Jugando con Android: Acciones

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes

realitzat per
Juan Ávila López del Castillo

i dirigit per
Ramon Baldrich

Escola d'Enginyeria
Sabadell, *Setembre* de *2012*

El sotasignat, ***Ramon Baldrich***
Professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció per

Juan Ávila López del Castillo

I per a que consti firma la present.
Sabadell, ***Setembre*** de ***2012***

Signat: ***Ramon Baldrich***

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: Jugando con Android: Acciones	
Autor: Juan Ávila López del Castillo	**Data:** Setembre 2012
Tutor: Ramon Baldrich	
Titulació: Enginyeria Tècnica en Informàtica de Sistemes.	

Paraules clau

- **Català:** Android, Java, càmera, codi QR.
- **Castellà:** Android, Java, cámara, código QR.
- **Anglès:** Android, Java, camera, QR-Code.

Resum del projecte

- **Català:**

  El projecte global estarà enfocat al aprenentatge del desenvolupament d'aplicacions per al sistema operatiu Android. Per això, es pretén realitzar una aplicació a traves de la qual es puguin adquirir coneixements dels diversos components que formen l'estructura de la plataforma. Amb això, aquest projecte pretén també ser una petita guia per aquelles persones que, com nosaltres, pretenen iniciar-se en el desenvolupament d'aplicacions per Android.

  L'aplicació desenvolupada serà un joc del gènere de lluita, per un o dos jugadors en el mateix terminal.

- **Castellà:**

  El proyecto global está enfocado al aprendizaje del desarrollo de aplicaciones para el sistema operativo Android. Para ello, se pretende realizar una aplicación a través de la cual se puedan adquirir conocimientos de los diversos componentes que forman la estructura de la plataforma. Con ello, este proyecto pretende también ser una pequeña guía para aquellas personas que, cómo nosotros, pretenden iniciarse en el desarrollo de aplicaciones para Android.

  La aplicación desarrollada será un juego del género lucha, para uno o dos jugadores en el mismo terminal.

- **Anglès:**

  The whole project is focused to learning to develop applications for Android operating system. To this end, is intended to make an application through which

we can acquire knowledge of the differents components that form the platform structure. Thus, this project is also intended as a brief guide for those who, like us, are intended to start to develop applications for Android.

The developed application will be a fighting game genre, for one or two players on the same terminal.

Índice

Índice de ilustraciones

1. Introducción

Este proyecto se realiza entre dos personas, en adelante, nos referiremos a la totalidad del proyecto como proyecto global. De esta manera diferenciaremos las veces que tratamos la parte expuesta en esta memoria de las veces que hablamos de la totalidad del proyecto.

1.1 Descripción

BattleCam es un juego diseñado para ser utilizado bajo el sistema operativo Android. El juego se incluye en el género lucha en dos dimensiones. Las batallas serán por turnos, en tercera persona y se llevan a cabo en escenarios reales, que son capturados mediante la cámara de nuestro dispositivo móvil. El juego será para uno o dos jugadores, que jugarán en el mismo terminal. Los diferentes personajes podrán ser seleccionados a través de nuestra cámara, utilizando códigos QR, o de forma manual. El objetivo principal del juego será vencer a nuestro rival antes de que nos derrote a nosotros.

El proyecto global está pensado de forma que sirva de referencia para que, cualquier persona que quiera realizar un proyecto similar, encuentre detalladas las herramientas necesarias para llevarlo a cabo.

Este proyecto está enfocado en la parte que no interactúa con el usuario, es decir, todas aquellas acciones que se realizan durante el juego, como por ejemplo, cálculos de parámetros o el sistema de lectura de códigos QR.

1.2 Motivación personal

En el futuro próximo, una vez terminados mis estudios, me gustaría poder trabajar de desarrollador de aplicaciones.

Con el día a día nos damos cuenta que, cada vez más, los teléfonos móviles y más concretamente los denominados *smartphones*, están cobrando mucho protagonismo en la vida de la gente.

Por estos motivos, considero básico aprender a desarrollar aplicaciones para los sistemas operativos de las plataformas móviles y, eso fue lo que me impulsó a elegir un proyecto relacionado con el tema.

Por otro lado, desde bien pequeño me apasionan los videojuegos, de manera que aunando todo lo anterior con mi pasión por los videojuegos, surgió la posibilidad de realizar este proyecto.

1.3 Estado del arte

El desarrollo de aplicaciones para dispositivos móviles se han disparado en los últimos años, gracias en parte, al gran protagonismo que están teniendo los videojuegos para los mismos.

Al existir tan gran variedad de videojuegos en el mercado de aplicaciones móviles, es difícil conocer el estado del arte actual. Se han buscado juegos similares dentro de las tiendas de aplicaciones más conocidas y únicamente hemos encontrado juegos del género lucha pero con trazos más convencionales. Por ello, suponemos que el proyecto global no hará grandes aportaciones al conocimiento que ya hay sobre la materia, pero sí que intentaremos explicar y documentar mejor algunos apartados.

1.4 Objetivos

El objetivo principal del proyecto global es aprender a desarrollar aplicaciones para el sistema operativo Android. Con este fin, el proyecto global se basará en el desarrollo de un juego completamente funcional, cuyos objetivos son los siguientes:

O1. Creación de códigos QR y lectura de éstos a través de la cámara del dispositivo, obteniendo así los datos leídos.
O2. Implementación de un sistema de reglas y juego.
O3. Poder empezar y acabar una partida.
O4. Creación de gráficos y representación de éstos en pantalla.
O5. Análisis del flujo de trabajo de navegación.
O6. Incluir efectos de sonido.

En esta memoria se tratarán los puntos anteriores que se encuentran subrayados.

1.5 Definiciones, acrónimos y abreviaciones

Definición, acrónimos y abreviaciones de términos utilizados en esta memoria.

- Actividad (*Activity*)[1]: Dentro del contexto de programación para Android, una actividad es cada pantalla de la aplicación que interactúa con el usuario.
- Android[2]: Sistema operativo basado en Linux, propiedad de *Google*, enfocado para su uso en dispositivos móviles.
- Android SDK[3]: Kit de desarrollo de software (*Android Software Development Kit*). Conjunto de herramientas necesarias para poder desarrollar aplicaciones Android.
- Apache Ant[4]: Herramienta de software usada en programación para procesos de automatización de compilación.
- Betatester: Usuario de programas cuyo fin es encontrar errores en software que está pendiente de terminar su fase de desarrollo.
- Código QR[5]: El código de respuesta rápida (*Quick Response Code*) se usa para almacenar información en un código de barras bidimensional.
- Java[6]: Lenguaje de programación orientada a objetos desarrollado por Sun Microsystems.
- Java JRE y JDK: Herramientas de desarrollo para Java.
- *Smartphone*[7]: Teléfono inteligente. Llamado así por su capacidad de usarse como un computador de bolsillo.

1.6 Estructura de la memoria

En este apartado está detallada la estructura del resto de la memoria.

- Estudio de viabilidad: Determinará la viabilidad del proyecto en función de los objetivos marcados. Estudio actual del sector, especificaciones y planificación del proyecto.
- Descripción técnica: Explicación de las herramientas utilizadas para el desarrollo del proyecto.
- Análisis: Descripción del funcionamiento del proyecto global.
- Diseño: Explicación del diseño de la aplicación y su funcionamiento.
- Pruebas: Batería de pruebas realizadas para comprobar el correcto funcionamiento de la aplicación.
- Conclusiones: Valoración personal sobre el desarrollo del proyecto, así como de los problemas que han surgido durante el desarrollo del mismo, posibles líneas de ampliación y comparativa del estudio previo con el realizado.
- Referencias electrónicas: Especificación de todas las fuentes de información utilizadas en el proyecto.

2. Estudio de viabilidad

En este capítulo, se estudiará la viabilidad del proyecto global en el mercado actual, así como sus requisitos, costes, riesgos y posibles alternativas.

2.1 Estudio del mercado

Al disponer de varios terminales con sistema operativo Android, la elección de sistema operativo para el cual desarrollar nuestra aplicación fue sencilla.

A continuación, se tuvo que tener en cuenta la fragmentación que existe en el mercado de dispositivos Android. A fecha de junio de 2012, la fragmentación es la siguiente:

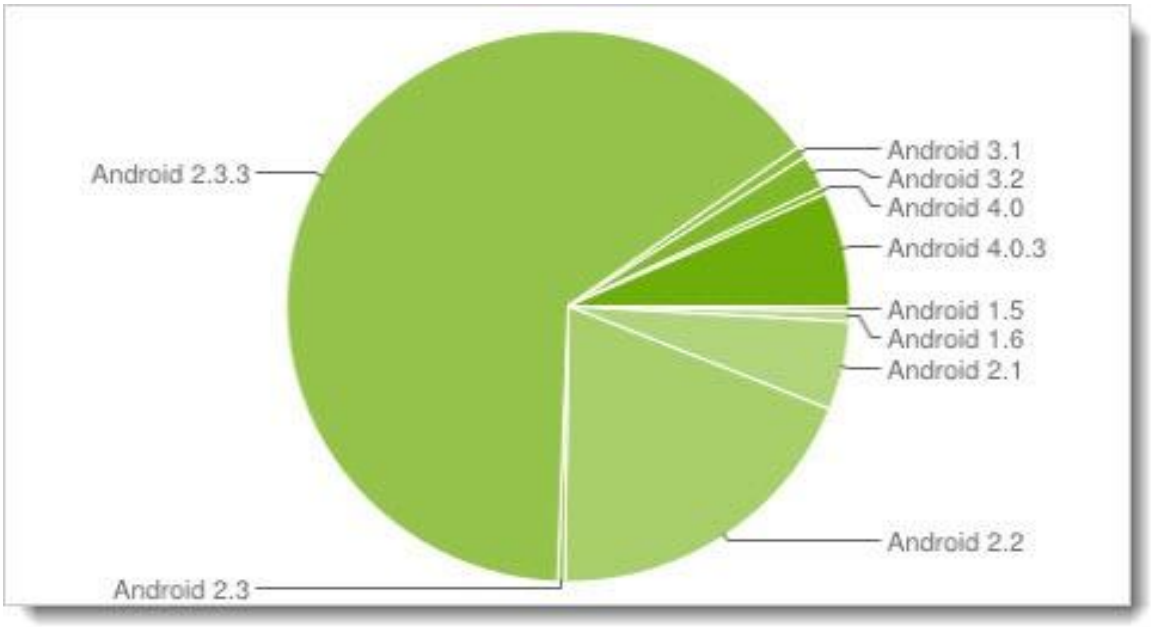

Figura 1: Fragmentación de Android en Junio 2012[8]

Teniendo en cuenta esta fragmentación, se decidió realizar nuestra aplicación para la versión 2.3.3, lo que nos permitirá gracias a la retrocompatibilidad que tiene Android, que nuestro juego llegue a ser completamente funcional en tres cuartas partes de los dispositivos que utilizan Android.

Se ha comprobado dentro del mercado de aplicaciones de Android la existencia de productos del mismo género, pero lo único que se ha encontrado son juegos de lucha convencionales. Estos juegos destacan por su calidad gráfica y por estar pensados para sacar el máximo rendimiento de las nuevas tecnologías.

En el proyecto global se ideó todo para crear una sensación de juego retro, utilizando gráficos en dos dimensiones, algo más típico de los juegos de hace dos décadas. A su vez,

la utilización de la cámara para la selección de personajes y también como escenario durante la partida, así como el tipo de combate (por turnos pero sin pausas), aportan unos toques de innovación.

2.2 Requisitos del sistema

En este apartado se detallarán los requisitos que tiene que cumplir el proyecto global. Se dividirán en requisitos funcionales (qué debe hacer) y no funcionales (cómo lo debe hacer). Dentro del apartado de requisitos funcionales, se clasificarán en: principales, opcionales y extras.

Los requisitos funcionales principales subrayados son los que se tratarán en esta memoria.

2.2.1 Requisitos funcionales

- Principales

RF 1. Aprendizaje de los módulos de la arquitectura de Android.

RF 2. Familiarización con el entorno de desarrollo de Android.

RF 3. Reconocimiento de las cartas a través de la cámara.

RF 4. Mostrar los gráficos por pantalla.

RF 5. Implementación del sistema de reglas.

RF 6. Poder empezar y acabar una partida.

RF 7. Incluir efectos de sonido.

- Opcionales

RF 8. Interconexión entre dos dispositivos.

RF 9. Selector de idiomas.

RF 10. Implementación con redes sociales.

- Extras

RF 11. Adaptación a otras plataformas.

RF 12. Añadir más estadísticas a los personajes de la plantilla.

RF 13. Establecer más reglas en el sistema de juego.

RF 14. Geolocalización (búsqueda de oponentes).

RF 15. Llegar a comercializarlo.

2.2.2 Requisitos no funcionales

RNF 1. Sencillo de jugar

RNF 2. Entretenido

RNF 3. Mayor compatibilidad

2.3 Catalogación de los requisitos

Aquí se detallan los requisitos del proyecto global en función de los objetivos marcados en el punto "1.4 Objetivos".

	RF 1	RF 2	RF 3	RF 4	RF 5	RF 6	RF 7	RF 8	RF 9	RF 10	RF 11	RF 12	RF 13	RF 14	RF 15	RNF 1	RNF 2	RNF 3
O1	X	X	X					X			X				X			X
O2	X	X	X		X	X		X			X	X	X		X	X	X	X
O3	X			X	X	X		X			X	X	X		X	X	X	X
O4	X	X	X	X				X			X	X			X		X	X
O5						X		X	X	X	X	X		X	X	X	X	X
O6	X	X					X	X			X				X			X

2.4 Viabilidad económica

Los recursos humanos utilizados en el desarrollo del proyecto global son: supervisor, analista, programador, diseñador y betatester. Éstos se encargarán de realizar todas las tareas necesarias y asumirán el coste a cargo de ellos mismos.

