Ingeniería Superior en Informática
Sistemas Informáticos

Aplicación para el día a día del entrenador deportivo

versión Android

Presentado por:

Pascual Bayo Esteller

Dirigido por:

Raúl Montoliu Colás

Castellón de la Plana, 16 de Junio de 2011

Resumen

El presente proyecto consiste en el desarrollo de una aplicación que será utilizada en teléfonos móviles táctiles, cuya finalidad es implementar un gestor de entrenamientos que facilite al usuario (desde el más aficionado hasta el entrenador deportivo) la tarea de planificar una sesión de ejercicios. Para ello se dispone de una tabla dividida en filas en las que se introducen los ejercicios deseados y su duración, pudiendo posteriormente ejecutar estos ejercicios con la ayuda de un cronómetro y otros elementos que incorpora la aplicación.

La aplicación ha sido implementada para el sistema operativo Android y se ha tenido en cuenta en todo momento la tecnología de pantalla táctil que incorporan los dispositivos móviles de última generación, lo cual agiliza tanto la introducción de los ejercicios como el seguimiento en la fase de ejecución de la sesión. Además, puesto que va a ser utilizada durante un entrenamiento, se ha diseñado para ser sencilla de usar y evitar que el usuario deba centrarse en ella más que en la propia actividad.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

Hoy en día, para muchas personas el deporte es una forma de vivir. El hecho de ser deportista profesional implica unas capacidades que permitan competir con otros deportistas y tener siempre presente el espíritu de superación. Y una buena forma de afrontar esto es planificar correctamente los métodos de trabajo y los entrenamientos. Por otra parte, vivimos en una sociedad en la que las nuevas tecnologías están a la orden del día, por lo que contamos con un amplio abanico de herramientas electrónicas que proporcionan ayuda en una gran variedad de campos y situaciones.

Cabe mencionar que la técnica utilizada hoy en día por la mayoría de entrenadores deportivos para planificar los entrenamientos consiste en escribir en una libreta la lista de ejercicios o actividades a realizar. Pero como se puede suponer, esto implica una serie de inconvenientes entre los que se encuentran:

1. Es necesario tener en todo momento la libreta y el rotulador a mano por si se necesita corregir algun aspecto de la actividad, lo cual conlleva tener ambas manos ocupadas.

2. Si se utiliza un único color de rotulador para elaborar la lista de ejercicios, se debe buscar en la lista de actividades en qué punto nos encontramos de la sesión, lo cual conlleva desviar la atención de la propia actividad. Si, por el contrario se utilizan diferentes colores para identificar los ejercicios, esto implica tener en todo momento todos los rotuladores.

3. Además del propio ejercicio, es necesario un cronómetro para llevar el tiempo de ejecución de cada una de las actividades, lo cual ya son un mínimo de 3 elementos (libreta, rotulador y cronómetro) los que necesitamos tener a nuestro alcance.

Por tanto, la duda que puede aparecer en este punto es: ¿Por qué no utilizar la tecnología para la correcta gestión de los entrenamientos de forma que podamos modificarlos con, por ejemplo, un simple click, en lugar de tener que utilizar una libreta y un rotulador? O, visto de otro modo, ¿por qué no aprovechar las funcionalidades que nos proporcionan los teléfonos móviles de última generación con su tecnología MultiTouch[1] para poder utilizar dicho teléfono como ayuda en la elaboración de entrenamientos? Será esta última idea la base sobre la que se desarrollará el presente proyecto.

Teniendo en cuenta lo dicho hasta ahora, los problemas mencionados derivados de la libreta y rotulador se pueden resolver de manera automática con un dispositivo móvil. Por ejemplo, la misma aplicación nos puede avisar mediante una señal sonora/vibratoria de que ha finalizado un ejercicio, marcando éste de un color diferente al resto para distinguirlo a primera vista. Si además se incorpora un cronómetro a la aplicación, se consigue que los problemas (2) y (3) desaparezcan automáticamente.

Por otra parte, utilizando una sola mano se puede interactuar con la aplicación en todos los aspectos, tanto para escribir el nombre de un ejercicio, como para modificar su duración o pausar

[1] `http://en.wikipedia.org/wiki/Multi-touch`

el cronómetro, con lo que se resuelve el problema (1).

Con estas ideas se llega a la conclusión de que desarrollar un gestor de entrenamientos para teléfonos móviles sería una gran ayuda para aquellas personas que, practicando deporte, necesitan tener cerca algun método de planificación de la sesión. Y como hemos visto, utilizar una libreta y un rotulador no es siempre la mejor opción.

Esta aplicación se desarrollará para el sistema operativo Android, por las numerosas ventajas que presenta, entre las que se encuentran:

- Android es un sistema operativo muy sencillo de instalar y con un gran potencial, por lo que cada día está presente en más elementos tecnológicos, ya sean teléfonos móviles, portátiles, marcos digitales o navegadores GPS.

- Android es de código abierto, lo cual tiene la ventaja de poder acceder al código de una aplicación para mejorarla o ampliar su funcionalidad. Otros sistemas operativos están restringidos a sus fabricantes, con lo que es más difícil depurar las aplicaciones.

- Al ser de código abierto existe una inmensa comunidad de diseñadores y desarrolladores que lo utilizan por todo el mundo, siendo de esta forma mucho más sencillo compartir soluciones o ideas o resolver dudas con las aplicaciones.

- Además, el hecho de ser de código abierto significa que no depende de fabricantes o proveedores, pudiéndolo instalar en cualquier operadora y liberando al usuario de tener que contratar una determinada operadora para poder utilizar su sistema operativo.

- Android es capaz de gestionar inteligentemente aplicaciones multitarea dejando en suspensión las que no se utilizan o dando más recursos a las que los necesitan para una correcta gestión de los recursos del dispositivo.

- Al existir una gran comunidad de usuarios de Android, hay muchas más aplicaciones implementadas, lo que permite el aprovechamiento de código eficiente para realizar funciones en nuestras aplicaciones.

Estas son las principales razones por las que se ha elegido Android como sistema operativo sobre el que se ejecutará la aplicación desarrollada.

1.2. ¿Por qué un gestor de entrenamientos?

Como ya se ha comentado, es de vital importancia una gestión correcta de los entrenamientos en el deporte, y hemos visto que utilizar libreta y rotulador no es la mejor forma de planificarlos. Además, donde mayor inconvenientes presenta este sistema es en la parte práctica de la sesión de ejercicios, en la que se debe estar concentrado en la actividad a realizar en lugar de buscar un ejercicio en una lista. Por tanto se llega a la conclusión de que una aplicación que permita hacer lo mismo que la libreta y el rotulador, pero de forma más automática y sencilla, mejoraría la forma de preparar las sesión de ejercicios. Es por esto que se decidió crear un sistema capaz de planificar y llevar a cabo la ejecución de las diferentes actividades que conforman una sesión de ejercicios.

1.3. Conceptos clave

Se define a continuación la terminología utilizada en la presente memoria para evitar ambigüedades en los siguientes términos:

- **Ejercicio / actividad**: cada uno de los diferentes esfuerzos físicos de los que consta un entrenamiento, como pueden ser correr, saltar, pivotar, etc.
- **Sesión de ejercicios**: conjunto de ejercicios incluidos en una misma planificación listos para ser ejecutados.
- **Gestor de entrenamientos**: un gestor de entrenamientos es la aplicación encargada de facilitar la tarea de planificar y ejecutar una sesión de ejercicios.
- **Duración**: la duración será el tiempo de ejecución asignado a cada ejercicio.
- **Sistema operativo**: conjunto de programas incorporados al teléfono móvil que permiten su uso de forma cómoda por parte del usuario.
- **Android**: sistema operativo que provee un conjunto de métodos y procedimientos utilizados en el presente proyecto.
- **Java**: lenguaje de programación orientado a objetos utilizado en este proyecto para, mediante las funciones incorporadas en Android, desarrollar la aplicación.
- **Usabilidad**: facilidad con la que se puede utilizar una herramienta u objeto con el fin de alcanzar un objetivo concreto. Este término estara presente durante todo el proyecto para recalcar que la aplicación debe ser sencilla de utilizar por el usuario final.
- **Aplicación instalable**: una aplicación instalable es un programa que puede ser utilizado en un dispositivo.
- **Thread**: aplicación ejecutada en segundo plano que interactúa con una aplicación principal.

1.4. Ámbito del proyecto

Este proyecto va dirigido a todas aquellas personas, deportistas profesionales o no, que quieran aprovechar las capacidades de su teléfono móvil con pantalla táctil para crear de forma cómoda sesiones de ejercicios. Además, no sólo permitirá crear la planificación de éstas, sino que el usuario podrá llevar a cabo dichos ejercicios ayudándose de la información visual y sonora que ofrece la aplicación.

Puesto que la aplicación está enfocada al ámbito deportivo, cualquier persona podrá utilizarla aunque no tenga conocimientos informáticos, ya que se ha desarrollado teniendo como principales objetivos la sencillez de uso y la máxima usabilidad posible.

Por otra parte, aunque este proyecto explicará la aplicación con ejemplos relacionados con el baloncesto, ésta puede ser utilizada para cualquier deporte en el que se requiera crear y ejecutar fácilmente una sesión de ejercicios. Además, la forma en que se ha diseñado hace posible que con pequeños cambios, pueda ser utilizada con otros fines, como por ejemplo crear listados de tareas y el tiempo previsto para su desarrollo.

1.5. Objetivos

El objetivo de este proyecto es el de desarrollar una aplicación para teléfonos móviles que incorporen el sistema operativo Android[2] que permita de forma sencilla y, lo mejor de todo, táctil, definir y ejecutar una sesión de ejercicios especificando la duración de cada uno.

Los principales objetivos del proyecto son los siguientes:

1. Realizar una aplicación que funcione como un gestor de entrenamientos, la cual debe permitir añadir ejercicios, modificar sus duraciones y ejecutarlos.

2. **Fácil de usar**: Posiblemente el principal objetivo del diseño de la aplicación. Se ha desarrollado de forma que pueda ser utilizada por cualquier persona en cualquier deporte. Por esto se ha estudiado meticulosamente el diseño de la interfaz para que la interacción con ella sea lo más intuitiva posible.

3. **Fácil de instalar**: Puesto que se ha desarrollado para cualquier tipo de usuario, no se deben requerir conocimientos avanzados para instalarla en el teléfono. En el apéndice 2.3 se explican los dos métodos existentes para poder utilizar la aplicación en el teléfono móvil.

4. **Escalable**: La aplicación debe ser escalable en cuanto a funcionalidades se refiere. Para esto se ha estructurado el código en funciones para facilitar la identificación de las distintas partes de la implementación.

5. **Aprender Android**: Un objetivo derivado de los anteriores es aprender las técnicas de programación de aplicaciones para Android, aprovechando sus capacidades para el desarrollo de aplicaciones táctiles.

6. **Poner en práctica lo aprendido en la carrera**: tras varios años cursando la carrera de Ingeniería Superior en Informática, este ha sido un buen proyecto para poner en práctica una gran cantidad de conceptos aprendidos en el transcurso de los años, como son las ideas generales de programación en Java o la programación estructurada, entre otros.

1.6. Herramientas utilizadas

- Microsoft Windows 7 [3]
- Eclipse [4]
- Software Development Kit (SKD) de Android [5]
- TexWorks [6]
- CmapTools [7]
- LaTeX [8]

[2] http://es.wikipedia.org/wiki/Android
[3] http://windows.microsoft.com/es-ES/windows7/products/home?os=win7
[4] http://www.eclipse.org/
[5] http://developer.android.com/sdk
[6] http://www.tug.org/texworks/
[7] http://cmap.ihmc.us/
[8] http://www.latex-project.org/

Para el desarrollo de este proyecto se ha utilizado el sistema operativo Microsoft Windows 7, en el que se han instalado las siguientes aplicaciones:

1. Eclipse: herramienta utilizada para toda la parte de programación debido a las innumerables ayudas que proporciona en la generación de código de manera rápida e intuitiva, además de incorporar un plugin para poder desarrollar aplicaciones para Android permitiendo su prueba mediante un emulador.

2. SDK de Android: puesto que se desarrolla una aplicación para el sistema operativo Android, es necesario utilizar su SDK para tener acceso a las funciones requeridas.

3. TexWorks: programa basado en LaTeX utilizado para la realización de esta memoria y elegido por sus ventajas para programar en este lenguaje. Este programa facilita la programación en LaTeX remarcando la sintaxis especial y mostrando los puntos en los que hay errores al compilar el documento para crear el archivo *pdf*.

4. CmapTools: herramienta cuya función es la creación de diagramas de flujo. Su gran ventaja es la facilidad de uso y la calidad de los diseños, además de estar disponible en versión gratuita. También permite exportar los diagramas a imagen y ofrece numerosas opciones para diseñar diagramas de todos los estilos.

5. LaTeX: posiblemente se trate de una de las mejores herramientas para la realización de documentos profesionales como memorias, informes u otros estilos. Uno de sus puntos fuertes es la representación de ecuaciones, aunque no se utilizará con ese fin en el presente documento.

1.7. Vista previa

En este punto se presenta la interfaz de la aplicación una vez terminado su desarrollo. Se comenta cada pantalla mediante una pequeña descripción de su cometido.

La figura 1.1 muestra la pantalla desde la que se accede a cada una de las dos secciones de las que consta la aplicación.

La pulsación de *Crear sesión* nos lleva a la figura 1.2, donde se introducen los ejercicios a realizar durante la sesión y se especifica su duración. Una vez se completa la sesión de ejercicios, se guarda utilizando el botón destinado a ello.

Finalmente, la figura 1.3 muestra la aplicación en el estado de *Ejecución*, donde el usuario se dispone a ejecutar los ejercicios introducidos. Como se puede apreciar, esta ventana consta de un cronómetro que permite al usuario saber en todo momento el tiempo restante del ejercicio en proceso.

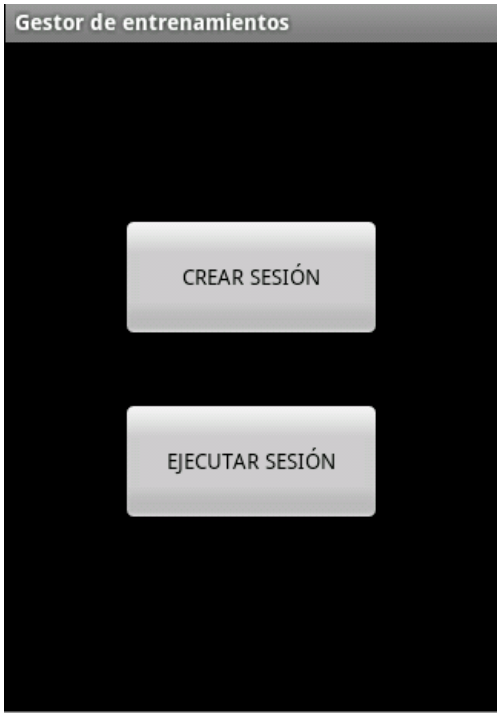

Figura 1.1.: Vista previa: Pantalla de planificación

Figura 1.2.: Vista previa: Pantalla de planificación

Figura 1.3.: Vista previa: Pantalla de ejecución

1.8. Organización de este documento

La presente memoria se divide en los siguientes apartados:

1. Resumen: breve introducción al contenido de esa memoria para que el lector se haga una idea del tema que se va a tratar.
2. Introducción: en este apartado se describen todos los elementos que han permitido poner en marcha el proyecto, tales como conceptos clave, herramientas utilizadas, objetivos del proyectos, y demás aspectos necesarios para asentar las bases de lo que sera este proyecto.
3. Programación en Android: puesto que el proyecto se ha desarrollado para móviles con sistema operativo Android, se ha incluido este apartado para explicar la forma de empezar, desde cero, a crear aplicaciones para Android.
4. Planificación: debido a que ha sido un proyecto que ha llevado una gran carga de trabajo, ha sido necesarios establecer una planificación en las tareas para llevar un seguimiento temporal del proyecto.
5. Desarrollo: en este apartado se explica el diseño y la implementación de las diferentes versiones que se han desrrollado.
6. Conclusiones y trabajo futuro: en esta sección se explica la consecución de los objetivos planteados al principio una vez la aplicación estaba finalizada y una serie de mejoras que se pueden llevar a cabo para dotar de más funcionalidades a la aplicación.
7. Apéndice A (Creación de un certificado de firma): en este apéndice se explica como crear un certificado para firmar nuestras aplicaciones Android, necesario para poder crear, a partir del código fuente, un ejecutable vinculado a nosotros como creadores.

8. Apéndice B (Guía de uso): manual de la aplicación donde se describe su funcionamiento y los elementos que conforman cada una de las pantallas.

9. Apéndices C al G: este conjunto de apéndices alberga todo el código fuente que ha sido necesario para la implementación de la aplicación en su versión final.

2. Programación en Android

2.1. ¿Qué necesito?

Una vez mencionados los elementos necesarios, vamos a ver como utilizarlos para poder empezar a programar.

Como punto de partida, hay que enlazar Eclipse con el SDK de Android para poder utilizar sus funciones y poder emular un simulador desde el mismo programa, lo cual es de gran ayuda para probar la aplicación cuando no se dispone de un teléfono móvil con dicho sistema operativo. Para ello se van a describir los pasos a seguir para descargar el plugin de Eclipse.

En primer lugar, puesto que el SDK se compone de funciones escritas en Java, para que funcione correctamente sera necesario instalar la última versión del Java Development Kit (JDK) [1].

Una vez instalado tanto Eclipse como el JDK, hay que descargar desde la web `http://developer.android.com/sdk` la versión para windows del SDK de Android. Existen dos versiones, una en archivo comprimido y otra como autoinstalable. Es recomendable descargar la segunda porque así será el mismo SDK quien buscará, en tiempo de instalación, las rutas del JDK y no habrá que modificar nada manualmente. Además, esta versión autoinstalable se encargará de mostrarnos al final de la instalación un conjunto de plataformas y componentes necesarios para poder utilizar el SDK. Bastará con darle a *Install* para instalarlos todos.

A continuación necesitamos el Android Development Tools, un plugin desarrollado para Eclipse que le proporciona un potente entorno de trabajo para desarrollar aplicaciones para Android. Para ello basta seguir los pasos que se comentan en el enlace `http://developer.android.com/sdk/eclipse-adt.html#installing`

El último punto es crear un dispositivo virtual (Android Virtual Device) que será el emulador en el que se puede probar la aplicación ejecutada desde Eclipse. En el enlace `http://developer.android.com/resources/tutorials/hello-world.html` se explica paso a paso cómo crear el AVD así como un pequeño tutorial para crear nuestra primera aplicación y comprobar que todos los pasos los hemos llevado a cabo correctamente.

En resumen, los pasos necesarios para empezar a programar una aplicación en Android son:

1. Descargar el Java Development Kit e instalarlo en el equipo en el que se vaya a desarrollar la aplicación.
2. Descargar e instalar el programa Eclipse.
3. Descargar e instalar el SDK de Android (preferiblemente, la version autoinstalable).
4. Seguir los pasos que se especifican en `http://developer.android.com/sdk/eclipse-adt.html#installing` para descargar e instalar el AVD, que nos permitirá ejecutar el emulador.

[1] `http://www.oracle.com/technetwork/java/javase/downloads`

5. A continuación, crear el AVD tal y como se especifica en el enlace `http://developer.android.com/resources/tutorials/hello-world.html`.

6. Finalmente ya estamos en condiciones de empezar a programar. En el enlace del paso 5 hay un tutorial para realizar una pequeña aplicación de prueba.

2.2. ¿Cómo se programa?

Una vez tenemos todos los elementos necesarios instalados, podemos empezar a programar. En primer lugar, si es la primera vez que ejecutamos el programa Eclipse, nos preguntará donde queremos crear nuestro directorio de trabajo: esto es la ruta en la que guardará todos los proyectos que realicemos con dicha herramienta. Se le indica la ruta y acto seguido carga el programa.

El primer paso es crear un proyecto Android, bajo el cual se creará de forma automática toda la jerarquía de clases y carpetas en las que se almacenarán los ficheros necesarios para poder ejecutarlo. De esta forma, el primer paso será ir a *File - New - Project*.

Nos saldrá una ventana para que indiquemos qué tipo de proyecto queremos utilizar, por lo que, si hemos seguido los pasos del apartado 2.1, deberemos tener una carpeta llamada Android dentro de la cual localizaremos *Android Project* y pulsaremos en *Siguiente*, con lo que se nos mostrará una ventana similar a la figura 2.1.

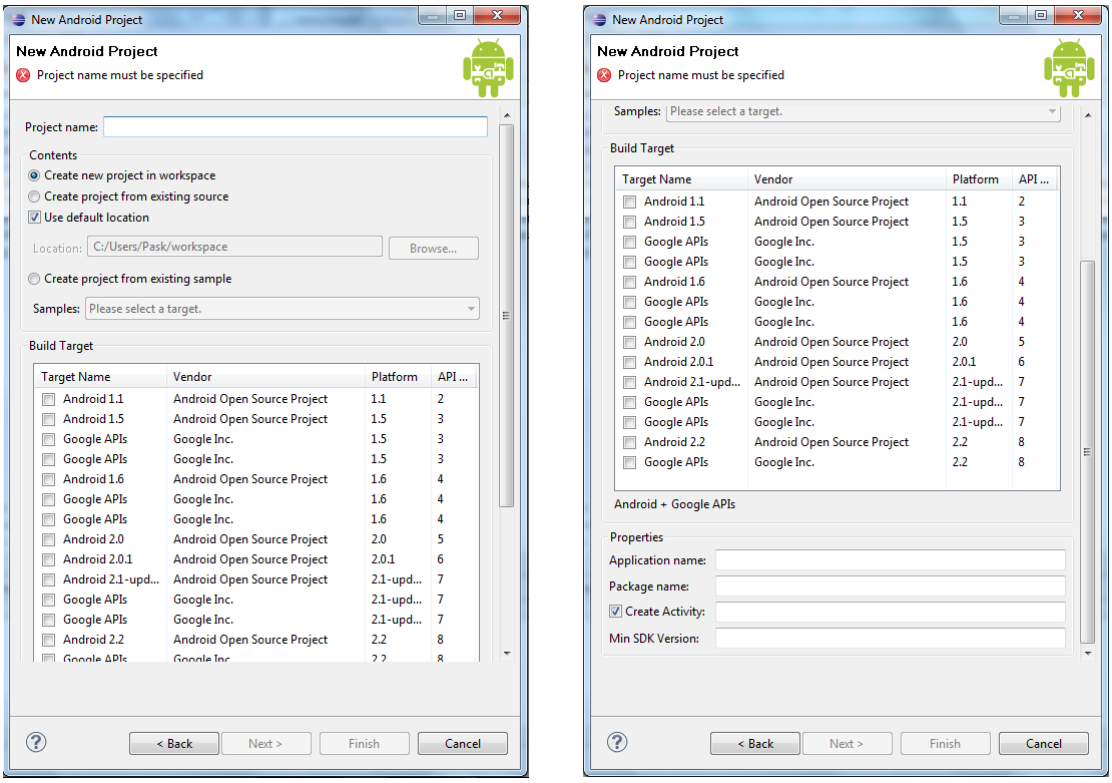

Figura 2.1.: Especificación de opciones del proyecto

Los principales elementos de la imagen 2.1 son los siguientes:

- *Project Name*: es el nombre del directorio que contendrá los archivos del proyecto.

- *Application Name*: este nombre será el que aparecerá en el teléfono móvil cuando ejecutemos la aplicación.
- *Package Name*: este campo se refiere al espacio de nombres. Todos los archivos que implementemos estarán contenidos dentro de la carpeta *src* en nuestro proyecto. En este campo tenemos que poner la jerarquía de nombres bajo la que queremos ubicar dichos archivos. Por ejemplo, utilizando el espacio de nombres "com.es.helloWorld", en la carpeta de nuestro proyecto tendremos, además de otras carpetas, la siguiente jerarquía: src/com/es/helloWorld.java.
- *Build Target*: En esta lista deberemos seleccionar el dispositivo AVD que hemos creado en el último paso del apartado 2.1
- *Create Activity*: marcando esta opción le estamos indicando que queremos que sea el programa quien nos cree la actividad básica para empezar a programar. Nótese que será útil que el nombre que le indiquemos a la actividad no contenga espacios, para así evitar posibles problemas que nos puedan surgir posteriormente al incluir paquetes en dicha actividad.

