, varios constructores para diferentes necesidades y el correspondiente método "get()*"* para cada atributo.
- **ItemAdapter.java:** dado que la mayoría de información se presenta a través de una *List view* (ver sub apartado 4.2.3), esta clase que extiende de ArrayAdapter<Item>, es el adaptador necesario para presentar la información. Puesto que cada fila de la lista es una vista, se tiene en cuenta la información a mostrar: una imagen a la izquierda junto a un texto, sólo un texto, un texto debajo de otro, etc. Según la información a mostrar, se selecciona una vista u otra (archivos XML en la carpeta */res/layout* [ver sub apartado 3.10.4]).
- **LoadingDialog.java:** clase que extiende de AsyncTask<Void, Void, Void>. Su función es la de mostrar un *ProgressDialog* (ver sub apartado 4.2.3) mientras se accede a Internet, se obtiene la información y se prepara para ser mostrada a través de una lista. Una vez la información está preparada, se oculta el mensaje. Todo eso se ejecuta en tres métodos:
    - **onPreExecute()**: se muestra el mensaje.

- **doInBackground():** se accede a Internet y se obtiene toda la información necesaria.
- **onPostExecute():** se crea y establece el adaptador de la lista. Se oculta el mensaje.

Para mostrar, por ejemplo, un listado con los sitios de interés cultural, se seguirían los siguientes pasos:

- **Crear una clase que extienda de *ListActivity* e implemente la interfaz *InterfaceLoading*:** como la mayor parte de la información se presenta a través de listas, crear una clase que extienda de *ListActivity* simplifica la codificación. Los métodos abstractos a codificar de la interfaz, sirven en este caso para definir el adaptador de la lista.
- **Crear una instancia de un objeto de la clase *LoadingDialog*:** indicando por parámetro, entre otros, el *script* al cual se desea conectar. Una vez creada la instancia, invocar el método "execute()" de la clase *LoadingDialog* para ejecutar la tarea de acceder a Internet, obtener y mostrar la información.

4.3.3. Recursos principales.

Ya se ha comentado anteriormente que cada versión ofrece unas ciertas funcionalidades para desarrollar una aplicación. A continuación se detallan las principales funcionalidades que se han utilizado para este proyecto:

- ***Linear layout*** (Fig. 4.3)**:** es un grupo de vista que alinea todos los elementos que contiene en una sola dirección, vertical u horizontalmente. Se puede especificar la dirección a través del atributo a*ndroid:orientation.*

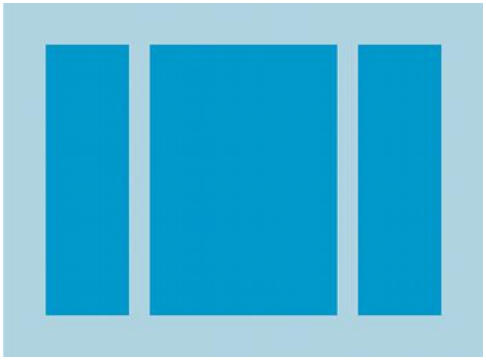

Fig. 4.3. *Linear layout*.

- ***List view*** (Fig. 4.4)**:** es un grupo de vista que muestra una lista de elementos desplazables. Los elementos de la lista se insertan automáticamente a la lista utilizando un adaptador que extrae el contenido de una fuente, tal como una consulta de matriz o base de datos, y convierte cada resultado en una vista que se coloca en la lista. Las listas utilizadas en la aplicación han sido personalizadas, ya que por defecto solo muestran una línea de texto por fila.

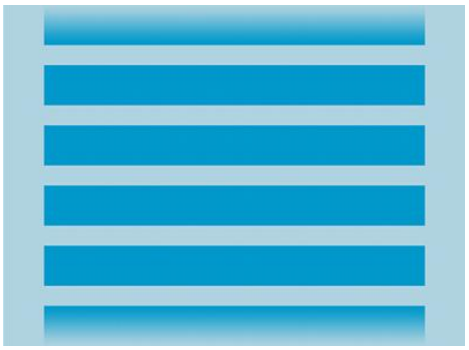

Fig. 4.4. *List view*.

- ***TabHost*** (Fig. 4.5)**:** es un contenedor para una vista de ventana con pestañas. Este objeto tiene dos hijos: un conjunto de etiquetas de pestañas donde el usuario hace clic para seleccionar una ficha específica, y un objeto *FrameLayout* que muestra el contenido de esa página.

Fig. 4.5. *TabHost*.

- ***ProgressDialog*** (Fig. 4.6)**:** cuando se realiza una tarea no visible para el usuario que puede demorar varios segundos, como acceder a Internet y obtener

información, es recomendable mostrarle al usuario qué se preparando la información y que no se impaciente. Para ello se muestra un *ProgressDialog* o mensaje de espera.

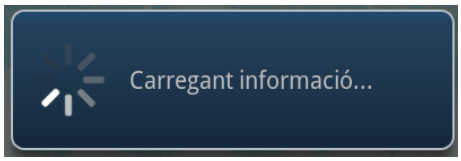

Fig. 4.6. *ProgressDialog*.

- **Menú de opciones** (Fig. 4.7)**:** aparece el menú cuando el usuario aprieta el botón físico "menú" del móvil. Es donde se incluyen acciones o opciones como "Ayuda", "Compartir", "Configuración", "Contacto", etc. Ha de ser configurado en cada pantalla o actividad en la que se desee mostrar.

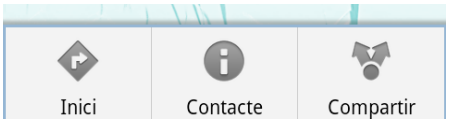

Fig. 4.7. Menú de opciones.

4.4. Base de datos.

Uno de los requerimientos principales es que la información que se muestre en la aplicación ha de ser la que se encuentre en la base de datos remota de la que se nutre la página web del ayuntamiento de Palafolls.

Después de analizar y realizar multitud de pruebas con la base de datos existente, se optó por crear una base de datos específica para la aplicación, debido a una ineficaz estructura (datos dispersos) y a los diferentes formatos en que la información estaba almacenada.

Se ha utilizado MySQL como gestor de la base datos y utilizado la herramienta de gestión phpMyAdmin.

4.4.1. Estructura.

La estructura de la base de datos específica para la aplicación es la que se muestra en la Fig. 4.8:

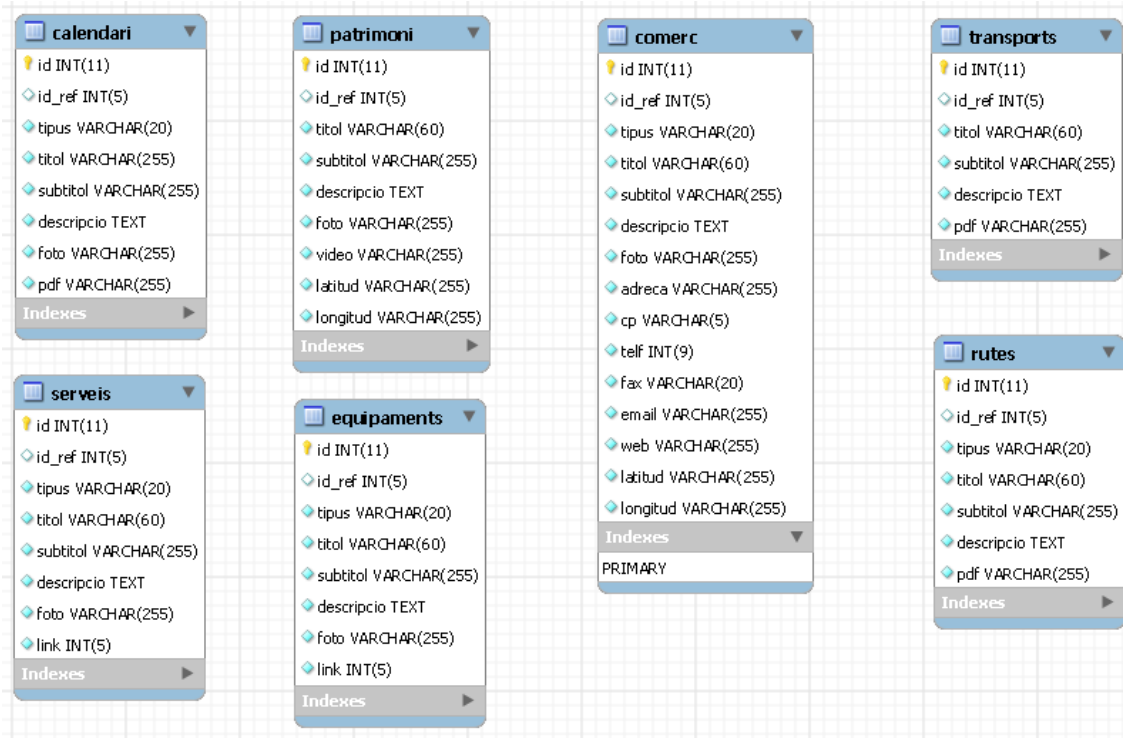

Fig. 4.8. Estructura de la base de datos.

Está formada por siete tablas para facilitar su mantenimiento una vez finalizado el proyecto. En todas las tablas se ha creado una columna, "id_ref", que vincula la información propia con la de la base de datos que se utiliza para la página web, que en las tablas que posee también tiene una columna con el mismo nombre. Al mismo tiempo servirá para insertar, borrar o modificar la información en ambas base de datos de una forma automatizada.

Todos los datos que se almacenan están en un formato de texto plano, para facilitar su manipulación, ya que en la base de datos de la página web en algunos casos contiene código HTML. Al mismo tiempo, la mayoría de columnas de las diversas tablas tienen el mismo nombre para facilitar su obtención y manipulación.

A continuación, se detalla la información que se almacena en cada tabla:

- **"Calendari":** datos del menú "Calendari" de la aplicación, que está dividido en cuatro apartados. La información de cada apartado se conoce a través de la columna "tipus", que contiene cuatro tipos de valores, uno para cada sección. Las demás columnas sirven para almacenar toda la información a mostrar.
- **"Patrimoni":** contiene la información del menú "Patrimoni" de la aplicación: el nombre del lugar, la breve descripción del listado ("subtitol"), la descripción, la ruta de donde se almacena la imagen del lugar, la dirección web de la video guía, y los valores de latitud y longitud para su situación en el mapa.
- **"Comerc":** alberga los datos del menú "Comerç" y "Restauració" de la aplicación. Dentro de cada menú hay varias categorías, que se ven reflejadas en la columna "tipus". El resto de columnas almacenan la información de los locales de la guía comercial y de restauración.
- **"Transports":** se almacenan la información del menú "Transports" de la aplicación: el nombre del servicio de transporte, la breve descripción del listado ("subtitol"), la descripción y la ruta del archivo PDF para descargar.
- **"Serveis":** contiene los datos de la sección "Serveis" del menú "Informació" de la aplicación. La columna "tipus" hace referencia a la categoría del servicio, y el resto de columnas contiene la información a mostrar del servicio.
- **"Equipaments":** contiene los datos de la sección "Equipaments" del menú "Informació" de la aplicación. Su funcionamiento es el mismo al de la tabla "Serveis", pero se han separado en dos para mejorar su manipulación.
- **"Rutes":** alberga la información del menú "Rutes" de la aplicación: el nombre de la ruta, una breve descripción del listado ("subtitol"), la descripción y la ruta del archivo PDF para descargar.

4.4.2. Conexión mediante *script.*

La conexión con la base de datos se realiza a través de un *script* PHP en el lado del servidor. Se han creado seis *scripts* para que sea más fácil de mantener una vez finalizado el proyecto. Para alojar estos archivos en el servidor FTP, se ha utilizado el software FileZilla Client.

El contenido de dichos *scripts* varia, pero en general su funcionamiento es el siguiente:

- Se establece una conexión a la base de datos (Fig. 4.9):

```
$connexio = mysql_connect("[redacted]", "[redacted]", "[redacted]");

if(!$connexio){
    die('No es pot connectar: ' . mysql_error());
}

mysql_select_db("palafoll_app", $connexio);
mysql_set_charset("utf8",$connexio);
```

Fig. 4.9. Conexión a la base de datos.

- Se ejecuta la sentencia deseada y se almacena la información para su mejor manipulación (ver Fig. 4.10):

```
$result = mysql_query("SELECT * FROM rutes WHERE tipus='list' ORDER BY titol ");

while ($row = mysql_fetch_assoc($result)) {
    $output[] = $row;
}
```

Fig. 4.10. Sentencia y almacenamiento de información.

- Se imprime el resultado en formato JSON y se cierra la conexión (Fig. 4.11):

```
print(json_encode($output));

mysql_close($connexio);
```

Fig. 4.11. Impresión resultado y cierre de conexión.

4.4.3. Clase ScriptBBDD.

Para que el dispositivo móvil se conecte al *script*, se ha creado una clase llamada ScriptBBDD, que contiene el siguiente atributo:

- **String rutaScript:** contiene como valor por defecto la ruta de Internet donde se encuentran los *scripts*.

Y los métodos:

- **ScriptBBDD(String script):** es el constructor de la clase, el cual concatena el valor que recibe como parámetro al atributo "rutaScript".

- **toJSON():** realiza una conexión a la dirección con el valor del atributo "rutaScript" y prepara la información en formato JSON.
- **getTitols():** invoca el método toJSON() y devuelve un vector con el valor de la columna "titol" de las tablas a las que se accede mediante el *script*.
  Están codificados varios métodos más, pero tienen una estructura muy similar, simplemente se diferencian de éste método por su nombre y el valor que se devuelve.

4.4.4. Obtención de datos.

Para poder obtener los datos almacenados en la base de datos, hay que seguir los siguientes dos pasos:

- Crear una instancia de un objeto de la clase ScriptBBDD, indicando por parámetro el nombre del archivo o *script* al cual se quiere realizar una conexión.
- Invocar al método correspondiente para obtener la información deseada.

4.5. *Widget.*

Un *widget* de escritorio es una mini aplicación diseñada para proveer información o mejorar una aplicación.

Uno de los requerimientos iniciales era el de, aparte de ofrecer información meteorológica en el *widget*, incluir las ultimas noticias publicadas y las entradas publicadas en el perfil de Facebook. Finalmente, sólo se ha podido implementar la función de ofrecer información meteorológica, debido a que un *widget* funciona diferente a una aplicación y que para leer las entradas de Facebook se tiene que trabajar con la API de Facebook, hecho que ha resultado imposible por falta de tiempo. Dado que no se han podido añadir las últimas noticias, se ha añadido un botón el cual redirige al usuario a la aplicación, al menú de RSS.

El resultado del *widget* desarrollado es el que se observa en la Fig. 4.12. Muestra una imagen de la situación meteorológica, la temperatura actual, la temperatura mínima y máxima, el escudo y nombre de la ciudad de Palafolls, la fecha actual, un botón para

actualizar la información (los *widget* se actualizan automáticamente cada 30 minutos) y un botón que redirige al usuario a las últimas noticias.

Fig. 4.12. *Widget* desarrollado.

Para crear el *widget* de la aplicación, se ha trabajado con la API de Yahoo! Weather (ver referencia [9]).

4.5.1. Clase Weather.

Esta clase contiene los siguientes atributos:

- **String unit_temp:** unidad de temperatura (ºF o ºC).
- **String temp:** valor de la temperatura.
- **String YahooWcode:** código que facilita la API de Yahoo sobre la situación meteorológica.
- **String MaxT:** valor de la temperatura máxima.
- **String MinT:** valor de la temperatura mínima

También cuenta con el siguiente método:

- **toString():** devuelve una cadena de texto con toda la información del objeto.