Los recursos de software y hardware utilizados constan de: un ordenador con el entorno de trabajo necesario para poder desarrollar en Android y un dispositivo móvil con el mismo sistema operativo, para probar las diferentes fases de la aplicación así como su versión final.

Tanto el sistema operativo utilizado como el resto de aplicaciones que configuran el software utilizado en el desarrollo, son totalmente gratuitos, por lo que, no supondrá ningún gasto adicional.

Si el objetivo de llegar a comercializar el producto final se encontrara entre los requisitos principales, se debería tener en cuenta que el incluir la aplicación en el mercado de aplicaciones de Android, tiene un coste de 25$, tal y como se indica en la página web[9].

Nombre	Descripción	Responsabilidad
Ramon Baldrich	Supervisor	• Participación en la definición, planificación y seguimiento del proyecto.
Javier Marta	Analista - Programador	• Asignar tareas. • Encargado de definir todo lo relacionado con el diseño del videojuego. • Implementación del diseño. • Introducir nuevos contenidos. • Desarrollo.
Juan Ávila	Analista - Programador	• Asignar tareas. • Encargado de definir todo lo relacionado con el sistema de reglas del videojuego. • Introducir nuevos contenidos. • Desarrollo.
Diseñador	Diseñador	• Encargado de diseñar botones, diálogos y personajes. • Creación de efectos de sonido.
Juan Ávila – Javier Marta	Betatester	• Realiza pruebas de las diferentes versiones del videojuego antes de su versión final y notifica los posibles fallos a los analistas – programadores.

2.5 Viabilidad técnica

En este apartado se detallarán los requisitos técnicos que deben cumplir los recursos de software y de hardware para el desarrollo del proyecto global, asimismo se detallarán las especificaciones técnicas mínimas que han de cumplir los dispositivos móviles que utilicen la aplicación final. En cualquier caso, se justificará la elección de software y hardware.

Recursos software

- Microsoft Windows 7
- Eclipse
- Java JRE y JDK
- Android SDK
- Microsoft Office

Se dispone de licencia gratuita de Microsoft Windows 7 y de Microsoft Office. El resto de recursos tienen licencia de software libre.

Para desarrollar aplicaciones para el sistema operativo Android existen varios programas, como pueden ser, Eclipse o Netbeans. La existencia de una mayor documentación para Eclipse ha sido determinante para decantarse por su uso.

Recursos hardware de PC

- Procesador Intel Core Duo @1,66GHz
- 1GB de memoria RAM
- 1GB libre en el disco duro

El software de desarrollo no requiere unas especificaciones mínimas para su correcto funcionamiento, pero se ha comprobado que con especificaciones inferiores, el tiempo que requiere para realizar operaciones aumenta considerablemente, haciendo más lento el desarrollo.

Recursos hardware de dispositivo móvil

- Procesador 700MHz
- 512MB de memoria RAM
- 20MB de memoria interna libre
- Pantalla de 3,5 pulgadas

El desarrollo de la aplicación, como se explicó en el apartado "2.1 Estudio de mercado", se ha realizado para la versión 2.3.3 y posteriores de Android. Esto hace que el hardware del dispositivo móvil deba ser suficiente como para poder utilizar esa versión con fluidez y, por lo tanto, si que existan unos requisitos mínimos.

Descripción física

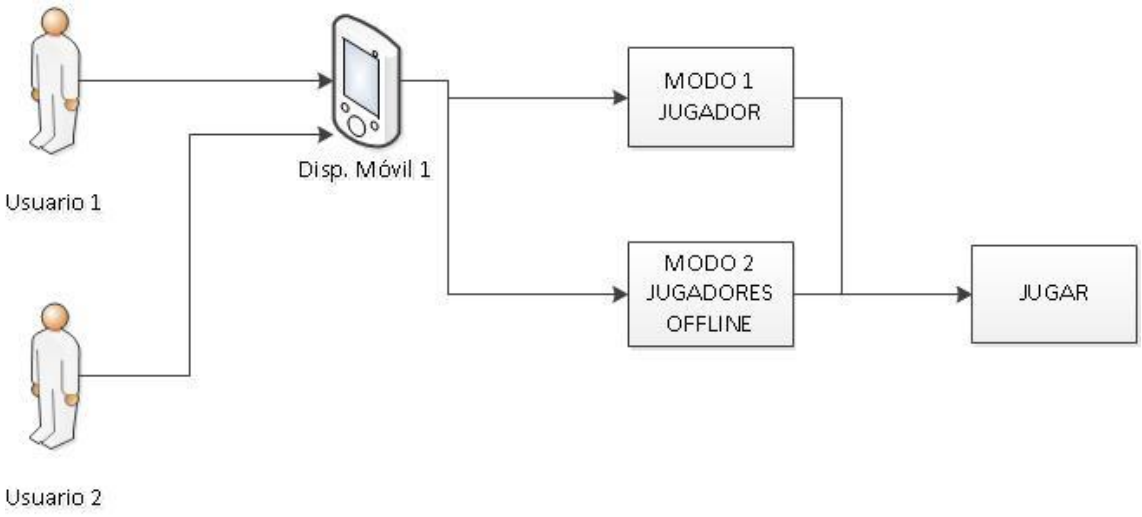

Figura 2: Descripción física de la aplicación

2.6 Viabilidad temporal y planificación

En este apartado se mostrará la previsión de la distribución de las tareas del proyecto global y la duración de las mismas.

Calendario previsto

La previsión temporal del proyecto global se realizó para que se extienda desde diciembre de 2011 a junio de 2012.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Proyecto**	**26,75 días**	**vie 02/12/11**	**mié 06/06/12**		
2	**Planificacion**	**27 días**	**vie 02/12/11**	**jue 12/01/12**		
3	Estudio de viabilidad	2 días	vie 02/12/11	lun 05/12/11		Javi;Juan
4	Estudio de requisitos	2 días	mié 07/12/11	vie 09/12/11	3	Javi;Juan
5	Reparto de tareas (diagrama Gantt)	3 días	lun 12/12/11	mié 14/12/11	4	Javi;Juan
6	Diseño del sistema de juego	5 días	jue 15/12/11	jue 12/01/12	5	Javi;Juan
7	**Desarrollo (1ª iteracion)**	**21 días**	**vie 13/01/12**	**jue 16/02/12**	**6**	
8	**Desarrollo del juego**	**11 días**	**vie 13/01/12**	**mié 01/02/12**		
9	Creacion de clases y metodos	5 días	vie 13/01/12	jue 19/01/12		Juan;Javi;Supervisor
10	Codificacion de algoritmos	6 días	vie 20/01/12	mié 01/02/12	9	Juan
11	Mostrar graficos en pantalla	5 días	vie 20/01/12	lun 30/01/12		Javi
12	Diseño de controles	10 días	mar 31/01/12	mié 15/02/12	11	Javi
13	Lectura de codigos QR	10 días	jue 02/02/12	jue 16/02/12	10	Juan
14	**Desarrollo (2ª iteracion)**	**17 días**	**vie 17/02/12**	**lun 12/03/12**	**12;13**	
15	Revision de requisitos	2 días	vie 17/02/12	lun 20/02/12		Javi;Juan;Supervisor
16	Programacion de controles	6 días	mar 21/02/12	mar 28/02/12	15	Juan
17	Mostrar controles en pantalla	3 días	mié 29/02/12	vie 02/03/12	16	Javi
18	Diseño de personajes	12 días	vie 17/02/12	lun 05/03/12		Diseñador
19	Mostrar personajes en pantalla	5 días	mar 06/03/12	lun 12/03/12	18;17	Javi
20	**Desarrollo (3ª iteracion)**	**24 días**	**mar 13/03/12**	**vie 13/04/12**	**19**	
21	Revision de requisitos	2 días	mar 13/03/12	mié 14/03/12		Javi;Juan;Supervisor
22	Programacion de acciones	7 días	jue 15/03/12	vie 23/03/12	21	Juan
23	Utilización de la camara	5 días	lun 26/03/12	vie 30/03/12	22	Javi;Juan
24	Version beta	4 días	mar 10/04/12	vie 13/04/12	23	Juan
25	**Desarrollo (4ª iteracion)**	**17 días**	**lun 16/04/12**	**mar 08/05/12**	**24**	
26	Revision de requisitos	2 días	lun 16/04/12	mar 17/04/12		Javi;Juan;Supervisor
27	Bateria de pruebas	3 días	mié 18/04/12	vie 20/04/12	26	Javi;Juan
28	Prueba por betatesters	6 días	lun 23/04/12	lun 30/04/12	27	Betatester
29	Revision	3 días	mar 01/05/12	jue 03/05/12	28	Javi;Juan
30	Finalizar desarrollo	3 días	vie 04/05/12	mar 08/05/12	29	Javi;Juan
31	Desarrollo memoria	10 días	mié 09/05/12	mar 22/05/12	30	Javi;Juan
32	Presentacion del proyecto	1 día	lun 04/06/12	lun 04/06/12		Javi;Juan
33	Final del proyecto	0 días	mié 06/06/12	mié 06/06/12		Javi;Juan

Figura 3: Planificación de tareas

Calendario temporal

En el calendario temporal vemos el desarrollo de las tareas en el tiempo. En la Figura 4 se han diferenciado los diferentes recursos con tramas y colores distintos. Con color azul se resalta las actividades realizadas exclusivamente por mí y detalladas en esta memoria. De color rojo y trama oblicua se puede observar el trabajo realizado por el otro programador. De color verde y trama vertical diferenciamos las tareas que se han realizado de forma conjunta. El diseñador se representa con una trama horizontal de color lila.

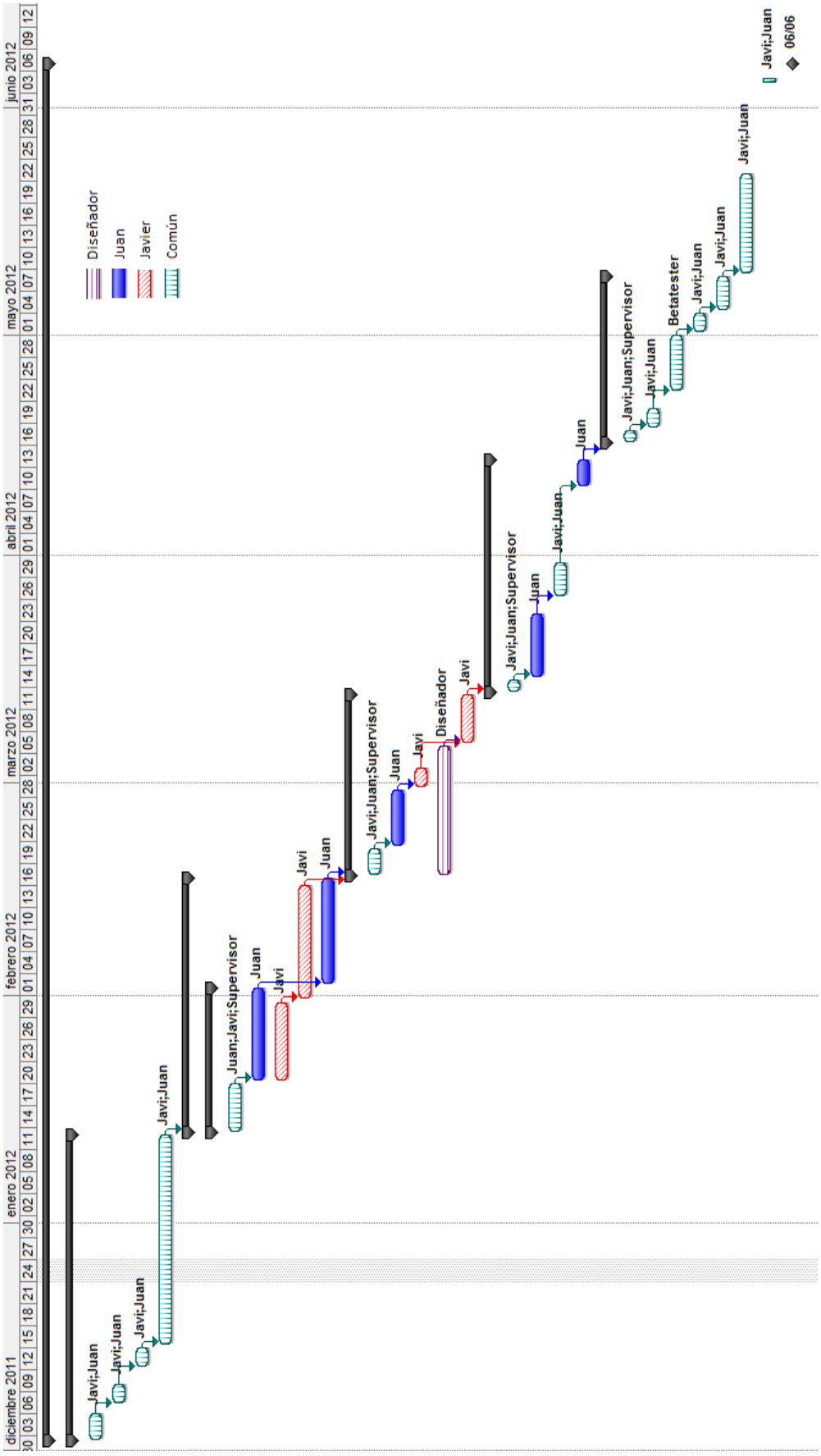

Figura 4: Diagrama temporal (de Gantt)

Los recursos humanos que aparecen en el diagrama son los siguientes:

- Supervisor: Encargado de la supervisión del proyecto, revisando que las tareas se vayan realizando dentro de los plazos establecidos.
- Juan y Javier: Desarrolladores del proyecto global.
- Diseñador: Encargado de realizar las tareas de diseño grafico.
- Betatester: Usuarios no finales encargados de probar las versiones beta de la aplicación en busca de posibles errores.