Para comprender mejor los conceptos anteriores, se muestra en la figura 2.2 un ejemplo de la estructura que generarían los siguientes valores:

- Project Name: HelloAndroid
- Package Name: com.example.helloandroid
- Create Activity: helloWorld

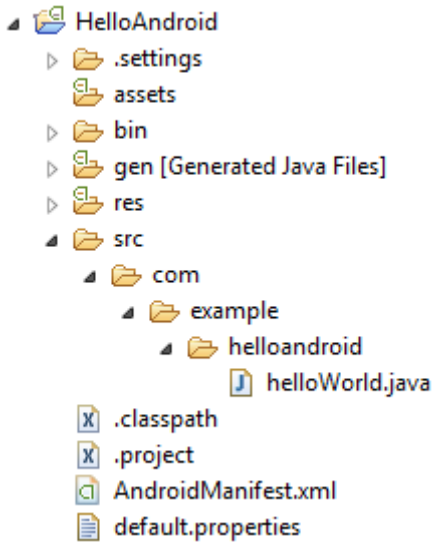

Figura 2.2.: Estructura de directorios creadas en un proyecto

Una vez rellenados estos campos, al pulsar en *Finish* se abrirá la ventana inicial de nuestro proyecto con la jerarquía de directorios indicada. Si se hace doble click sobre helloWorld.java veremos las instrucciones iniciales básicas para poder ejecutarla en el emulador, pero es un buen punto de partida para comprobar que realmente estamos en disposición de empezar a implementar código sobre una versión estable. Se puede ver el código generado automáticamente

haciendo doble click en el archivo helloWorld.java del navegador de Eclipse, mostrado en la figura 2.3

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class helloWorld extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /* Código de la aplicación */