4.5.2. Clase WidgetWeather.

Esta clase contiene el código principal que hace funcionar al *widget*. Como se ha comentado en el apartado 4, un *widget* no funciona igual que una aplicación; una de las diferencias es que las clases que se utilizan extienden de la clase *AppWidgetProvider* y no de la clase *Activity*. Tiene los siguientes métodos:

- **onUpdate(Context context…):** es el método que se invoca cuando se crea el *widget*. Invoca al método get_data.

- **get_data(Context ctx, AppWidgetManager appwidg):** invoca los métodos queryYahooWeather, convertStringToDocument, parseWeather, y obtiene la información meteorológica a mostrar a través de una instancia de un objeto de la clase Weather. Una vez obtenida la información, actualiza el *widget* con los nuevos datos.
- **actualizarWidget(Context context…):** actualiza el *widget* con la información que recibe por parámetro.
- **imgTiempo(int yahoowcode):** según el valor del parámetro que recibe (el código de Yahoo sobre la situación meteorológica), devuelve el identificador de la imagen a mostrar.
- **queryYahooWeather(String ciudadID):** se realiza una petición al servicio web de Yahoo, indicando la ciudad que se desea. Retorna el resultado de esa petición, toda la información que ofrece la API sobre esa ciudad.
- **convertStringToDocument(String src):** convierte la información devuelta por el método QueryYahooWeather en un objeto del tipo Document, para su mejor manipulación.
- **parseWeather(Document srdDoc):** se encarga de analizar sintácticamente toda la información del objeto Document creado por el método convertStringToDocument, y extraer únicamente los datos necesarios. Devuelve una instancia de un objeto de la clase Weather.
- **onReceive(Context context…):** actualiza la información que se muestra si se aprieta el botón de actualizar.

4.5.3. Documento widget_provider.xml.

Ésta archivo define las cualidades esenciales del *widget*, como las dimensiones mínimas, el *layout* inicial y el período de actualización. Ha de ser un archivo XML y estar ubicado en la carpeta */res/xml*.

4.5.4. Implementación.

Una vez implementadas las clases anteriores, el archivo referenciado en el sub apartado 4.3.3., y definido el archivo *layout* con el contenido de la interfaz del *widget*, hay que definirlo en el archivo AndroidManifest.xml. En la Fig. 4.13 se muestra como:

```
<receiver
    android:name=".Widget.WidgetWeather"
    android:exported="false"
    android:label="Palafolls Meteo" >
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE"
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_weather_provider" />
</receiver>
```

Fig. 4.13. Definición del *widget* en el archivo AndroidManifest.xml.

De esta forma, el *widget* ya estará disponible para ser ubicado en el escritorio del dispositivo, a través del nombre definido en el atributo a*ndroid:label.*

5. Manual para el mantenimiento.

Debido al carácter público de este documento, aquellos datos de carácter privado (contraseñas, direcciones IP…) se ocultarán con un sombreado.

Para realizar este manual, se asume que el usuario posee un nivel medio en conocimientos de programación y ha estudiado a fondo la integridad de la documentación. La totalidad del proyecto Android contiene comentarios para facilitar su compresión y mantenimiento.

5.1. Base de datos y acceso.

La base de datos está situada en el servidor: **“[redacted]”**. Para poder acceder se necesita la cuenta con *login*, **“[redacted]”**, y contraseña, **“[redacted]”**.

Al realizar la conexión, se encontrarán dos bases de datos:

- **palafoll_palafolls:** la base de datos que contiene la información de la página web. Con la cuenta anteriormente mencionada, en esta base de datos sólo se tienen permisos de lectura.
- **palafoll_app:** la base de datos que contiene la información de la aplicación. Se tienen todos los permisos sobre esta base de datos

Los *scripts* utilizados para acceder a la base de datos a través de la aplicación se encuentran en el servidor: “[redacted]”. La cuenta necesaria para acceder, tiene como *login* “[redacted]”, y contraseña “[redacted]”.

Tanto la estructura y la descripción de la base de datos de la aplicación, el funcionamiento de los *scripts* y la vía para conectarse a través de los *scripts*, está descrito en el apartado 4.3.

5.2. Proyecto Android.

5.2.1. Recursos de la aplicación.

Las imágenes utilizadas se encuentran en las carpetas *“res/drawable-hdpi”*, *“res/drawable-ldpi”*, *“res/drawable-mdpi”*, y *“res/drawable-xhdpi”*, para cada una de las

resoluciones comentadas en el apartado 3.3. Las imágenes tienen que tener el mismo nombre en cada una de las carpetas.

El menú de la aplicación que se muestra al apretar el botón físico de menú del dispositivo, está declarado en el archivo *res/menu/menu_app.xml*, como se observa en la Fig. 5.1:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/opcion_inici"
        android:icon="@android:drawable/ic_menu_directions"
        android:orderInCategory="1"
        android:title="Inici">
    </item>
    <item
        android:id="@+id/opcion_sobre"
        android:icon="@android:drawable/ic_menu_info_details"
        android:orderInCategory="2"
        android:title="Contacte">
    </item>
    <item
        android:id="@+id/opcion_compartir"
        android:icon="@android:drawable/ic_menu_share"
        android:orderInCategory="3"
        android:title="Compartir">
    </item>