2.7 Estructura desplegada del desarrollo

Fases	Descripción
Iniciación	Incluye las actividades que definen el proyecto, la asignación y matriculación.
Planificación	Estudio de viabilidad y plan de proyecto.
Análisis	Análisis de los requisitos funcionales y no funcionales.
Diseño	Diseño del videojuego.
Desarrollo	Fases de desarrollo del videojuego.
Test y pruebas	Fase de pruebas del videojuego.
Generación de documentos	Realizar la documentación del proyecto. Incluye manuales de usuario y la memoria del proyecto.
Cierre del proyecto	Se firma la aceptación y se cierra el proyecto.
Defensa del proyecto	Defensa del proyecto frente al jurado.

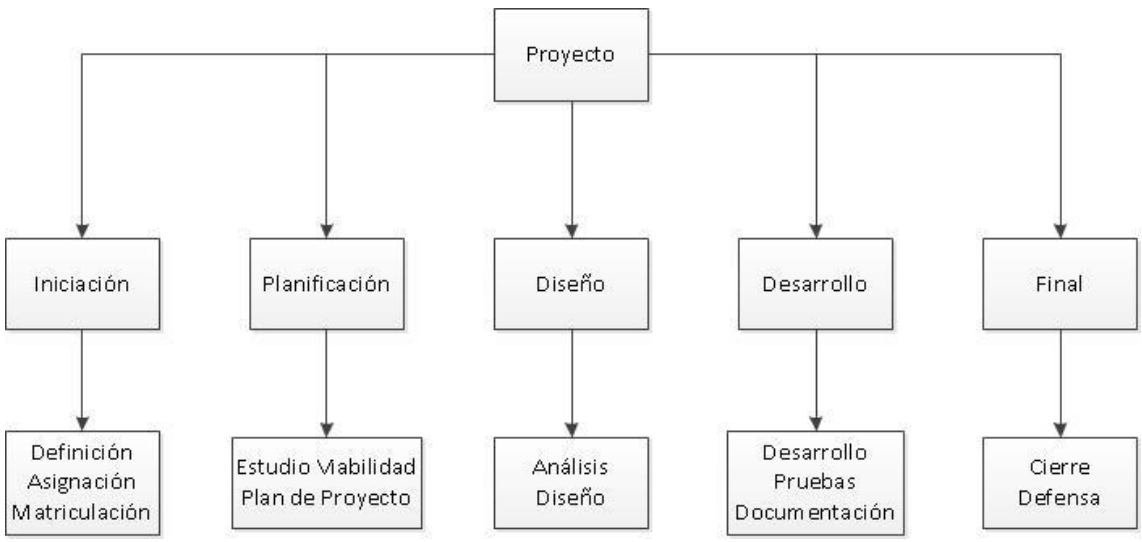

Figura 5: Fases de desarrollo

2.8 Hitos del proyecto

En la siguiente tabla se observa los principales hitos del proyecto y las fechas en las que se deberían haber realizado.

Nombre	Descripción	Fecha
Inicio	Matriculación	14/10/2011
Estudio de Viabilidad	Aprobación	02/12/2011
Plan de Proyecto	Aprobación	07/12/2011
Análisis	Aprobación	24/12/2011
Diseño	Aprobación	15/12/2011
Cierre	Aceptación	22/05/2012
Defensa	Evaluación	06/06/2012

2.9 Evaluación de riesgos

Al ser la primera vez que se realiza un proyecto de esta envergadura, la cantidad de riesgos puede ser elevada. En esta sección se realizará una evaluación de los riesgos del proyecto, el impacto que pueden tener sobre éste y las posibles soluciones a los mismos.

Lista de riesgos

1. Planificación temporal optimista: No se acaba en la fecha prevista.
2. Falta de alguna tarea necesaria: No se cumplen los requisitos del proyecto global.
3. Incumplimiento de leyes: Retraso en el desarrollo del proyecto.
4. Cambio de requisitos: Retraso en el desarrollo y en el resultado final.
5. Herramientas de desarrollo inadecuadas: Retraso en el desarrollo y menos calidad.
6. Abandono del proyecto antes de su finalización: Pérdidas económicas.

Catalogación de riesgos y posibles soluciones

Riesgo	Probabilidad	Impacto	Solución
1	Alta	Crítico	Aplazar o quitar alguna funcionalidad.
2	Alta	Crítico	Modificar la planificación.
3	Media	Crítico	Modificar la planificación y rediseñar todo el contenido que incumpla leyes.
4	Alta	Crítico	Replantear la planificación. Asumir perdidas por retrasos.
5	Baja	Crítico	Mejorar la formación. Previsión de alternativas.
6	Media	Catastrófico	No tiene solución.

2.10 Alternativas y selección de soluciones

Se detallarán las posibles alternativas al desarrollo de alguna de las tareas del proyecto global.

Alternativa 1

ZXing ("*Zebra crossing*")[10], es una librería de código abierto que permite el procesado de imágenes de códigos de barras de diversos formatos. Está realizada en Java y su principal propósito es ser usada en móviles para el escaneo y decodificado de códigos de barras.

Alternativa 2

El rol del diseñador se realiza por parte de otro de los recursos humanos, realizando todas las tareas asignadas a éste.

Solución propuesta

La primera alternativa permite que se recorten horas de desarrollo en el proyecto y al ser librería de código abierto, no supone gastos adicionales.

La segunda alternativa supondría invertir tiempo en aprendizaje de herramientas de diseño, lo que supondría una demora en el desarrollo de otras tareas y probablemente en la entrega del proyecto.

Por consiguiente, se aplicará el uso de la primera alternativa pero se descarta completamente la segunda, ya que, el gasto que provoca es menor que las pérdidas de tiempo de desarrollo por parte de otro recurso y, menguaría la calidad del producto resultante.

2.11 Planificación de costes

En este apartado se realizará una aproximación al coste total del proyecto, desglosando costes por recursos.

Estimación del coste de recursos humanos

Recursos humanos	Valoración	Horas	Coste (en €)
Supervisor	50€/hora	24	1.200€
Juan Ávila	15€/hora	648	9.720€
Javier Marta	15€/hora	608	9.120€
BetaTester	10€/hora	48	480€
Diseñador	15€/hora	96	1.440€
TOTAL			**21.960€**

En esta tabla podemos ver detallados, en función de las horas empleadas por cada uno de los recursos en el proyecto global, los costes de los mismos expresados en euros.

Estimación de coste de recursos de software y hardware

Recursos	Unidades	Precio (en €)/ Unidad	Coste (en €)
PC Programador	2	300€	600€
Dispositivo móvil	2	250€	500€
TOTAL			**1.100€**

En esta tabla se detalla el coste de los recursos de hardware. Al utilizar software libre, los recursos de software no tienen coste.

Estimación de costes indirectos y otros costes

En el caso de querer comercializar el producto, el coste para poder distribuirlo en la tienda de aplicaciones de Android es de 25$, que deberíamos sumar al coste del proyecto global.

No se contemplan otros costes.

Resumen y análisis de estimación de costes

Coste total del desarrollo = 23.060€

El coste del proyecto es elevado, pero como ya se explicó en el apartado "2.4 Viabilidad económica", los recursos humanos asumirán el coste a cargo de ellos mismos.

2.12 Valoración y conclusión

Una vez se ha finalizado el estudio de viabilidad, tras contemplar las ventajas y desventajas que tiene la realización del proyecto global, se ha decidido que el mismo es viable y que puede pasar a su fase de desarrollo.

3. Marco tecnológico

Con el fin de obtener un mejor entendimiento del proyecto, se explicarán las herramientas utilizadas, así como alguno de sus componentes.

3.1 Java

Lenguaje de programación orientada a objetos, desarrollado por *Sun Microsystem*s en 1995. Para desarrollar es necesario utilizar sus herramientas JRE y JDK. El primero proporciona bibliotecas básicas y la máquina virtual de Java. El JDK proporciona el kit de desarrollo de aplicaciones para Java.

Java es necesario para crear cualquier aplicación para Android.

3.2 Android

Sistema operativo basado en Linux, propiedad de *Google*, enfocado para su uso en dispositivos móviles.

Arquitectura Android

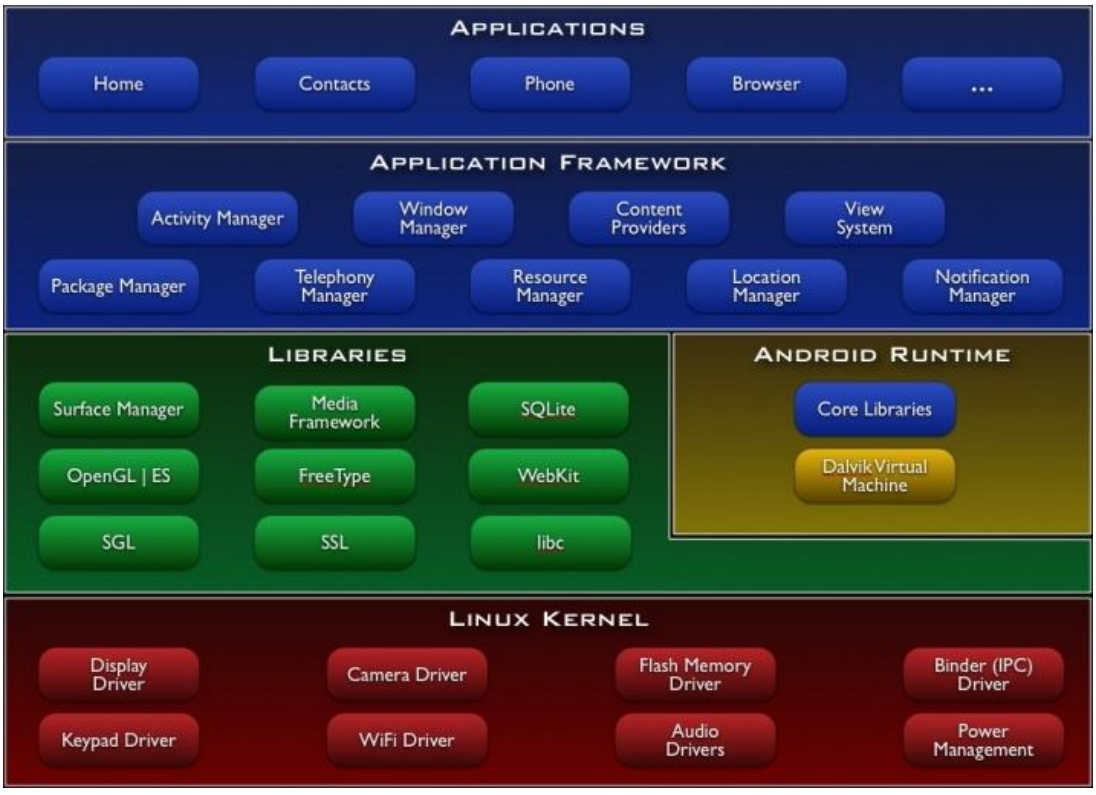

Figura 6: Arquitectura de Android[11]

Antes de desarrollar una aplicación se ha de conocer bien la arquitectura del sistema operativo para el cual se desarrolla. Como se puede observar en la Figura 6, Android se divide en 5 capas. La arquitectura Android es una arquitectura de pila, es decir, cada capa utiliza los elementos de la capa inferior para realizar sus funciones.

- Linux Kernel: Núcleo del sistema operativo y a su vez capa encargada de gestionar los accesos al hardware por parte de las aplicaciones. Esto permite que las aplicaciones accedan a los componentes sin necesidad de conocer características concretas de los mismos. La versión del *kernel* que incorpora Android es la 2.6, similar a la utilizada en cualquier distribución Linux para PC, pero adaptada para dispositivos móviles.
- Librerías (*Libraries*): También conocidas como bibliotecas, normalmente están compiladas para la arquitectura de hardware específica del dispositivo móvil. Son utilizadas para evitar codificar cada vez aplicaciones de uso frecuente y dotar a las mismas de una mayor funcionalidad.
- Entorno de ejecución de Android (*Android Runtime*): Su componente principal es la máquina virtual *Dalvik*, encargada de ejecutar aplicaciones no nativas de Android. Estas aplicaciones suelen estar codificadas en Java. Está formado también por librerías que incluyen funcionalidades de Java.
- Marco de aplicación (*Application Framework*): Esta capa contiene las clases que se van a utilizar por parte de las aplicaciones. Estas clases suelen ser librerías de Java interpretadas por la máquina virtual *Dalvik*.
- Aplicaciones (*Applications*): Capa superior formada por las aplicaciones. Incluye todas las aplicaciones del dispositivo.

El objetivo principal, como se ha explicado, es aprender a desarrollar aplicaciones para Android. Por este motivo, se considera esencial trabajar un gran número de componentes de la arquitectura de Android. Los componentes trabajados en este proyecto son:

- Controlador de la cámara (*Camera driver*): Controlador encargado de proporcionar acceso del software a la cámara del hardware. Indispensable su uso para la lectura de códigos.
- Librerías del núcleo (*Core libraries*): Librerías esenciales pero no nativas de Android. Se utiliza para añadir las funcionalidades propias de Java.
- Máquina virtual *Dalvik* (*Dalvik virtual machine*): Encargada de ejecutar aplicaciones no nativas de Android.
- Administrador de paquetes (*Package manager*): Parte encargada de realizar la compilación en la extensión propia de las aplicaciones Android (.apk).
- Administrador de recursos (*Resource Manager*): Este componente se utiliza para acceder a todos los elementos de la aplicación que se utilizan en el código, como por ejemplo, imágenes, sonidos y archivos de texto.
- Vistas (*View System*): Utilizado para construir interfaces de usuario.
- Aplicaciones: Cualquier otra aplicación utilizada por la aplicación en desarrollo.