    }
}
```

Figura 2.3.: Código inicial creado por la aplicación

Se explican a continuación el contenido de los diferentes ficheros que componen un proyecto Android:

- **Android Manifest**: En este fichero se declaran todas las especificaciones de nuestra aplicación. Todas las ventanas que formen nuestra aplicación deberán estar aquí declaradas, indicando para cada una de ellas la siguiente información básica (se puede encontrar una descripción completa de este fichero en la dirección `http://developer.android.com/guide/topics/manifest/manifest-intro.html`.
  1. Nombre del fichero que contiene la actividad. Este nombre sirve como un identificador único de la aplicación.
  2. Qué procesos contendrán los componentes de la aplicación.
  3. Qué permisos tiene la aplicación para acceder a partes protegidas del API e interactuar con otras aplicaciones.
  4. Título de la actividad (texto que aparece en la parte superior de la ventana al ser ejecutada).
  5. Configuración de cambios de orientación de pantalla: se puede indicar que la aplicación no cambie su orientación si el teléfono pasa de vertical a horizontal o viceversa.
  6. Temas de estilos personalizados que se pueden utilizar en diferentes aplicaciones.

  En resumen, lo que hay que recordar al desarrollar una aplicación es incluir los permisos de ejecución (acceso a Internet, a la cámara, etc.) y declarar todos los componentes de la aplicación (Activity, Service, etc.).
- **XML**: en este fichero se define la estructura de cada actividad. Es el homólogo a los CSS del lenguaje HTML. Cada actividad debe tener su propio XML con el diseño de su interfaz. Además del diseño, se pueden definir atributos como la posición, forma, color, nombre,

eventos, etc.

Aunque se puede crear la interfaz sin usar el XML haciendo uso de las clases y métodos que crean los elementos, es aconsejable utilizar el XML para facilitar la depuración del código y evitar mezclar las partes que implementan elementos con las partes que modifican el aspecto de éstos.

Existen dos atributos que deben tener todos los elementos que se creen en el xml, y son *android:layout_width* y *android:layout_height*, mediante los cuales se especifica el ancho que tendrá el elemento. Estos atributos aceptan tres posibles valores:

- *fill_parent*: el elemento ocupará el mismo ancho que su elemento padre. Esto es: si el elemento padre ocupa toda la pantalla, aunque este elemento no contenga texto, también ocupará toda la pantalla.
- *wrap_content*: el elemento ocupa el ancho necesario para albergar su texto contenido, sin importar las medidas del padre. Es decir, un elemento que contenga una cadena de 10 caracteres ocupará el doble de ancho en pantalla que el mismo elemento con una cadena de 5 caracteres.
- La medida exacta que queremos que ocupe, que puede estar expresada en las siguientes unidades: *píxeles (px)*, *pulgadas (in)*, *milímetros (mm)*, *puntos (pt)*, *píxeles independientes de densidad (dp)* y *píxeles independientes de escala (sp)*.

- **Código**: este es el fichero *.java*, la parte en la que se programa nuestra aplicación, con el conjunto de clases que, compiladas, ejecutan el proyecto.

- **Drawables**: aquí van las imágenes utilizadas en nuestra aplicación. Se dividen en 3 tipos dependiendo de la resolución de la pantalla (en dpi [2]) de los dispositivos.
  - *hdpi*: High dpi. Imágenes de mayor calidad para poder ser mostradas en teléfonos móviles con una resolución de pantalla alta.
  - *ldpi*: Low dpi. Imágenes con una calidad baja que se utilizarán en los teléfonos con resoluciones inferiores.
  - *mdpi*: Medium dpi. Imágenes con una calidad media para ser utilizadas en una resolución estándar en los móviles.

  Las imágenes deben estar en las 3 carpetas con el mismo nombre para facilitar su reconocimiento al sistema operativo. Por ejemplo, si tenemos la imagen *vacaciones.png* en ldpi y en mdpi y se intenta mostrar en un dispositivo a la mayor resolución, en primer lugar buscará la imagen en la carpeta hdpi, porque es la que se corresponde con la resolución del teléfono. Como no se encuentra en esa carpeta, el sistema operativo bajará un nivel de detalle, mostrando en pantalla la imagen de la carpeta mdpi, pero a la mayor resolución, lo cual puede provocar un pixelado que distorsione la imagen. Por eso es importante tener cada imagen en las 3 carpetas con sus apropiadas resoluciones, para que no haya distorsiones derivadas del pixelado.

- **Raw**: Aquí van todos los demás elementos que no sean imágenes, como clips de video, sonidos, documentos, etc, que podrán ser enlazados posteriormente desde el código de nuestra aplicación.

[2] `http://en.wikipedia.org/wiki/Dots_per_inch`

Una vez implementada la aplicación, llega el momento de exportarla a un archivo *.apk*, que será el que reconocerán los teléfonos móviles Android y podrán instalar. Para generar este fichero *.apk*, hay que exportar el proyecto a un archivo que deberá ser firmado para identificarnos de manera única con la aplicación. Vamos a ver estos pasos detalladamente:

1. El primer paso será la creación del certificado de firma, que está explicada detalladamente en el apéndice A.

2. Llegados a este punto, tenemos el proyecto en Eclipse y un certificado propio para poder firmarlo. Para exportar el fichero, deberemos ir a *File - Export*. En la ventana que se abre, pulsamos en Android y a continuación en *Export Android Application*, llegando a la figura 2.4

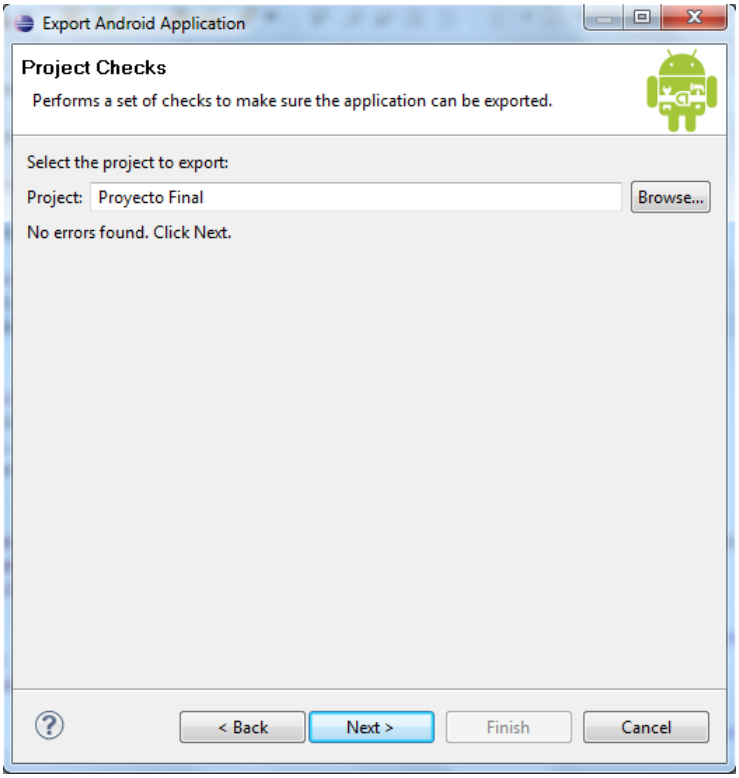

Figura 2.4.: Exportar el proyecto: paso 1

3. Como se muestra en la imagen 2.4, nos pregunta por el nombre del proyecto que queremos exportar. Tras escribir su nombre exacto, pulsar en Next.

4. Llegados a este punto es cuando debemos firmar la aplicación (figura 2.5). Para ello pulsamos en *Browse* para acceder al certificado creado en el primer punto. Introducimos la contraseña del almacén de llaves que hemos indicado en el proceso de creación y pulsamos en Next.

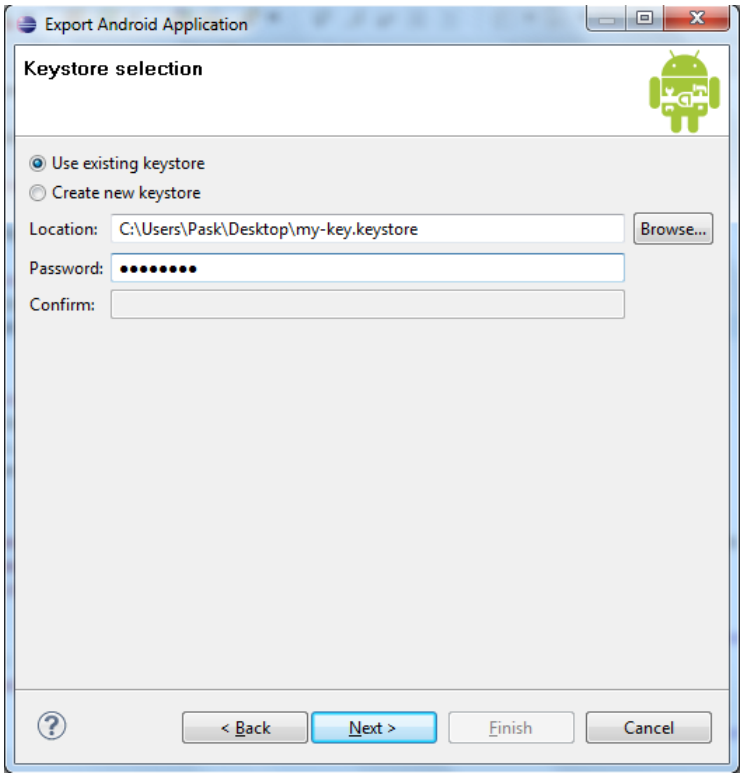

Figura 2.5.: Exportar el proyecto: paso 2

5. Como se puede apreciar en la imagen 2.6, el siguiente paso es escoger el nombre con el que queremos firmar el proyecto. Basta con seleccionarlo del desplegable e introducir la contraseña utilizada para su creación. Si en el repositorio de certificados tenemos más de uno, en esta lista aparecerá el nombre de todos los certificados creados. Una vez elegido el que nos interesa para firmar la aplicación, deberemos utilizar la contraseña para crearlo, no la del repositorio.

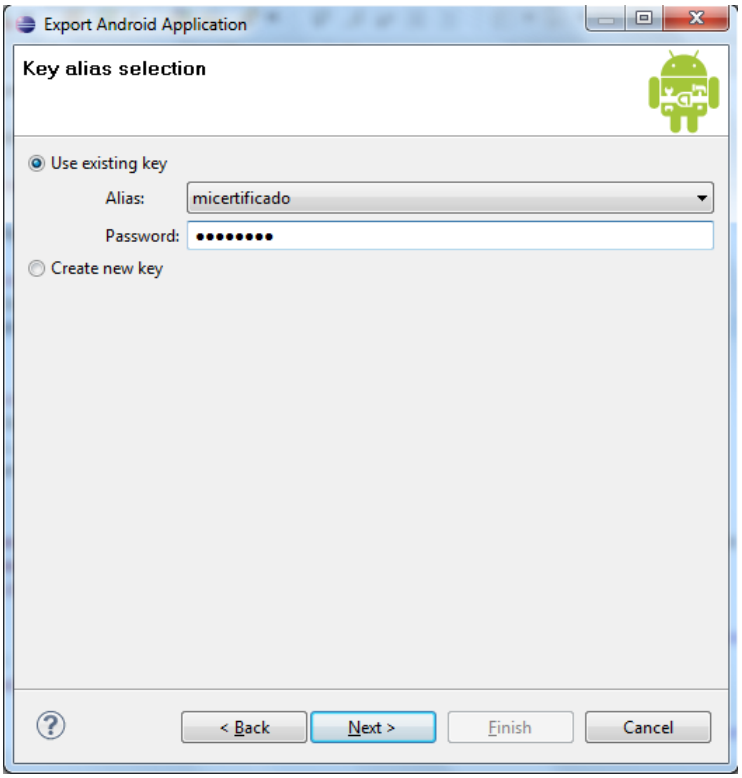

Figura 2.6.: Exportar el proyecto: paso 3

6. En este último paso elegimos el directorio donde se exportará el proyecto y, si no ha habido ningún error, el programa nos muestra el fichero que se creará y la fecha de expiración del certificado, la cual dependerá del valor indicado en el parámetro *validity* en el momento de crearlo. Estos datos se pueden apreciar en la figura 2.7

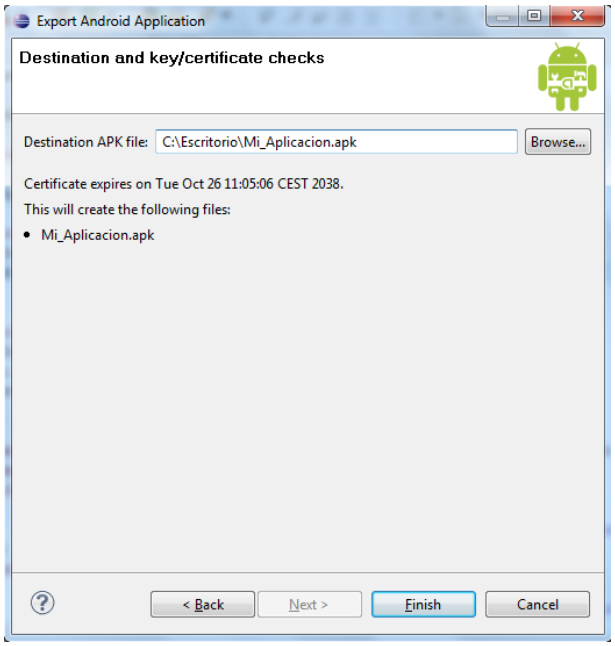

Figura 2.7.: Exportar el proyecto: paso 4

7. Finalmente, tras pulsar *Finish* se generará, en el directorio especificado, el archivo *.apk* que estará listo para ser instalado en cualquier teléfono móvil que incorpore el sistema operativo Android.

2.3. ¿Cómo se instala esta aplicación en el teléfono Android?

La forma de instalar la aplicación desarrollada en el teléfono móvil es idéntica a la del resto de aplicaciones. En este apartado se describirá la forma de instalar la aplicación desarrollada, aunque los pasos son extrapolables a cualquier otra aplicación.

En primer lugar habrá que asumir que tenemos el fichero *.apk* correctamente creado, ya sea subido en un servidor online o en nuestro propio ordenador.

Existen dos formas de instalarla en el dispositivo. Es importante tener presente que, como la aplicación no proviene del Android Market, es necesario activar los orígenes desconocidos en el teléfono para que reconozca otras aplicaciones. Esta opción aparece en la ruta *Ajustes - Aplicaciones - Orígenes desconocidos*. Vamos a ver a continuación, para esta aplicación en concreto, los pasos a seguir para instalarla en el teléfono:

1. Acceso online: consiste en acceder a la url `http://www4.uji.es/~al066469/android` mediante el navegador web del teléfono móvil y pulsar encima de la imagen, lo cual descargará la aplicación en el teléfono y mediante el navegador de archivos se podrá seleccionar para instalarla automáticamente. Es interesante remarcar que, a pesar de ser una aplicación totalmente funcional, con diferentes ventanas y eventos, se ha conseguido que, una vez instalada, ocupe 524 KB aproximadamente, lo cual facilita su descarga online y hace que los tiempos de instalación y ejecución sean cortos en comparación con otras aplicaciones.

2. Acceso local: si tenemos la aplicación en nuestro ordenador, lo primero que hay que hacer es conectar el teléfono a éste. De esta forma podremos transferir el fichero a la tarjeta de memoria del dispositivo para, una vez finalizada la transferencia, utilizar cualquier administrador de archivos que proporcione el teléfono móvil para seleccionar el archivo que acabamos de transferir y comenzar con su instalación.

Una vez instalada aparecerá como una aplicación más en el menu del teléfono, pudiendo seleccionarla y ejecutarla.

Si no disponemos de un teléfono móvil con android, podemos utilizar el emulador de Eclipse, el cual hará las funciones de teléfono virtual para poder probar nuestro proyecto. Para ello, tras pulsar *Ctrl + F11*, o ir a *Project - Build Project* para lanzar el compilador, primero cargará el emulador, y una vez aparezca la *Home*, instalará la aplicación. Aunque este proceso le puede llevar algún tiempo, una vez esté cargado el emulador las sucesivas veces que compilemos el proyecto únicamente instalará la aplicación, por lo que finalizará en unos pocos segundos.

3. Planificación

Con el fin de llevar un seguimiento de la duración del proyecto, se elaboró una tabla con las diferentes tareas de las que consta, junto con una breve descripción de su cometido. Acto seguido se presentará la planificación prevista y real de estas tareas, explicando, si fuese necesario, las diferencias notables en los tiempos de ejecución de cada una de ellas.

A continuación se presentan los días festivos (excluyendo fines de semana) y las fechas en las que, por un motivo justificado, no se pudo avanzar el proyecto:

- 1 de noviembre del 2010: Día de todos los Santos.
- 23 de diciembre del 2010 al 6 de enero del 2011: vacaciones de Navidad.
- 7 de enero al 27 de enero del 2011: ausencia del tutor del proyecto por viaje.
- 27 de enero al 8 de febrero del 2011: ausencia del tutor por enfermedad.
- 28 de marzo al 3 de abril del 2011: fiestas de la Magdalena.
- 21 de abril al 25 de abril del 2011: fiestas de Pascua.

3.1. Identificación de las tareas

Tras el estudio de los requisitos que debía cumplir el proyecto, se decidieron las tareas que reflejasen de forma precisa el proceso de desarrollo del proyecto. Así, la estructura básica que se definió para cada versión del proyecto fue la siguiente:

1. Requisitos
2. Diseño
3. Implementación
4. Pruebas

Con esto, se obtuvieron las tareas mostradas en la tabla 3.1.

Tabla 3.1.: Tareas en la realización del proyecto

Fase	**Tareas**
Planificación	Definición de objetivos Especificación de las tareas Instalación de las herramientas a utilizar Familiarización con el entorno de trabajo
Versión 1	Especificación de requisitos de la versión 1 Diseño de la interfaz Implementación Pruebas
Versión 2	Especificación de requisitos de la versión 2 Rediseño de la interfaz Estudio de las transiciones entre ventanas Implementación Pruebas
Conclusiones	Instalación y pruebas en un dispositivo real Estudio de posibles mejoras
Documentación	Elaborar la guía de uso Realización de la memoria final Elaborar la presentación del proyecto

Se explica a continuación el objetivo de cada una de las tareas:

- *Definición de objetivos*: En esta fase, tras dos reuniones con el tutor, se acordaron los objetivos que se pretendían alcanzar con la realización del presente proyecto, así como características que debía cumplir la aplicación en su versión final.

- *Especificación de las tareas*: A partir de los objetivos obtenidos en el punto anterior se acordaron las tareas en las que había que dividir el proyecto para su correcta gestión.

- *Instalación de las herramientas a utilizar*: Adquisición de los elementos necesarios para empezar a desarrollar el proyecto.

- *Familiarización con el entorno de trabajo*: Una vez instaladas las herramientas que se iban a utilizar, fue necesario un proceso de aprendizaje para tener una idea de cómo se programa una aplicación en Android. No sólo bastaba con realizar una pequeña aplicación que mostrase un mensaje por pantalla, sino que había que comprender perfectamente el funcionamiento y estructura de directorios creados puesto que el proyecto iba a ser considerablemente complicado para alguien que nunca antes había programado ni en Java ni para Android.

- *Especificación de requisitos de la versión 1*: Para la primera versión se definieron unos requisitos básicos que nos permitiesen desarrollar una primera aproximación de lo que estábamos buscando.

- *Diseño de la interfaz*: Este fue un punto fuerte ya que no bastó con una interfaz mediante iconos o botones, sino que debido al tamaño limitado de las pantallas de los teléfonos móviles, había que buscar la mejor estructuración de elementos para aprovechar al máximo las dimensiones de la pantalla del dispositivo.

- *Implementación*: A partir del diseño de la interfaz, en esta tarea se creó todo el código que genera primea versión de la aplicación.
- *Pruebas*: Como trabajo obligatorio, había que cerciorarse de que el código escrito hacía su función perfectamente, sin ningún tipo de problemas. Y para eso se realizaron pruebas en el emulador de Eclipse.
- *Especificación de requisitos de la versión 2*: A partir de la primera versión se pudo comprobar que era necesario ampliar considerablemente la funcionalidad de la aplicación para poder utilizarla en una sesión de ejercicios, por lo que se establecieron los requisitos que debía cumplir la segunda versión del proyecto.
- *Rediseño de la interfaz*: Se comprobó que algunos elementos de la interfaz no eran necesarios y se podía prescindir de ellos, con lo que su eliminación de la pantalla acarreaba un mayor espacio aprovechable de la misma. De este modo, se diseñó una nueva estructura de la interfaz para la segunda versión.
- *Estudio de las transiciones entre ventanas*: Puesto que en esta segunda versión iban a haber dos partes claramente diferenciadas (planificación de la sesión y ejecución de la sesión), había que estudiar la forma de interactuar entre ellas, el paso de mensaje de una a otra, etc.
- *Implementación*: Cuando quedó clara la transición entre ventanas y el diseño de la nueva interfaz se procedió a la generación del código necesario.
- *Pruebas*: Al igual que en la versión 1 se definieron unas pruebas para comprobar que el código implementado funcionaba como se esperaba.
- *Instalación y pruebas en un dispositivo real*: Tras probar la segunda versión de la aplicación en el emulador y comprobar que no había errores fuera de control, se probó la aplicación en un teléfono móvil que será donde se ejecutará finalmente.
- *Estudio de posibles mejoras*: Con el proyecto finalizado se estudiaron las posibles mejoras y ampliaciones que se podrían implementar para dotar de una mayor funcionalidad y robustez a la aplicación.
- *Elaborar la guía de uso*: Puesto que el proyecto va dirigido a todo el público en general, era necesario elaborar una guía de uso que ayudase al usuario a entender y aprovechar al máximo las posibilidades de la aplicación.
- *Realización de la memoria final*: Con el objetivo de documentar todo el proyecto, tanto el código como la planificación de las tareas o la guía de uso, se elaboró la presente memoria.
- *Elaborar la presentación del proyecto*: Para dar por terminado el proyecto, se elaboraron una serie de diapositivas en las que se describía el proceso de desarrollo y características del proyecto para ser explicados en la presentación final.

3.2. Seguimiento del proyecto

3.2.1. Planificación prevista

A partir de la figura 3.1 vemos que la fecha de inicio del proyecto fue el 28 de Octubre del 2010, y la previsión de finalizarlo era el 9 de Mayo del 2011. De esta forma, para completar las 300 horas estipuladas para su realización se necesitaría un ritmo de trabajo de unas 4 horas diarias. A continuación se muestra esta planificación desglosada por tareas:

Task Name	Duration	Start	Finish
Proyecto	**87 days**	**Thu 28/10/10**	**Mon 09/05/11**
Planificación	**12 days**	**Thu 28/10/10**	**Wed 17/11/10**
Definición de objetivos	2 days	Thu 28/10/10	Tue 02/11/10
Especificación de las tareas	3 days	Tue 02/11/10	Fri 05/11/10
Instalación de las herramientas a utilizar	4 days	Thu 28/10/10	Thu 04/11/10
Familiarización con el entorno de trabajo	8 days	Thu 04/11/10	Wed 17/11/10
Versión 1	**37 days**	**Thu 28/10/10**	**Fri 11/02/11**
Especificación de requisitos de la versión 1	2 days	Thu 28/10/10	Tue 02/11/10
Diseño de la interfaz	2 days	Tue 02/11/10	Thu 04/11/10
Implementación	23 days	Wed 17/11/10	Wed 09/02/11
Pruebas	2 days	Wed 09/02/11	Fri 11/02/11
Versión 2	**24 days**	**Fri 11/02/11**	**Tue 22/03/11**
Especificación de requisitos de la versión 2	3 days	Fri 11/02/11	Wed 16/02/11
Rediseño de la interfaz	2 days	Thu 17/02/11	Mon 21/02/11
Estudio de las transiciones entre ventanas	2 days	Mon 21/02/11	Wed 23/02/11
Implementación	15 days	Wed 23/02/11	Fri 18/03/11
Pruebas	2 days	Fri 18/03/11	Tue 22/03/11
Conclusiones	**7 days**	**Tue 22/03/11**	**Fri 08/04/11**
Instalación y pruebas en un dispositivo real	4 days	Tue 22/03/11	Tue 05/04/11
Estudio de posibles mejoras	3 days	Tue 05/04/11	Fri 08/04/11
Documentación	**19 days**	**Fri 08/04/11**	**Mon 09/05/11**
Elaborar la guía de uso	3 days	Fri 08/04/11	Wed 13/04/11
Realización de la memoria final	10 days	Wed 13/04/11	Fri 29/04/11
Elaborar la presentación del proyecto	6 days	Fri 29/04/11	Mon 09/05/11

Figura 3.1.: Planificación prevista de las tareas

Se adjunta a continuación el diagrama de Gantt donde se pueden ver las relaciones entre las diferentes tareas de forma gráfica. Las zonas grisáceas se corresponden a periodos en los que por un motivo u otro no se pudo avanzar con el proyecto.

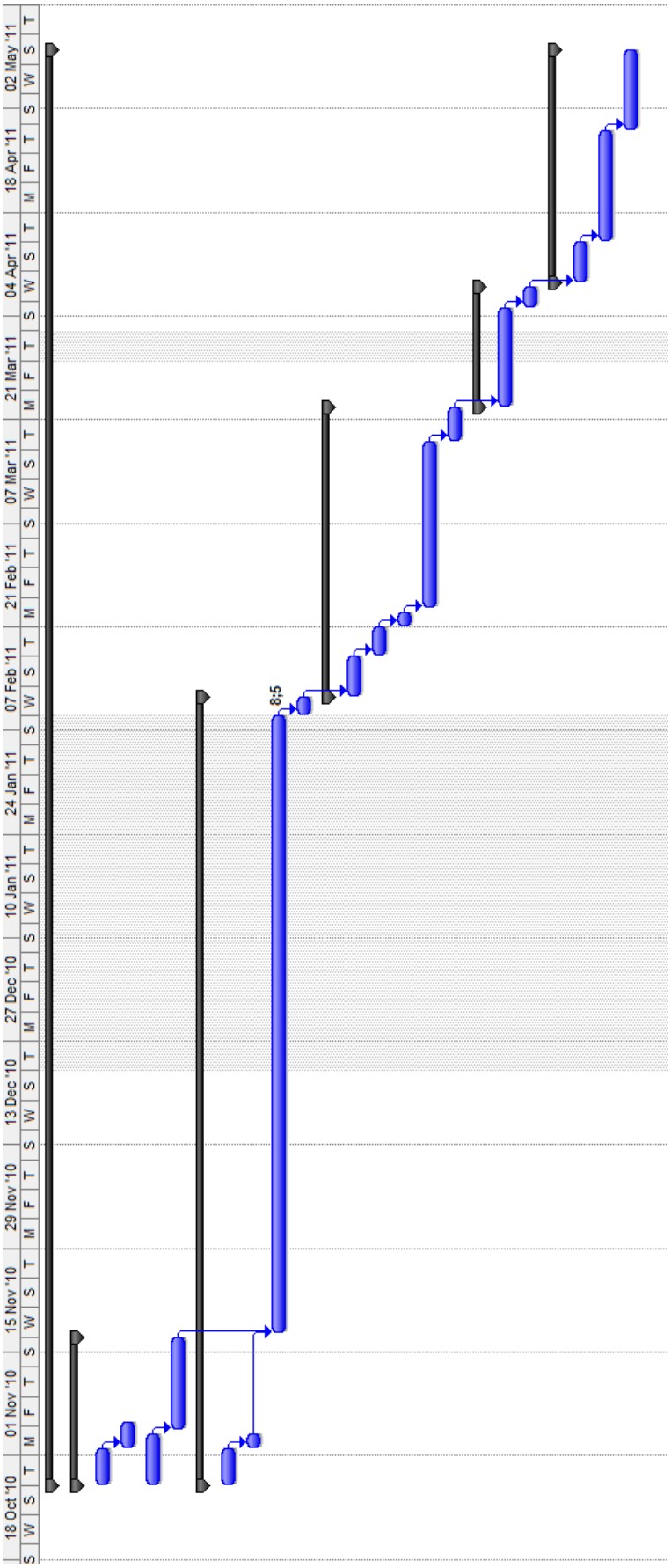

Figura 3.2.: Diagrama de Gantt previsto

3.2.2. Planificación real

Una vez finalizada la planificación prevista de las tareas, se empezó a desarrollar el proyecto, anotando la fecha de inicio y de fin de cada una de las tareas con el objetivo de saber sus duraciones reales.

Hay que tener en cuenta que no se pudo cumplir con las 4 horas diarias de trabajo por motivos de tiempo, por lo que se tuvo que reducir este tiempo repercutiendo en la duración del proyecto, como se verá más adelante. De esta forma, el tiempo de dedicación por día al proyecto fue de 3 horas aproximadamente.

La figura 3.3 muestra la planificación real de las tareas y las fechas de inicio y de fin reales entre las que se desarrolló el proyecto.

Task Name	Duration	Start	Finish
Proyecto	**110 days**	**Thu 28/10/10**	**Tue 14/06/11**
Planificación	**16 days**	**Thu 28/10/10**	**Tue 23/11/10**
Definición de objetivos	2 days	Thu 28/10/10	Tue 02/11/10
Especificación de las tareas	1 day	Tue 02/11/10	Wed 03/11/10
Instalación de las herramientas a utilizar	2 days	Thu 28/10/10	Tue 02/11/10
Familiarización con el entorno de trabajo	14 days	Tue 02/11/10	Tue 23/11/10
Versión 1	**53 days**	**Thu 28/10/10**	**Wed 09/03/11**
Especificación de requisitos de la versión 1	2 days	Thu 28/10/10	Tue 02/11/10
Diseño de la interfaz	3 days	Tue 02/11/10	Fri 05/11/10
Implementación	36 days	Wed 24/11/10	Tue 08/03/11
Pruebas	1 day	Tue 08/03/11	Wed 09/03/11
Versión 2	**35 days**	**Wed 09/03/11**	**Tue 10/05/11**
Especificación de requisitos de la versión 2	3 days	Wed 09/03/11	Mon 14/03/11
Rediseño de la interfaz	3 days	Tue 15/03/11	Fri 18/03/11
Estudio de las transiciones entre ventanas	2 days	Fri 18/03/11	Tue 22/03/11
Implementación	24 days	Tue 22/03/11	Thu 05/05/11
Pruebas	3 days	Thu 05/05/11	Tue 10/05/11
Conclusiones	**3 days**	**Wed 11/05/11**	**Mon 16/05/11**
Instalación y pruebas en un dispositivo real	2 days	Wed 11/05/11	Fri 13/05/11
Estudio de posibles mejoras	1 day	Fri 13/05/11	Mon 16/05/11
Documentación	**19 days**	**Mon 16/05/11**	**Tue 14/06/11**
Elaborar la guía de uso	1 day	Mon 16/05/11	Tue 17/05/11
Realización de la memoria final	12 days	Tue 17/05/11	Fri 03/06/11
Elaborar la presentación del proyecto	6 days	Mon 06/06/11	Tue 14/06/11

Figura 3.3.: Planificación real de las tareas

Y el diagrama de Gantt obtenido para la duración real de las tareas es el mostrado en la figura 3.4

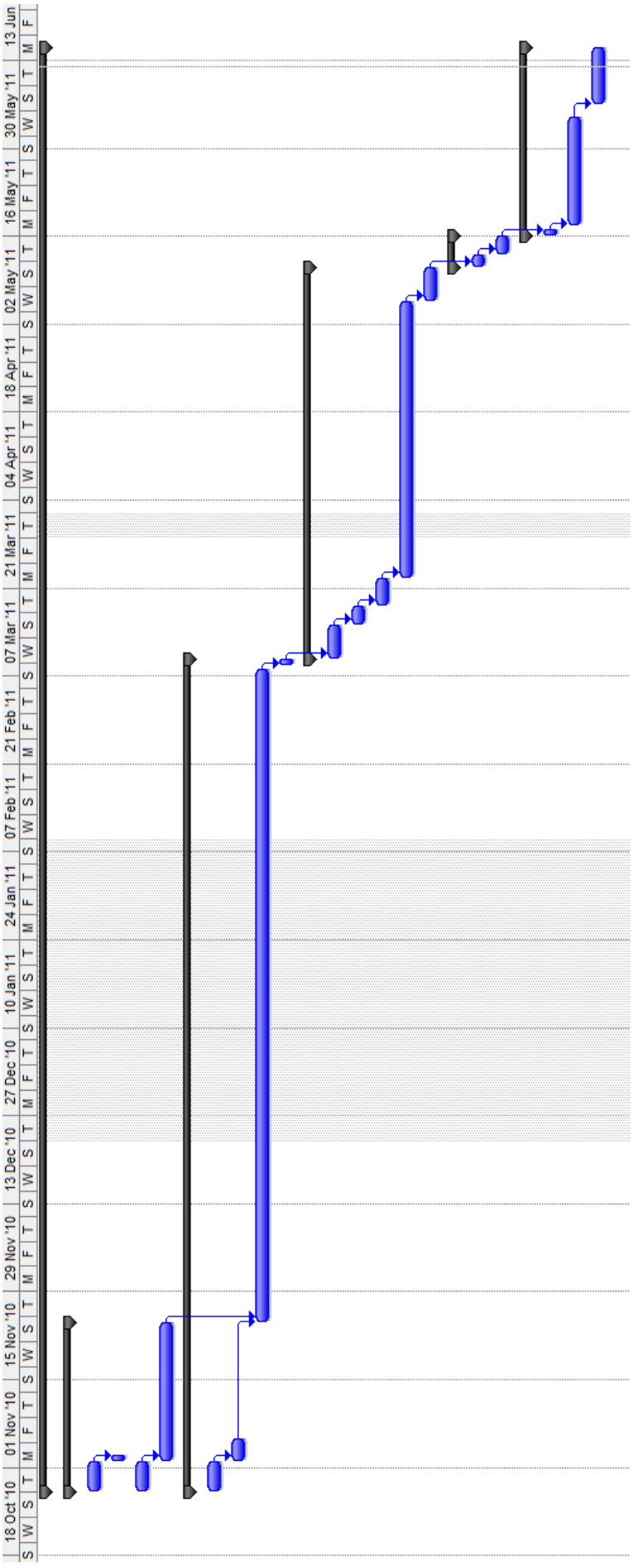

Figura 3.4.: Diagrama de Gantt real

Cabe mencionar que la tarea de *Familiarización con el entorno de trabajo* requirió prácticamente el doble del tiempo del previsto debido a que, en un principio, se pensó que sería más sencilla la programación de aplicaciones Android mediante Java, lo cual provocó un exceso de confianza que se tradujo, posteriormente, en un mayor trabajo de aprendizaje para tener claros los conceptos principales de la programación.

Como se puede apreciar, la tarea que más tiempo consumió fue la implementación de la versión 1. Esto concuerda con lo previsto puesto que al ser la primera versión de la aplicación, era de esperar que consumiese más tiempo debido a la falta de conocimientos en el lenguaje de programación Java. Además, el incremento sustancial del tiempo de ejecución de esta tarea se debe a que se ideó un diseño para esta primera versión, pero a la hora de programarlo y ejecutarlo se llegó a la conclusión de que no era el más adaptable a la ventana del teléfono móvil, por lo que hubo que cambiarlo y, con ello, la implementación del mismo, lo cual requiso más tiempo.

Por otra parte, aunque la segunda tarea más duradera es la implementación de la versión 2, lo cual también coincide con los resultados previstos, era de esperar que su duración fuese inferior a la de la versión 1, ya que, aunque requería más programación, ya se tenía soltura en el desarrollo de la aplicación y parte del código se pudo aprovechar.

A partir de la información de las figuras 3.1 y 3.3 se llega a la conclusión de que el proyecto se ha retrasado un mes y 5 días. El motivo es debido a que desde el 7 de enero al 8 de febrero no se pudo contactar con el tutor del proyecto por motivos justificados, por lo que la fecha de finalización se ha resentido. Además, hay que tener en cuenta los periodos vacacionales en los que no se pudo avanzar con la realización del proyecto como se esperó en un principio, lo cual también influyó negativamente en la fecha de finalización del mismo. De esta forma, podemos concluir que el tiempo real de desarrollo del proyecto ha sido de 110 días. Sabiendo que por término medio se dedicaron 3 horas al día para la realización de éste, obtenemos que el tiempo de desarrollo del proyecto, medido en horas, ha sido de 330 horas.

4. Desarrollo

En este apartado se explicará el proceso de desarrollo del proyecto, concretando los requisitos acordados para cada versión y explicando tanto las pautas seguidas para su implementación como las ideas seguidas para definir el diseño de la interfaz.

Se empezó con una primera versión totalmente funcional que sirvió de base para definir, posteriormente, la versión final. Ésta engloba las mismas funcionalidades que la primera versión más otras mejoras que se consideraron importantes tras analizar el resultado de la primera versión.

4.1. Desarrollo de la versión 1

4.1.1. Análisis de requisitos

Esta primera versión de la aplicación constará de las funcionalidades básicas que permitan utilizarla para planificar una sesión de ejercicios. De este modo, un requisito indispensable será permitir al usuario introducir los diferentes ejercicios y su duración. Además, a modo informativo, se añadirán otros elementos que facilitarán la comprensión de la información introducida en pantalla.

Éstos y otros requisitos se definen a continuación:

- *Añadir ejercicio*: el usuario debe poder añadir actividades para componer la sesión se ejercicios. Puesto que la pantalla tiene unas dimensiones reducidas, el procedimiento seguido para introducir un nuevo ejercicio será el siguiente:

  1. Hacer una pulsación corta encima de la franja en la que se desee introducir la actividad.
  2. Automáticamente la franja mostrará los controles básicos para introducir el ejercicio y modificar su duración. Por lo tanto, se deberá pulsar encima de la caja de texto para introducir el nombre del ejercicio, utilizando el teclado tipo QWERTY que mostrará el terminal.
  3. Una vez introducido el ejercicio, se puede pulsar el botón *Atrás* del teléfono para ocultar el teclado en pantalla, con lo que se mostrará la sesión de ejercicios con la nueva actividad introducida.
  4. Además, se ha definido esta caja de texto como un elemento *Autocompletable*, lo que significa que cuando se hayan escrito un mínimo de 3 caracteres, mostrará una lista de ejercicios predefinidos cuyo nombre comience de la misma forma que el texto introducido hasta ese momento, pudiendo seleccionar cualquiera de ellos pulsando en el deseado.

- *Eliminar ejercicio*: de la misma forma que se pueden agregar nuevos ejercicios, también se deben poder eliminar. Para ello será suficiente una pulsación larga encima de la zona no utilizada de ese ejercicio. Esto es, el espacio que aparece debajo de los botones de cambio de la duración o en el texto de la franja horaria que aparece a la izquierda de la celda.

- *Cambiar las duraciones*: otro requisito importante es el de permitir al usuario modificar el tiempo de ejecución de los diferentes ejercicios. La aplicación debe permitir tanto aumentar como disminuir esta duración pulsando en los botones que aparecen al lado de cada ejercicio.

- *Mostrar la duración del ejercicio*: Se trata de un requisito indispensable en toda sesión de ejercicios que el usuario sepa cuánto tiempo le va a dedicar a cada actividad. Este tiempo se actualizará cada vez que el usuario modifique la duración del ejercicio en cuestión.

- *Tamaños de celda*: A modo de información visual que permita al usuario comparar duraciones entre ejercicios, se ha definido como un requisito variar la altura de las celdas que contienen los ejercicios proporcionalmente a su duración. Así se facilita la tarea de comparación de las duraciones entre ejercicios.

- *Franjas horarias*: Para que el usuario sepa a qué hora empezar cada actividad se muestra también las franjas horarias en rangos de 10 minutos. Nótese que aunque una actividad empiece a una hora en concreto, si se modifica su duración también se debería modificar el rango de horas asociado. No se implementó así porque, como se ha comentado, esta primera versión simplemente es una guía hacia lo que será la aplicación final, por lo que no se consideró importante actualizar estos elementos.

Estos requisitos son los que debía cumplir la primera versión, la cual no se consideró terminada hasta que no incluyó estas funcionalidades y se testeó para detectar que no contenía errores de ningún tipo.

Por otra parte, es importante tener en cuenta que la aplicación se desarrolló para que fuese sencilla de usar por el usuario final, que podrá ser cualquier persona tenga o no conocimientos informáticos, por lo que la forma de implementar estos requisitos se hizo intentando maximizar la usabilidad de la aplicación para que el usuario pueda utilizarla de forma cómoda y con el menor esfuerzo posible.

4.1.2. Diseño de la interfaz

Añadir ejercicio

En primer lugar cabe decir que se siguió un diseño cíclico, con el objetivo de poder ir adecuando el proyecto (tanto la interfaz como la implementación) a las necesidades o requisitos que fueron surgiendo a medida que se ejecutaba y se detectaban carencias en la aplicación. De esta forma, se comenzó con una versión a la que se llegó tras un proceso de análisis de los requisitos explicados en el punto 4.1.

El objetivo inicial del diseño fue dotar de una interfaz sencilla e intuitiva a la aplicación. Por tanto, se pensó en diseñarla de forma tabular. Es decir, esta idea se basaba en tener en la parte superior de la pantalla (ocupando el 30 % superior) una lista de ejercicios, cada uno representado mediante una pequeña imagen diferente. En el 70 % inferior se disponía una tabla a modo de calendario en el que poder arrastrar las actividades para ir confeccionando la sesión de ejercicios.

Pronto apareció un problema derivado de esta estructuración: además de agregar los ejercicios al calendario, había que añadir también botones que permitiesen cambiar su duración, lo cual hacía que únicamente cupiesen unos pocos ejercicios en pantalla dificultando la creación de la sesión de manera sencilla.

Esto hizo que se estudiase una reestructuración del contenido en la interfaz, la cual se decidió que siguiese el mismo diseño utilizado en Google Calendar[1]. La ventaja de esta disposición es que no se tiene una lista de ejercicios en pantalla, sino únicamente el calendario. Por tanto, pulsando en una franja horaria, aparecen los elementos necesarios para introducir el texto deseado, lo cual permite utilizar toda la superficie de la pantalla para mostrar la sesión de ejercicios.

Llegados a este punto en el que se tuvo la primera parte de la interfaz decidida, había que estudiar cómo se iba a mostrar cada ejercicio y sus botones de duración para aprovechar al máximo el espacio de la celda, teniendo en cuenta que había que dejar una zona libre de elementos para permitir el borrado de la actividad. Por esto se decidió dividir cada celda en diversas partes:

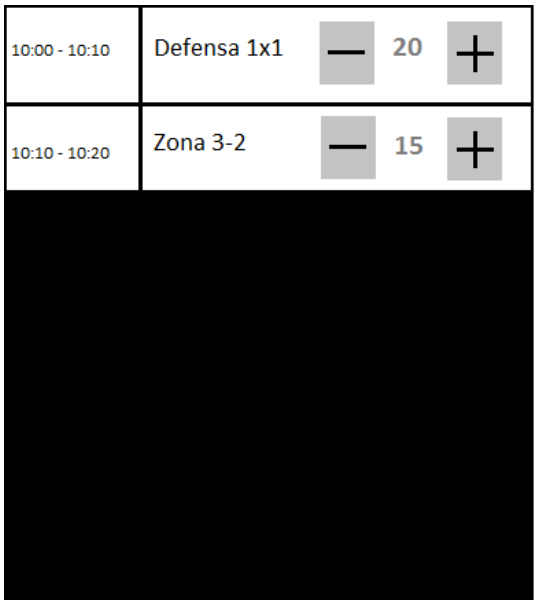

Figura 4.1.: Estructura de la pantalla

Como se puede apreciar en la figura 4.1, cada celda se divide en las siguientes partes:

- A la izquierda de la celda se mostraba la hora de inicio y de fin de esa celda en concreto. Estos valores empezaban a contar a partir de las 10 de la mañana y se incrementaban de 10 en 10 minutos.
- La parte de la derecha de la celda se dividía en otras 2 secciones:
  - Ejercicio: está en la zona izquierda, y es donde se escribía el texto del ejercicio.
  - Duraciones: es la parte en la que se gestiona la duración del ejercicio, mediante los botones de incremento y decremento de ésta y la casilla que muestra la duración actual.

Eliminar ejercicio

Como se ha comentado, es necesario poder eliminar los ejercicios introducidos. Pero debido a las limitaciones en cuanto a tamaño de la pantalla se refiere, se hace complicado añadir un

[1] `http://www.google.com/intl/es/googlecalendar/tour.html`

botón que tenga esta función puesto que esto generaría una interfaz excesivamente comprimida. Por este motivo se ha diseñado de forma que para eliminar un entrenamiento sea suficiente hacer una pulsación larga (3 segundos) encima de la zona comprendida entre los botones de duración o encima del texto que muestra la franja horaria de cada celda.

Modificación de la duración del ejercicio

Como ya se ha dicho, es de vital importancia poder cambiar el tiempo de ejecución de los ejercicios. Por eso se decidió añadir dos botones con imágenes de los símbolos *mas* y *menos* puesto que el cerebro asocia antes una imagen con su significado que un texto, lo cual mejora la usabilidad y hace la interfaz más intuitiva.

La caja de texto con la duración del ejercicio se decidió ubicar entre ambos botones para mejorar su visualización a la hora de modificar la duración.

Variación de la altura de la celda

Con el fin de facilitar la comparación entre las duraciones de los entrenamientos, se ha definido el diseño de forma que la altura de las celdas que contienen los diferentes ejercicios varíen en función de la duración de los mismos. Así, la altura de cada celda era igual a 40 píxeles que se sumaban a la duración del ejercicio. Por ejemplo, una actividad de 20 minutos ocupa 40+20 = 60 píxeles de altura, mientras que una de 10 minutos ocupa 40+10 = 50 píxeles. La variación de la duración de los ejercicios se hace mediante dos botones de incremento y decremento en rangos de 5 minutos.

Scroll

Aunque no se definió un requisito como tal, debido a su fácil implementación y la comodidad que otorga al usuario para introducir más ejercicios, se creó un scroll vertical para poder añadir más ejercicios de los que se pueden mostrar en pantalla. De esta forma, arrastrando el scroll hacia arriba o hacia abajo se pueden añadir más actividades. Además, esto permite no tener un límite, a priori, de la cantidad de ejercicios que se pueden introducir. Este límite estará marcado por la cantidad que se haya especificado en el código de la aplicación, aunque podría tratarse de una futura mejora poder definirlo cuando se crea la propia sesión.

Anotaciones

Aunque no se trata de un requisito como tal, la forma de implementar la aplicación permitió introducir ejercicios que podrían ser tratados de *Anotaciones*. Esto significa que se podía reducir la duración de un ejercicio hasta los 0 minutos, lo que significa que el ejercicio no tiene duración, pero se puede asimilar su uso a una anotación que podría ser interesante comentar mientras se ejecuta una sesión de ejercicios.

La figura 4.2 muestra un ejemplo de la aplicación en esta primera versión pudiendo identificar los siguientes elementos:

1. Botones de cambio de la duración del ejercicio.
2. Duración actual del ejercicio.
3. Información horaria de cada celda.

4. Zona libre para poder eliminar el ejercicio.

Además de los elementos anteriores también se puede comprobar que la altura de las celdas es ligeramente mayor dependiendo de la duración de los ejercicios.

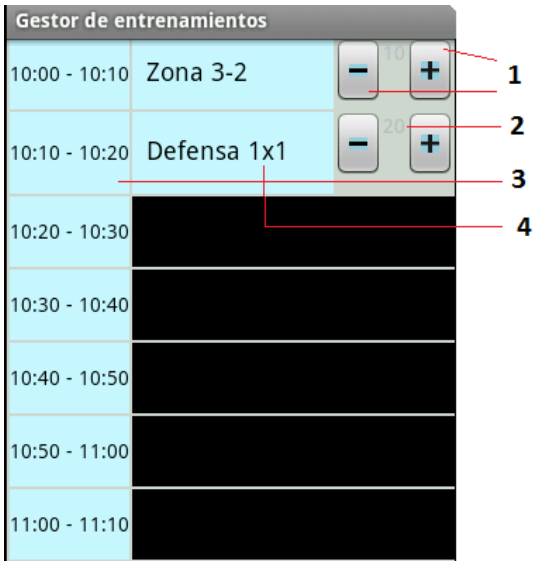

Figura 4.2.: Versión 1

4.1.3. Implementación

Esta primera versión hace uso de una clase en la que se crea el código que generará la aplicación. Para soportar la interactuación por parte del usuario se definen los métodos *OnClickListener* y *OnLongClickListener* asignados a la clase que serán utilizados por elementos creados en ella.

Puesto que hay que crear nuevos elementos dinámicamente, el método seguido consiste en definir un rango de valores para cada tipo de elemento, asignando a la variable *id* el último identificador asignado. Por ejemplo, empezando a contar desde el valor 0, las filas de la tabla tienen un identificador con valor *id+1000*. De esta forma, cada vez que se crea una nueva fila (o lo que es lo mismo, se añade un nuevo ejercicio), su identificador estará comprendido entre 1000 y 1999. El motivo por el cual se ha comenzado en 1000 es para dejar libres los identificadores entre 1 y 999 para elementos fijos en la pantalla, como puede ser la caja de texto con la hora de finalización o componentes que se necesiten añadir en futuras mejoras. Esto tiene la ventaja de que a partir de un identificador de una fila, podemos acceder a los elementos contenidos en ella sumándole la constante del elemento requerido. Los rangos establecidos son los siguientes:

- Filas: id+1000
- Celda izquierda de la fila: id+2000
- Celda derecha de la fila: id+3000
- Botón incremento de duración: id+4000
- Botón decremento de duración: id+5000
- Indicador duración: id+6000
- Caja de texto con el nombre del ejercicio: id+7000
- Lista de ejercicios: id+9000

El gráfico 4.3 muestra la relación entre las funciones implementadas.

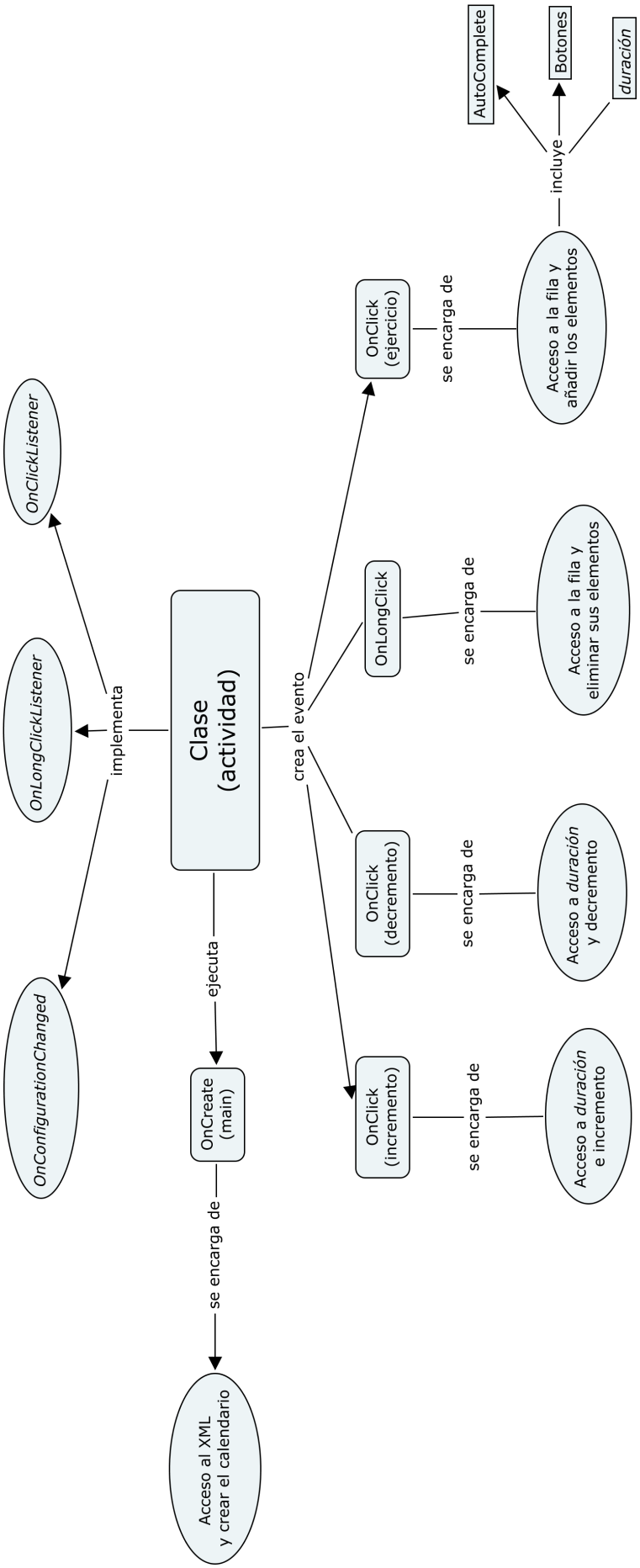

Figura 4.3.: Gráfico de funciones - versión 1

En primer lugar se definió la tabla principal en el xml. Para ver su estructura, basta con observar el código4.4 en el que se han eliminado los atributos de los elementos para que se identifique mejor la estructura.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <ScrollView>
                <TableLayout>
                                    <TableRow>
                                    </TableRow>
                            </TableLayout>
        </ScrollView>
</LinearLayout>
```