</menu>
```

Fig. 5.1. Definición del menú de la aplicación.

Por cada elemento del menú se define:

- ***android:id:*** el identificar del elemento.
- ***android:icon:*** la imagen que se mostrará como icono.
- ***android:orderInCategory:*** la posición en la que se mostrará el elemento.
- ***android:title:*** el nombre del elemento.

En el directorio *res/values* se encuentran los siguientes archivos de recursos:

- ***colors.xml:*** contiene los colores utilizados en la aplicación.
- ***strings.xml*:** contiene las cadenas de texto utilizadas.
- ***styles.xml:*** engloba los estilos aplicados.

En los menús de pestañas utilizados, se muestran unas imágenes orientativas. Cuando se cambia de pestaña, también se cambia de imagen. La definición de qué imagen se ha de

mostrar cuando una pestaña está activada o no, se encuentra en los ficheros de la carpeta *res/drawable.*

5.2.2. Archivos de *layout.*

A continuación se especifican que contiene cada archivo XML de *layout.* También en que clase se aplican; ésta tarea se realiza a través del método *setContentView(int layoutResID)* de la actividad, indicando por parámetro la referencia del archivo *layout* obtenida a través de la clase R (ver apartado 3.10.4). Ordenados alfabéticamente.

- **activity_agenda_mensual.xml:** contiene un campo de texto y un botón. Aplicado en la clase "AgendaMensualActivity.java" del paquete "com.palafolls.palafollejant.Calendari".
- **activity_contacte.xml:** contiene varios campos de texto. Implementado en la clase "ContacteActivity.java" del paquete "com.palafolls.palafollejant".
- **activity_main.xml:** contiene las imágenes y los campos de texto del menú principal. Implementado en la clase "MainActivity.java" del paquete "com.palafolls.palafollejant".
- **activity_tabs.xml:** se define la estructura de cualquier actividad que utilice un menú de pestañas. Las clases que lo implementan son: "AgendaActivity.java", "InformacioActivity.java" y "OciRestauracioActivity.java" del paquete "com.palafolls.palafollejant".
- **activity_xarxes.xml:** implementa un campo de texto y dos botones de imágenes. Aplicado en la clase "XarxesActivity.java".
- **fitxa_calendari.xml:** contiene un campo de texto para mostrar la descripción de un evento definido en la agenda de actividades. Aplicado en la clase "FitxaCalendari.java" del paquete "com.palafolls.palafollejant.Objetos".
- **fitxa_comerc_contacte.xml:** contiene varios campos de texto para mostrar los datos de contacto de un comercio. Aplicado en la clase "FitxaComercContacte.java" del paquete "com.palafolls.palafollejant.Objetos".
- **fitxa_comerc.xml:** se define un campo de texto para la descripción de un comercio. Aplicado en la clase "FitxaComerc.java" del paquete "com.palafolls.palafollejant.Objetos".

- **fitxa_informacio_link.xml:** implementa un campo de texto para la descripción de elemento del menú "Equipaments". Aplicado en la clase "FitxaInformacioLink.java" del paquete "com.palafolls.palafollejant.Objetos".
- **fitxa_informacio.xml:** contiene un campo de texto para la descripción de elemento del menú "Equipaments" y un botón para abrir la ficha del servicio o equipamiento asociado. Aplicado en la clase "FitxaInformacio.java" del paquete "com.palafolls.palafollejant.Objetos".
- **fitxa_patrimoni.xml:** implementa un campo de texto para la descripción de un sitio declarado patrimonio cultural, así como un botón para visualizar su video guía y un segundo botón para ubicarlo en un mapa. Se aplica en la clase "FitxaPatrimoni.java" del paquete "com.palafolls.palafollejant.Objetos".
- **fitxa_ruta.xml:** contiene un campo de texto para la descripción de un recorrido por la ciudad o cercanías, y un botón para poder descargar un archivo PDF con la ruta. Se aplica en la clase "FitxaRuta.java" del paquete "com.palafolls.palafollejant.Objetos".
- **fitxa_transport.xml:** se define un campo de texto para la descripción de un servicio de transporte, a la vez que un botón para descargar un tríptico informativo. Aplicado en la clase "FitxaTransport.java" del paquete "com.palafolls.palafollejant.Objetos".
- **list_header.xml:** contiene la estructura para aquellas pantallas que muestran únicamente una lista, es decir, una *List view*.
- **list_row_foto_titol.xml:** define la estructura de una fila de una lista donde se muestra una foto y un texto como nombre del elemento.
- **list_row_foto.xml:** especifica la disposición de una fila de una lista donde se muestra una foto, un texto como nombre del elemento y un texto como breve descripción.
- **list_row_text.xml:** define la disposición de una fila de una lista en caso de que solo se muestre el nombre del elemento y una breve descripción.
- **list_row_titulo.xml:** se define la estructura de una fila de una lista dónde sólo se muestra un texto como nombre del elemento.
- **lista_texto_custom.xml:** especifica el contenido de aquellas listas cuyos elementos sólo muestran un campo de texto.

- **lista.xml:** contiene un *List view* para aquellas pantallas dónde muestra una lista pero debajo de un menú de pestañas.
- **widget_weather.xml:** define el aspecto visual del *widget* meteorológico de la aplicación.

5.2.3. Código fuente.

Antes de continuar, aclarar que el funcionamiento general del código fuente es el siguiente: se lista una serie de elementos con información de un objeto de la clase "Item.java". Al seleccionar un elemento, se lanza otra clase (actuando como ficha del elemento) enviando a través de un *Intent* una información más amplia sobre el elemento. Utilizando este patrón, facilita el uso y mantenimiento del proyecto.

A continuación se especifica el contenido y funcionamiento de cada uno de los archivos de la carpeta *src*, clasificados por los paquetes en los que están ubicados.

Paquete "com.palafolls.palafollejant":

- **AgendaActivity.java:** es la actividad del menú "Agenda" de la aplicación. Esta clase extiende de *TabActivity* para implementar un menú de pestañas. Dicho menú, se personalizada a través de un método para cambiar su color.
- **ComercActivity.java:** extiende de la clase *ListActivity* para facilitar su implementación. Es la actividad del menú "Comerç". Se configura el listado de categorías de comercios y se redirige a la actividad pertinente al seleccionar un elemento de la lista.
- **ContacteActivity.java:** extiende de la clase *Activity* y representa la pantalla del menú "Contacte" del menú de opciones de la aplicación. Se codifican los eventos de llamada, enviar email y abrir página web, según se presionen los campos de texto pertinentes. Al mismo tiempo se obtiene la versión de la aplicación definida en el archivo *AndroidManifest.xml* para ser mostrada.
- **InformacioActivity.java:** extiende de la clase *TabActivity* para implementar un menú de pestañas. Ésta clase corresponde al menú "Informació" de la aplicación.

- **MainActivity.java:** extiende de la clase *Activity* y corresponde a la pantalla inicial de la aplicación. Según que elemento del menú principal seleccione el usuario, se redirige a la actividad adecuada.
- **OciRestauracioActivity.java:** extiende la clase *TabActivity* y define el comportamiento del menú de pestañas. Corresponde al menú "Restauració" inicial.