En la Figura 7 se puede observar todos los componentes utilizados en el proyecto global, así como los recursos humanos que los llevan a cabo.

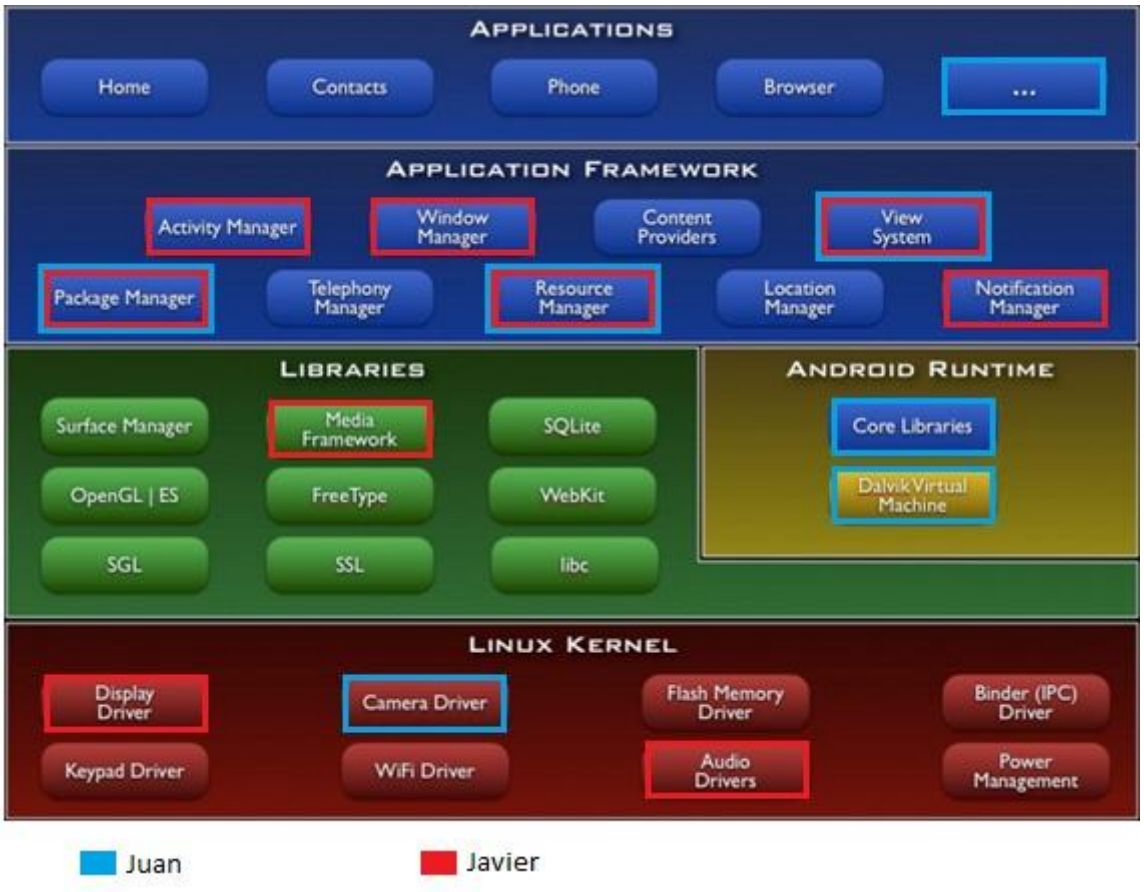

Figura 7: Arquitectura de Android utilizada

Estructura de Android dentro del entorno de trabajo

Tras explicar cómo se estructura Android a nivel arquitectural, se pasa a definir la estructura existente en el entorno de trabajo. En este proyecto, como se detalló anteriormente, se desarrolla en Eclipse.

Dentro de un proyecto Android nos encontramos una serie de archivos y carpetas. Cada uno de ellos desarrolla una función específica dentro del proyecto y aloja unos determinados archivos. Pasamos a explicar con detalle los más relevantes en el desarrollo:

- Carpeta “src”: Dentro de esta carpeta se encuentra todo el código fuente de la aplicación. Por defecto se crea un paquete y dentro de éste, la clase java asociada a la actividad principal. Con cada actividad que se cree, se generará automáticamente una clase Java asociada a la misma dentro de esta carpeta. También puede contener clases que no están asociadas a ninguna actividad.
- Carpeta “gen”: En esta carpeta se generan archivos automáticamente. Contiene una clase de definición de variables y métodos autogenerados llamada R.java.

- Carpeta “assets”: Esta carpeta se usa para incluir al proyecto archivos que no necesitan ser compilados.
- Carpeta “bin”: Los archivos de esta carpeta se crean al compilar la aplicación y contiene por ejemplo el archivo “.apk”, necesario para instalar la aplicación en dispositivos móviles Android.
- Carpeta “res”: Esta carpeta contiene los archivos requeridos por nuestra aplicación y está formada por diferentes carpetas.
    - “*Drawable*” sirve para colocar archivos de gráficos. Al estar tan diversificado el mercado de móviles con sistema operativo Android se requieren diferentes carpetas “*drawable*” según las especificaciones de las pantallas.
    - “*Layout*” es donde se encuentran definidas cada una de las pantallas que va a tener la aplicación en XML.
    - “*Menu*” contiene el fichero “menu.xml” que es el encargado de configurar los diferentes menús que aparecen en la aplicación al pulsar la tecla de menú del dispositivo móvil.
    - “*Values*” es donde se aloja, por ejemplo, el archivo “strings.xml” que contiene, entre otros, todos los mensajes que va a mostrar la aplicación.
- *AndroidManifest.xml*: Archivo principal de configuración de la aplicación. En él, por ejemplo, se incluyen permisos, se referencian librerías, se incluye la versión de Android y se configura la pantalla que se ejecutará por defecto.

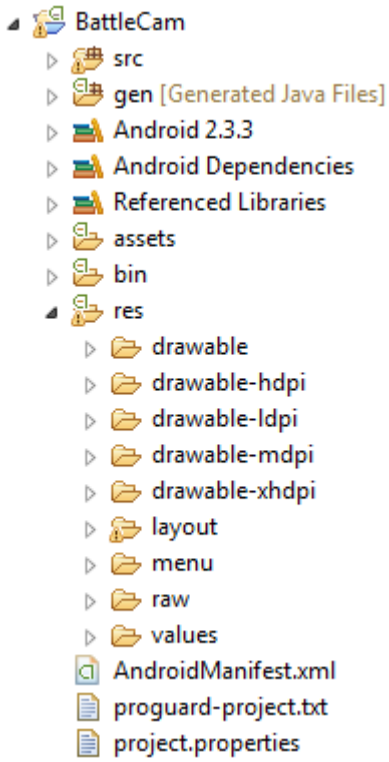

Figura 8: Estructura de este proyecto en Eclipse

Como se puede observar en la Figura 8, en la estructura de este proyecto, existen también las librerías propias de la versión de Android 2.3.3, las dependencias de librerías (*Android Dependencies*) y las librerías externas que se utilizarán (*Referenced Libraries*).

3.3 Actividad

Una actividad, en el contexto de desarrollo de aplicaciones para Android, es cada pantalla de la aplicación que interactúa con el usuario. Cada actividad lleva asociada una clase de Java que, a su vez, extiende la clase "*Activity*".

La clase "*Activity*" es la encargada de crear la ventana donde se puede poner la interface de usuario y que responderá ante las acciones del sistema operativo o del usuario.

Ciclo de vida de una actividad

Toda aplicación Android funciona con un ciclo de vida controlado por el sistema operativo. El ciclo de vida depende de cómo se encuentran las actividades de la aplicación en cada momento. Por ello, se debe tener en cuenta cómo funciona el ciclo de vida básico de una actividad.

El sistema operativo almacena las actividades en pila. Al empezar una nueva actividad, ésta se posiciona en la parte superior de la pila y se ejecuta, dejando las demás actividades tapadas, parcial o completamente. El resto de actividades se ejecutarán cuando no tengan ninguna otra actividad por encima.

Desde que se crea una actividad hasta que se destruye puede encontrase en 4 estados diferentes:

- Activa (*Running*): La actividad está arriba de la pila, se está ejecutando, se está mostrando y el usuario puede interactuar con ella.
- Pausada (*Pause*): La actividad ha pasado a segundo plano pero sigue visible porque otra actividad se ha puesto delante en la pila pero no la tapa del todo. Si se necesita liberar recursos, el sistema puede destruir esta actividad.
- Parada (*Stopped*): La actividad ha pasado a segundo plano y está completamente tapada. Igual que en el estado anterior, si el sistema necesita liberar recursos puede destruirla.
- Destruida (*Destroyed*): La actividad ya no está disponible y se han liberado sus recursos.

Métodos para la gestión del ciclo

Los métodos de gestión del ciclo se utilizan para ejecutar operaciones cuando una actividad va cambiando de estados. Los métodos que se utilizan son los siguientes:

- *onCreate()*: Método llamado cuando se crea la actividad por primera vez.
- *onRestart()*: Se llama a este método después de que la actividad haya sido parada, para que empiece nuevamente.
- *onStart()*: Método llamado cuando la actividad pasa a ser visible para el usuario.
- *onResume()*: Se llama cuando la actividad empieza a interactuar con el usuario.
- *onPause()*: Método utilizado cuando el sistema va a resumir una actividad previa. Se usa principalmente, por ejemplo, para parar animaciones y cosas que puedan hacer que la CPU consuma.
- *onStop()*: Llamado cuando la aplicación ya no sea visible para el usuario, porque otra actividad se está ejecutando y tapa a ésta.
- *onDestroy()*: Último método llamado antes de que la actividad sea destruida.

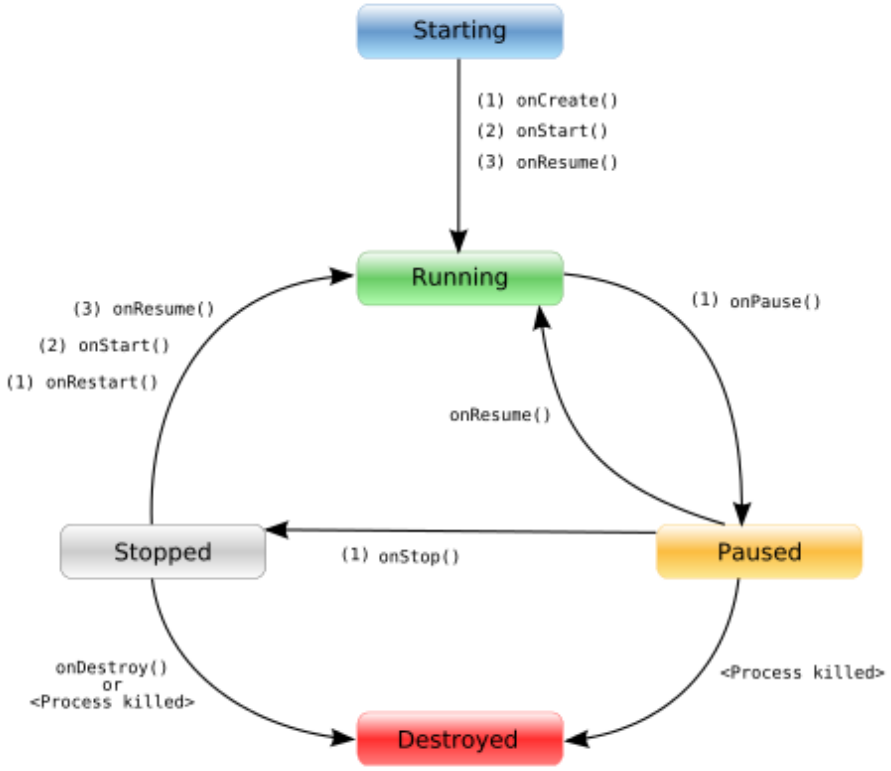

Figura 9: Ciclo de vida de una actividad y métodos de gestión.[12]

3.4 Servicios

Existen componentes que se ejecutan en Android y no tienen interacción con el usuario, estos componentes se llaman servicios[13]. Android incorpora un gran número de servicios, pero también se pueden añadir nuevos, declarándolos en el AndroidManifest.xml.

Ciclo de vida de un servicio

Los ciclos de vida de los servicios tienen dos posibles formas y, dependiendo a través de qué método se llame al servicio, se ejecutará un ciclo u otro. Las posibles formas son:

- Empezado (*Started*): Este servicio se ejecutará hasta que finalice.
- Vinculado (*Bound*): Un componente de la aplicación se vincula a este servicio para interactuar con el mismo.

Métodos para la gestión del ciclo

Los métodos más importantes que se usan para la gestión del ciclo de los servicios son:

- *onCreate()*: Método llamado para establecer la configuración del servicio.
- *onStartCommand()*: Es usado cuando un componente requiere que el servicio se inicie. Con este método el servicio se ejecutará en segundo plano indefinidamente.
- *onBind()*: Este método es usado cuando un componente se quiere vincular al servicio. Al vincularse, el componente que hace de cliente se puede comunicar con el servicio.
- *onDestroy()*: Cuando no es necesario seguir ejecutando el servicio se ejecuta este método.