Figura 4.4.: Estructura del XML de la versión 1

Se compone de un LinearLayout que engloba todo el contenido de la pantalla. Éste no es más que un ScrollView que nos permitirá tener más entrenamientos de los que caben verticalmente en la pantalla y un TableLayout que hará la función de contenedor para ir almacenando los sucesivos ejercicios.

En cuanto a la implementación de la clase principal se refiere, se podría decir que la aplicación utiliza un vector donde cada posición es un componente View que almacena el contenido de una actividad. Así, inicialmente este View está vacío (al empezar la sesión no hay ejercicios) y cada vez que se introduce un ejercicio, la función *OnClick* aplicada a la celda se encarga de añadir al View los elementos restantes.

De esta forma, la ejecución de la aplicación se puede describir como sigue:

1. Cuando se ejecuta el programa, se accede al TableLayout definido en el XML y se le añaden tantas filas como se le haya indicado en la variable *num_entrenos*.

2. Al mismo tiempo que se van creando las filas, se les asigna a cada una un *OnClickListener*, evento que permite interactuar con ellas para que hagan lo que nos interese al hacer una pulsación encima de ellas. En este caso, en cada click, se obtiene el identificador de la fila que se ha pulsado y se accede al objeto que representa la fila para añadirle los nuevos elementos (botones de incremento y decremento, TextView con AutoComplete para los ejercicios y casilla con la duración). La forma de añadirlos es la comentada en la sección 4.1.3: dependiendo del elemento del que se trate se suma una constante u otra al identificador de la fila.

3. Para los botones de *más* y *menos*, se define otro evento *OnClickListener*, de forma que su pulsación tiene como efecto la actualización de la duración del ejercicio y actualización de la altura de la fila en función de la duración actual.

Finalmente, para eliminar un ejercicio, se incorpora un evento *OnLongClick* a cada una de las filas de la tabla, y su activación se realiza cuando se mantiene pulsada dicha fila durante 3 segundos, ya que en caso contrario se trata como una pulsación normal. En el caso de una pulsación larga,

se accede a la fila y se obtiene su identificador. Con éste, puesto que los identificadores tienen todos un patrón consistente en sumar una constante dependiendo del tipo de elemento que se trate, se accede a los botones de *más* y *menos*, al TextView con el nombre del ejercicio y al campo con la duración y se eliminan de la fila mediante la función *removeViewAt*, a la que se le pasa como parámetro el índice del elemento dentro de la vista que se quiere eliminar.

4.1.4. Pruebas

Puesto que no se dispuso de un teléfono real en el que probar esta primera versión de la aplicación, las pruebas se realizaron sobre el emulador lanzado en Eclipse. Éstas consistieron en:

1. Comprobar que la pulsación de los botones de incremento y decremento tienen los siguientes efectos:
   - Cambio de la duración del ejercicio.
   - Actualización del tamaño de la celda acorde con la nueva duración.
2. Comprobar que una pulsación larga encima de la zona gris o franja horaria de una fila provoca:
   - Desaparición del ejercicio, junto sus botones y campo de duración.
   - Restaurar la altura por defecto de la fila.
3. Comprobar que al reducir la duración de un ejercicio, esta no disminuye de 0.
4. Comprobar qué ocurre si se hace una pulsación larga en una fila que no contiene ningún ejercicio o se hace una pulsación corta en una fila que ya contiene un ejercicio.

Los resultados de las pruebas anteriores fueron los siguientes:

1. La pulsación de los botones de incremente y decremento cumple con su función, actualizando la duración del ejercicio e incrementando o disminuyendo la altura de la celda, con el añadido de que la altura no disminuye de la establecida inicialmente. De esta forma, el ejercicio siempre da la sensación de estar ubicado dentro de la fila.
2. Si se hace una pulsación larga en la fila se pueden dar dos situaciones:
   - Si se pulsa encima del cuadro de texto con el nombre del ejercicio, el comportamiento no está implementado, dependiendo del teléfono móvil la acción a realizar. En este caso en particular, en el que se hace una pulsación larga encima del nombre del ejercicio, se despliega un menu de opciones que permiten copiar el texto, cortarlo, añadirlo al diccionario, ...
   - Si se pulsa encima de la franja horaria o de la zona gris, el comportamiento es el esperado: desaparece el ejercicio de la lista dejando la fila libre para poder ser utilizada más adelante por otro ejercicio.
3. Efectivamente, la duración mínima de un ejercicio es de 0 minutos, no permitiendo rebajarla en ningún caso.
4. Si se hace una pulsación larga encima de una fila vacía no ocurre nada, siendo ignorada y no afectando al estado de la aplicación. Si, por el contrario, se hace una pulsación corta en una fila que ya contiene una actividad, se crea un nuevo ejercicio encima de la fila, sobreescribiendo el que ya había.

4.2. Desarrollo de la versión 2

4.2.1. Análisis de requisitos

Una vez finalizada la versión 1, se analizó su interfaz para intentar ampliar sus funcionalidades o redistribuir los elementos para mejorar la interacción con la aplicación.

Por ejemplo, se vio que no era necesario mostrar la hora de inicio y de finalización en cada ejercicio, ya que es una información redundante por el hecho de tener una casilla de duración en el propio ejercicio. Por esto se eliminó la franja horaria de las celdas, permitiendo aprovechar el espacio para aumentar la longitud del nombre de la actividad. Además, puesto que en este momento no había ninguna información acerca de la hora de inicio ni de fin, se decidió añadir esta información a la aplicación, mostrando una pequeña fila en la parte superior del calendario en la que se especificaba la hora de inicio y la de finalización, siendo ésta última actualizada automáticamente cada vez que se modificaba la duración de un ejercicio.

Además, uno de los cambios más importantes consistió en diferenciar las dos etapas en una sesión de ejercicios. En primer lugar, antes de empezar con la actividad física que requiere una sesión de ejercicios, es necesario planificarla para decidir qué actividades realizar y la duración de cada una. Una vez hecho esto, se puede pasar a la ejecución de las actividades, pero no antes de haberlas planificado. Por esto se estableció como un requisito importante poder diferenciar entre la fase de planificación y la de ejecución.

De este modo, tras la pantalla inicial habría que poder seleccionar a qué fase queremos ir, mediante un botón cuya función es ejecutar la aplicación de dicha fase. En la planificación se siguió un estilo similar a la versión 1, aunque con los cambios comentados al principio de este apartado. Ya en la etapa de ejecución, la finalidad es la de ejecutar los ejercicios planeados, donde se debe saber en todo momento qué ejercicio se está ejecutando, el tiempo total destinado a su ejecución y el tiempo restante para finalizarlo.

También se decidió, en la fase de planificación, incluir un tiempo de recuperación entre ejercicios para que el usuario no tenga que pausar manualmente la sesión. De esta forma éste no se debe preocupar al acabar una actividad, ya que la misma aplicación comienza un tiempo de descanso antes de empezar con el siguiente ejercicio.

En resumen, los requisitos que debía cumplir esta segunda versión fueron los siguientes:

1. Distinguir entre una etapa de planificación, en la que se pueden introducir los ejercicios y modificar sus duraciones y otra etapa de ejecución, donde el usuario no puede modificar los ejercicios y el cometido de esa fase no es otro que la puesta en práctica de los ejercicios elegidos.

2. Respecto a la fase de planificación:
   - Conocer la hora de inicio y de fin de la sesión de ejercicios. Esta información será únicamente visual, ya que no podra ser modificada por el usuario. Así, a medida que se van añadiendo actividades a la sesión actual se podrá ir viendo la hora de fin.
   - Puesto que se distinguen dos etapas, será necesario almacenar la información de la planificación para que este disponible en la de ejecución. Por esto un requisito será poder guardar una sesión de ejercicios para su posterior ejecución.

- Modificación de la duración de los ejercicios, mediante dos botones de incremento y decremento del tiempo en rangos de 5 minutos. Además, la pulsación de alguno de estos botones implica la actualización de la hora de finalización de la sesión.
- Mostrar la duración de cada ejercicio para saber en todo momento el tiempo destinado a cada una de las actividades.
- Añadir el ejercicio deseado, bien sea utilizando el teclado en pantalla del dispositivo o la lista desplegable de los ejercicios predefinidos.

3. Respecto a la fase de ejecución:
   - Saber en todo momento el ejercicio que se esta realizando, donde para ello se mostrará de un color diferente al resto.
   - Conocer el tiempo total destinado a la realización de un ejercicio. Este tiempo es el seleccionado en la fase de planificación.
   - Saber el tiempo restante para finalizar un ejercicio o el periodo de descanso. Este tiempo deberá mostrarse en un cronómetro en la parte superior de la aplicación para estar siempre visible y facilitar su uso al usuario.
   - Poder pausar la sesión o finalizarla. Para ello se dotará al cronómetro de dos botones que ejecutarán estas acciones.
   - Reproducir una señal acústica en el cambio entre ejercicios para una mayor comodidad por parte del usuario en la ejecución de la sesión, liberándolo de la tarea de estar pendiente del cronómetro para dar por finalizado un ejercicio.

4.2.2. Diseño de la interfaz

Puesto que hay que tener en cuenta las dos etapas mencionadas anteriormente, en esta versión se ha rediseñado la interfaz. Ahora constará de 3 partes claramente diferenciadas, de modo que al ejecutar la aplicación, nos aparece una ventana con dos botones claramente diferenciados: uno para crear una nueva sesión de ejercicios y otro para ejecutar una sesión previamente almacenada en un fichero de texto. Este fichero se creará automáticamente cuando se pulse el botón de *Guardar sesión* en la fase de planificación, como se verá más adelante. El objetivo de esta ventana inicial es el de poder elegir en qué etapa del entrenamiento está el usuario: la forma usual de elaborar el entrenamiento será acceder primero a la fase de planificación para crear una nueva sesión de ejercicios y, posteriormente, entrar en la fase de ejecución donde llevar a cabo los ejercicios seleccionados.

La aplicación no permite entrar en la fase de ejecución sin pasar, previamente, por la de planificación, mostrando un aviso de que no se han podido cargar los los ejercicios si se da este caso. Por el contrario, si se guarda una sesión, ésta podrá ser ejecutada tantas veces como se desee sin tener que volver a crearla de nuevo. Es decir, el fichero creado con los ejercicios estará disponible en todo momento mientras la aplicación no se haya cerrado desde que se creó la sesión. Esto significa que si se sale de la aplicación, la próxima vez que se ejecute habrá que crear de nuevo una sesión de ejercicios porque la anterior habrá sido borrada.

El diseño de esta ventana inicial se puede ver en la figura 4.5.

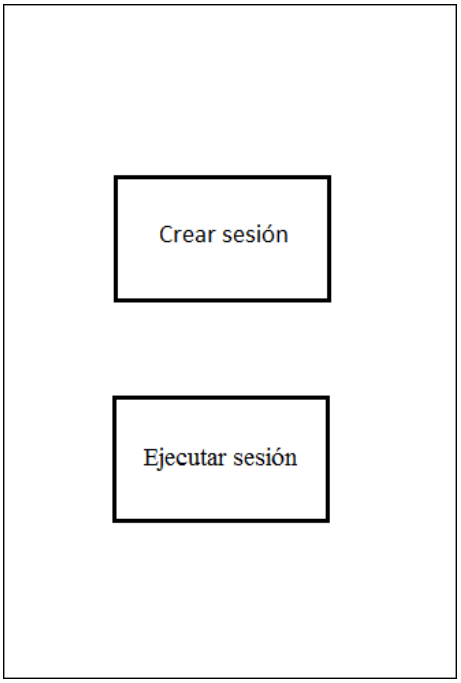

Figura 4.5.: Versión 2 - pantalla inicial

La fase de planificación tiene la estructura mostrada en la figura 4.6. Como se puede apreciar, se han eliminado las franjas horarias incluidas en la versión 1, lo cual ha permitido aumentar el espacio para el ejercicio. Además, se han añadido nuevos elementos, como son el botón de guardado de la sesión o la fila especial informativa de la hora de inicio y fin de la sesión.

Además, se ha modificado el comportamiento de las filas al aumentar o disminuir la duración de un ejercicio. En la primera versión, no quedaba clara la relación entre un ejercicio de 20 minutos y otro de 10, puesto que la altura no era proporcional. El cambio que se ha llevado a cabo en esta versión es el de aumentar o disminuir la altura de la fila de forma que un ejercicio de 20 minutos tendrá una altura de fila del doble que un ejercicio de 10 minutos, lo cual facilita al usuario comparar los ejercicios. El proceso seguido para ello se comentará en el apartado 4.2.3.

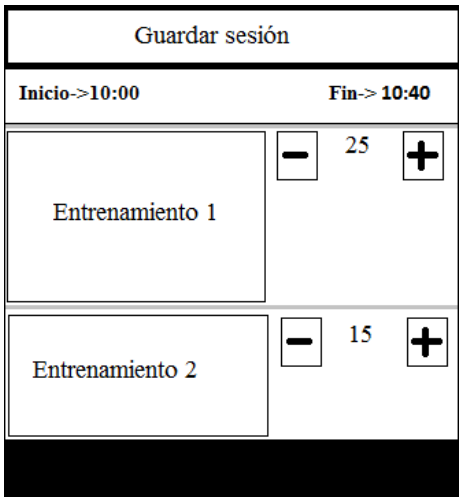

Figura 4.6.: Versión 2 - planificación de la sesión

Para finalizar con el diseño, se procede a explicar la ventana de ejecución. El diseño inicial de esta ventana es el mostrado en la figura 4.7.

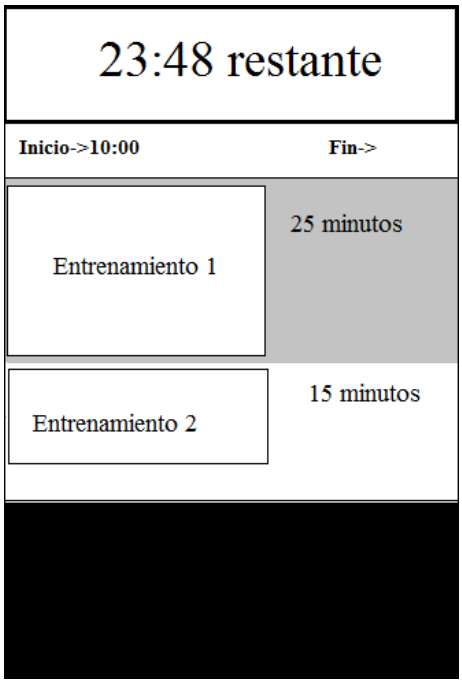

Figura 4.7.: Versión 2 - ejecución de la sesión de ejercicios. Diseño inicial

Tras una primera implementación de esta ventana se llegó a la conclusión de que había que modificarla para ajustarla más al uso que el usuario final le iba a dar. Por ejemplo:

1. En esta fase se puede prescindir de la fila informativa de la hora de inicio y de fin, ya que

esta información ya se tenía en la fase de planificación y el usuario, cuando está ejecutando los ejercicios, debe centrarse en el propio ejercicio más que en la hora de fin de la sesión completa. Por tanto se consideró que esta información era prescindible por lo que se eliminó de la vista y se dejó como importante en la fase de planificación.

2. ¿Qué ocurre si se necesita pausar la sesión por algún motivo? Era necesario añadir botones de pausa y de fin al cronómetro, pensando en que el usuario puede ser un entrenador deportivo que está utilizando la aplicación para mandar realizar los ejercicios a los jugadores. Por tanto, es muy probable que necesite hacer correcciones, aclaraciones o comentarios y para ello deba pausar el ejercicio en curso. Por ello se añadieron ambos botones que permiten pausar o finalizar la sesión indiferentemente del punto en el que se encuentre, ya sea en mitad de un ejercicio o de un periodo de descanso.

3. Aunque los colores blanco y gris sirven perfectamente para distinguir los ejercicios, no son suficientes para representar también el periodo de descanso, pues el usuario deberá poder distinguir sin problemas en qué fase está del entrenamiento. Por eso se decidió cambiar la combinación de colores a la siguiente:
   - Fondo azul para el ejercicio en curso.
   - Fondo gris para el resto de ejercicios.
   - Fondo rojo en el cronómetro para el periodo de descanso

Puesto que cada fase tiene su función, se ha decidido que una vez se ha guardado la sesión en la etapa de planificación no se van a poder modificar las duraciones de los ejercicios. Por lo tanto, en la etapa de ejecución se han eliminado los botones de las duraciones así como se ha deshabilitado la edición en el TextView del nombre del ejercicio, ya que el cometido de esta etapa es pasar a la realización de los ejercicios. Además se muestra el nombre del ejercicio en mayúsculas para facilitar su lectura al usuario en caso de estar en movimiento en el momento de leerlo.

El diseño de esta fase se ha realizado similar al de la fase de planificación para facilitar el reconocimiento de los elementos en pantalla por parte de los usuarios con conocimientos básicos de uso del teléfono móvil.

Los cambios realizados en esta ventana han sido los siguientes:

- El botón de guardar la sesión se ha sustituido por un cronómetro y dos botones, que permitirán al usuario modificar el flujo de ejecución de la sesión pausándola o finalizándola cuando lo requiera. El cronómetro muestra en todo momento el tiempo restante de ejecución del ejercicio actual, y en los periodos de descanso se establece este tiempo de 15 segundos. Además, en el paso de un ejercicio a otro se cambia automáticamente el tiempo a mostrar, liberando al usuario de toda interacción con el teléfono que pudiese entorpecer el desarrollo de los ejercicios.

- Respecto a los ejercicios planificados, éstos se muestran en una lista con el fondo gris con letras negras. Sin embargo, el ejercicio que se está llevando a cabo en ese momento está pintado con un color azul de fondo para diferenciarlo de los demás. Además, el fondo del cronómetro es de color verde mientras se realiza un ejercicio, cambiando de color en los periodos de descanso para dejar claro en todo momento en que parte del entrenamiento se encuentra el usuario.

- En los periodos de descanso, como se ha comentado, el fondo del entrenamiento se vuelve rojo. También se muestra una notificación durante un breve espacio de tiempo para que el usuario pueda ver que se encuentra en periodo de reposo.
- A modo de notificaciones sonoras y táctiles cabe mencionar que se emite un sonido al finalizar tanto los ejercicios como los descansos, para informar al usuario que empieza un nuevo periodo. Además, también se ha implementado una ligera vibración del teléfono para maximizar las ayudas que pueda recibir el usuario en cuanto a información de la sesión se refiere.

.

Tras los cambios comentados respecto a la primera versión, se puede ver el nuevo diseño en la figura 4.8.

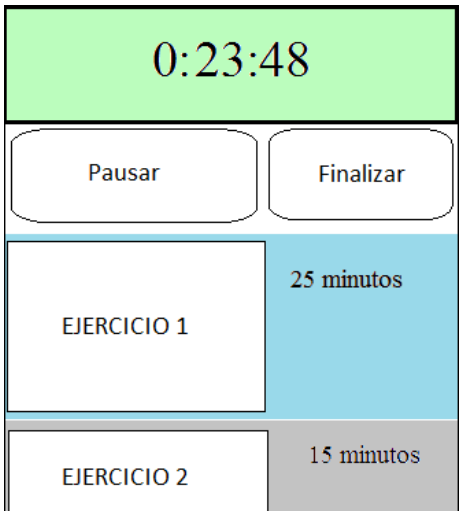

Figura 4.8.: Versión 2 - rediseño de la pantalla de ejecución

4.2.3. Implementación

Como se ha comentado en el apartado anterior, existen 3 ventanas en esta segunda versión, por lo que se implementa cada una en una clase diferente para facilitar su depuración. De esta forma, se tienen las siguientes clases:

1. Inicio: es la clase inicial de la segunda versión, la que implementa la pantalla de inicio de la aplicación. Desde ella se implementan los accesos a las dos ventanas restantes como se verá posteriormente.