- **PatrimoniActivity.java:** corresponde al menú "Patrimoni". Extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading* para implementar una lista con los sitios declarados como patrimonio cultural. Al hacer clic sobre un elemento, se lanza la actividad "FitxaPatrimoni.java".
- **RSSActivity.java:** esta clase extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading* para implementar una lista con las últimas noticias publicadas en la página web. Al seleccionar una noticia, se abre la página web dónde se está redactada la plenitud de la noticia.
- **TransportActivity.java:** extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading* para mostrar un listado con los servicios de transporte. Al hacer clic en un elemento de la lista, se lanza la actividad "FitxaTransport.java". Ésta clase corresponde al menú inicial de "Transports".
- **XarxesActivity.java:** extiende de la clase *Activity* y corresponde a la pantalla del menú "Xarxes socials". Configura los botones que hay en dicha interfaz para redirigir al usuario al perfil del Ayuntamiento a la red social correspondiente.

Paquete "com.palafolls.palafollejant.BBDD":

- **ScriptBBDD.java:** el funcionamiento de esta clase está descrito en el sub apartado 4.3.3.

Paquete "com.palafolls.palafollejant.Calendari":

- **AgendaMensual.java:** esta clase extiende de la clase *Activity*. En ella se codifica la descarga de las actividades programadas en el actual mes, después de apretar un botón.
- **CalendariListActivity.java:** extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading.* Se listan todas actividades de la categoría escogida a

través del menú de pestañas. Al seleccionar un evento o actividad, se lanza la actividad "FitxaCalendari.java".

Paquete "com.palafolls.palafollejant.Comerc":

- **ComercListActivity.java:** extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading.* Se listan los comercios de una categoría: esto es, los de los apartados "Comerç" y "Restauració" del menú inicial. Al seleccionar un comercio, se lanza la actividad "FitxaComerc.java".

Paquete "com.palafolls.palafollejant.Informacio":

- **InformacioListActivity.java:** esta clase extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading*. Su finalidad es la de mostrar todos los elementos de la categoría de un servicio o equipamiento, según la opción del menú de pestañas seleccionada del menú principal "Informació". Al seleccionar uno de los elementos, se lanza la actividad "FitxaInformacio.java".
- **EquipamentsActivity.java:** extiende de la clase *ListActivity* y muestra las diferentes categorías de equipamientos del menú "Equipaments" de la pantalla de "Informació" del menú principal. Al seleccionar una categoría, lanza la actividad "InformacioListActivity.java".
- **ServeisInfoActivity.java:** su configuración es la misma que la clase "EquipamentsActivity.java", excepto que muestra las categorías de servicios del menú "Serveis" de la pantalla "Información" del menú principal.

Paquete "com.palafolls.palafollejant.Objetos":

- Las clases: **ConexionInternet.java, InterfaceLoading.java, Item.java, ItemAdapter.java** y **LoadingDialog.java** están descritas en el sub apartado 4.2.1.
- El resto de clases de esta actividad, extienden de la clase *Activity*. En ellas, se obtiene la información almacenada en un *Bundle*, recibida a través de un *Intent* y se asigna a los campos de texto pertinentes.

Paquete "com.palafolls.palafollejant.OciRestauracio":

- **OciRestauracioListActivity.java:** extiende de la clase *ListActivity* e implementan la interfaz *InterfaceLoading*. En esta clase se listan los locales de ocio o de restauración del menú inicial "Restauració". Al seleccionar un local, se lanza la actividad "FitxaComerc.java".

Paquete "com.palafolls.palafollejant.Rutes":

- **PeuActivity.java:** extiende de la clase *ListActivity* e implementa la interfaz *InterfaceLoading*. Se listan las rutas que se pueden realizar por el municipio y al escoger una de ellas, se lanza la actividad "FitxaRuta.java".

Paquete "com.palafolls.palafollejant.Widget":

- El funcionamiento de las dos clases que contiene este paquete, **Weather.java** y **WidgetWeather** está especificado en los sub apartados 4.4.1 y 4.4.2.

Al mismo tiempo, se hace saber que en aquellas clases que no implementan un menú de pestañas, se codifica la configuración del menú de opciones de la aplicación. Tampoco las que correspondan al menú de opciones.

En todas aquellas clases en las cuales se accede a Internet para mostrar información, se crea un objeto de la clase "ConexionInternet.java" y se comprueba que exista conexión a Internet invocando al método "isOnline()". Si el resultado que devuelve es afirmativo (*true*) se muestra la información.

5.2.4. *Widget.*

La implementación del *Widget* de la aplicación está explicada en el apartado 4.4.

5.2.5. Archivo AndroidManifest.xml.

Del archivo "AndroidManifest.xml" hay que tener en cuenta los atributos:

- ***android:versionCode.*** Representa el número de versión de la aplicación. En cada revisión se debería de incrementar el valor en 1.
- ***android:versionName.*** Contiene información que el usuario puede visualizar. El formato del valor del atributo es <mayor>.<menor>.<micro>, siendo cada

uno de estos campos un valor numérico que comienza en cero y se incrementa de uno en uno. Se debe actualizar este valor para poder publicar la aplicación en Google Play.

- ***android:minSdkVersion.*** Designa el nivel API mínimo para que un dispositivo pueda ejecutar la aplicación. Para conocer el nivel API de una versión Android, basta con dirigirse a la página web indicada en la referencia [3].

Como se detalla en el sub apartado 3.10.4, en este archivo se declaran las actividades utilizadas por la aplicación y los permisos necesarios para que funcione correctamente.

5.3. Publicación de actualizaciones.

Para publicar actualizaciones de la aplicación, seguir las instrucciones detallas en el capítulo 8.

Los datos necesarios para la firma digital son los siguientes:

- Clave del almacén de contraseñas: [redacted].
- Nombre de la clave utilizada: [redacted].
- Contraseña de la clave: [redacted].

6. Manual para el uso.

6.1. Inicio.

Una vez se inicie la aplicación, aparecerá el menú principal de la aplicación (Fig. 6.1), a través del cual se podrá navegar por los diferentes apartados:

Fig. 6.1. Menú principal de la aplicación.

6.2. Menú “Agenda”.

Al hacer clic en el botón de Agenda, se abrirá una pantalla mostrando en la parte superior un menú de pestañas y debajo del mismo la información pertinente a cada menú (Fig. 6.2). Esto es, información acerca de eventos y/o actividades programadas para el mes actual, de carácter festivo, cultural y ferial. En el menú de actividades mensuales, se muestra un breve texto informativo y un botón a través del cual se descargará un archivo en formato PDF con las actividades del mes (Fig. 6.2).

Fig. 6.2. Menú de actividades mensuales.

En el resto de menús, se mostrará un listado con el nombre de las diferentes actividades, un breve texto descriptivo, en caso de haber, y una imagen referente al evento (Fig. 6.3).

Fig. 6.3. Listado de actividades feriales.

Al seleccionar una de ellas, se abrirá una pantalla con la descripción de la actividad (Fig. 6.4).

Fig. 6.4. Descripción de la actividad.

6.3. Menú "Patrimoni".