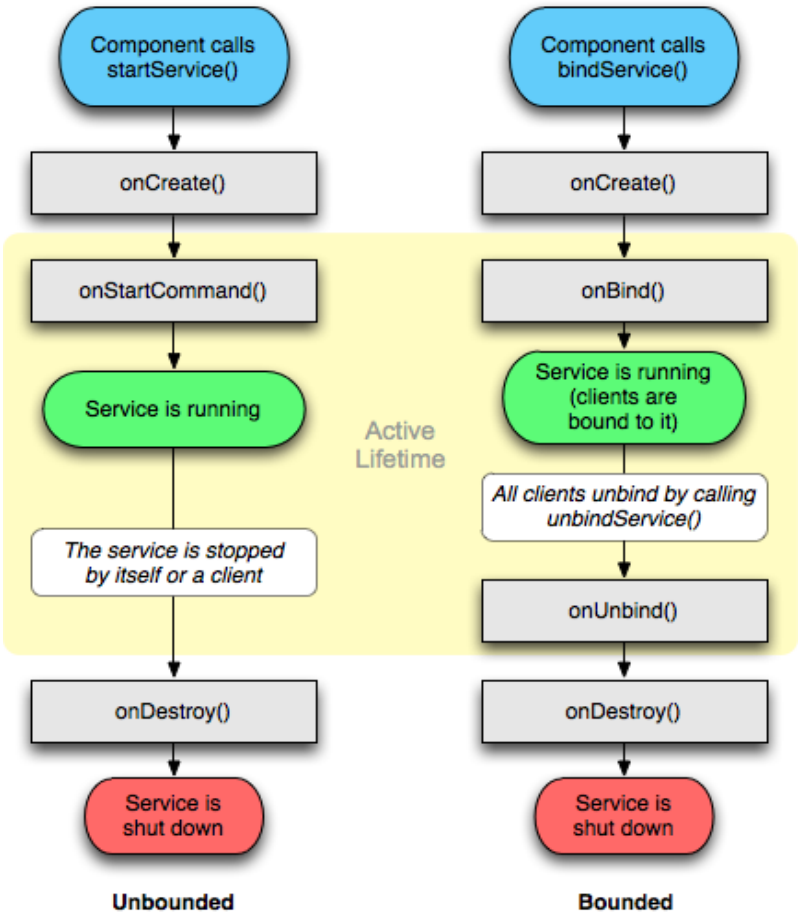

Figura 10: Ciclos de vida de los servicios

4. Análisis

En este apartado se realizará el estudio previo al desarrollo del proyecto global. Se llevará a cabo un análisis de los posibles casos de uso de las diferentes partes de la aplicación.

4.1 Diagrama de casos de uso

Es un diagrama que representa el comportamiento del sistema desde el punto de vista del usuario. En el diagrama podemos observar las partes que desarrollarán cada uno de los miembros del equipo del proyecto global.

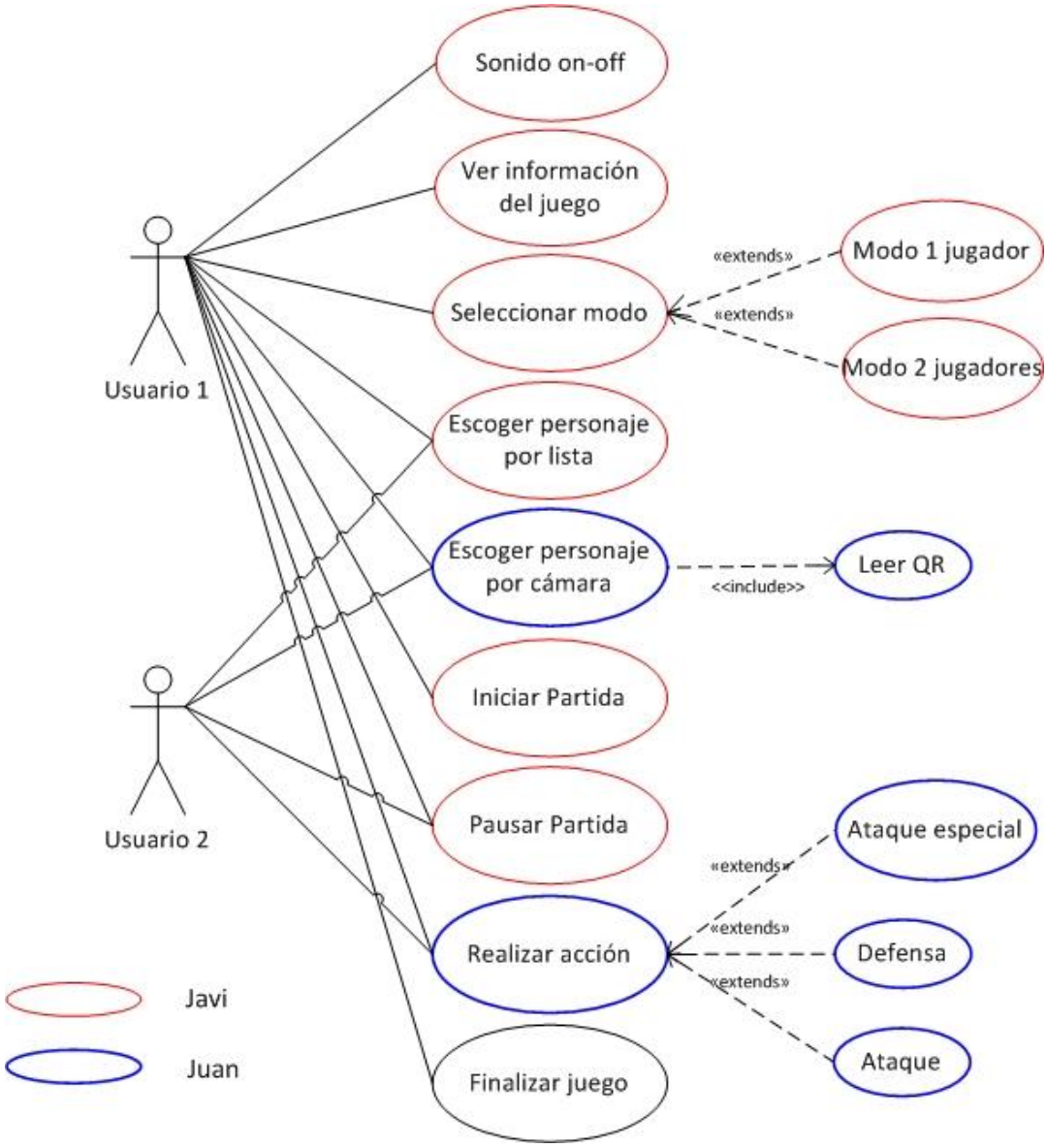

Figura 11: Diagrama de casos de uso

4.2 Casos de uso

Se especificarán los pasos que deberán realizarse para llevar a cabo los procesos expuestos en el diagrama de casos de uso. Con este fin y como paso previo, se han de definir los actores que intervendrán en el uso de la aplicación y para ello se utilizará la siguiente estructura:

- Actor: Identificador del actor.
- Descripción: Breve descripción del actor.
- Relaciones: Posibles relaciones del actor con otros actores del sistema.
- Casos de uso: Listado de los casos de uso en los que interviene el actor.

Actor	Usuario 1
Descripción	Es el actor principal y representa cualquier persona que utilice la aplicación. Desarrolla el papel del jugador número uno y es el encargado de iniciar la aplicación y seleccionar varias opciones disponibles.
Relaciones	Se relaciona con el Usuario 2 en el modo de dos jugadores.
Casos de uso	Ver información del juego, Cambiar sonido, Seleccionar modo, Escoger personaje por lista, Escoger personaje por cámara, Iniciar partida, Pausar partida, Realizar acción, Finalizar juego.

Actor	Usuario 2
Descripción	Actor principal que participa en algunas partes de la aplicación. Desarrolla el papel de segundo jugador en el modo de dos jugadores.
Relaciones	Se relaciona con el Usuario 1 en el modo de dos jugadores.
Casos de uso	Escoger personaje por lista, Escoger personaje por cámara, Realizar acción.

Actor	Sistema
Descripción	Actor secundario que representa a la aplicación.
Relaciones	Se relaciona con los demás actores durante el transcurso de toda la aplicación.
Casos de uso	Ver información del juego, Cambiar sonido, Seleccionar modo, Escoger personaje por lista, Escoger personaje por cámara, Iniciar partida, Pausar partida, Realizar acción, Finalizar juego.

Una vez definidos los actores, se pasa a detallar la estructura de los casos de uso que se llevan a cabo en este proyecto. Será la siguiente:

- Nombre: Nombre del caso de uso.
- Actores: Actores que participan en el caso de uso.
- Descripción: Breve descripción del caso de uso.

- Precondiciones: Condiciones que deben cumplirse para poderse dar este caso de uso.
- Flujo principal: Serie de pasos que tienen lugar durante el transcurso del caso de uso.
- Flujo alternativo: Pasos alternativos que pueden ocurrir durante el transcurso del caso de uso. Pueden existir más de un flujo alternativo, en este caso se enumerarán.

Nombre	Escoger personaje por cámara
Actores	Usuario 1, Usuario 2, Sistema
Descripción	El Usuario 1 y/o el Usuario 2 eligen personaje mediante la cámara del dispositivo móvil.
Precondiciones	El Usuario 1 debe estar en la pantalla de selección de personajes del modo de uno o dos jugadores. El Usuario 2 debe estar en la pantalla de selección de personajes del modo de dos jugadores.
Flujo principal	1. El Usuario 1 pulsa el icono de selección de personaje utilizando la cámara. 2. El Sistema ejecuta el lector de códigos QR. 3. El Usuario 1 posiciona el código elegido. 4. El sistema muestra el resultado asociado al código. 5. El Usuario 1 pulsa el botón empezar.
Flujo alternativo 1	1. El Usuario 2 pulsa el icono de selección de personaje utilizando la cámara. 2. El Sistema ejecuta el lector de códigos QR. 3. El Usuario 2 posiciona el código elegido. 4. El sistema muestra el resultado asociado al código. 5. El Usuario 1 pulsa el botón empezar.
Flujo alternativo 2	1. El Usuario 1 y/o el Usuario 2 quieren volver a elegir personaje. 2. El Usuario 1 y/o el Usuario 2 repiten los pasos del 1 al 5 del flujo principal.

Nombre	Realizar acción
Actores	Usuario 1, Usuario 2, Sistema
Descripción	El Usuario 1 y/o el Usuario 2 realizan acciones durante la partida.
Precondiciones	El Usuario 1 y/o el Usuario 2 deben estar en la pantalla de lucha del modo de un jugador, o bien, en la pantalla de lucha del modo de 2 jugadores.
Flujo principal	1. El Usuario 1 pulsa el botón de “Ataque”, “Defensa” o “Ataque Especial” (si éste está disponible).

	2. El Sistema genera una acción aleatoria entre las disponibles. 3. El Sistema calcula los daños, actualiza los valores de los personajes y lleva a cabo las animaciones pertinentes. 4. El Sistema se queda listo para recibir una nueva acción.
Flujo alternativo 1	1. Se repiten los pasos 1, 2 y 3 del flujo principal. 2. El Sistema detecta una vida igual a 0 y acaba la partida, proclamando un vencedor.
Flujo alternativo 2	1. El Usuario 1 pulsa el botón de “Ataque”, “Defensa” o “Ataque Especial” (si éste está disponible). 2. El Usuario 2 pulsa el botón de “Ataque”, “Defensa” o “Ataque Especial” (si éste está disponible). 3. El Sistema calcula los daños, actualiza los valores de los personajes y lleva a cabo las animaciones pertinentes. 4. El Sistema se queda listo para recibir una nueva acción.
Flujo alternativo 3	1. El Usuario 1 pulsa el botón de “Ataque”, “Defensa” o “Ataque Especial” (si éste está disponible). 2. El Usuario 2 pulsa el botón de “Ataque”, “Defensa” o “Ataque Especial” (si éste está disponible). 3. El Sistema calcula los daños, actualiza los valores de los personajes y lleva a cabo las animaciones pertinentes. 4. El Sistema detecta una vida igual a 0 y acaba la partida, proclamando un vencedor.

Nombre	Finalizar partida
Actores	Usuario 1, Sistema
Descripción	Finalización de la partida actual y salida del juego.
Precondiciones	El Usuario 1 debe estar en la pantalla de selección de modo.
Flujo principal	1. El Usuario 1 pulsa el botón físico de retroceso del dispositivo móvil. 2. El Sistema muestra una ventana de confirmación. 3. El Usuario 1 pulsa el botón “Sí”. 4. El Sistema cierra la aplicación.
Flujo alternativo 1	1. El Usuario 1 pulsa el botón físico “Home” del dispositivo móvil en cualquier momento. 2. El Sistema sale de la aplicación dejándola en segundo plano.
Flujo alternativo 2	La aplicación se encuentra en la pantalla de lucha de cualquier modalidad. 1. Se repiten los pasos del 1 al 3 del flujo principal. 2. El sistema acaba el enfrentamiento y muestra la pantalla de selección de modo. 3. Se repiten los pasos del 1 al 4 del flujo principal.

5. Diseño e implementación

Aquí se llevará a cabo la explicación del diseño del proyecto global y de la implementación de la parte específica de este proyecto.

5.1. Diseño

En este apartado se explicarán los diferentes flujos de navegación que tiene el proyecto global. Aunque el diseño no se trata como algo primordial en el proyecto, la aplicación se ha desarrollado intentando que el diseño sea fácil y amigable para el usuario.