2. Planificación: es una clase similar a la de la versión 1 salvo por los cambios de diseño comentados. Su función es implementar la ventana de planificación que servirá, cuando se guarde la sesión, para devolver el nombre del fichero en el que se han guardado los ejercicios.

3. Ejecución: esta clase recibe como entrada el nombre del fichero mencionado, y a partir de él reconstruye el calendario adaptándolo para la fase de ejecución.

Tras una breve introducción a las distintas clases, se presentan a continuación de un modo más detallado. En primer lugar se muestra la estructura de la clase de la ventana de Inicio. Esta clase se encarga de llamar a las clases de planificación y ejecución, además de mostrar mensajes por pantalla al usuario. Estos mensajes son los siguientes:

- Error al cargar los ejercicios: Como se ha comentado anteriormente, para entrar en la fase de ejecución es necesario haber guardado, desde que se ejecutó la aplicación, al menos una sesión se ejercicios, ya que en caso contrario no existe el fichero del que cargar los ejercicios. Por esto, si el usuario intenta lanzar la ventana de ejecución recibirá un mensaje de aviso de dicha situación, evitando así que aparezca un error que haría que la aplicación finalizase.

- Una vez se ha creado la planificación de los ejercicios, es necesario almacenarla en un fichero para que pueda ser cargada más adelante. Esto conlleva la apertura del fichero, recorrer la lista de ejercicios y, finalmente, cerrar el fichero. Este proceso se encierra en una sentencia *try / catch*, de modo que si no se produce ningún error, se vuelve a la pantalla de inicio mostrando un mensaje de que el fichero se ha creado correctamente.

- Cuando finaliza la ejecución de una sesión de ejercicios se vuelve a la pantalla de inicio, mostrando un aviso de que la sesión ha finalizado.

Esta pantalla inicial se ha implementado como un *RelativeLayout* para poder ubicar los botones en el lugar deseado dentro de la interfaz. Además, se les ha dado unas medidas que faciliten su pulsación, como se ve en la figura 4.9.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android">
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical">
            <Button
                android:id="@+id/nueva"
                android:onClick="nueva"
                android:text="CREAR SESIÓN"
                />
            <Button
                android:id="@+id/ejecuta"
                android:onClick="ejecuta"
                android:text="EJECUTAR SESIÓN"
                />
    </RelativeLayout>
</RelativeLayout>
```

Figura 4.9.: Versión 2 - xml de la ventana de Inicio

En la figura anterior se han eliminado los atributos de diseño para mostrar únicamente los siguientes campos, que son los que definen el estilo y comportamiento del botón:

- *android:id*: el nombre que se le da al elemento mediante el cual se puede acceder desde el código.

- *android:onClick*: atributo que indica qué función ejecutará el elemento cuando se active el evento *onClick* sobre él.

- *android:text*: texto que se mostrará encima del botón.

Los atributos que comienzan por *xmlns:android (XML NameSpace)* son obligatorios para todos los contenedores de elementos. Estos son los *LinearLayout*, *RelativeLayout*, *TableLayout*, etc. En otras palabras, son los que estan preparados para englobar conjuntos de elementos, que a su vez pueden ser otros contenedores. Su singificado es indicar al contenedor que será usado por android y no por otra función. Se puede encontrar una breve pero completa descripción de para qué se usa el xmlns en la siguiente dirección:

`http://stackoverflow.com/questions/1181888/what-does-xmlns-in-xml-mean/1181936#1181936`

Tras presentar el xml, se muestra a continuación el diagrama de clases y función de esta ventana inicial.

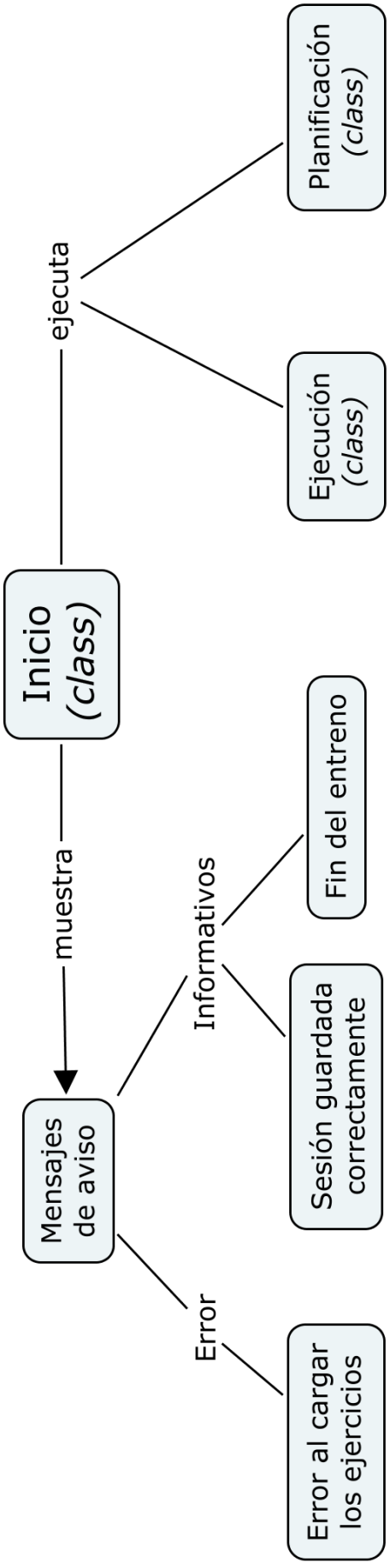

Figura 4.10.: Versión 2 - estructura de clases de la ventana de Inicio

El código necesario para implementar esta ventana se puede entender observando el siguiente fragmento.

```
1  OUTPUT: pantalla principal
2  PROCEDIMIENTO:
3      nueva_sesion = new Button()
4      nueva_sesion.asigna(ClickListener) -> class Planifica()
5      ejecuta_sesion = new Button()
6      ejecuta_sesion.asigna(ClickListener) -> class Ejecuta()
```

Figura 4.11.: Versión 2 - orden de implementación en la pantalla de Inicio

El funcionamiento es sencillo. Lo que se hace en la ventana inicial es inicializar dos botones que apuntan a las ventanas de planificación y ejecución, para ejecutarlas cuando se active el evento *OnClick* sobre el botón correspondiente.

Tras explicar la ventana inicial, se procede con la fase de planificación. La siguiente figura muestra la estructura de clases de la ventana de planificación. Tal y como se puede ver, la clase implementa 3 tipos de métodos:

1. *OnClickListener*: evento asociado a una pulsación normal, lo que equivaldría a un click de ratón en un ordenador. La forma de utilizarlo es indicar en la cabecera de la clase, que se va a implementar el evento *OnClick*, y en el código se hace la llamada asignándolo a tantos elementos como sea necesario.

2. *OnLongClickListener*: evento asociado a una pulsación larga en la pantalla, la cual se activa tras pulsar 3 segundos seguidos en el mismo punto. Al igual que el evento *OnClickListener*, se indica en la cabecera de la clase que se va a referenciar este evento y en el código se le asocian los elementos que nos interesen que respondan ante pulsaciones largas.

3. *OnConfigurationChanged*: este método está relacionado con el giro del teléfono, de la posición vertical a horizontal o viceversa. Si no se especifica este método, el giro del teléfono implica girar el contenido de la pantalla, lo cual borra el estado de la aplicación y se pierden los cambios hechos hasta el momento. Mediante esta función se indica que en caso de giro de la pantalla, el nuevo estado sea el mismo que el anterior.

La parte del *main* coincide con la de la primera versión. Simplemente se carga el diseño contenido en el xml y se crean las filas del calendario mediante un bucle que iterará tantas veces como se haya indicado en la variable *num_ entrenos*. El pseudo-código asociado se muestra en la figura 4.12.

```
1  OUTPUT: calendario
2  PROCEDIMIENTO:
3      vista = Load(xml)
4      Calendario.create(vista)
5      num_ejercicios := 0
6      boton_guardar.asigna(ClickListener)
7      MIENTRAS (num_ejercicios < total_ejercicios):
8          fila.create()
9          fila.asigna(ClickListener, LongClickListener)
10         Calendario.append(fila)
```

Figura 4.12.: Versión 2 - orden de implementación en la fase de Planificación

El evento especial que implementa esta fase y que no estaba en la primera versión es el relacionado con el boton de guardado de la sesión. Al pulsar este botón, se ejecuta su evento *onClick* y se comprueba la cantidad de ejercicios introducidos. Si es 0, se muestra un mensaje de aviso. Si no, se recorre la lista de ejercicios y se almacenan en el fichero. Antes de explicar este proceso, se mencionan algunas variables utilizadas en él:

- *num_ entrenos*: como ya se ha comentado, esta variable contiene el número de filas que contiene el calendario.
- *idEntrenos*: array donde cada componente es un entero que puede valer 1 o 0. Si vale 1, significa que esa fila del calendario contiene un ejercicio, y si vale 0 significa que no hay ninguna actividad en esa fila del calendario.

El primer paso es recorrer todas las filas del calendario, lo cual se consigue con un bucle donde se actualiza una variable con el número de fila. Dependiendo de su valor en el vector *idEntrenos* se ejecuta un código u otro:

- Si su valor es 0, significa que no hay ningun ejercicio en esa fila, y por tanto se puede pasar a la siguiente.
- Si su valor es 1, hay un ejercicio en esa fila del calendario. Por lo tanto, como las filas se numeran desde 0 hasta *num_ entrenos*, teniendo el índice y sabiendo que los identificadores de todos los elementos se han creado sumando constantes que dependen del elemento, se accede fácilmente al campo del nombre de la actividad (9003+índice) y al campo de la duración (6003+índice). Los valores de 6003 y 9003 se corresponden a la constante propia de cada elemento (6000 o 9000) más el límite inferior a partir del cual se comienzan a asignar identificadores. La ventaja de esto es que se le asigna el identificador 2 al botón de guardado de la sesión para identificarlo más rápidamente y el identificador 1 queda libre para poder utilizarlo en alguna mejora futura. Así, con estos datos, se inserta una nueva fila en el fichero que contiene el nombre del ejercicio, la duración y la fila del calendario (para poder reproducir en la fase de ejecución el mismo orden que se ha planificado). El patrón seguido es separar estos tres campos por tres guiones seguidos.

El resultado final es que se almacena en un fichero una línea por cada ejercicio, donde las líneas incluyen información del orden del ejercicio, su nombre y su duración.

Por último, se cierra el fichero y se devuelve a la pantalla inicial su nombre, que será utilizado más adelante en la fase de ejecución para cargar los ejercicios. El nombre dado al fichero es la fecha actual, con la idea de poder crear una base de datos a modo de mejora en la que se puedan almacenar las fechas y los ejercicios en cada una.

Toda esta información se representa de forma gráfica en el siguiente diagrama.

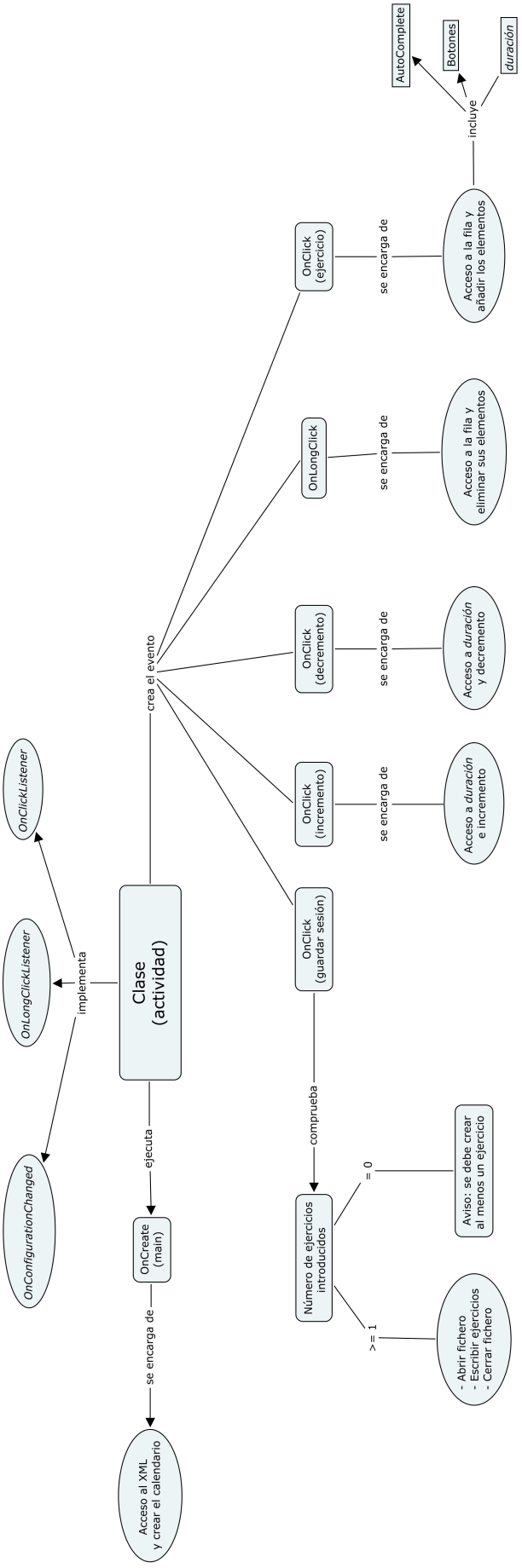

Figura 4.13.: Versión 2 - estructura de clases de la ventana de Planificación

Por último, se procede a explicar la implementación de la ventana de ejecución. Se muestra en la figura 4.14 en pseudo-código la estructura de la clase para dar una idea general y a continuación se explican los pasos que se siguen.

```
INPUT: fichero_con_ejercicios
OUTPUT: ejecución de los ejercicios
PROCEDIMIENTO:
    ejercicios = Load(fichero)
    Calendario.create() <- ejercicios
    thread_cronometro.inicia()
    MIENTRAS (ejercicio_actual != ultimo_ejercicio):
        decrementa_cronometro(1segundo)
        SI (horas=0 AND minutos=0 AND segundos=0):
            alarma.ejecuta()
            SI (estamos_descansando): #viene un ejercicio
                ejercicio_actual = ejercicio_actual.siguiente()
                actualiza_cronometro(ejercicio_actual.duracion)
                thread_fondo_cronometro.actualiza()
            SINO: #viene descanso
                actualiza_cronometro(15segundos)
                thread_fondo_ejercicio.actualiza()