Se muestra un listado con los sitios declarados como patrimonio cultural de la ciudad de Palafolls (Fig. 6.5). Por cada emplazamiento se muestra el nombre, un breve texto descriptivo, en caso de disponer, y una imagen referente al lugar.

Fig. 6.5. Listado de sitios declarados patrimonio cultural.

Al hacer clic en algún elemento de la lista, se abre una pantalla mostrando (Fig. 6.6): la descripción del lugar, un botón para visionar una vídeo guía del sitio dando opción al usuario con que aplicación visionarla y un botón que al oprimirlo abrirá un mapa con la localización del emplazamiento.

Fig. 6.6. Información a cerca de un lugar declarado patrimonio cultural.

6.4. Menú RSS.

Esta sección muestra un listado con el título de las últimas noticias publicadas en la web del Ayuntamiento de Palafolls (Fig. 6.7). Al hacer clic en alguna de ellas, se abrirá una página web con el contenido de la noticia.

Fig. 6.7. Listado de noticias.

6.5. Menú "Comerç".

Expone un listado con los diferentes tipos de comercios de la ciudad, como se observa en la Fig. 6.8.

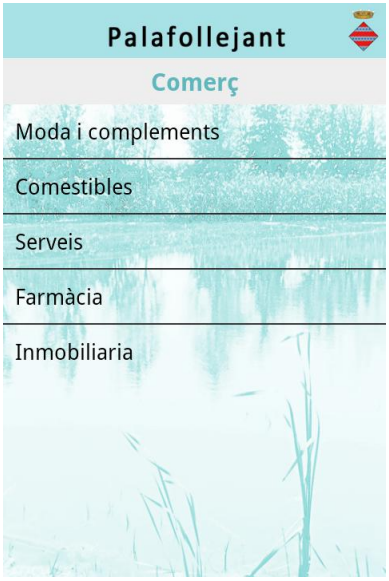

Fig. 6.8. Listado de las categorías de comercios.

Al hacer clic en una categoría de comercio, se mostrará un listado con los comercios de esa categoría (Fig. 6.9). De cada comercio se muestra: el nombre, un breve texto descriptivo, en caso de haber, y una imagen referente al negocio.

Fig. 6.9. Listado de comercios de la misma categoría.

Al hacer clic en un negocio, se observa una descripción del negocio en caso que haya y un botón para ver los datos de contacto y su localización en el mapa. Ver Fig. 6.10.

Fig. 6.10. Descripción de un comercio.

Al ver los datos de contacto (Fig. 6.11), se podrá hacer clic en los valores de "Telèfon", "Email" y "Web". Se abrirá la función de llamar por teléfono, un listado de aplicaciones a través de las cuales enviar un correo electrónico y una página web con la dirección correspondiente, respectivamente.

Fig. 6.11. Información de contacto de un comercio.

6.6. Menú "Informació".

Se muestra en la parte superior de la pantalla un menú de pestañas y debajo de este la información del menú (ver Fig. 6.12). Hay dos secciones: los servicios que ofrece el Ayuntamiento de Palafolls y los equipamientos de la ciudad donde se realizan los servicios, ambos clasificados por categorías.

Fig. 6.12. Contenido del menú "Informació".

Por cada categoría, un listado de los servicios o equipamientos correspondientes. Al seleccionar un ítem se abre una pantalla con la descripción del servicio o equipamiento. En caso de que un servicio este asociado a un equipamiento, mediante el botón que aparece en la ficha del servicio (ver Fig. 6.13), se podrá ver la información del equipamiento, y viceversa.

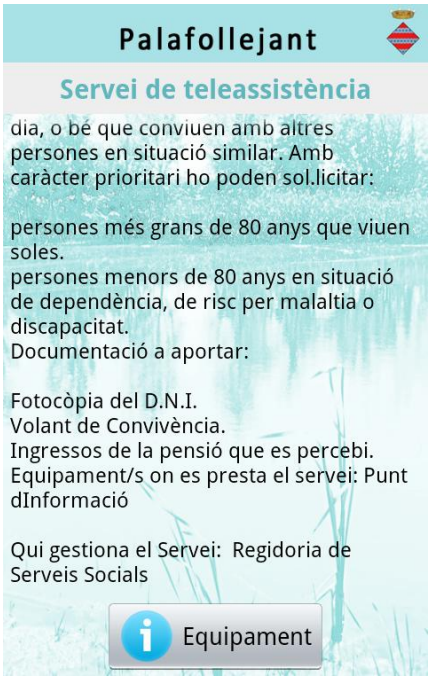

Fig. 6.13. Descripción de un servicio.

6.7. Menú "Restauració".

Se expone un menú de pestañas con los locales de ocio y restauración (Fig. 6.14). En cada menú, un listado con el nombre, un breve texto descriptivo en caso de haber y una imagen referente de todos los establecimientos.

Fig. 6.14. Contenido del menú "Restauració".

Su funcionamiento es el mismo que el menú "Comerç" (ver apartado 6.5), por cada local se muestra su descripción y se pueden ver los datos de contacto y su localización en el mapa.

6.8. Menú "Rutes".

El usuario contemplará un listado con las rutas más interesantes de Palafolls. Por cada ruta, una descripción de su recorrido así como un botón a través del cual se descargará un mapa con todo el itinerario, como se muestra en la Fig. 6.15.

Fig. 6.15. Descripción de una ruta.

6.9. Menú "Transports".

Se muestra un listado con distintos servicios de transporte. Al hacer clic en uno de los transportes, se presenta una descripción del servicio y un botón para descargarse un tríptico informativo, en caso de disponer de uno. Ver Fig. 6.16.

Fig. 6.16. Descripción de un servicio de transporte.

6.10. Menú "Xarxes socials".

Se muestran botones cuya función es servir de enlace al usuario hacia los perfiles del Ayuntamiento en las diferentes redes sociales. (Ver Fig. 6.17).

Fig. 6.17. Redes sociales en la que está presente el Ayuntamiento de Palafolls.

6.11. Menú de opciones.

Si se aprieta el botón físico de menú del dispositivo en cualquiera de las pantallas de la aplicación, aparece en la parte inferior el menú de opciones (Fig. 4.7). Las posibilidades que se ofrecen son:

- Dirigirse al menú principal.
- Ver la información de contacto del Ayuntamiento, así como el número de la versión de la aplicación (Fig. 6.18). Se puede hacer clic en los datos de teléfono, email y página web.

Fig. 6.18. Información de contacto del Ayuntamiento.

- Hacer saber a quien se desee la existencia de la aplicación. Se ofrecerá un listado de aplicaciones para poder compartir la experiencia.

7. Pruebas.

Conforme se iba desarrollando la aplicación, se ha ido testeando las funcionalidades implementadas. Los dispositivos utilizados a lo largo del desarrollo han sido los siguientes, destacando su procesador y la densidad de pantalla:

- Samsung Galaxy S Plus: versión Android 2.3.3 y densidad de pantalla HDPI.
- Samsung Galaxy Ace: versión Android 2.3.3 y densidad de pantalla MDPI.
- Dispositivo virtual de Android SDK: configurado con una densidad de pantalla LDPI y versión Android 4.0.

Se ha intentado probar la aplicación en el mayor número de dispositivos pese a que no se disponían de recursos económicos para adquirir nuevos dispositivos. Se ha tenido en cuenta la versión Android y la densidad de pantalla de cada dispositivo.

Tanto a lo largo del desarrollo como una vez la aplicación estaba completamente desarrollada, ésta ha sido testeada por personas ajenas a su desarrollo para obtener una opinión objetiva de la aplicación, así como posibles errores o mejoras.

Los principales defectos que han encontrado los usuarios que han probado la aplicación han sido:

- **Tiempo de espera:** el tiempo medio de espera es de 3.55 segundos, un tiempo que el usuario puede consentir. Sin embargo, en algún apartado específico el tiempo de espera aumenta hasta los 8 segundos debido a que se tiene que obtener mayor información de Internet y prepararla para ser mostrada.
- **Interfaz gráfica:** tanto la maquetación como la paleta de colores utilizada.

8. Publicación en Google Play.

Una vez la aplicación ha sido desarrollada, el Ayuntamiento de Palafolls ha sido el encargado de publicar la aplicación en Google Play. Destacar que el peso final de la aplicación es de 2.40MB, un tamaño aceptable para los usuarios. El tamaño de las aplicaciones es un tema importante para los usuarios debido al pequeño tamaño de la memoria interna de los dispositivos actuales.

Para distribuir una aplicación Android, el canal para llegar más fácilmente al usuario es la tienda oficial de aplicaciones de Google, más conocida como Google Play.

Para publicar una aplicación, antes hay que realizar unos pasos previos:

- **Revisar el archivo *AndroidManifest.xml*.** Hay que verificar que la aplicación esté bien configurada: que sólo contenga los permisos necesarios, esté bien definida la versión mínima de Android en la cual se podrá ejecutar la aplicación, comprobar el nombre y la versión de la aplicación, etc.