Pantalla inicial

Figura 12: Pantalla inicial de *BattleCam*

Esta es la pantalla que nos encontramos al ejecutar la aplicación (Figura 12). Contiene el logotipo del juego y un único botón para acceder a la siguiente pantalla. Si mientras permanecemos en esta pantalla el usuario pulsa el botón físico de retroceso del dispositivo móvil, la aplicación terminaría.

Pantalla de menú (o de selección de modo)

En esta pantalla se ofrecen al usuario los diferentes modos de juego que están disponibles (Figura 13). También se puede controlar desde aquí el sonido, pulsando sobre el icono del altavoz. Si el icono está de color verde, el sonido estará activado, si por el contrario el icono es rojo, el sonido estará deshabilitado.

Figura 13: Pantalla de menú (o selección de modo)

Si se pulsa el botón físico de menú del dispositivo móvil aparecerá un menú emergente con el botón para acceder a un apartado de información acerca del proyecto global (Figura 14). Para salir de dicha vista, únicamente se debe pulsar el botón físico de retroceso.

Figura 14: Ventana de información

Si pulsamos el botón físico de retroceso se ejecutará una nueva ventana de confirmación de salida de la aplicación (Figura 15). En caso afirmativo, la aplicación se cerrará completamente, pero si pulsamos el botón "No", la ventana desaparecerá y se volverá a la pantalla de menú.

Figura 15: Ventana de confirmación de salida

Según el botón que pulsemos (un jugador o dos jugadores), se pasará a una pantalla u otra, pero en ambos casos, el usuario será enviado (o los usuarios serán enviados) a una pantalla de selección de personaje.

Pantalla de selección de personaje

La variación entre las pantallas de selección de personajes es mínima. Dejando a un lado el título, la diferencia más significativa que podemos observar es que, en el modo de un jugador, el único personaje que podemos seleccionar es el jugador 1 y en el modo de dos jugadores, se pueden seleccionar tanto el jugador 1 como el 2 (Figura 16).

Figura 16: Pantallas de selección de personajes

En el caso de que se quiera hacer una selección manual de personajes, se pulsara el botón de selección manual en cualquiera de los dos modos de juego, lo que hará aparecer una ventana con la lista de personajes disponibles (Figura 17).

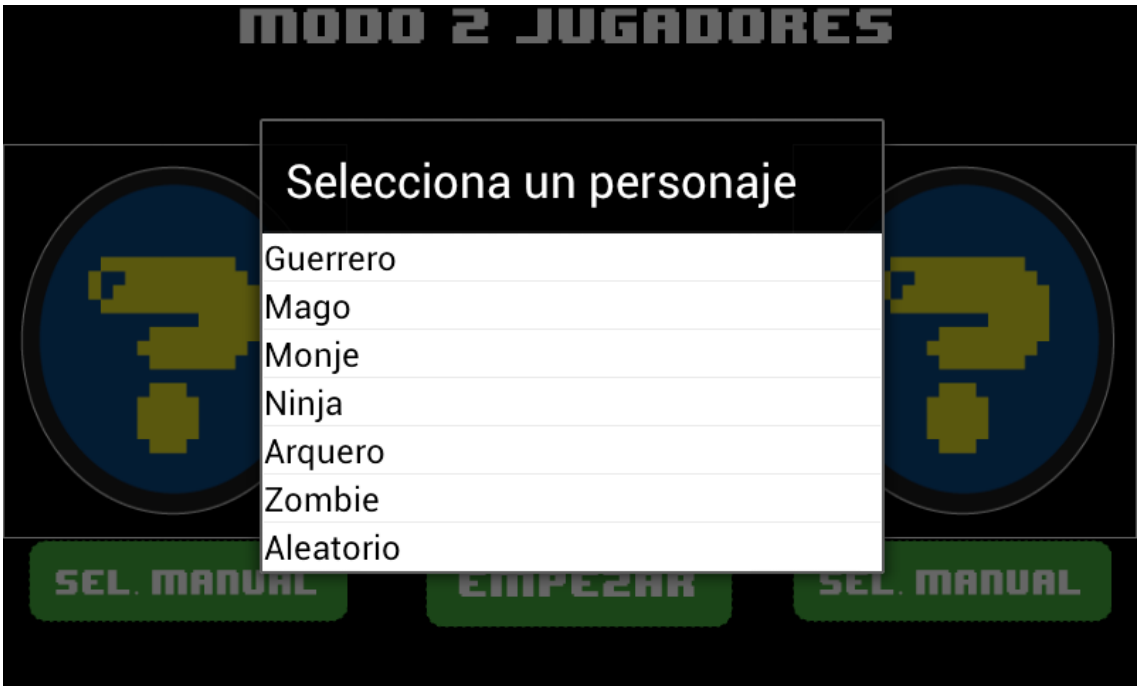

Figura 17: Ventana de selección manual de personajes

El botón de empezar no estará activo en ninguno de los flujos hasta que todos los jugadores hayan seleccionado un personaje para jugar. En caso de darle antes, aparecerá una ventana con un mensaje informativo (Figura 18).

Figura 18: Ventanas de información

En el modo de dos jugadores el texto cambia a plural.

El icono de color azul con el interrogante amarillo, además de mostrar la selección de personaje realizada, es el botón que se pulsará cuando queramos acceder a la función de escanear un código QR para seleccionar el personaje con el que se desea jugar.

Pantalla de lectura de códigos QR

Esta pantalla habilitará la cámara para utilizarla de lector de códigos. El código se situará en el recuadro interior de la pantalla de lectura. La posición del código no es importante siempre y cuando no aparezca parcialmente tapado.

Figura 19: Pantalla de lectura de códigos QR

Pantalla de lucha

En esta pantalla transcurrirá toda la acción del juego, por lo tanto, es una parte esencial del proyecto. La ausencia de botones para controlar al segundo jugador en el modo de un jugador, es el único componente diferenciador que se puede apreciar a simple vista entre el modo de un jugador y el modo de dos jugadores.

Figura 20: Pantallas de lucha

Como se observa en la Figura 20, cada jugador tiene 6 elementos en pantalla, que son:

- Personaje: Gráfico en dos dimensiones del personaje seleccionado en la pantalla anterior. Realizará pequeñas animaciones, según la acción escogida. El personaje del jugador 1 se mostrará en la parte izquierda de la pantalla y el personaje del jugador 2 o del sistema en la derecha.
- Barra de vida: Se muestra una barra de color verde y un corazón en la parte superior de la pantalla para cada jugador. Según va bajando la vida la barra se hace más pequeña hasta llegar a 0.

- Barra de energía: Se muestra una barra en la parte superior de la pantalla acompañada por una estrella para cada jugador. Inicialmente la barra esta a 0 y según se recarga se vuelve de color amarillo.
- Botón "Ataque": Botón que se muestra en la parte inferior de la pantalla. En el modo de un jugador solo existe un botón de este tipo, mientras que, en el modo de dos jugadores hay un botón para cada uno de los jugadores.
- Botón "Ataque especial": Botón que se muestra en la parte inferior de la pantalla. Se habilita únicamente en el caso de que la barra de energía este al máximo y se deshabilita con un uso, hasta que la energía vuelve a estar al máximo. En el modo de un jugador solo existe un botón de este tipo, mientras que, en el modo de dos jugadores hay un botón para cada uno de los jugadores.
- Botón "Defensa": Botón que se muestra en la parte inferior de la pantalla. En el modo de un jugador solo existe un botón de este tipo, mientras que, en el modo de dos jugadores hay un botón para cada uno de los jugadores.

Si en cualquier momento uno de los usuarios pulsa el botón físico de retroceso del dispositivo móvil, aparecerá una ventana para confirmar la finalización de la partida. Si en esta ventana se pulsa "Sí" se regresará a la pantalla de menú (o selección de modo), si por el contrario se pulsa "No" la partida se reanudará.

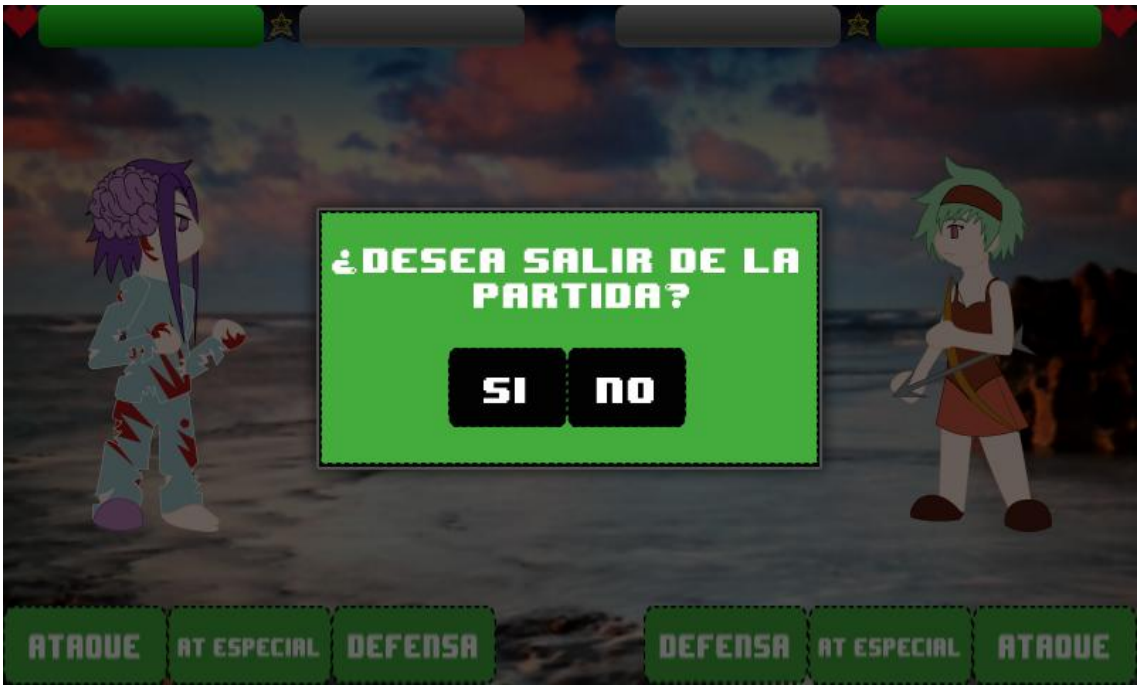

Figura 21: Ventana de confirmación de fin de partida en la pantalla de lucha

Una vez finalizada la partida, se muestra una ventana que nos muestra el jugador que ha vencido y un único botón para aceptar. Si pulsamos este botón se regresará a la pantalla de menú o selección de modo.

Figura 22: Pantalla de partida finalizada

5.2 Implementación

En el siguiente apartado se explicará la implementación que se ha realizado en este proyecto de los componentes necesarios.

5.2.1 Configuración del entorno de trabajo

Para empezar la implementación, se debe instalar el programa de desarrollo deseado que, en el caso de este proyecto, es Eclipse. A continuación, se instalará Java junto con sus herramientas JRE y JDK, posteriormente, se instala y actualiza el Android SDK. Para este proyecto se instalará también Apache Ant.

Una vez instalados todos los componentes, se tienen que configurar las variables de entorno del sistema operativo (Figura 23). Para ello, se crearán o modificarán las siguientes variables:

- JAVA_HOME: En esta variable se tiene que poner la ruta del directorio donde se encuentra el JDK de Java.
- ANT_HOME: En esta variable se tiene que poner la ruta donde se ha descomprimido el directorio de Apache Ant.
- Classpath: En esta variable únicamente se tiene que poner un ";".
- Path: En esta variable se tiene que añadir la ruta de los directorios "bin" del JDK de Java y del Apache Ant.

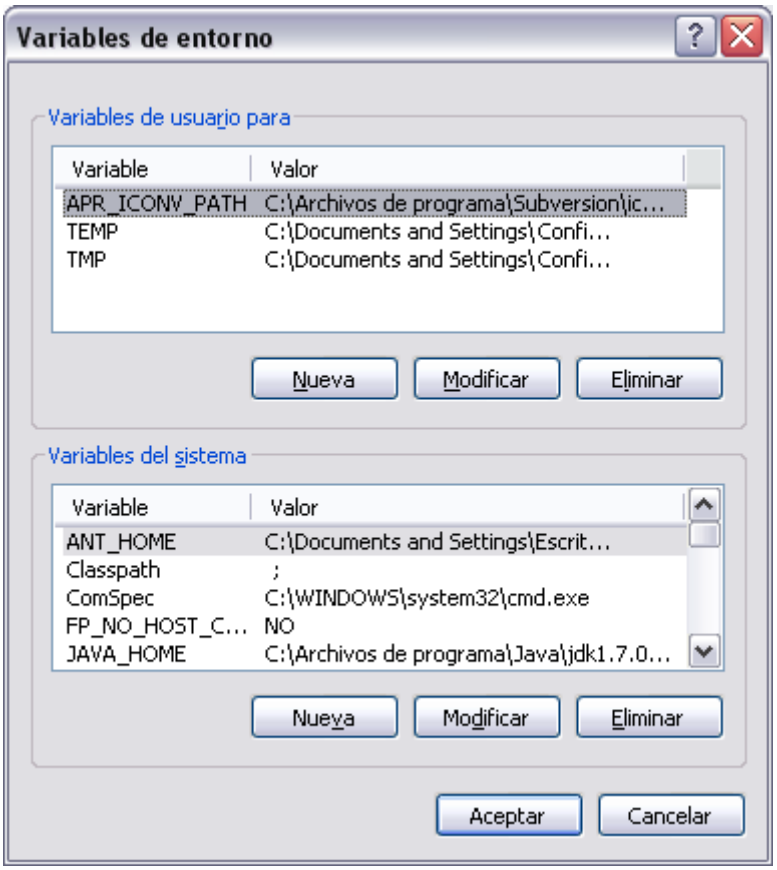

Figura 23: Variables de entorno

5.2.2 Lectura de códigos QR y *ZXing*

Como se mencionó en el apartado "2.10 Alternativas y selección de soluciones" de esta memoria, para la lectura de códigos QR se utilizaría como complemento una librería externa llamada *ZXing*. Existen varias formas de usar *ZXing*, por ejemplo, utilizándolo como aplicación externa lectora de códigos, o bien, implementándolo en el proyecto como librería, consiguiendo así que el mismo proyecto pueda leer códigos.