    alarma_fin.ejecuta()
    thread_cronometro.finaliza()
```

Figura 4.14.: Versión 2 - orden de implementación en la fase de Ejecución

1. Se obtiene el nombre del fichero que le ha pasado la ventana de planificación y lo abre en modo lectura.

2. Se accede al xml y se carga el cronómetro, sus botones y el calendario.

3. Para cada linea del fichero, se obtiene la fila en la que se ha introducido el ejercicio, el nombre del mismo y la duración, y con estos datos se accede a la misma fila del calendario cargado y se escriben. Puesto que en este punto no se ha definido ningún evento *OnClickListener* sobre la fila, una pulsación (ya sea corta o larga) no tiene ningún efecto.

4. Una vez se han cargado todos los ejercicios en el calendario, se asigna el evento *OnClickListener* a los botones de pausa y finalización de la sesión del cronómetro.

5. En este punto se inician los threads que gestionarán el funcionamiento de la aplicación. Todos ellos se ejecutan en segundo plano y se van llamando desde el programa principal según se requiera su ejecución. Su cometido es el siguiente:

    - *Descanso (mensaje_ descanso)*: Este thread es encargado de mostrar un mensaje informativo al usuario de que se encuentra en un periodo de descanso. Se ha implementado mediante un thread porque es la forma idónea de delegar una tarea para que la aplicación siga ejecutando el resto de hilos. Si no se hubiese hecho así, al ejecutar el mensaje de descanso posiblemente se hubiese ralentizado el cronómetro, con lo cual el tiempo marcado no sería el correcto.

- *Cronometraje (handler_ cron)*: la función de este thread es estar ejecutándose en segundo plano y actualizar cada segundo el cronómetro. Para ello se ejecuta en un bucle infinito donde las variables *horas*, *minutos* y *segundos* almacenan en todo momento estos parámetros para ser mostrados al usuario en forma de reloj digital. De este modo, cada segundo se resta una unidad a la variable *segundos* y se comprueba si ha llegado a 0, restando una unidad a los *minutos* y propagando esta comprobación hacia las *horas*. Si se llega al punto en que las 3 variables son 0 estamos en el punto de haber finalizado un ejercicio o un periodo de descanso. En este caso se ejecuta el evento de alarma para notificar al usuario y entran en juego las variables *descansando* y *entreno_ actual*. La primera determina el estado en el que se encuentra la aplicación, mientras que la segunda almacena qué ejercicio se ejecuta en todo momento. Así, mediante estas variables podemos comprobar si ha finalizado el último ejercicio de la sesión, un ejercicio intermedio, o un periodo de descanso, cuyo procedimiento se explica a continuación:

  - Si *descansando* tiene valor *cierto* y *entreno_ actual* coincide con la cantidad de ejercicios, ha finalizado la sesión. Esto conlleva ejecutar el servicio del vibrador con un patrón definido, se ejecuta el sonido de fin de sesión y finalmente, se interrumpe el thread y se llama a la función *finish()*, que se encargará de liberar los recursos que consumía la actividad y volver a la ventana inicial, donde se muestra un mensaje de que la sesión ha finalizado.

  - Si *descansando* tiene valor *cierto* y no estamos en el último ejercicio, se ejecuta el sonido de alarma informativo de cambio de actividad, se cambia el valor de la variable *descansando* a *falso* y se hace una llamada al thread que cambiará el color de fondo de las filas del cronómetro. También se llama al thread que intercambia el color de fondo del cronómetro, de rojo a verde.

  - Si *descansando* tiene valor *falso* es porque hemos estado ejecutando un ejercicio, por lo que actualizamos el cronómetro a 15 segundos, cambiamos los colores de fondo y se pone a *cierto* la variable *descansando*.

- *Fondo del cronómetro (fondo_ cron)*: el objetivo de este thread es poner a rojo o verde el color de fondo del cronómetro dependiendo del valor de la variable *descansando*.

- *Fondo del calendario (actualiza_ fila)*: a medida que van finalizando los ejercicios, estos van pasando de color azul a gris. Inicialmente, el primer ejercicio de la lista está con un fondo azul, mientras que los restantes tienen un fondo grisáceo. Cuando finaliza un ejercicio y sus 15 segundos de descanso, éste pasa a color gris y el siguiente de la lista cambia a color de fondo azul. Esto facilita en gran medida al usuario identificar el ejercicio que está llevando a cabo.

La figura 4.15 muestra el diagrama seguido en la implementación de la clase de la fase de ejecución. Como se puede ver, la clase principal inicia los cuatro threads y los deja corriendo en segundo plano, para pasar ésta a la creación del calendario y asignar los eventos a los botones.

La forma de interactuar el hilo principal con los 4 restantes es mediante el paso de mensajes vacíos. Se llama a la función *sendEmptyMessage* aplicada sobre el thread que necesitamos ejecutar y así se consigue establecer la comunicación entre la clase principal y cada uno de los threads en background.

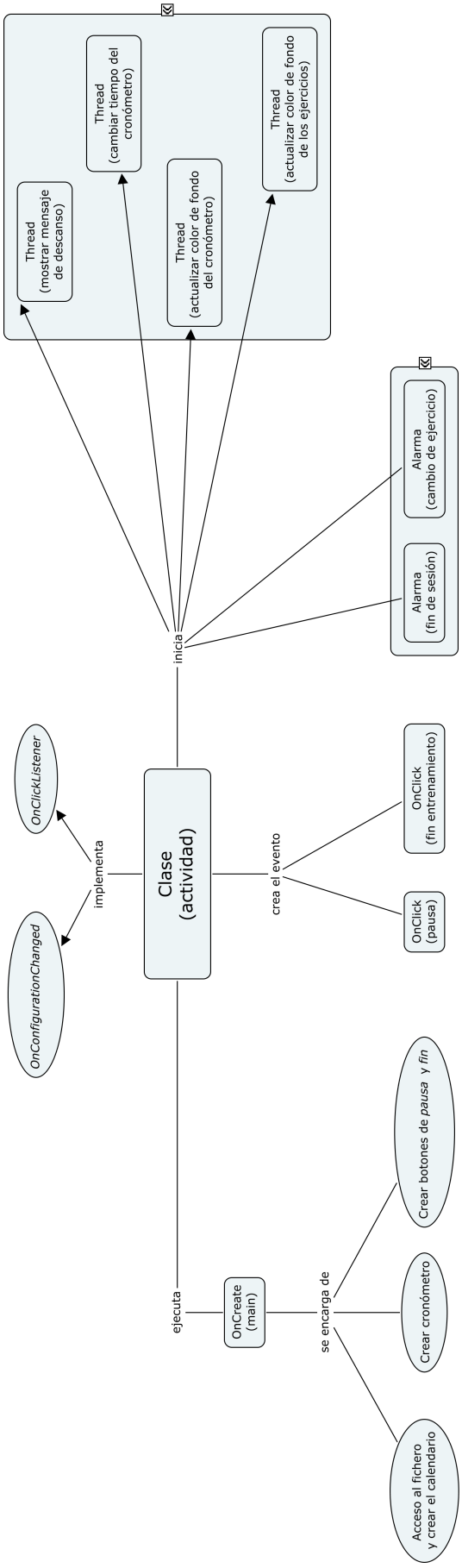

Figura 4.15.: Versión 2 - estructura de clases de la ventana de Ejecución

En la siguiente figura se muestra el flujo de llamadas entre el programa principal, el thread y los diferentes manejadores.

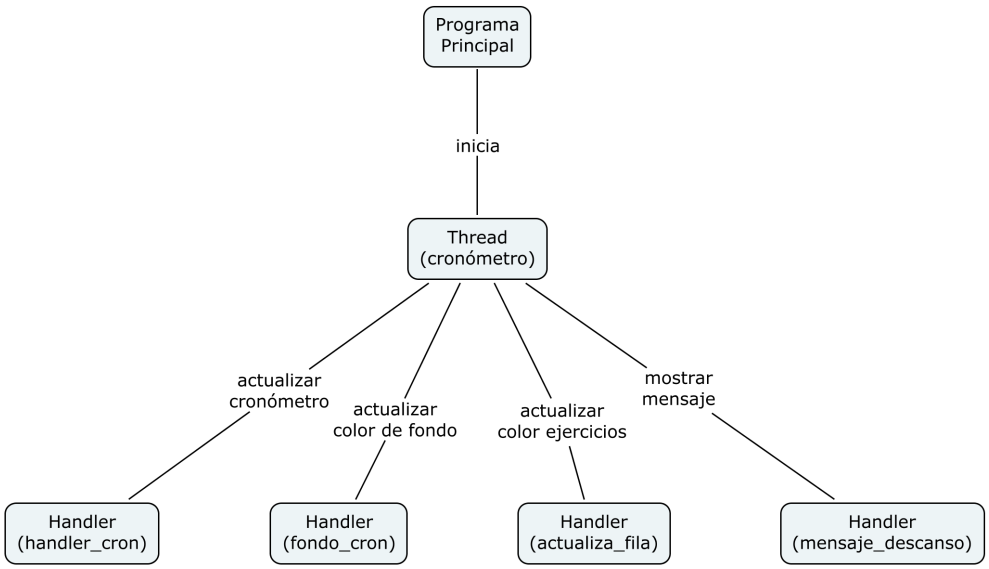

Figura 4.16.: Diagrama de flujo

4.2.4. Pruebas

Al igual que en la versión 1, no se disponía de un teléfono móvil en el cual probar la aplicación, así que todas las pruebas se realizaron sobre el emulador. Con esta segunda versión, además de las pruebas de la versión 1, también se pasaron las siguientes:

- Comprobar que el fichero se crea correctamente en la fase de planificación.
- Comprobar que se muestra un aviso cuando se guarda la sesión con 0 ejercicios.
- Comprobar que desde la pantalla de inicio se puede acceder tanto a la vista de planificación como a la de ejecución y desde ellas se vuelve a la de inicio.
- Cuando se pulsa el botón para guardar la sesión de ejercicios, comprobar que se muestra el mensaje de que se ha guardado correctamente.
- El botón de pausar el cronómetro detiene la cuenta atrás, al igual que la vuelve a iniciar con una segunda pulsación.
- El botón de Finalizar sale de la vista de ejecución y vuelve a la pantalla de inicio, mostrándose un mensaje de finalización de la sesión de ejercicios.
- Tras cada actividad se reproduce un sonido como señal acústica.
- Tras cada ejercicio se activa el vibrador.
- Tras cada ejercicio se ejecuta un periodo de 15 segundos de descanso.
- Al finalizar el último periodo de descanso se reproduce un sonido diferente y se reproduce una vibración.

- Tras cada actividad, se muestra un mensaje de *Descansando...*.
- Tras cada periodo de descanso, el ejercicio actual pasa a color gris y el siguiente a color azul.
- Durante cada periodo de descanso, el color de fondo del cronómetro cambia a color rojo.
- Comprobar que cuando se finaliza la sesión antes de terminarla de la forma habitual, se interrumpe el *thread* que ejecuta el cronómetro y no se reproducen los sonidos.
- Comprobar que los nombres de los ejercicios aparecen en mayúsculas.
- Comprobar si aparece un mensaje al intentar entrar en la fase de ejecución sin haber creado, previamente, una sesión de ejercicios.

Tras pasar las pruebas, los resultados obtenidos fueron satisfactorios, ejecutándose siempre la aplicación como debería y mostrando los avisos en el momento oportuno. Los eventos asociados a los botones también respondieron correctamente, tanto el de pausar el cronómetro como el de finalizar la sesión. Respecto al cambio de colores de fondo, no hubo ningún problema, cambiando el color de la aplicación al finalizar los periodos de descanso o al cambiar de ejercicio. Los sonidos y la vibración tampoco ocasionaron problemas, funcionando todos a la perfección.

El único fallo que se detectó fue con la penúltima prueba, referente a la finalización del thread cuando terminaba la aplicación. En el momento de ejecutar la prueba no se pasó satisfactoriamente debido a que no se llamaba a la función *interrupt()*, encargada de interrumpir la ejecución de un thread. Por tanto, no se paraba su ejecución y aunque la aplicación no estaba en funcionamiento, el thread seguía ejecutándose. La solución fue añadir dicha instrucción justo antes de salir de la ventana.

4.3. Aplicación final

En este apartado se va a describir el aspecto final y las funcionalidades que presenta la aplicación diseñada. En este punto la aplicación se encuentra terminada y las posibles ampliaciones se tratarán en el apartado de futuras mejoras. Se presenta a continuación (figuras 4.17, 4.18, 4.19 y 4.20) de manera conjunta los cuatro estados en los que puede estar la aplicación para tener una visión global de la misma, y acto seguido se explicará brevemente cada pantalla.

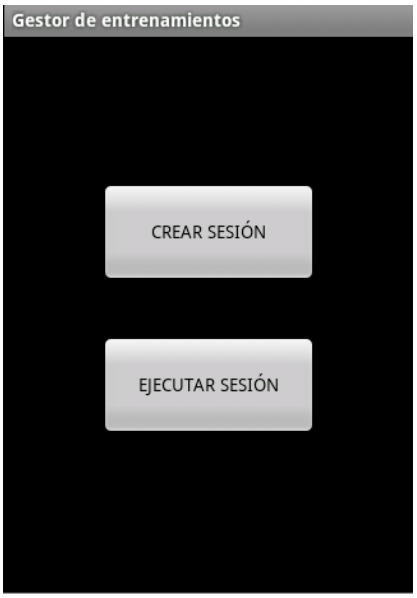

Figura 4.17.: Pantalla de inicio

Figura 4.18.: Vista de planificación

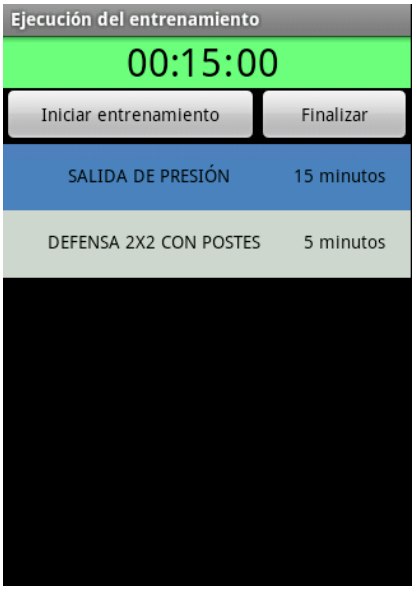

Figura 4.19.: Vista de ejecución

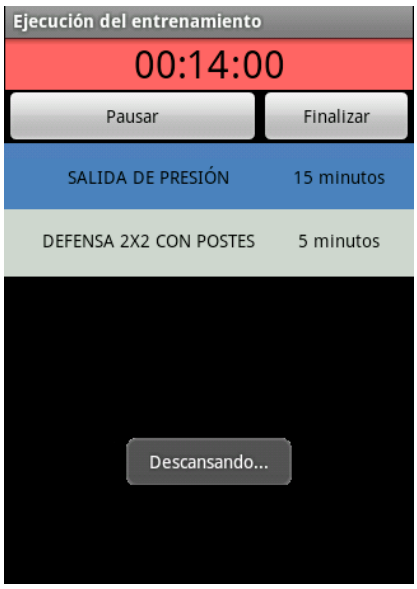

Figura 4.20.: Periodo de descanso

La figura 4.17 muestra la ventana principal que aparece cuando se ejecuta la aplicación. Desde ella se puede acceder a la planificación de una sesión o a la ejecución de una sesión.

La siguiente figura mostrada (4.18) corresponde a la fase de planificación, en la que se añaden ejercicios, se modifica su duración y se almacenan para ejecutarlos en la siguiente fase.

La figura 4.19 se corresponte a la fase de ejecución, donde a partir de los ejercicios guardados en la pantalla anterior, se crea el calendario del entrenamiento y se ejecutan en el orden establecido. El diseño de esta pantalla se ha optimizado para que sea lo más fácil y minimalista posible de cara al usuario final, ya que se ha supuesto que durante un ejercicio no se dispone de mucho tiempo para mirar el calendario, ver el ejercicio actual o tener constancia de cuanto queda para pasar al siguiente ejercicio. Por eso se han utilizado colores que facilitan la lectura de todos los elementos, se han escrito los ejercicios con letras mayúsculas (lo cual también otorga un mejor acabado al diseño) y se han añadido botones que permiten pausar o finalizar la sesión de ejercicios.

Por último, la figura 4.20 muestra el estado de la fase de ejecución en un periodo de descanso. Éste tiene lugar después de cada ejercicio y su duración es de 15 segundos, independientemente del tiempo que haya durado el ejercicio. En caso de requerir más tiempo para preparar el próximo ejercicio, se puede hacer uso del botón de pausa del cronómetro.

4.4. Memoria final

Para el desarrollo de esta memoria se ha utilizado LaTeX por las ventajas que ofrece en la creación de documentos y el diseño profesional de los trabajos generados, como se ha comentado en el apartado 1.6. Por otra parte, se ha dividido en diferentes secciones que facilitan su organización, como se explican a continuación:

- Introducción: donde se definen los objetivos y propósitos del proyecto
- Programación en Android: donde se explican detalladamente los pasos a seguir para comenzar a programar aplicaciones para Android.
- Planificación: donde se establece el tiempo de desarrollo del proyecto
- Desarrollo del proyecto: donde se comentan detalladamente las dos versiones de las que consta el proyecto y, para cada una de ellas, se explica el análisis de requisitos, el diseño de la interfaz, las estrategias seguidas en su implementación, las pruebas realizadas y resultados de las pruebas.
- Conclusiones y trabajos futuros: donde se indica si se han logrado los objetivos y las mejoras que se podrían llevar a cabo con el fin de mejorar la funcionalidad de la aplicación.

4.5. Presentación

La presentación del proyecto se basará en una explicación de unos 20 minutos utilizando diapositivas para presentar todas las características de la aplicación. Se comenzará con una introducción a modo de motivación sobre los contenidos que se describirán. Posteriormente se explicarán con detalle las versiones implementadas, destacando las diferencias entre ellas y el motivo de estos cambios. Finalmente se comentarán algunas mejoras que se podrían llevar a cabo sobre la aplicación y las principales dificultades que han aparecido a la hora de realizar el proyecto.

5. Conclusiones y trabajo futuro

La principal conclusión que se puede sacar una vez finalizado el proyecto es que se han conseguido los objetivos planteados inicialmente. Uno de los ellos era crear una aplicación capaz de gestionar un entrenamiento, desde el punto inicial de añadir ejercicios a una lista modificando sus duraciones hasta el punto de ejecutarlos empleando para ello un cronómetro y retroalimentando al usuario. Además de cumplir este objetivo, se ha conseguido que la aplicación sea sencilla y, sobretodo, cómoda de utilizar.

Además, es fácil de instalar, al igual que cualquier aplicación Android y la ventaja que tiene es que es escalable, de forma que no es complicado hacer cambios en la implementación debido a su estructura de clases.

Por otra parte, un objetivo derivado era poner en práctica las técnicas aprendidas en la carrera para aprender la programación en Android, objetivo que se ha conseguido gracias a las bases de programación en Java, temas de usabilidad y desarrollo de estructuras de datos que se han ido viendo a lo largo de la carrera.

La aplicación cumple todos los requisitos definidos en las versiones y es totalmente funcional tanto en el emulador utilizado para su desarrollo como en teléfonos móviles con sistema operativo Android que és, realmente, donde se va a ejecutar.

Respecto a los conocimientos previos requeridos, han sido necesarias nociones de Java adquiridas en la asignatura II72 (Sistemas Distribuidos) y nociones de gestión y planificación de proyectos (II32 - Gestión y Desarrollo de Proyectos Informáticos), además de todos los conocimientos relacionados con la programación orientada a objetos aprendidos en las asignaturas de programación de los primeros cursos de la carrera. Aún así, debido a la complejidad del proyecto, se han consultado todo tipo de recursos online, como páginas web sobre Java, Android, foros, etc.

A nivel personal ha sido una experiencia muy enriquecedora en cuanto a la adquisición de conocimientos se refiere, pues me ha permitido adentrarme en la programación orientada a dispositivos móviles aprendiendo a apreciar las diferencias visuales entre las pantallas de ordenador y las de dispositivos móviles a la hora de implementar programas destinados a cada una. También me ha servido para aprender el lenguaje de programación Java y a utilizar una de las herramientas más usadas en la actualidad para la programación de aplicaciones como es Eclipse.

5.1. Trabajo futuro

La aplicación puede ser ampliada para dotarla de nuevas funcionalidades que la permitan abarcar más aspectos de una sesión de ejercicios. Por tanto, algunas de las mejoras que se podrian llevar a cabo para mejorar tanto el rendimiento como las capacidades se comentan a continuación.

- Acceso a una base de datos con las últimas sesiones para poder cargarlas posteriormente. De esta forma se podría implementar la aplicación con un selector de sesiones creadas,

pudiendo elegir alguna de ellas y evitar el paso por la fase de planificación. Además también se podria compartir esta base de datos con otros teléfonos móviles mediante algún sistema de autenticación, teniendo como resultado una base de datos compartida entre usuarios donde cada uno pondría al servicio del resto sus sesiones de ejercicios.

- Scroll que deslice la ventana hasta el ejercicio en ejecución, ya que en el estado actual de la aplicación, aquellos ejercicios que se encuentran ocultos por no caber en pantalla, cuando se ejecutan no se desplaza el scroll hasta su posición, sino que permanecen ocultos siendo necesario este desplazamiento de forma manual.

- Capacidad para poder añadir anotaciones a los ejercicios. Se podría implementar mediante un pequeño botón en cada ejercicio de modo que al pulsarlo apareciese una ventana donde introducir comentarios e información relacionada con el ejercicio, para tenerlos disponibles en la fase de ejecución.

- Optimización de recursos. Aunque se ha intentado que el código esté estructurado y bien comentado para facilitar la escalabilidad y su depurado, hay algunos elementos que pueden ser optimizados. Como por ejemplo, utilizar un solo thread para cambiar los colores de fondo de la aplicación en la fase de ejecución. Esto tendría la ventaja de tener un sólo proceso en segundo plano (aparte del que gestiona el cronometraje) lo cual aprovecharía de mejor manera los recursos del sistema.

- Poder definir manualmente la duración del tiempo de descanso para adaptar al máximo la fase de ejecución al entrenamiento individualizado de cada usuario.

A. Creación de un certificado de firma

En este apéndice se explca paso a paso el proceso para crear un certificado de firma que nos permitirá poner nuestro sello a las aplicaciones que implementemos para Android.

La duda que nos puede surgir es: ¿Para qué tengo que firmar mi aplicación?. La respuesta es porque las aplicaciones y firmwares solo pueden instalarse en los móviles Android si han sido previa y debidamente firmados. Esto es una medida de seguridad que otorga validez al archivo aunque en la práctica es más un requisito que una garantía.

Existen en internet una gran cantidad de videotutoriales donde se explica cómo firmar nuestras aplicaciones. La diferencia con este manual es que aquí se explica detalladamente cada paso para tener constancia en todo momento de lo que se está haciendo. Además, en este anexo se especifica cómo crear una firma sin necesidad de instalar programas auxiliares como aconsejan muchos tutoriales online.

Si tenemos instalado correctamente el Java Development Kip, tendremos disponible el script *keytool*, el cual se encuentra dentro del directorio *bin* en la carpeta en la que se ha instalado el JDK. Así, el primer paso será abrir una terminal de comandos, acceder al directorio *bin* dentro de la ruta en la que se instaló el JDK y escribir el siguiente comando (es el mismo tanto para PC como para MAC):

keytool -genkey -v -keystore my-key.keystore -alias micertificado -keyalg RSA -validity 10000

Con este comando estamos indicando que vamos a crear un certificado almacenado en el fichero *my-key.keystore*, con el nombre *micertificado*, utilizando el algoritmo de cifrado RSA[1] y con una duración de 10000 días.

A continuación se explica con detalle cada paso relacionado con el comando anterior:

[1] `http://es.wikipedia.org/wiki/RSA`

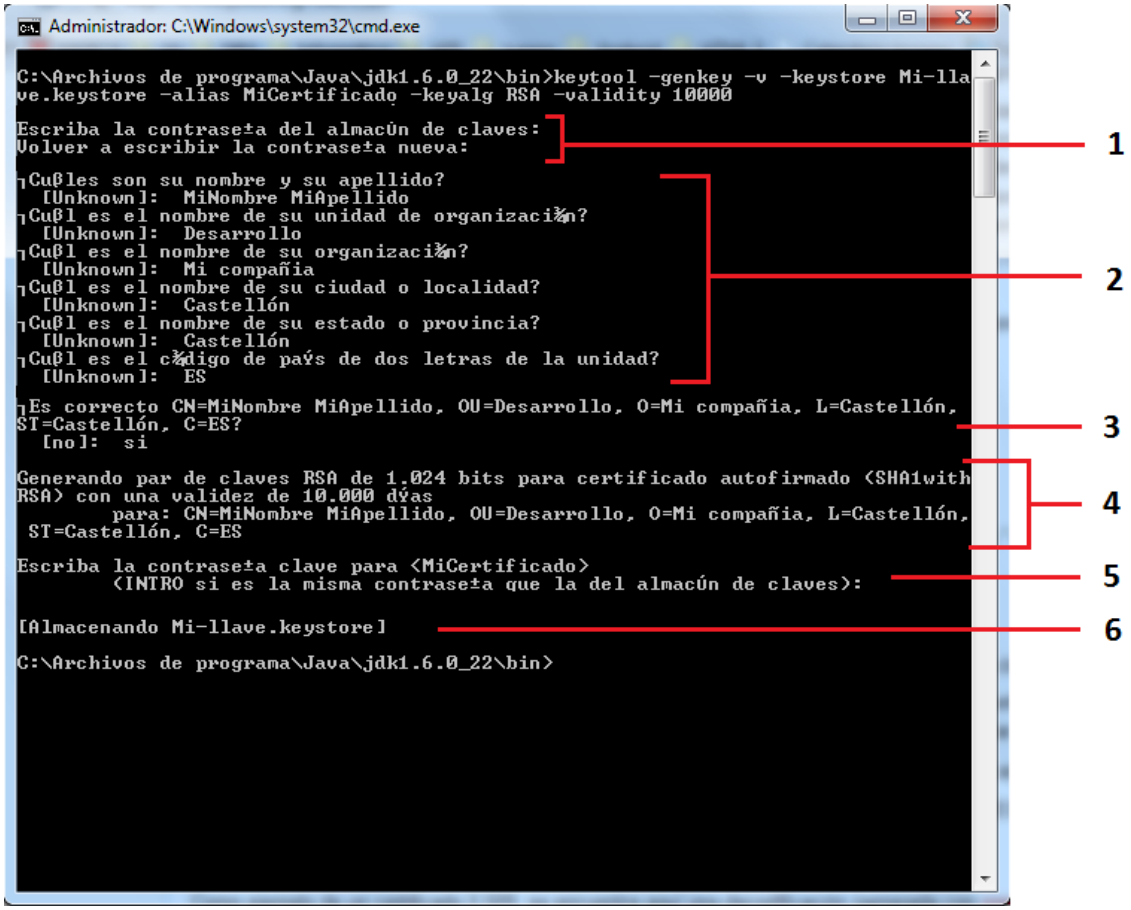

Figura A.1.: Pasos para la creación del certificado

1. La contraseña del almacén de claves es, por decirlo de alguna forma, la puerta de acceso al repositorio en el que tenemos nuestros certificados. Por esto es importante tener una buena contraseña con el fin de evitar el acceso a personas no autorizadas.

2. Este es el proceso de creación del certificado. Nos pregunta algunos datos personales que se añadirán al certificado y nos vinculará con él. Puesto que el certificado no va a estar verificado por ninguna autoridad certificadora, podemos poner los datos que queramos, reales o no, aunque si el certificado se va a utilizar para firmar aplicaciones importantes (como las que se encuentra en el Android Market[2]), se requerirán los datos auténticos.

3. Tras introducir los datos, se nos muestran para comprobar que son correctos. En caso afirmativo, escribir *si* y pulsar *Intro*.

4. En este punto es donde se crea el certificado mediante la información recopilada en los puntos anteriores. Nos indica que va a utilizar 1024 bits para generar la clave RSA y que la duración del certificado será de 10.000 días. También muestra el receptor del mismo.

5. Este es uno de los puntos más importantes, pues aquí deberemos introducir la contraseña

[2] `https://market.android.com/`

que protegerá el certificado. Es recomendable que ésta tenga, como mínimo, 6 caracteres entre los que haya números y letras alternados.

6. Finalmente, si todo el proceso se ha ejecutado según lo previsto, nos muestra el fichero creado en el directorio desde el cual hemos ejecutado el comando anterior. En la figura A.1, éste se llama Mi-llave.keystore y estará almacenado en la carpeta *bin*. Para mayor seguridad, deberemos guardar el certificado en un lugar seguro ya que en caso de que alguien tuviese acceso a él y conociese las claves, podria firmar aplicaciones bajo nuestra identidad.

B. Guía de uso

La aplicación está desarrollada íntegramente para poder ser utilizada de forma táctil, por lo que se caracteriza por un funcionamiento sencillo y cómodo, a la par que útil para la correcta gestión de las sesiones de entrenamiento.

B.1. Pantalla de inicio de la aplicación

Cuando se ejecuta la aplicación se muestra la pantalla de inicio, desde la que podemos acceder a las restantes partes del programa. Las partes en las que se divide se muestran en la siguiente imagen:

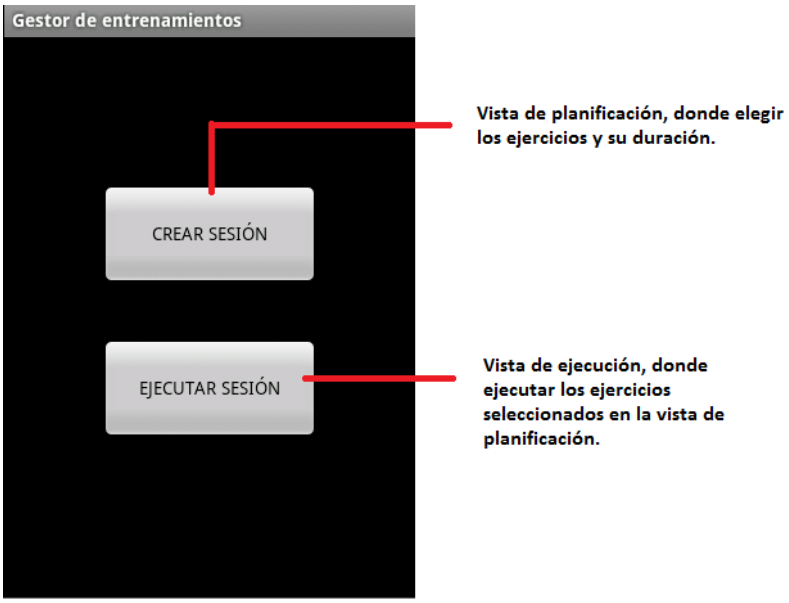

Figura B.1.: Manual: Pantalla de inicio

Puesto que el primer paso para gestionar una sesión de entrenamiento es seleccionar los ejercicios deseados y sus duraciones, en primer lugar habrá que pulsar sobre el botón *Crear sesión*, el cual nos llevara a la siguiente pantalla.

Si se pulsa inicialmente el botón de *Ejecutar sesión*, obtendremos un aviso de que no se han podido cargar los ejercicios, y es que, como se ha comentado anteriormente, es necesario haber creado primero el fichero con los ejercicios para poder cargarlo en la fase de ejecución.

B.2. Pantalla de planificación

Esta es la sección en la que se deben escoger los ejercicios y sus duraciones. Una vez hayamos terminado, habrá que pulsar el botón superior para almacenar la sesión y poder ejecutarla posteriormente. Las partes de las que consta esta pantalla son las siguientes:

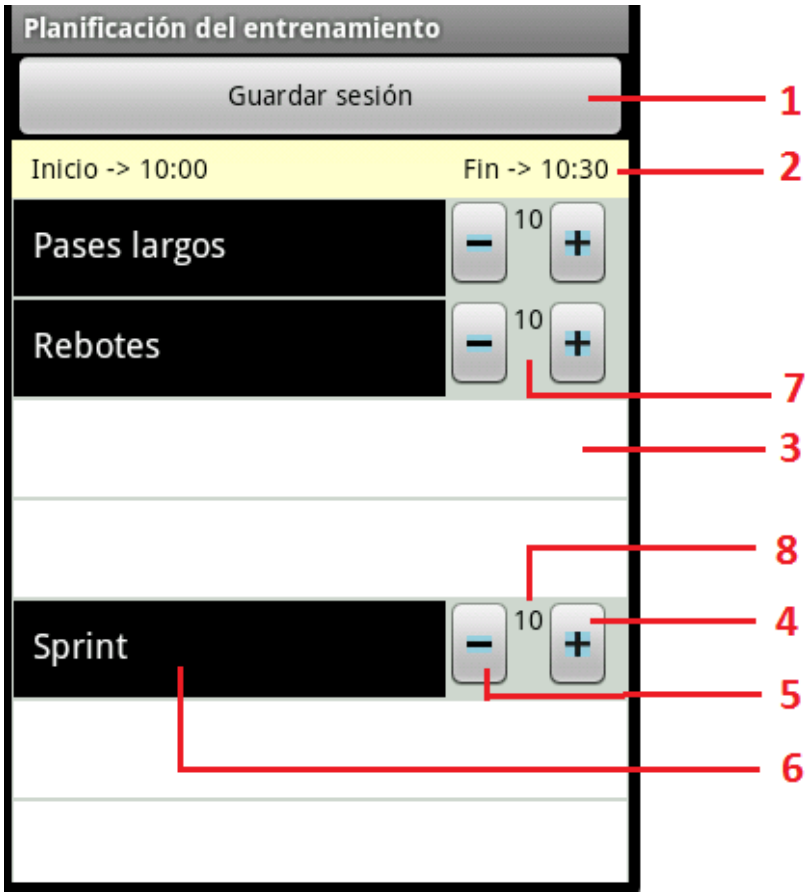

Figura B.2.: Manual: Pantalla de planificación

- 1: Botón de guardado de la sesión. Pulsándolo, se almacena la sesión de ejercicios que hemos introducido, finaliza la parte de planificación y nos aparece la pantalla inicial. En caso de no haber seleccionado ningún ejercicio aparecerá una ventana de aviso.
- 2: Indicador de la hora de finalización de la sesión. Dependiendo de la duración de todos los ejercicios seleccionados la hora de finalización variará.
- 3: Las filas simbolizan los ejercicios disponibles. Pulsando en cada fila nos aparecerá un cuadro de texto en el que escribir el nombre del ejercicio, dos botones para aumentar y disminuir la duración del ejercicio y la propia duración.
- 4: Pulsando este botón se incrementa la duración del ejercicio en intervalos de 5 minutos. Además, se actualiza la hora de fin y la altura de la propia fila.

- 5: Análogo al anterior pero reduciendo la duración.
- 6:Existe una lista predefinida de ejercicios que nos aparecerán si escribimos sus primeras letras en el cuadro de texto. Obviamente, no es obligatorio seleccionar uno de estos ejercicios por defecto, pudiendo escribir cualquier ejercicio que nos interese.
- 7: Pulsando durante 3 segundos en esta zona gris se elimina el ejercicio.
- 8: Indicador de la duración de cada ejercicio

Además la ventana dispone de un scroll vertical que nos permitirá añadir más ejercicios de los que caben en pantalla. Aunque las zonas blancas son espacios disponibles en los que añadir entrenamientos, estos se pueden insertar en cualquier fila, pudiendo alternar ejercicios con franjas vacías como se muestra en la imagen anterior.

B.3. Pantalla de ejecución

Esta pantalla es similar a la de planificación pero se han omitido algunos elementos para facilitar la identificación de los ejercicios y la duración. Se explican a continuación los elementos de la interfaz:

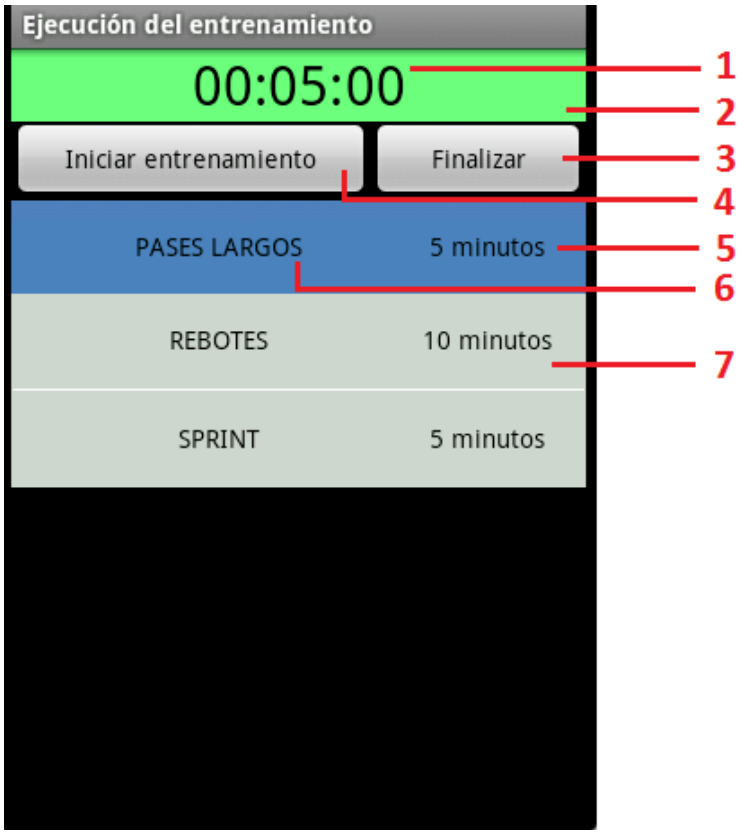

Figura B.3.: Manual: Pantalla de ejecución

- 1: El cronómetro. Uno de los elementos principales de esta fase, pues será la cuenta atrás que nos permitirá saber el tiempo restante del ejercicio actual.

- 2: Fondo del cronómetro. Puede adoptar dos colores: verde, cuando se está ejecutando un ejercicio, o rojo cuando se está en un periodo de descanso, como se puede apreciar en las figuras B.3 y B.4.
- 3: El botón de finalizar sirve para dar por terminada la sesión de entrenamiento actual. Su pulsación implica salir de la fase de ejecución y nos aparece la pantalla de inicio, desde la que poder crear otra sesión o salir del programa.
- 4: El botón de iniciar el entrenamiento sólo muestra este mensaje al inicio de la sesión, y su función es la que indica su nombre: comenzar la cuenta atrás del primer ejercicio. Una vez iniciada, el texto mostrado cambia por *Pausar* como se muestra en la figura B.4, cuya función sera pausar la sesión en el punto en el que se encuentre, ya sea durante un entrenamiento o durante el descanso.
- 5: Indicador de la duración del ejercicio. Puesto que esta etapa se centra en la ejecución de la sesión, se ha desactivado su edición. Por tanto, este valor no variará en toda la sesión.
- 6: Aquí aparece el nombre del ejercicio en mayúsculas para poder diferenciarlo más fácilmente en caso de estar realizando alguna actividad que implique el movimiento del teléfono, como puede ser correr, saltos, etc.
- 7: El color de fondo de los ejercicios también irá cambiando automáticamente dependiendo del ejercicio en el que nos encontremos.

La siguiente imagen muestra el estado de la aplicación durante un periodo de descanso, con el fondo del cronómetro de color rojo y un aviso del periodo.

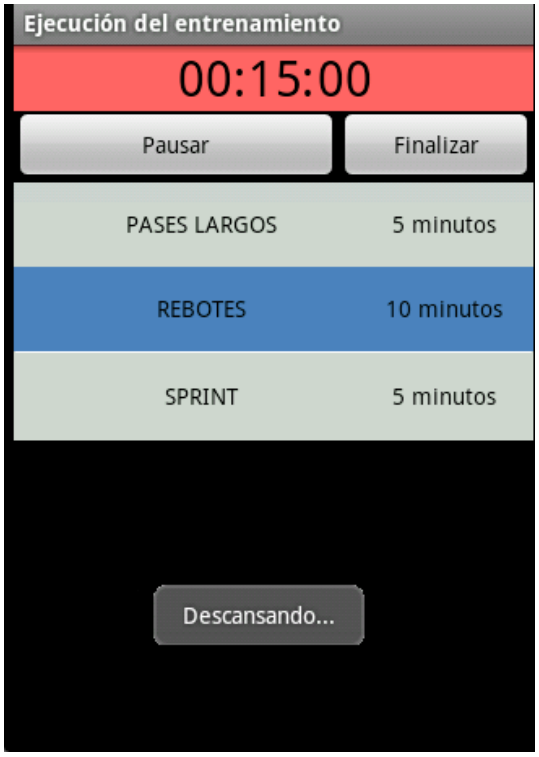

Figura B.4.: Manual: Pantalla de ejecución (descansando)

Una vez acaba de ejecutarse la sesión, se produce un sonido informativo y se activa el vibrador del teléfono móvil y se vuelve a la pantalla de inicio, desde la que se puede volver a ejecutar la misma sesión o empezar a planificar una nueva.

C. XML de la versión 1

En este apéndice se muestra el código del xml de la versión 1 en el que se pueden apreciar, además de los atributos mencionados en el apartado 2.2, otros destinados a la ubicación en pantalla de los elementos, como *layout_margin*, *gravity* o *stretchColumns* entre otros.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="fill_parent"
android:layout_width="fill_parent">
    <ScrollView android:layout_height="fill_parent"
    android:layout_width="fill_parent">
    <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/tl"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"
              android:background="#CCD4CF"
              android:layout_margin="1dip"
              android:stretchColumns="1" >
             <TableRow>  <TextView
                       android:id="@+id/texto1"
                       android:layout_width="fill_parent"
                       android:layout_height="30px"
                       android:textColor="#000000"
                       android:background="#C7F6FF"
                       android:gravity="center"
                       android:layout_marginTop="2dip"
                  android:layout_marginRight="2dip"
                  android:layout_marginLeft="2dip" />
                  <TextView
                        android:id="@+id/ent1"
                        android:layout_width="fill_parent"
                        android:layout_height="30px"
                        android:background="#ffffff"
                        android:layout_marginTop="2dip"
                        android:layout_marginBottom="2dip"
                        android:gravity="center"
                        android:textColor="#0000FF"   />
                       <TextView
                            android:id="@+id/ent2"
                            android:layout_width="200px"
                            android:layout_height="30px"
                            android:background="#ffffff"
                            android:layout_marginTop="2dip"
```

```
                                android:layout_marginBottom="2dip"
                                android:gravity="center"
                                android:textColor="#0000FF" />
</TableRow> </TableLayout></ScrollView> </LinearLayout>
```