- **Firma digital de la aplicación.** Por suerte, este paso es sencillo gracias al uso de un asistente de Eclipse. Para firmar la aplicación Eclipse utiliza un almacén de claves por defecto. El almacén de claves es donde se guarda el certificado digital. Es necesaria la generación de una clave privada que identifica al desarrollador y que es fundamental para crear la relación de confianza con el usuario de la aplicación. De esta manera, los usuarios están seguros que la aplicación proviene de quien dice ser.

  Este mecanismo de firma digital es más simple dado que el desarrollador es el que crea la clave privada y siempre utilizará la misma. Por tanto, cada vez que se crea una actualización de la aplicación se debe utilizar la misma clave privada.

  Para realizar la firma digital de la aplicación, hay que seguir los siguientes pasos:

  - Hacer clic en el botón secundario sobre el proyecto y seleccionar la opción *Export* (exportar).

- En el menú *Export*, hacer clic en Android para visualizar y elegir *Export Android Application***.** Esto hará que se genere un archivo ".apk" (*Android package)*, que es el instalador de la aplicación.
- El siguiente paso es la creación de un almacén de claves. Elegir la opción *Create new keystore* si es la primera vez que se publica una aplicación, o *Use existing keystore* si se está publicando una actualización. Si es la primera vez (Fig. 8.1), hay que seleccionar una ubicación del ordenador en donde se ubicará el almacén de claves y una contraseña para gestionarlo. Si se está publicando una actualización, seleccionar el almacén de claves e introducir su clave.

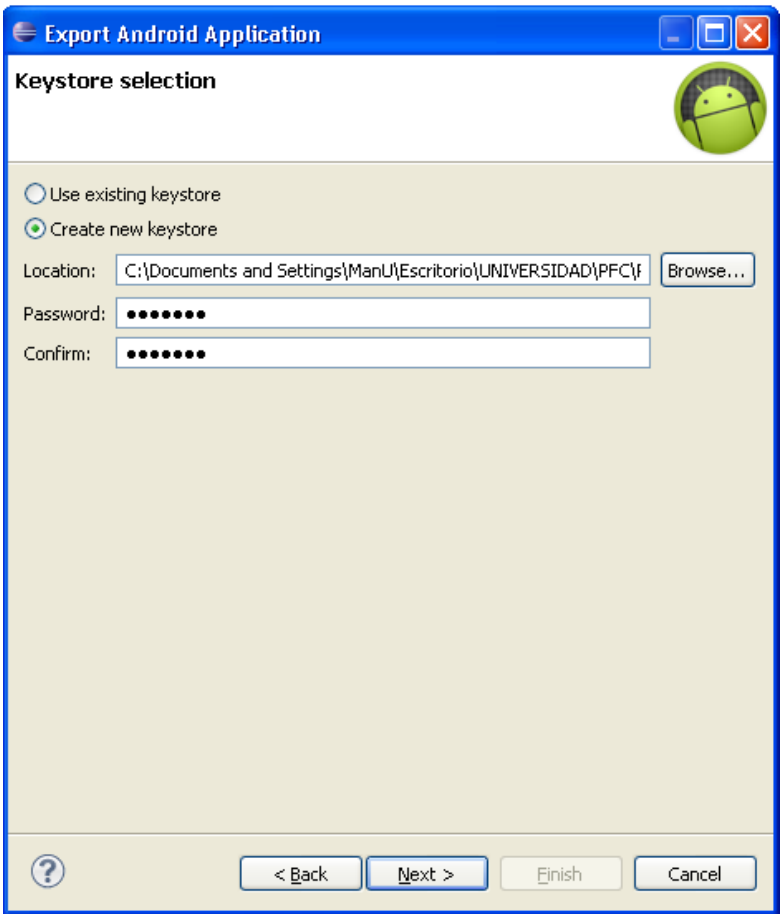

Fig. 8.1. Creación de almacén de claves.

- En la pantalla de *Key Creation* (Fig. 8.2) se introducen los detalles de la clave, incluyendo la información de la empresa u organización. Para

Google Play la validez debe ser como mínimo hasta 22/10/2033. El alias es una cadena que identifica la clave que se está creando.

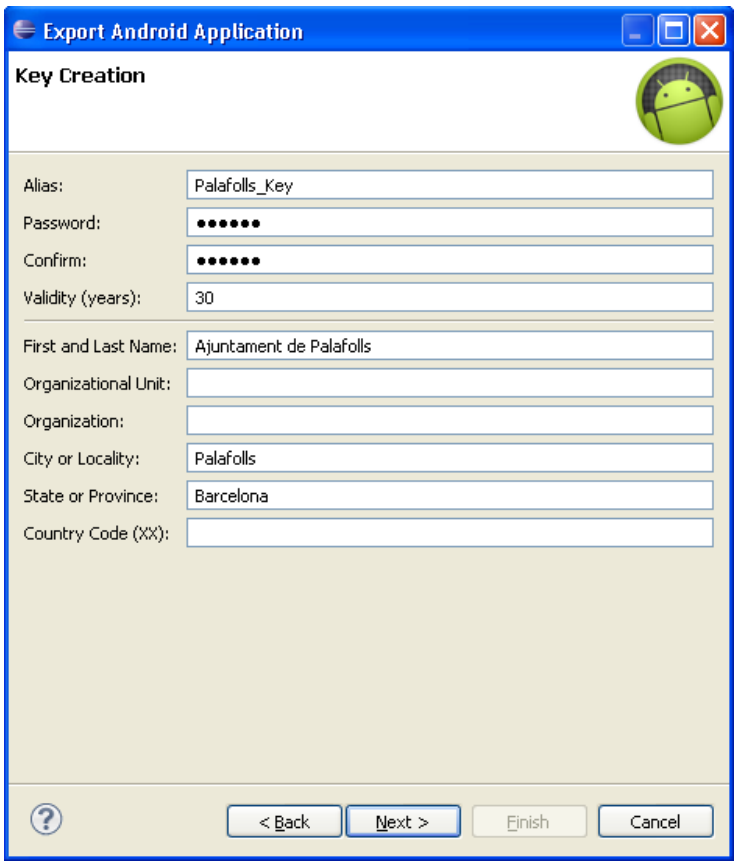

Fig. 8.2. Detalles de la clave.

- Hacer clic en el botón *Next* (siguiente).
- En la siguiente pantalla introducir la ubicación en donde se almacenará el archivo ".apk" de la aplicación.
- Hacer clic en el botón *Finish* (finalizar).
- El archivo ".apk" resultante será el que se distribuya en Google Play.

Una vez se tiene la aplicación lista para su distribución, hay que seguir los siguientes pasos para publicarla en Google Play:

Darse de alta en el servicio Google Play como desarrollador.

- Acceder a la dirección: http://market.android.com/publish utilizando una cuenta de Google (Gmail).

- La primera vez que se acceda al servicio se mostrará el asistente para la creación de una nueva cuenta de desarrollador. Introducir los datos que se soliciten. Pulsar "Siguiente".
- Leer y aceptar el Acuerdo de distribución para desarrolladores
- Para darse de alta como desarrollador de Android y poder publicar aplicaciones en Google Play, se deberá abonar una única cuota de 25$. Para pagar dicha cuota se podrá utilizar el servicio de Google Checkout o con tarjeta. En el segundo caso, pulsar "Siguiente".
- Se mostrará el detalle de la factura con un artículo llamado "Android – *Developer Registration Free for* XXX". Introducir los datos de la tarjeta de crédito para realizar el pago, así como la dirección de facturación dónde llegará la correspondiente factura.
- Si todo es correcto, el asistente mostrará la siguiente ventana, indicando que el pedido se ha enviado a Google Play. Para continuar con el proceso, pulsar "Vuelve al sitio de desarrolladores de Google Play para completar el registro".
- El asistente indicará que el registro ha concluido, con el mensaje "Se ha aprobado tu registro en Google Play. Ahora puedes subir y publicar aplicaciones de software en Google Play". A partir de este momento ya se podrá utilizar la cuenta registrada para publicar aplicaciones.

Publicar la aplicación.

- Acceder con la cuenta de desarrollador creada a la dirección: https://market.android.com/publish/Home.
- Pulsar el enlace "Subir aplicación".
- Se mostrará la ventana de selección de fichero ".apk". Seleccionar el archivo ".apk" de la aplicación previamente firmado digitalmente. Pulsar el botón "Publicar".
- Si es correcto el archivo ".apk" y cumple todos los requisitos, el asistente mostrará el botón "Guardar" y los datos del archivo (nombre de la aplicación, nombre de la versión, código de la versión, permisos necesarios, etc.). Pulsar el botón "Guardar".
- Tras subir el archivo, pulsar el enlace "Activar" para introducir los datos necesarios para publicarlo en Google Play. Desde aquí se podrá activar o

desactivar la publicación de la aplicación, si por ejemplo, se ha detectado algún error y no se desea que pueda descargar hasta haberlo solucionado.

- Introducir todos los datos requeridos en la pestaña "Información de producto" para la nueva aplicación. Los datos necesarios son:
    - Mínimo dos capturas de pantalla de la aplicación.
    - Icono de la aplicación.
    - Si no se desea que la aplicación sea promocionada fuera de Google Play, marcar la casilla "No promocionar mi aplicación salvo en Google Play y en los sitios web o para móviles propiedad de Google".
    - Se podrá elegir varios idiomas para añadir la descripción de las funciones de la aplicación. El inglés es obligatorio. En este punto se solicitará:
        - Título de la aplicación, inferior a 30 caracteres.
        - Descripción detallada (hasta 4.000 caracteres).
        - Cambios recientes. Si es una actualización, se podrá indicar las nuevas mejoras.
        - Si se ha añadido un vídeo promocional, se podrá añadir un texto promocional.
        - Seleccionar el tipo de aplicación que más se ajuste a los ofrecidos en un desplegable.
        - Seleccionar la categoría que más se ajuste de un desplegable.
    - Clasificación de contenido: marcar si la aplicación es para todos los públicos o contiene algún tipo de contenido para mayores.
    - Precio: indicar si la aplicación será gratuita o de pago.
    - Si se ha elegido que la aplicación sea de pago, en la casilla "Precio determinado", introducir el precio que ha de tener la aplicación. Pulsando el botón "Autocompletar" calculará el precio para los diferentes países en los que se desea publicarla.
    - También se indicará el número aproximado de modelos de dispositivos Android que soportarán la aplicación según los filtros indicados en el archivo "AndroidManifest.xml".
    - Por último, introducir los datos de contacto: sitio web, correo electrónico y teléfono.

- Una vez introducidos los datos, pulsar el botón “Guardar” de la parte superior derecha. El asistente comprobará que los datos sean correctos, indicando posibles errores. Si no hay errores, guardará los datos especificados al fichero “.apk” subido.
- Pulsar en el botón “Publicar” para publicar definitivamente la aplicación en Google Play.
- El botón indicará que se está publicando.
- Tras finalizar, mostrará en el apartado “Todos los elementos de Google Play” el estado “Publicada”. Desde éste apartado, se podrá realizar un seguimiento del número de instalaciones que se realicen, posibles errores, comentarios de los usuarios, etc.

9. Valoración económica.

9.1. Costes de recursos humanos.

Como se observa en la Tabla 9.1, el coste total de los recursos humanos es de 2.676,67€.

Concepto	Horas	Precio/hora (€)	Total (€)
Formación (Programador júnior)	80	16	106,67