No se pretende depender de programas externos para el funcionamiento del proyecto global, por lo tanto, el primer paso es la implementación de *ZXing* como librería en nuestro proyecto.

Los pasos que se deben realizar para la integración de *ZXing* son los siguientes:

1. Obtener código fuente de ZXing.
2. Crear el núcleo (*core*) de ZXing.
3. Crear proyecto ZXing para usar como librería.
4. Integración de ZXing en el proyecto global.

Todos estos pasos se encuentran detallados en el manual "*Integrate ZXing barcode scanner into your Android app natively using Eclipse*"[14]. Por este motivo no se reproducirá la explicación en esta memoria.

Una vez integrado, es conveniente explicar para que sirven las líneas de código integradas en nuestra aplicación.

```
Intent intent = new Intent("com.google.zxing.client.android.SCAN");
intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
startActivityForResult(intent, 0);
```

Estas tres líneas de código sirven para crear un enlace con una actividad nueva del tipo "*SCAN*" de la librería *ZXing*, indicarle el tipo de código que se espera recibir e inicializar la actividad esperando resultado de la misma.

El segundo parámetro del método "*startActivityForResult()*" se utiliza para asignar un código de resultado al objeto que se escanee. Este parámetro es importante si se desea realizar más de una lectura de códigos y que éstas se almacenen para luego ser tratadas durante el transcurso de una misma actividad.

El siguiente paso es capturar el resultado y para ello usaremos el siguiente código:

```
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
       if (requestCode == 0) {
          if (resultCode == RESULT_OK) {
             String contents = intent.getStringExtra("SCAN_RESULT");
             String format = intent.getStringExtra("SCAN_RESULT_FORMAT");
             // Handle successful scan
          }
          else if (resultCode == RESULT_CANCELED) {
             // Handle cancel
          }
       }
}
```

Esta función comprobará, para la actividad creada con "*startActivityForResult()*", el código asignado y si la lectura ha sido valida. Si el resultado es válido lo introducirá, en este caso, en la variable "*contents*" y el formato del mismo en la variable "format". A continuación, ya se pueden procesar los datos recibidos como sea conveniente.

5.2.3 Lectura de ficheros XML

En este proyecto se utiliza un fichero XML para almacenar las estadísticas de cada uno de los personajes disponibles. Para la correcta lectura del archivo, debe tener una estructura determinada, que será como la siguiente:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="guerrero">
        <item>100</item>
        <item>50</item>
        <item>35</item>
        <item>25</item>
    </string-array>

</resources>
```

Como se puede observar, primero se encuentra la línea de configuración de XML y a continuación el apartado "*resources*". Es dentro de éste apartado donde se introducirán todos los componentes que se necesiten para la aplicación.

Para poder leer los parámetros de manera sencilla, se utilizarán "*string-array*". El identificador de cada "*string-array*" es el campo nombre (*name*) y dentro de cada componente se introducirán los objetos (*item*) que sean necesarios.

Una vez se conoce el formato del archivo con los todos los campos necesarios, se procederá a explicar la forma de leer los datos.

El primer paso es la declaración de un tipo de variable "*ArrayAdapter*" dentro de la clase que se desee utilizar, tal y como se muestra:

```
private ArrayAdapter<CharSequence> adapter1;
```

A continuación, se recuperarán los datos guardados en el fichero XML y se transformarán al tipo de dato necesario para el proyecto.