D. Código de la versión 1

Tras mostrar el xml de la primera versión se expone a continuación el código utilizado para su implementación.

```
package pask.proyectofinal;

import pask.proyectofinal.R;

/* Clases generales*/
import android.app.Activity;
import android.os.Bundle;
import android.content.res.Configuration;

/* Clases para construir la interfaz
 * Layout, Table, Cajas de texto, botones, etc */
import android.graphics.Color;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.EditText;
import android.widget.FrameLayout.LayoutParams;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

/* Clases para los eventos onClick */
import android.view.View.OnClickListener;
import android.view.View.OnLongClickListener;

public class v1 extends Activity implements OnClickListener, OnLongClickListener{
    int alto_celda=50; // altura predefinida de cada celda
    int id=1; // Contador de los identificadores de elementos de la interfaz
    int num_entrenos=30; // Numero de franjas disponibles

    /* Variables para definir el estilo de los elementos de la interfaz */
    LayoutParams ff = new LayoutParams(LayoutParams.FILL_PARENT,
                                       LayoutParams.FILL_PARENT);
    LayoutParams ww = new LayoutParams(LayoutParams.WRAP_CONTENT,
```

```
                                                    LayoutParams.WRAP_CONTENT);
LayoutParams wf = new LayoutParams(LayoutParams.WRAP_CONTENT,
                                                    LayoutParams.FILL_PARENT);
LayoutParams fw = new LayoutParams(LayoutParams.FILL_PARENT,
                                                    LayoutParams.WRAP_CONTENT);

/* Array de entrenos por defecto */
String[] entrenos = {
        "1 vs 1",
        "3 vs 3",
        "Pases largos",
        "Pases cortos",
        "Bandeja",
        "Pick&Roll",
        "Contraataques",
        "Defensa",
        "Run&Gun" };

    /* Hora de inicio: 10:00  */
    int minIni=0;
    int minFin=1;
    int horaIni=10;
    int horaFin=10;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.v1);
    TableLayout tabla = (TableLayout) findViewById(R.id.tl);

    /* Crear las filas */
    for(int i=0;i<num_entrenos;i++){
            /* Crear los elementos de la fila y asignarles un id*/
            TableRow tr = new TableRow(this);
            tr.setOnClickListener(this);
            tr.setOnLongClickListener(this);
            tr.setId(id+1000);

            RelativeLayout izq = new RelativeLayout(this);
            izq.setId(id+2000);

            TextView ti = new TextView(this);
            ti.setId(id+7000);

            RelativeLayout der = new RelativeLayout(this);
            RelativeLayout der2 = new RelativeLayout(this);

            id++;
```

```
                    /* Actualizar la hora de fin de la sesion */
                    if (minIni==5){
                                    horaFin=horaFin+1;
                            minFin=0;
                    }

                    String hora = horaIni+":"+minIni+"0 - "+horaFin+":"+minFin+"0";
                    ti.setText(hora);
                    minFin=minFin+1;
                    if (minIni==5){
                            minIni=0;
                                horaIni=horaIni+1;
                    }
                    else  minIni=minIni+1;

                    /* Propiedades visuales de la interfaz */
                    ti.setWidth(85); // Ancho del textview para la franja horaria
                    ti.setGravity(Gravity.CENTER_HORIZONTAL);
                    ti.setBackgroundColor(Color.rgb(199, 246, 255));
                    ti.setTextColor(Color.BLACK);
                    ti.setLayoutParams(fw);
                    ti.setHeight(alto_celda);
                    ti.setGravity(Gravity.CENTER);
                    izq.setBackgroundColor(Color.rgb(204, 212, 207));
                    izq.setPadding(2,0,2,2);
                    izq.addView(ti);
                    izq.setGravity(Gravity.CENTER);
                    tr.addView(izq);

                    /* Estilo de la celda derecha */
                    der.setBackgroundColor(Color.WHITE);
                    der.setMinimumHeight(alto_celda);
                    der2.setBackgroundColor(Color.WHITE);
                    der2.setMinimumHeight(alto_celda);
                    tr.addView(der);
                    tr.addView(der2);
                    tabla.addView(tr,fw);
                }

        } // fin on-create

/* Evento ejecutable en cada click:
* Crear el ejercicio y propiedades visuales */
public void onClick(View v){
    RelativeLayout lista =  new RelativeLayout(this);

    /* AutoComplete con lista de entrenos predefinidos */
    AutoCompleteTextView ent = new AutoCompleteTextView(this);
```

```
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                                        this,
                                        android.R.layout.simple_dropdown_item_1line,
                                        entrenos);
        ent.setThreshold(3);
        ent.setAdapter(adapter);
        ent.setSingleLine(true);
        ent.setLayoutParams(fw);
        ent.setHeight(50);
        ent.setId(v.getId()+8000);
        ent.setBackgroundColor(Color.rgb(199, 246, 255));

        /* Crear botones de duracion */
        LinearLayout tiempo = new LinearLayout(this);
        tiempo.setOrientation(LinearLayout.HORIZONTAL);
        ImageButton menos = new ImageButton(this);
        ImageButton mas = new ImageButton(this);
        TextView dur = new TextView(this);
        dur.setId(v.getId()+5000); // v tiene la 1000+
        mas.setId(v.getId()+3000);
        menos.setId(v.getId()+4000);
        dur.setHeight(30);
        dur.setHorizontallyScrolling(true);
        dur.setText("10");

        dur.setSingleLine(true);
        menos.setImageResource(R.drawable.menos);
        mas.setImageResource(R.drawable.mas);
        menos.setOnClickListener(decrementa);
        mas.setOnClickListener(incrementa);

        /* Incluir todos los elementos a la vista */
        tiempo.addView(menos,ww);
        tiempo.addView(dur,ww);
        tiempo.addView(mas,ww);
        lista.setGravity(Gravity.CENTER);
        lista.addView(ent);
        ((ViewGroup) v).removeViewAt(1);
        ((ViewGroup) v).removeViewAt(1);
        ((ViewGroup) v).addView(lista);
        ((ViewGroup) v).addView(tiempo);
}

/* Evento a ejecutar ante un click largo */
public boolean onLongClick(View v){
        RelativeLayout der = new RelativeLayout(this);
        RelativeLayout der2 = new RelativeLayout(this);
        der.setBackgroundColor(Color.WHITE);
        der.setMinimumHeight(alto_celda);
```

```
        der2.setBackgroundColor(Color.WHITE);
        der2.setMinimumHeight(alto_celda);

        /* Eliminar la fila */
        ((ViewGroup) v).removeViewAt(1);
        ((ViewGroup) v).removeViewAt(1);
        ((ViewGroup) v).addView(der);
        ((ViewGroup) v).addView(der2);
        TextView izq = (TextView) findViewById(v.getId()+6000);
        izq.setHeight(alto_celda);
        v.setMinimumHeight(alto_celda);
        return true;
}

/* Evento del boton de incremento de la duracion */
private OnClickListener incrementa = new OnClickListener() {
        public void onClick(View v) {
            TextView et = (TextView) findViewById(v.getId()+2000);

            /* Sumar 5 minutos */
            et.setText(String.valueOf(Integer.parseInt(et.getText().toString())+5));

            /* Incrementar el alto de la celda si es mayor que 11*/
            if(Integer.parseInt(et.getText().toString()) > 11){
                    TableRow t = (TableRow) findViewById(v.getId()-3000);
                    TextView izq = (TextView) findViewById(v.getId()+3000);
                    TextView ent = (TextView) findViewById(v.getId()+5000);
                    ent.setHeight(38+(Integer.parseInt(et.getText().toString())));
                    izq.setHeight(38+(Integer.parseInt(et.getText().toString())));
                    t.setMinimumHeight(38+(Integer.parseInt(et.getText().toString())));
            }
        }
};

/* Evento del boton de decremento de la duracion */
private OnClickListener decrementa = new OnClickListener() {
        public void onClick(View v) {
            TextView et = (TextView) findViewById(v.getId()+1000);
            int duracion = Integer.parseInt(et.getText().toString());
            // controlar las duraciones negativas
            if(duracion-1 <= 0) et.setText("0");
            else
                et.setText(String.valueOf(duracion)-5));

            /* Decrementar el alto de la celda si es mayor  que 10 */
            if(Integer.parseInt(et.getText().toString()) > 10){
                    TableRow t = (TableRow) findViewById(v.getId()-4000);
                    TextView ti = (TextView) findViewById(v.getId()+2000);
                    EditText ent = (EditText) findViewById(v.getId()+4000);
```

```
                    ent.setHeight(38+(Integer.parseInt(et.getText().toString())));
                    ti.setHeight(38+(Integer.parseInt(et.getText().toString())));
                    t.setMinimumHeight(38+(Integer.parseInt(et.getText().toString(
                }
            }
};

/* Evitar perder los datos al girar la pantalla */
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
}
}// fin -class-
```

E. XML de la versión 2

Se muestra a continuación todo el código que comprenden los xml de la versión 2. En primer lugar se muestra el referente a la pantalla inicial, luego se muestra en xml de la ventana de planificación y por último el de la pantalla de ejecución.

E.1. Pantalla de inicio

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout   xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:background="#000000"
    android:layout_width="fill_parent"
    android:gravity="center">
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:orientation="vertical"
        android:layout_centerHorizontal="true">
        <Button
                android:id="@+id/nueva"
            android:onClick="nueva"
            android:layout_width="170px"
            android:layout_height="80px"
            android:text="CREAR SESI\'ON"
            android:layout_y="90px"   />
        <Button
            android:id="@+id/ejecuta"
            android:layout_below="@+id/nueva"
            android:layout_marginTop="40px"
            android:onClick="ejecuta"
            android:layout_width="170px"
            android:layout_height="80px"
            android:text="EJECUTAR SESI\'ON"
            android:layout_y="220px"       />
    </RelativeLayout>
</RelativeLayout>
```

E.2. Pantalla de planificacion

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout  xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_height="fill_parent"
      android:layout_width="fill_parent">
      <Button
            android:id="@+id/guardaSesion"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Guardar sesion"  />
           <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            android:layout_y="45px">
          <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
                  android:id="@+id/tl"
                  android:layout_height="wrap_content"
                  android:layout_width="wrap_content"
                  android:background="#CCD4CF"
                  android:stretchColumns="0" />
          </ScrollView>
</AbsoluteLayout>
```

E.3. Pantalla de ejecucion

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout  xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_height="fill_parent"
      android:layout_width="fill_parent">
      <RelativeLayout
          xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_height="fill_parent"
          android:layout_width="fill_parent">
          <TextView
               android:id="@+id/ConTiempo"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
                    android:textColor="#000000"
                    android:background="#6EFE7F"
               android:textSize="30sp"
               android:gravity="center" />
              <Button
                    android:id="@+id/BtnAccion"
                    android:layout_below="@id/ConTiempo"
                    android:text="Iniciar entrenamiento"
                    android:layout_width="200px"
                    android:layout_height="45px"/>
                   <Button
                    android:id="@+id/BtnDetener"
                    android:layout_below="@id/ConTiempo"
                    android:layout_toRightOf="@id/BtnAccion"
                    android:text="Finalizar"
                    android:layout_width="fill_parent"
                    android:layout_height="45px"/>
          </RelativeLayout>
          <ScrollView
                xmlns:android="http://schemas.android.com/apk/res/android"
                     android:layout_height="fill_parent"
               android:layout_width="fill_parent"
               android:layout_y="85px">
              <TableLayout
                    xmlns:android="http://schemas.android.com/apk/res/android"
                    android:id="@+id/tl"
                    android:layout_height="wrap_content"
                    android:layout_width="wrap_content"
                    android:background="#CCD4CF"
                    android:stretchColumns="0" />
</ScrollView></AbsoluteLayout>
```

F. Codigo de la version 2

Por último, para acabar con el código de la aplicación, se muestra la implementación de cada una de las distintas partes que componen la versión final. Se muestra primero el codigo de la ventana de inicio y posteriormente se mostrará la implementación de las ventanas de planificación y ejecución, respectivamente.

F.1. Pantalla de Inicio

```
package pask.proyectofinal;

import pask.proyectofinal.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.View;
import android.widget.Toast;

public class v2inicio extends Activity{

     String nombre = "";

     /* Llamada a la actividad de planificacion */
     public void nueva(View target) {
          Intent i = new Intent(this, v2planifica.class);
          startActivityForResult(i, 1);
    }

    /* Llamada a la actividad de ejecucion */
    public void ejecuta(View target) {
         Intent i = new Intent(this, v2ejecuta.class);
         i.putExtra("nombre", nombre);
         if(nombre.length() == 0){
                 Toast.makeText(getApplicationContext(),
                                 "No se han podido cargar los ejercicios.",
                                 Toast.LENGTH_LONG).show();
         }
         else
            startActivityForResult(i, 2);
    }

    protected void onActivityResult(int requestCode,
```

```
                                        int resultCode,
                                        Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(resultCode==RESULT_OK && requestCode==1){
                nombre = data.getStringExtra("valor");
             String msg = "Sesion guardada correctamente!";
                Toast.makeText(getApplicationContext(), msg,
                                Toast.LENGTH_SHORT).show();
        }
        else{
              ((Vibrator)getSystemService(VIBRATOR_SERVICE)).vibrate(1000);
              String msg = "Fin del entrenamiento!";
              Toast.makeText(getApplicationContext(), msg,
                              Toast.LENGTH_SHORT ).show();
        }
    }

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.v2inicio);

    } // on-create


}
```

F.2. Vista de planificación

```
package pask.proyectofinal;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.Calendar;
import java.util.GregorianCalendar;
import android.app.Activity;
import android.content.Intent;
import android.content.res.Configuration;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnLongClickListener;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.FrameLayout.LayoutParams;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;


public class v2planifica extends Activity implements
        OnClickListener, OnLongClickListener{

        /* Estilos de los elementos */
        LayoutParams ff = new LayoutParams(LayoutParams.FILL_PARENT,
                                           LayoutParams.FILL_PARENT);
        LayoutParams ww = new LayoutParams(LayoutParams.WRAP_CONTENT,
                                           LayoutParams.WRAP_CONTENT);
        LayoutParams wf = new LayoutParams(LayoutParams.WRAP_CONTENT,
                                           LayoutParams.FILL_PARENT);
        LayoutParams fw = new LayoutParams(LayoutParams.FILL_PARENT,
                                           LayoutParams.WRAP_CONTENT);

        /* Lista inicial de entrenos*/
        String[] entrenos = {
            "1 vs 1",
```

```
            "3 vs 3",
            "Pases largos",
            "Pases cortos",
            "Bandeja",
            "Pick&Roll",
            "Contraataques",
            "Defensa",
            "Run&Gun"
    };

        int num_entrenos=15;
        int[] idEntrenos = new int[num_entrenos];
        int alto_celda=50;
        int id=3; //1-> etiqueta Fin;  2-> boton de guardar;
        int minutos=0; // Minutos en la hora de Fin

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.v2planifica);

        for(int i=0;i<num_entrenos;i++){
                idEntrenos[i]=0;
        }

        /*Boton para guardar la sesion*/
        Button guarda = (Button) findViewById(R.id.guardaSesion);
        guarda.setId(2);
        guarda.setOnClickListener(this);

        /* Aceso a la tabla general */
        TableLayout tabla = (TableLayout) findViewById(R.id.tl);

        /* Propiedades de la primera fila de la tabla (inicio y fin)*/
        TableRow primera = new TableRow(this);
        TextView inicio = new TextView(this);
        TextView fin = new TextView(this);
        inicio.setText("Inicio -> 10:00");
        inicio.setPadding(10, 2, 10, 2);
        fin.setText("Fin -> 10:00");
        fin.setId(1);
        primera.setMinimumHeight(30);
        primera.setGravity(Gravity.CENTER);
        fin.setPadding(10, 2, 10, 2);
        inicio.setTextColor(Color.BLACK);
        fin.setTextColor(Color.BLACK);
        primera.setBackgroundColor(Color.rgb(255, 255, 203));
```

```
            /* Agregar la primera fila a la tabla */
            primera.addView(inicio);
            primera.addView(fin);
            tabla.addView(primera);

            /* Crear las filas de la tabla*/
            for(int i=0;i<num_entrenos;i++){
            /* Crear la fila, asignarle evento click, longClick e id*/
            TableRow tr = new TableRow(this);
            tr.setOnClickListener(this);
            tr.setOnLongClickListener(this);
            tr.setId(id+1000);
            id++;

            /* Texto del entreno*/
            RelativeLayout izq = new RelativeLayout(this);
            TextView ti = new TextView(this);
            ti.setWidth(50);
            ti.setHeight(alto_celda);
            izq.setBackgroundColor(Color.WHITE);
            izq.addView(ti);
            izq.setGravity(Gravity.CENTER);
            tr.setPadding(1, 1, 1,1);
            tr.addView(izq);

            /* Vista que incluye los botones de + y - y la duracion del entreno*/
            RelativeLayout der = new RelativeLayout(this);
            der.setBackgroundColor(Color.WHITE);
            der.setMinimumHeight(alto_celda);
            tr.addView(der);

            /* Agregar la fila a la tabla*/
            tabla.addView(tr,fw);
           }
} // on-create

/* Evento onClick de las filas*/
public void onClick(View v){
/* Si el id de v es 2, guardamos la sesion
 * si no, se ha pulsado en una fila (id>1000) */
     if(v.getId() == 2){
         int num_ejercicios = 0;
         for(int i=0;i<num_entrenos;i++)
          if(idEntrenos[i]==1)
              num_ejercicios++;