Análisis y estudio de mercado (Programador júnior)	10	16	160,00
Programación (Programador júnior)	110	16	1.760,00
Redacción de la documentación (Administrativo)	65	10	650,00
TOTAL	**265**		**2.676,67€**

Tabla 9.1. Coste de los recursos humanos.

La amortización de la formación se ha establecido en un período de 3 años realizando 4 proyectos al año. Su coste total es de 1.280€ (80 horas multiplicado por el salario de programador júnior, 16€), por lo que se imputa un coste de 106,67€ tras realizar un proyecto.

Se ha establecido en 16€/hora el salario de un programador júnior y de 10€/hora de un administrativo.

9.2. Amortización de equipos y software.

El coste total de la amortización de las herramientas utilizadas es de 127,41€. Aquellas herramientas cuyo coste es de 0€, significa que su distribución es gratuita. Ver Tabla 9.2.

Concepto	Coste
Ordenador	83,33€
Eclipse Indigo	0€

Java JDK	0€
Android SDK	0€
AVT (*Android Development Tools*)	0€
Microsoft Office Hogar 2007	10,83€
EasyPHP 12.1	0€
phpMyAdmin	0€
FileZilla Client	0€
Samsung Galaxy S Plus	33,25€
TOTAL	**127,41€**

Tabla 9.2. Coste de la amortización de las herramientas utilizadas.

La amortización del ordenador se ha establecido en un período de 3 realizando 4 proyectos anuales. Como su coste inicial fue de 1.000€, se imputa un coste de 83,33€ tras realizar un proyecto (1.000€ dividido entre 12 proyectos en total, equivale a 83,33€ por proyecto). Asimismo, se incluye el sistema operativo Windows XP.

En el caso del software Microsoft Office Hogar 2007, con un coste inicial de 129,99€, siguiendo los mismos pasos que para la amortización del ordenador su coste es de 10,83€.

Para el dispositivo Samsung Galaxy S Plus, con un coste inicial de 399€ y siguiendo el razonamiento del coste del ordenador, se imputa un coste de 33,25€.

9.3. Gastos indirectos.

Como gastos indirectos se tiene en cuenta un porcentaje del 15% sobre los costes anteriores.

9.4. Coste total del proyecto.

Como se observa en la Tabla 9.3, el coste final del desarrollo del proyecto es de 3.224,69€.

Concepto	Coste
Recursos humanos	2.676,67€
Amortizaciones	127,41€
Subtotal	2.804,08€
Gastos indirectos (15%)	420,61€
TOTAL	**3.224,69€**

Tabla 9.3. Coste total del proyecto.

10. Conclusiones.

Para desarrollar aplicaciones para dispositivos móviles es recomendable tener un mínimo de conocimiento y experiencia, ya que si no cuesta empezar a programar. La motivación personal de trabajar en el sector de desarrollo de aplicaciones y el autodidactismo sobre la materia han facilitado la consecución del proyecto.

Para realizar una buena gestión del proyecto, desde el inicio del mismo se debe mantener una buena y constante comunicación con el cliente. Una de las trabas que ha surgido, ha sido la falta de instrucciones por parte del cliente, lo que ha provocado que el desarrollador trabaje sin seguir unas indicaciones específicas. Al tratarse de una aplicación móvil, la interfaz gráfica es una parte fundamental del proyecto. En este caso, el cliente sólo ha proporcionado el diseño de la pantalla inicial de la aplicación, por lo que se ha tenido que generar la interfaz gráfica con el paso del tiempo sufriendo diversos cambios.

10.1. Posibles mejoras.

Pese a que la interfaz gráfica se ha ido modificando en todo momento para mejorar su apariencia, es uno de los aspectos a mejorar en un futuro. Se tendría que haber realizado un diseño inicial de la aplicación en el cual basarse y de esta manera el resultado final hubiese sido de mayor calidad.

Un aspecto sobre el cual reflexionar, es el acceso a la información. Ocho de los nueve menús principales de la aplicación, han de acceder a Internet para ofrecer información al usuario. Esto implica tiempo de espera, en ocasiones más prolongado de lo habitual y no todos los clientes están dispuestos a esperar tanto. Una de las posibles mejoras sería disminuir el tiempo de espera o bien almacenar la información de manera local en el dispositivo y por tanto el tiempo sería nulo. No toda la información se podría almacenar localmente, pero los datos de varios menús que acceden a Internet podrían estarlo ya que no es información susceptible a cambios constantes.

Junto con la propuesta anterior, estaría relacionada la de mejorar la estructura de la base de datos que nutre a la página web del Ayuntamiento de Palafolls, al mismo tiempo que el formato de la información guardada. De esta manera, se ahorraría la duplicidad de

información almacenada. Actualmente, si se desea añadir, eliminar o modificar información de la página web que aparezca en la aplicación, se han de realizar sentencias que afecten a ambas base de datos.

Otro de los conceptos a mejorar sería que toda la información siguiese siempre el mismo formato o estructura. Por ejemplo: que todos los locales de restauración tuviesen un texto descriptivo no superior a 140 caracteres, ya que actualmente algunos locales tienen un texto descriptivo y otros no. Esto se debe a que la información estaba pensada para una página web y no para una aplicación móvil.

En la reunión inicial con el cliente, se planteó el desarrollar la aplicación tanto para Android como para iOS. Se descartó por los motivos mencionados en el apartado 3.4. No obstante, una de las mejoras sería el desarrollo de la aplicación para iOS. Para ello se podría desarrollar la aplicación con el lenguaje nativo o bien utilizar la herramienta J2ObjC de Google. Esta herramienta que ha sido analizada en las fases iniciales del proyecto, tiene como función convertir código Java a Objective-C. No obstante se descartó su uso debido a que acababa de ser publicada y estaba en fase de pruebas. Actualmente sigue en fase de pruebas.

Otra posible mejora sería la de incorporar la opción de poder escoger el idioma en el cual se muestra la información. Es un aspecto que no ha sido planteado a lo largo del desarrollo, pero de esta manera la aplicación llegaría a más usuarios. Para aplicar esta característica, se tendrían que seguir los siguientes pasos:

- Crear el directorio *Res/values*-(código facilitado por la normativa ISO 639-1 del idioma deseado), e introducir en él todos los recursos de texto utilizados, en su correspondiente idioma.
- Adaptar la base de datos para almacenar la información a mostrar en los diversos idiomas.
- Crear un menú de configuración en la aplicación, para que el usuario pueda escoger en qué idioma desea ver la información.

11. Referencias.

[1] Scott McCracken, *Android. Curso de desarrollo de aplicaciones.* Inforbooks.

[2] http://blog.flurry.com/bid/85911/App-Developers-Signal-Apple-Allegiance-Ahead-of-WWDC-and-Google-I-O Peter Farago, App Developers Signal Apple Allegiance Ahead of WWDC and Google I/O. Thu, Jun 07, 2012.

[3] http://developer.android.com/intl/es/about/dashboards/index.html, Platform Versions.

[4] http://www.emol.com/noticias/tecnologia/2012/09/12/560134/mark-zuckerberg-usar-html5-fue-el-mayor-error-que-hayamos-cometido.html Emol. Miércoles, 12 de septiembre de 2012.

[5] http://www.oracle.com/technetwork/java/javase/downloads, Java JDK.

[6] http://www.eclipse.org/downloads, Eclipse.

[7] http://developer.android.com/sdk/index.html, Android SDK.

[8] http://dl-ssl-google-com/android/eclipse, Android ADT.

[9] http://developer.yahoo.com/weather/, API Yahoo! Weather.