```
adapter1 = ArrayAdapter.createFromResource(
       this, R.array.guerrero, android.R.layout.simple_spinner_item);
adapter1.setDropDownViewResource(android.R.layout.simple_dropdown_item_1line);
atk1=Integer.parseInt(adapter1.getItem(0).toString());
def1=Integer.parseInt(adapter1.getItem(1).toString());
matk1=Integer.parseInt(adapter1.getItem(2).toString());
mdef1=Integer.parseInt(adapter1.getItem(3).toString());
```

La primera línea de código se utiliza para cargar, en la variable del tipo "*ArrayAdapter*" declarada con anterioridad, el contenido del "*string-array*" del fichero XML con el nombre deseado (en este caso: guerrero). Con el método "getItem()" seleccionaremos los diferentes parámetros guardados dentro del "string-array" elegido.

5.2.4 Mecánica de juego

Al tratarse de un juego, se ha desarrollado un sistema de reglas de juego básicas y unas condiciones que se deben implementar. A continuación, se explicará cómo se ha implementado el sistema en las diferentes clases.

Modo de un jugador

En este caso, únicamente se requiere la interacción de un usuario con el sistema, por lo tanto, cada acción que realice el usuario debe tener una respuesta del sistema.

En el desarrollo de una partida, el usuario puede realizar tres acciones, que son:

- Ataque
- Ataque Especial
- Defensa

Al pulsar sobre cualquiera de estos botones se lanzará la función que determina la acción a realizar por el sistema. En función de las acciones del usuario y del sistema, se calcularán los datos necesarios, se actualizarán los parámetros y se visualizarán las animaciones y los cambios en pantalla.

Modo de dos jugadores

En este modo, se requiere que los dos usuarios seleccionen la acción a realizar y no se escoge ninguna acción de forma automática.

En el desarrollo de una partida, cada uno de los usuarios puede realizar tres acciones, que son:

- Ataque
- Ataque Especial
- Defensa

No se dispone de un sistema de alertas para conocer el turno de cada jugador, porque se pretendía tener una mayor fluidez durante el juego. Debido a este motivo, cuando un usuario selecciona una acción, se debe esperar a que el contrario escoja acción a realizar.

Para lograr este propósito, se crearán unas variables globales que permitirán conocer en todo momento, cuantas acciones se han seleccionado y el tipo de éstas.

Una vez seleccionadas las acciones de los dos jugadores se llamará a una función encargada de calcular los datos necesarios. A continuación, se actualizarán los parámetros y se visualizarán las animaciones pertinentes.

Cálculos

Esta clase se utiliza para calcular los datos necesarios para el juego. Consta únicamente de una función, que se debe llamar pasándole como parámetros los dos personajes elegidos (con todas sus estadísticas) y las acciones seleccionadas de cada usuario.

La función retornará el daño recibido de un personaje cada vez que sea llamada.

6. Pruebas

En este apartado se especificarán las pruebas realizadas al proyecto global, detallando las realizadas en este proyecto.

6.1 Batería de pruebas

Serie de pruebas realizadas al proyecto global.

- Prueba 1: Ciclo total del juego.
- Prueba 2: Funcionamiento paralelo de la música.
- Prueba 3: Correcta visualización de todos los elementos en pantalla.
- Prueba 4: Funcionamiento de acciones.
- Prueba 5: Correcta lectura y captura de códigos QR.

A continuación, se detallarán las pruebas que se encuentran subrayadas, que son las correspondientes a este proyecto.

Identificador	Prueba 4
Descripción	Funcionamiento de acciones
Detalles	Modo de un jugador Se ha comprobado el funcionamiento de las tres posibles acciones que se pueden llevar a cabo en este modo para todos los personajes. Además, se ha probado el funcionamiento de respuesta del sistema. Modo de dos jugadores En este caso, se ha probado que funcionen correctamente las restricciones de turnos y que todas las posibles combinaciones de acciones se puedan ejecutar. Estas pruebas se han realizado para todos los personajes posibles.
Resultado	Modo de un jugador Funcionamiento correcto de todas las acciones para todos los personajes. El sistema responde correctamente a todas las acciones del usuario. Modo de dos jugadores La restricción de turnos ha funcionado correctamente y se pueden realizar todas las combinaciones de acciones. Se ha comprobado para todos los personajes.
Observaciones	Se podría cambiar estadísticas de algunos personajes para generar más variedad en el juego.

Identificador	Prueba5
Descripción	Correcta lectura y captura de códigos QR
Detalles	Se ha probado a leer y capturar una variedad de códigos QR, entre los que existen algunos códigos que no corresponden a ningún personaje del juego. En estos casos, por diseño, la aplicación debería detectar estos códigos como personaje aleatorio.
Resultado	La lectura de los códigos QR que contenían el nombre de algún personaje valido, funcionaban correctamente. En el caso de los códigos que no contenían un nombre valido, la aplicación generaba un error.
Solución	Se ha revisado el código de la aplicación y se ha corregido el error. Tras repetir la prueba, no hay más errores.
Observaciones	No hay observaciones.

6.2 Dispositivos de prueba

En este apartado se detallarán los dispositivos utilizados para la realización de las pruebas de este proyecto.

HTC Desire Z

Especificaciones del dispositivo:

- Procesador 800MHz
- 512MB de memoria RAM
- Pantalla de 3,7 pulgadas
- 1,5 GB de memoria interna
- Sistema operativo Android 2.3.3

En este dispositivo la aplicación no ha presentado ningún tipo de error de compatibilidad. Todas las funciones se han realizado correctamente.

Samsung Galaxy SII I9100

Especificaciones del dispositivo:

- Procesador *dual core* 1,5GHz
- 1GB de memoria RAM
- Pantalla de 4,3 pulgadas
- 16 GB de memoria interna
- Sistema operativo Android 4.0.3

En este dispositivo la aplicación no ha presentado ningún tipo de error de compatibilidad. Todas las funciones se han realizado correctamente.

7. Conclusiones

En este apartado se realizará una valoración personal del proyecto, así como los problemas surgidos durante la realización, posibles líneas de ampliación y comparativa entre el estudio previo del proyecto y el realizado.

7.1 Objetivos

En este apartado se considerará si los objetivos marcados en el apartado "1.4 Objetivos" para este proyecto se han cumplido.

O1. Creación de códigos QR y lectura de estos a través de la cámara del dispositivo, obteniendo así los datos leídos.

O2. Implementación de un sistema de reglas y juego.

O3. Poder empezar y acabar una partida.

Estos tres objetivos se han cumplido íntegramente durante el transcurso de este proyecto. El objetivo principal del proyecto global, que era aprender a desarrollar aplicaciones para el sistema operativo Android, también se ha logrado.

7.2 Desviaciones y dificultades

En este apartado se comentarán las dificultades y desviaciones sufridas en este proyecto.

Previsión temporal

La previsión temporal demasiado optimista, ha sido uno de los puntos más negativos en el desarrollo del proyecto global. Debido a esta mala previsión, se ha retrasado el proyecto enormemente, hasta el punto de no ser posible presentarlo en la convocatoria de junio.

El motivo principal es no haber tenido en cuenta el curso académico y la posibilidad de no poder emplear el tiempo necesario al proyecto. Por este motivo el proyecto se ha retrasado desde Enero hasta Junio.

Planteamiento del proyecto

En un principio se quería realizar una aplicación basada en realidad aumentada. Este planteamiento se descartó debido a la poca documentación y alternativas gratuitas existentes.

7.3 Planificación temporal real

En este apartado detallaremos como ha sido la planificación temporal real del proyecto global teniendo en cuenta desvíos, retrasos y reducciones temporales.

Planificación y calendario temporal real

El calendario temporal tiene una duración de diciembre de 2011 a septiembre de 2012. En la Figura 24 observamos la duración real de las tareas.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Proyecto**	**202 días**	**vie 02/12/11**	**mié 19/09/12**		
2	**Planificacion**	**26 días**	**vie 02/12/11**	**mié 11/01/12**		
3	Estudio de viabilidad	2 días	vie 02/12/11	lun 05/12/11		Javi;Juan
4	Estudio de requisitos	2 días	mié 07/12/11	vie 09/12/11	3	Javi;Juan
5	Reparto de tareas (diagrama Gantt)	3 días	lun 12/12/11	mié 14/12/11	4	Javi;Juan
6	Diseño del sistema de juego	4 días	jue 15/12/11	mié 11/01/12	5	Javi;Juan
7	**Desarrollo (1ª iteracion)**	**127 días**	**jue 12/01/12**	**jue 12/07/12**	**6**	
8	Creacion de clases y metodos	8 días	jue 12/01/12	jue 21/06/12	6	Juan;Javi;Supervisor
9	Utilización de la camara	5 días	vie 22/06/12	jue 28/06/12	8	Javi;Juan
10	Mostrar graficos en pantalla	5 días	jue 12/01/12	jue 21/06/12	6	Javi
11	Diseño de controles	3 días	vie 22/06/12	mar 26/06/12	10	Javi
12	Lectura de codigos QR	10 días	vie 29/06/12	jue 12/07/12	9	Juan
13	**Desarrollo (2ª iteracion)**	**15 días**	**vie 29/06/12**	**jue 19/07/12**	**9;10**	
14	Revision de requisitos	2 días	vie 29/06/12	lun 02/07/12	9;10	Javi;Juan;Supervisor
15	Mostrar controles en pantalla	3 días	vie 29/06/12	mar 03/07/12	11	Javi
16	Diseño de personajes	7 días	mié 04/07/12	jue 12/07/12	15	Diseñador
17	Mostrar personajes en pantalla	5 días	vie 13/07/12	jue 19/07/12	16	Javi
18	**Desarrollo (3ª iteracion)**	**21 días**	**vie 20/07/12**	**vie 17/08/12**	**17;14**	
19	Revision de requisitos	2 días	vie 20/07/12	lun 23/07/12	17;14	Javi;Juan;Supervisor
20	Codificacion de algoritmos	6 días	vie 20/07/12	vie 27/07/12	12	Juan
21	Programacion de controles	4 días	lun 30/07/12	jue 02/08/12	20	Juan
22	Programacion de acciones	7 días	vie 03/08/12	lun 13/08/12	21	Juan
23	Version beta	4 días	mar 14/08/12	vie 17/08/12	22	Juan
24	**Desarrollo (4ª iteracion)**	**12 días**	**lun 20/08/12**	**mar 04/09/12**	**19;23**	
25	Revision de requisitos	2 días	lun 20/08/12	mar 21/08/12	19;23	Javi;Juan;Supervisor
26	Bateria de pruebas	2 días	mié 22/08/12	jue 23/08/12	25	Javi;Juan
27	Prueba por betatesters	3 días	vie 24/08/12	mar 28/08/12	26	Betatester
28	Revision	3 días	mié 29/08/12	vie 31/08/12	27	Javi;Juan
29	Finalizar desarrollo	2 días	lun 03/09/12	mar 04/09/12	28	Javi;Juan
30	Desarrollo memoria	10 días	mié 05/09/12	mar 18/09/12	29	Javi;Juan
31	Presentacion del proyecto	1 día	mié 19/09/12	mié 19/09/12	30	Javi;Juan
32	Final del proyecto	0 días	mié 19/09/12	mié 19/09/12	31	Javi;Juan

Figura 24: Planificación de tareas real

Se han realizado cambios respecto a la planificación temporal y se ha movido alguna tarea de sitio. A continuación, en la Figura 25 se puede observar el diagrama temporal (de Gantt) real. Con color azul se resalta las actividades realizadas exclusivamente por mí y detalladas en esta memoria. De color rojo y trama oblicua se puede observar el trabajo realizado por el otro programador. De color verde y trama vertical diferenciamos las tareas que se han realizado de forma conjunta. El diseñador se representa con una trama horizontal de color lila.

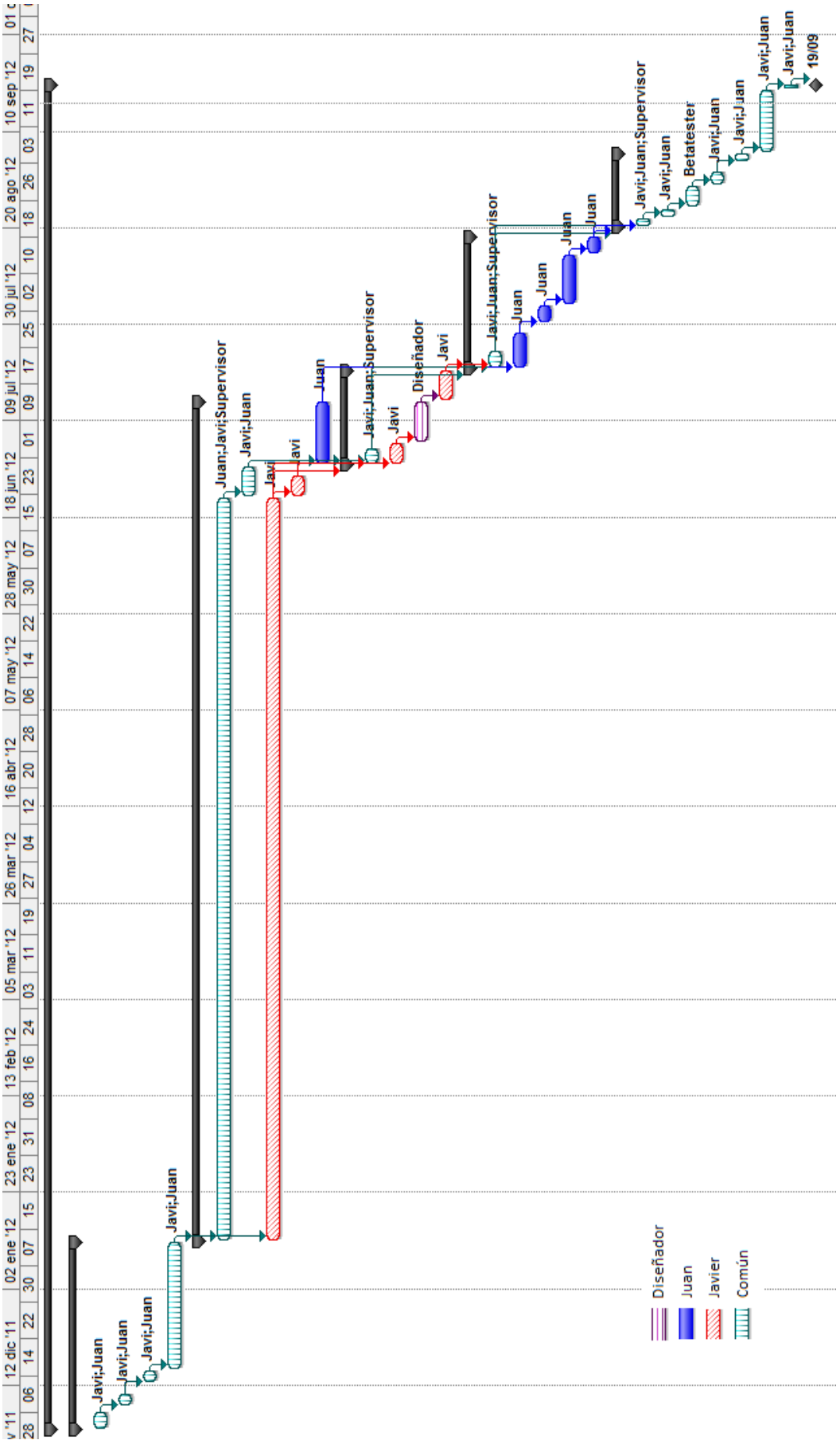

Figura 25: Diagrama temporal (de Gantt) real

Como se puede apreciar en la Figura 25, existe un aplazamiento desde enero a junio de 2012. Esto es debido a la mala previsión temporal comentada en el apartado "7.2 Desviaciones y dificultades" de esta memoria. Exceptuando este aplazamiento, la duración del resto de tareas no ha tenido grandes retrasos, incluso en algún caso, se ha logrado acabar alguna tarea antes de lo planificado.

Con el fin de mostrar los desfases comentados, se ha creado una tabla en la que se puede observar la duración real de todas las tareas realizadas y la previsión de duración hecha en el estudio de viabilidad del proyecto global.

	1ª Planificación	Planificación real	Desfases
Planificación	27 días	26 días	-1 día
Estudio de viabilidad	2 días	2 días	0 días
Estudio de requisitos	2 días	2 días	0 días
Reparto de tareas (diagrama Gantt)	3 días	3 días	0 días
Diseño del sistema de juego	5 días	4 días	-1 día
Creación de clases y métodos	5 días	8 días	+3 días
Codificación de algoritmos	6 días	6 días	0 días
Mostrar gráficos en pantalla	5 días	5 días	0 días
Diseño de controles	10 días	3 días	-7 días
Lectura de códigos QR	10 días	10 días	0 días
Revisión de requisitos	2 días	2 días	0 días
Programación de controles	6 días	4 días	-2 días
Mostrar controles en pantalla	3 días	3 días	0 días
Diseño de personajes	12 días	7 días	-5 días
Mostrar personajes en pantalla	5 días	5 días	0 días
Revisión de requisitos	2 días	2 días	0 días
Programación de acciones	7 días	7 días	0 días
Utilización de la cámara	5 días	5 días	0 días
Versión beta	4 días	4 días	0 días
Revisión de requisitos	2 días	2 días	0 días
Batería de pruebas	3 días	2 días	-1 día
Prueba por betatesters	6 días	3 días	-3 días
Revisión	3 días	3 días	0 días
Finalizar desarrollo	3 días	2 días	-1 día
Desarrollo memoria	10 días	10 días	0 días
Presentación del proyecto	1 día	1 día	0 días
Final del proyecto	0 días	0 días	0 días
	Total		-18 días

En color azul se resaltan las tareas realizadas exclusivamente en este proyecto. En color rojo se muestran las tareas realizadas por la otra parte. El resto son tareas comunes.

Como se puede observar, si se ignora el aplazamiento del proyecto global, el resultado es que el proyecto global hubiese tenido un adelanto de 18 días.

7.4 Ampliaciones

En este apartado se incluirán posibles vías de ampliación para el proyecto global. Las posibles ampliaciones son:

Ampliar el número de dispositivos Android compatibles

Gracias a la retrocompatibilidad de Android, no existen problemas de compatibilidad con versiones del sistema operativo superiores a la versión para la que se ha desarrollado la aplicación. Por ello, si se desea ampliar el número de dispositivos Android compatibles, se han de realizar cambios para poder llegar a versiones anteriores a la versión 2.3.3. Esto supondría una revisión total de requisitos y posiblemente se tendrían que hacer grandes cambios en el código.

Portar la aplicación a otras plataformas

El objetivo de aprender sigue vigente, es por ello que la posibilidad de portar la aplicación a otras plataformas es una vía de ampliación muy válida. Existen varios sistemas operativos para móviles a los que se podría hacer el porte, pero supondría revisar toda la aplicación, cambiando casi íntegramente el código.

Añadir funcionalidades al juego

Se pueden añadir funcionalidades al juego para seguir aprendiendo elementos de Android, como por ejemplo, poder jugar en línea con oponentes que se encuentren en un cierto rango. Ampliar la aplicación por esta vía supondría un menor esfuerzo, ya que no sería necesario retocar el código ya realizado.

7.5 Valoración personal

Este proyecto (y por extensión el proyecto global) me ha permitido cumplir uno de mis sueños, que era realizar un juego propio desde cero.

Me gustaría destacar el aprendizaje adquirido, ya que no es solo el conocimiento sobre la materia obtenido, sino que también he aprendido valiosas lecciones. He visto con frustración cómo nos veíamos obligados a posponer el proyecto global si queríamos aprobar el resto de asignaturas de la carrera, dejando pasar una convocatoria. He visto con alegría cómo se iban consiguiendo los objetivos marcados y el juego iba tomando forma. Me he dado cuenta de que apartados que me parecían inservibles, como la planificación o la documentación, son en realidad las cosas más útiles.

Por todos estos motivos, me siento muy orgulloso y satisfecho con el resultado final del proyecto.

8. Referencias electrónicas

[1] *Activity*. Android developers [fecha de consulta: agosto de 2012]. Disponible en <http://developer.android.com/reference/android/app/Activity.html>

[2] Colaboradores de Wikipedia. *Android*. Wikipedia, La enciclopedia libre, 2012 [fecha de consulta: julio de 2012]. Disponible en: <http://es.wikipedia.org/wiki/Android>

[3] *Android SDK*. Android developers [fecha de consulta: julio de 2012]. Disponible en <http://developer.android.com/sdk/index.html>

[4] The Apache Software Foundation. *Apache Ant*. The Apache Software Foundation, mayo de 2012, [fecha de consulta: julio de 2012]. Disponible en <http://ant.apache.org/>

[5] Colaboradores de Wikipedia. *Código QR*. Wikipedia, La enciclopedia libre, 2012 [fecha de consulta: julio de 2012]. Disponible en: <http://es.wikipedia.org/wiki/Codigo_QR>

[6] Colaboradores de Wikipedia. *Java*. Wikipedia, La enciclopedia libre, 2012 [fecha de consulta: julio de 2012]. Disponible en: <http://es.wikipedia.org/wiki/Java_(lenguaje_de_programación) >

[7] Colaboradores de Wikipedia. *Teléfono inteligente*. Wikipedia, La enciclopedia libre, 2012 [fecha de consulta: julio de 2012]. Disponible en: <http://es.wikipedia.org/wiki/Smartphone>

[8] Bruce Berls. *Android fragmentation*. Bruceb consulting, junio de 2012 [fecha de consulta: julio de 2012]. Disponible en: <http://www.brucebnews.com/2012/06/android-fragmentation-aka-wheres-my-ice-cream-sandwich/>

[9] Google. *Registro de desarrolladores*. Google Play para desarrolladores, 2012 [fecha de consulta: septiembre de 2012]. Disponible en: <http://support.google.com/googleplay/android-developer/bin/answer.py?hl=es&answer=113468&topic=2365624&ctx=topic>

[10] Denso Wave, inc. *ZXing (Zebra crossing)*. Denso Wave, inc, 2011 [fecha de consulta: julio de 2012]. Disponible en: <http://code.google.com/p/zxing/>

[11] *Android architecture*. Android developers, 2010 [fecha de consulta: julio de 2012]. Disponible en: <http://developer.android.com/images/system-architecture.jpg>

[12] Telekita. *Ciclo de vida de una activity*. Diario de un proyecto Android, febrero de 2012 fecha de consulta: julio de 2012]. Disponible en: <http://www.javahispano.org/storage/android/imagenes/Ciclo%20de%20Vida%20Android.png>

[13] *Services*. Android developers [fecha de consulta: septiembre de 2012]. Disponible en < http://developer.android.com/guide/components/services.html >

[14] Damian Flannery. *Integrate ZXing barcode scanner into your Android app natively using Eclipse.* Damian Flannery's Blog, junio de 2011 [fecha de consulta: septiembre de 2012]. Disponible en <http://damianflannery.wordpress.com/2011/06/13/integrate-zxing-barcode-scanner-into-your-android-app-natively-using-eclipse/>

Firmado: ***Juan Ávila López del Castillo***