          /* Aviso por si hay 0 entrenos */
         if(num_ejercicios==0){
                String msg =  "Debes introducir al menos un ejercicio!";
```

```
            Toast.makeText(getApplicationContext(),msg,Toast.LENGTH_LONG).show();
            return;
          }

          /* Nombre del fichero: dia-mes-anyo.txt*/
          Calendar c = new GregorianCalendar();
          String dia, mes, anyo;
          int d = c.get(Calendar.MONTH)+1;
          dia = Integer.toString(c.get(Calendar.DAY_OF_MONTH));
          mes = Integer.toString(d);
          anyo = Integer.toString(c.get(Calendar.YEAR));
          String nombre = dia+"-"+mes+"-"+anyo+".txt";

          try {
                FileOutputStream fOut = openFileOutput(nombre,MODE_WORLD_READABLE);
                OutputStreamWriter osr = new OutputStreamWriter(fOut);

                for(int i=0;i<num_entrenos;i++){
                /* Si idEntrenos[i] vale 1, en esa fila hay un ejercicio */
                if(idEntrenos[i]==1){
                   /* A partir del valor de "i" accedemos al id
                    * del entreno y su duracion */
                   int ident = i+9003;
                   int iddur = i+6003;
                   TextView ent = (TextView) findViewById(ident);
                   TextView dur = (TextView) findViewById(iddur);
                   String e = ent.getText().toString();
                   String duracion = dur.getText().toString();

                   /* texto__duracion__identificador */
                   osr.write(e+"___"+duracion+"___"+i+"\n");
          }
           }
          /* Asegurar que se han escrito todos los datos antes de cerrar el fichero */
          osr.flush();
          osr.close();
        }
        catch (FileNotFoundException e) {e.printStackTrace();}
        catch (IOException e) {e.printStackTrace();}

        /* Se devuelve el nombre del fichero a la actividad inicial */
        Intent intent = getIntent();
        intent.putExtra("valor", nombre);
        setResult(RESULT_OK,intent);
        finish();
      }
      else{
          /* Crear nuevo entreno */
           RelativeLayout entreno =  new RelativeLayout(this);
```

```
/* Propiedades visuales del TextView */
AutoCompleteTextView ent = new AutoCompleteTextView(this);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                       this,
                       android.R.layout.simple_dropdown_item_1line,
                       entrenos);
ent.setAdapter(adapter);
ent.setThreshold(3);
ent.setSingleLine(true);
ent.setLayoutParams(fw);
ent.setHeight(alto_celda);

/* Fondo azul con letras blancas*/
ent.setBackgroundColor(Color.BLACK);
ent.setTextColor(Color.WHITE);

   /* Asignar id*/
   ent.setId(v.getId()+8000);
   idEntrenos[v.getId()-1003]=1;
   id++;

   /* Incluir la vista del entreno */
   entreno.addView(ent);

   /* Etiqueta con la duracion*/
   LinearLayout tiempo = new LinearLayout(this);
   tiempo.setOrientation(LinearLayout.HORIZONTAL);
   TextView dur = new TextView(this);

   /* Botones de mas y menos*/
   ImageButton menos = new ImageButton(this);
   ImageButton mas = new ImageButton(this);

   /* Asignar ids*/
   dur.setId(v.getId()+5000); // v tiene la 1000+
   mas.setId(v.getId()+3000);
   menos.setId(v.getId()+4000);

   /* Por defecto, duracion de 10 minutos*/
   dur.setText("10");
   dur.setTextColor(Color.BLACK);
   dur.setSingleLine(true);
   menos.setImageResource(R.drawable.menos); // imagen del mas
   mas.setImageResource(R.drawable.mas);     // imagen del menos
   menos.setOnClickListener(decrementa);
   mas.setOnClickListener(incrementa);
   dur.setGravity(Gravity.CENTER);
```

```
        /* Agregar botones y duracion a la vista de tiempo*/
        tiempo.addView(menos,ww);
        tiempo.addView(dur,ww);
        tiempo.addView(mas,ww);

        /* Acceso al elemento con id=1 --> Fin: */
        TextView fin = (TextView) findViewById(1);
        minutos += 10; // incrementamos en 10 los minutos

        /* Si tenemos 50 minutos, ponerlos a 0 */
       int aux = minutos;
       String aux2 = "";
        while(aux-60 > 0) aux -= 60;
        if (aux==5) aux2="0"+aux;
        else aux2=aux+"";
        if (aux==60) aux2="00";

        /* Si minutos > 60, incrementar en uno la hora */
       int hora = 10+(minutos/60);
       fin.setText("Fin -> "+hora+':'+aux2);

        /* Incluir entreno y botones a la fila actual */
       ((ViewGroup) v).removeViewAt(0);
        ((ViewGroup) v).removeViewAt(0);
        ((ViewGroup) v).addView(entreno);
        ((ViewGroup) v).addView(tiempo);
    }
}

public boolean onLongClick(View v){
        /* Un click largo elimina el entreno*/
        idEntrenos[v.getId()-1003]=0;
        RelativeLayout der = new RelativeLayout(this);
        RelativeLayout der2 = new RelativeLayout(this);
        der.setBackgroundColor(Color.WHITE);
        der.setMinimumHeight(alto_celda);
        der2.setBackgroundColor(Color.WHITE);
        der2.setMinimumHeight(alto_celda);
        TextView et = (TextView) findViewById(v.getId()+5000);
        TextView fin = (TextView) findViewById(1);

        /* Restar los minutos a la duracion total del entreno */
        minutos -= Integer.parseInt(et.getText().toString());
        int aux = minutos;
        String aux2 = "";
        while(aux-60 > 0) aux -= 60;
        if ((aux==5) || (aux==0) ) aux2="0"+aux;
        else aux2=aux+"";
        if (aux==60) aux2="00";
```

```
            int hora = 10+(minutos/60);
            fin.setText("Fin -> "+hora+':'+aux2);

            /* Eliminar entreno y botones y dejar el fondo en blanco */
            ((ViewGroup) v).removeViewAt(0);
            ((ViewGroup) v).removeViewAt(0);
            ((ViewGroup) v).addView(der);
            ((ViewGroup) v).addView(der2);
            v.setMinimumHeight(alto_celda);
            return true;
        }

    /* Sumar 5 minutos */
    private OnClickListener incrementa = new OnClickListener() {
            public void onClick(View v) {
                    TextView et = (TextView) findViewById(v.getId()+2000);

                    /* Incrementar la duracion */
                    int duracion = Integer.parseInt(et.getText().toString());
                    et.setText(String.valueOf(duracion)+5));

                    /* Actualizar etiqueta Fin*/
                    TextView fin = (TextView) findViewById(1);
                    minutos += 5;
                    int aux = minutos;
                    String aux2 = "";
                    while(aux-60 > 0) aux -= 60;
                    if (aux==5) aux2="0"+aux;
                    else aux2=aux+"";
                    if (aux==60) aux2="00";
                    int hora = 10+(minutos/60);
                    fin.setText("Fin -> "+hora+':'+aux2);

                    /* Incrementar el alto de la celda si es mayor que 10*/
                    if(Integer.parseInt(et.getText().toString()) > 10){
                            TableRow t = (TableRow) findViewById(v.getId()-3000);
                            TextView ent = (TextView) findViewById(v.getId()+5000)
                            ent.setHeight(ent.getHeight()+25);
                            t.setMinimumHeight(ent.getHeight()+25);
                    }
            }
    };

    /* Restar 5 minutos */
    private OnClickListener decrementa = new OnClickListener() {
            public void onClick(View v) {
                    TextView et = (TextView) findViewById(v.getId()+1000);

                    /* Controlar las duraciones negativas*/
```

```
                if(Integer.parseInt(et.getText().toString())-5 < 0){
                        et.setText("0");
                }

                else{
                        int duracion = Integer.parseInt(et.getText().toString();
                        et.setText(String.valueOf(duracion)-5));
                        /* Actualizar etiqueta Fin*/
                        TextView fin = (TextView) findViewById(1);
                        minutos -= 5;
                        int aux = minutos;
                        String aux2 = "";
                        while(aux-60 > 0) aux -= 60;
                        if ((aux==5) || (aux==0) ) aux2="0"+aux;
                        else aux2=aux+"";
                        if (aux==60) aux2="00";
                        int hora = 10+(minutos/60);
                        fin.setText("Fin -> "+hora+':'+aux2);
                }

                /* Decrementar el alto de la celda si es mayor  que 10*/
                if(Integer.parseInt(et.getText().toString()) >= 10){
                        TableRow t = (TableRow) findViewById(v.getId()-4000);
                        TextView ent = (TextView) findViewById(v.getId()+4000);
                        ent.setHeight(ent.getHeight()-25);
                        t.setMinimumHeight(ent.getHeight()-25);
                }
        }
};

/* Mantener los datos al girar la pantalla */
@Override
public void onConfigurationChanged(Configuration newConfig){
    super.onConfigurationChanged(newConfig);
}
}
```

F.3. Vista de ejecución

```
package pask.proyectofinal;

/* Clases generales del proyecto */
import pask.proyectofinal.R;
import android.app.Activity;
import android.os.Bundle;
import android.content.res.Configuration;

/* Clases externas para el manejo de ficheros */
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

/* Clase para la reproduccion de los archivos de sonido */
import android.media.MediaPlayer;

/* Clases para el manejo de los Threads */
import android.os.Handler;
import android.os.Message;

/* Clases para la construccion de la interfaz
 * Layout, Table, Cajas de texto, botones, etc */
import android.graphics.Color;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.FrameLayout.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

/* Clase para el uso del vibrador */
import android.os.Vibrator;

/* Clase para mostrar mensajes en pantalla */
import android.widget.Toast;

public class v2ejecuta extends Activity{
     int alto_celda=50; // altura predefinida de cada celda

     /*  sonido de cambio de entreno */
```

```
MediaPlayer mediaPlayer = new MediaPlayer();

/* sonido de fin de la sesion */
MediaPlayer player_fin = new MediaPlayer();
int id=3; // Contador de los identificadores de elementos de la interfaz
int num_entrenos=15; // Cantidad de entrenos que mostrados

int entreno_actual = 0; // En que entreno nos encontramos

Toast descanso; // Variable global para mostrar el mensaje de Descansando...
Vibrator vib;    // Variable global para utilizar el vibrador

/* Array con las duraciones de los ejercicios */
String[] duraciones = new String[num_entrenos];

/* Array con los identificadores de los ejercicios */
int[] identificadores = new int[num_entrenos];
int ejercicios = 0; // Cantidad de ejercicios que se van a ejecutar

/* Variables para definir el tamanyo de los elementos de la interfaz */
LayoutParams ff = new LayoutParams(LayoutParams.FILL_PARENT,
                                   LayoutParams.FILL_PARENT);
LayoutParams ww = new LayoutParams(LayoutParams.WRAP_CONTENT,
                                   LayoutParams.WRAP_CONTENT);
LayoutParams wf = new LayoutParams(LayoutParams.WRAP_CONTENT,
                                   LayoutParams.FILL_PARENT);
LayoutParams fw = new LayoutParams(LayoutParams.FILL_PARENT,
                                   LayoutParams.WRAP_CONTENT);

/* Variables para acceder y modificar el cronometro */
public TextView txtTiempo;
String cronometro;

/* Variables globales con el estado del cronometro*/
public boolean detenido;
public boolean pausado;
public boolean enmarcha = false;

public boolean descansando = false; // Indicador de los periodos de descanso
int cron_minutos=0, segundos=0, horas=0; // Valores del cronometro

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    /* Creacion de la interfaz */
    setContentView(R.layout.v2ejecuta);
    TableLayout tabla = (TableLayout) findViewById(R.id.tl);
```

```
        for(int i=0;i<num_entrenos;i++){

            /* Crear una nueva fila */
            TableRow tr = new TableRow(this);

            /* El id de las filas siempre mayor que 1000 */
            tr.setId(id+1000);
            id++;

            /* Display con el nombre del ejercicio */
            RelativeLayout izq = new RelativeLayout(this);
            TextView ti = new TextView(this);

            /* Parametros visuales del display */
            ti.setWidth(50);
            ti.setHeight(alto_celda);
            izq.setBackgroundColor(Color.WHITE);
            izq.addView(ti);
            izq.setGravity(Gravity.CENTER);
            tr.setPadding(1, 1, 1,1);
            tr.addView(izq);

            /* Display con la duracion del ejercicio */
            RelativeLayout der = new RelativeLayout(this);
            der.setBackgroundColor(Color.WHITE);
            der.setMinimumHeight(alto_celda);
            tr.addView(der);
            tabla.addView(tr,fw);
        }

    /* Acceso al cronometro y botones */
    txtTiempo = (TextView) findViewById(R.id.ConTiempo);
        final Button btn = (Button) findViewById(R.id.BtnAccion);
        final Button btnDetener = (Button) findViewById(R.id.BtnDetener);

        detenido = false;
        pausado = true;

        /* Inicio del thread que controla el cronometro */
        thread.start();

        /* Boton de pausa/reanudacion de la sesion */
        btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(btn.getText().equals("Iniciar ejercicios")){
                pausado = false;
                btn.setText("Pausar");
            }
```

```
                else{
                    if(btn.getText().equals("Pausar")){
                        pausado = true;
                        btn.setText("Continuar");
                    }
                    else{
                        pausado = false;
                        btn.setText("Pausar");
                    }
                }
            }
        });

        /* Acciones de finalizar la sesion */
        btnDetener.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                detenido = true;
                pausado = true;
                thread.interrupt();
                finish();
            }
        });

    }

/* Thread encargado de la gestion del cronometro */
Thread thread = new Thread(){
    public void run(){
        String min,sec;

            /* Cargar los entrenamientos */
            actualiza(0);

        /* Actualizar el cronometro */
        if(Integer.parseInt(duraciones[entreno_actual]) <10)
             min ="0"+duraciones[entreno_actual];
            else
             min = duraciones[entreno_actual];
            txtTiempo.setText("00:"+min+":00");
                segundos = Integer.parseInt(duraciones[entreno_actual]);
                if(segundos < 10) sec = "0"+segundos;
                else sec = segundos+"";

            while(!detenido){
                while(!pausado){
                    try {
                        if (segundos == 00 && (cron_minutos>0 || horas>0)) {
                            segundos = 59;
```

```
cron_minutos--;
 }
 if (cron_minutos == 00 && horas>0) {
     cron_minutos = 59;
     horas--;
 }

 if(cron_minutos < 10) min=""+cron_minutos;
 else min = ""+cron_minutos;
 if(segundos <10) sec="0"+segundos;
 else sec = ""+segundos;

 cronometro = "0"+horas + ":0" + min + ":" + sec;
 handler_cron.sendEmptyMessage(0);
 segundos--;
 this.sleep(1000);

 if(horas == 00 && cron_minutos == 00 && segundos==00){
       if(descansando==true){
          if(entreno_actual == ejercicios-1){
                    /* Fin del entreno */
               detenido = true;
               pausado = true;

               cronometro = "00:00:00";

       /* Actualizar colores de la interfaz */
            handler_cron.sendEmptyMessage(0);
       actualiza_fila.sendEmptyMessage(0);
       long[] patron = {0,500,500,500,500,2500};
       vib = (Vibrator) getSystemService(
                  VIBRATOR_SERVICE);
       vib.vibrate(patron,-1);

  /* Sonido de fin de sesion */
       finalSesion();
       this.sleep(2500);
       player_fin.stop();
       thread.interrupt();
       finish();
 }
 else{
       fondo_cron.sendEmptyMessage(0);
       pausado = true;
       alarma();
       descansando = false;
       actualiza_fila.sendEmptyMessage(0);

       segundos = Integer.parseInt(
```

```
                                        duraciones[entreno_actual]);
                                if(cron_minutos <10) min="0"+cron_minutos;
                                else min = ""+cron_minutos;
                                if(segundos <10) sec="0"+segundos;
                                else sec = ""+segundos;
                                cronometro = horas + "0:" + min + ":" + sec;
                                handler_cron.sendEmptyMessage(0);
                                pausado = false;
                            }
                        }
                        else{ // 15 segundos de reposo
                            descansando = true;
                            fondo_cron.sendEmptyMessage(0);
                            long[] patron = {0,2000,500,2000,500,2000};
                            vib = (Vibrator) getSystemService(
                                    VIBRATOR_SERVICE);
                            vib.vibrate(patron,-1);
                            if(cron_minutos <10) min="0"+cron_minutos;
                    else min = ""+cron_minutos;
                            cronometro = horas + "0:" + min + ":00";
                            handler_cron.sendEmptyMessage(0);
                            mensaje_descanso.sendEmptyMessage(0);
                            ((Vibrator)getSystemService(VIBRATOR_SERVICE)).
                                                        vibrate(1000);
                            alarma();
                            segundos = 15;
                            if(cron_minutos <10) min="0"+cron_minutos;
                            else min = ""+cron_minutos;
                            if(segundos <10) sec="0"+segundos;
                            else sec = ""+segundos;
                            cronometro = horas + "0:" + min + ":" + sec;
                            handler_cron.sendEmptyMessage(0);
                            detenido = false;
                            descansando = true;
                        }
                     }
                     } catch (Exception ex) {ex.printStackTrace();}
                    }// Fin while(!pausado)
                }// Fin while(!detenido)
    }// Fin run
};

/* Actualizar cronometro */
private Handler handler_cron = new Handler(){
    public void handleMessage(Message msg){
            txtTiempo.setText(cronometro);
            if(descansando == true)
                    txtTiempo.setBackgroundColor(Color.rgb(255,101,101));
            else
```

```
                txtTiempo.setBackgroundColor(Color.parseColor("#6EFE7F"));
    }
};

/* Actualizar background cronometro */
private Handler fondo_cron = new Handler(){
    public void handleMessage(Message msg){
            if(descansando == true)
                txtTiempo.setBackgroundColor(Color.rgb(255,101,101));
            else
                txtTiempo.setBackgroundColor(Color.parseColor("#6EFE7F"));
    }
};

private Handler mensaje_descanso = new Handler(){
        public void handleMessage(Message msg){
            descanso = Toast.makeText(getApplicationContext(),
                                      "Descansando...",
                                      Toast.LENGTH_LONG);
            descanso.show();
        }
};

public Handler actualiza_fila = new Handler(){
        public void handleMessage(Message msg){
            /* Actual a gris */
                View vista = (View) findViewById(
                                            identificadores[entreno_actual]);
            vista.setBackgroundColor(Color.parseColor("#CCD4CF"));
            TextView ent = (TextView) findViewById(vista.getId()+8000);
            ent.setBackgroundColor(Color.parseColor("#CCD4CF"));

            entreno_actual++;

           /* Actualizar poner el siguiente ejercicio a verde */
            if(entreno_actual != ejercicios){
                View siguiente = (View) findViewById(
                                            identificadores[entreno_actual]);
                siguiente.setBackgroundColor(Color.rgb(79, 129, 189));
                TextView ent2 = (TextView) findViewById(
                                                siguiente.getId()+8000);
                ent2.setBackgroundColor(Color.rgb(79, 129, 189));

            }
        }
};

/* Crear la sesion */
public void actualiza(int valor){
```

```
Bundle extras = getIntent().getExtras();
String nombre = "";
boolean inicial=true;
if(extras!=null)
  nombre = extras.getString("nombre");

try {
  InputStream instream = openFileInput(nombre);
   if (instream != null) {
        InputStreamReader inputreader = new InputStreamReader(instream);
        BufferedReader buffreader = new BufferedReader(inputreader);
        String line;
        int cont = 0;

        while (( line = buffreader.readLine()) != null){

            String[] datos = line.split("___");
            int i = Integer.parseInt(datos[2].toString());
            String ent = datos[0];
            String min = datos[1].toString();
            duraciones[cont] = min;
            ejercicios++;

            identificadores[cont] = i+1003;
            cont++;
            View v = (View) findViewById(i+1003);
            RelativeLayout entreno =  new RelativeLayout(this);

            TextView ent1 = new TextView(this);
            ent1.setId(9003+i);
            ent1.setSingleLine(true); // una sola linea
            ent1.setLayoutParams(fw);
            ent1.setHeight(alto_celda);
            ent1.setWidth(50);
            ent1.setGravity(Gravity.CENTER);

            if(inicial == true){
                  inicial = false;
                  v.setBackgroundColor(Color.rgb(79, 129, 189));
                  ent1.setBackgroundColor(Color.rgb(79, 129, 189));
            }
            else
                  ent1.setBackgroundColor(Color.parseColor("#CCD4CF"));

            ent1.setTextColor(Color.BLACK);
            ent = ent.toUpperCase();
            ent1.setText(ent);

            /* Agregar la vista del entreno */
```

```
                    entreno.addView(ent1);

                    /* Etiqueta con la duracion*/
                    LinearLayout tiempo = new LinearLayout(this);
                    tiempo.setOrientation(LinearLayout.HORIZONTAL);
                    TextView dur = new TextView(this);

                    /* Asignar la duracion establecida */
                    dur.setText(min+" minutos");
                    dur.setTextColor(Color.BLACK);
                    dur.setPadding(0,15,20,0);
                    tiempo.setGravity(Gravity.CENTER);
                    dur.setGravity(Gravity.CENTER);

                    /* Incluir duracion a la vista de tiempo */
                    tiempo.addView(dur,ww);

                    /* Agregar el entreno y botones a la fila actual */
                        ((ViewGroup) v).removeViewAt(0);
                        ((ViewGroup) v).removeViewAt(0);
                    ((ViewGroup) v).addView(entreno);
                    ((ViewGroup) v).addView(tiempo);

                }
            }
        catch (FileNotFoundException e){ e.printStackTrace();}
        catch (IOException e){e.printStackTrace();}
}

/* Alarma de fin de sesion */
public void finalSesion(){
     player_fin = MediaPlayer.create(this, R.raw.finsesion);
     player_fin.start();
}

/* Alarma de fin de ejercicio */
public void alarma(){
     mediaPlayer = MediaPlayer.create(this, R.raw.alarm);
     mediaPlayer.start();
}

/* Evitar perder el estado actual de la interfaz al girar la pantalla */
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
}
}
```

G. Android Manifest

Por último, se muestra el fichero *android_manifest* que como se ha comentado anteriormente, es de vital importancia puesto que habrá que especificar las ventanas que incluye nuestra aplicación, además de otros elementos como la rotación de la pantalla, nombre de la aplicación, etc.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="pask.proyectofinal"
      android:versionCode="1"
      android:versionName="1.0">
      <uses-permission android:name="android.permission.VIBRATE" />
    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">
        <activity android:name=".v2inicio"
                          android:configChanges="keyboard|keyboardHidden|
                                                 orientation"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="v2planifica"
                          android:label="Planificacion del entrenamiento">
            </activity>
                <activity android:name="v2ejecuta"
                                  android:label="Ejecucion del entrenamiento">
                </activity>

    </application>


</manifest>
```

Bibliografía

[1] Layouts en Android
http://mobiforge.com/designing/story/understanding-user-interface-android-part-1-layou

[2] Views en Android
http://mobiforge.com/designing/story/understanding-user-interface-android-part-2-views

[3] Gestos: arrastrar
http://www.zdnet.com/blog/burnette/how-to-use-multi-touch-in-android-2-part-5-implemer
1789?tag=mantle_skin;content

[4] TableLayout
http://huuah.com/using-tablelayout-on-android

[5] Mover un objeto por la pantalla
https://groups.google.com/group/desarrolladores-android/web/
conocimientos-bsicos-mover-un-objeto-por-la-pantalla?hl=es

[6] Layouts y sus propiedades
http://www.sgoliver.net/blog/?p=1341

[7] Introducción a Android
http://www.htcmania.com/showthread.php?t=134170

[8] Widgets en Android
http://www.droiddraw.org/widgetguide.html

[9] Ejemplos de programas
http://myandroidwidgets.googlecode.com/svn/trunk

[10] Tutorial: desarrollo de aplicaciones
http://viva-moore.blogspot.com/2010/07/tutorial-desarrollo-de-aplicaciones.
html

[11] Cajas de mensaje
http://patricktools.com/d/android/23message_boxes

[12] Click Listeners
http://android-developers.blogspot.com/2009/10/ui-framework-changes-in-android-16.
html

[13] Ejemplo de Intent
http://marakana.com/forums/android/examples/65.html

[14] Uso de ficheros
http://www.android-spa.com/viewtopic.php?t=4416

[15] Paso de parametros en Intents
`https://www.android-spa.com/viewtopic.php?t=2941&sid=3fe681a258d7c90745af3020f226a503`

[16] Uso del vibrador del teléfono
`http://android.konreu.com/developer-how-to/vibration-examples-for-android-phone-development`

[17] 10 razones por las que utilizar Android
`http://www.tudosisgeek.com/10-razones-para-usar-android-ventajas/`

[18] Usabilidad
`http://es.wikipedia.org/wiki/Usabilidad`

[19] Certificados con Java
`http://www.androiddevelopment.org/tag/keytool/`