Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Aplicación Android de rutinas de entrenamiento adaptadas al usuario usando SQLite y JSON

Autor: Mirian Franco Maireles

Tutor: María Teresa Ariza Gómez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera

Ingeniería de Telecomunicación

Aplicación Android de rutinas de entrenamiento adaptadas al usuario usando SQLite y JSON

Autor:

Mirian Franco Maireles

Tutor:

María Teresa Ariza Gómez

Profesor Titular

Departamento de Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera: Aplicación Android de rutinas de entrenamiento adaptadas al usuario usando SQLite y JSON

Autor: Mirian Franco Maireles
Tutor: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Afortunada soy, pues muchas son las personas a las que debo y quiero agradecer por un motivo u otro a lo largo de esta difícil y maravillosa etapa. El agradecimiento más grande se lo doy a mis padres, Antonio y Reme, gracias por toda una vida llena de trabajo y sacrificio para que mi hermano y yo tuviésemos la mejor educación. Desde las clases de música hasta nuestras respectivas carreras, habéis invertido y apostado siempre por nuestra formación. Ni un momento lo habéis dudado, aunque algún que otro año no hubiera beca, siempre habéis sabido tirar hacia adelante y superar todas las adversidades, por eso no sólo os doy las gracias, también toda mi admiración porque sois los mejores padres del mundo. Gracias a mi hermano, Antonio, por ser durante toda mi vida un ejemplo a seguir, y por su implicación directa en este proyecto, te admiro por el gran profesional que eres, pero también por la persona humilde y trabajadora en que te has convertido. No quiero dejar pasar este momento tan importante en mi vida sin mencionar a mi hermana Nuria, que la vida nos arrebató tan injustamente, siempre estarás presente en mis pensamientos y siempre te querré estés donde estés.

En segundo lugar quiero agradecer a mi pareja, Jesús Ángel con los que llevo compartiendo casi siete maravillosos años. Llegaste justo en el momento en que más lo necesitaba. Gracias por tu apoyo incondicional, por tus consejos y por tus palabras de aliento en los momentos más duros de esta etapa.

Gracias también a mi tutora Mª Teresa, por darme la oportunidad de adentrarme en el maravilloso mundo de la programación Android, y por todos sus consejos y correcciones.

Gracias a toda mi familia, primos, cuñada, tíos, y abuelos, los que están y los que desgraciadamente no, siempre he recibido vuestro apoyo y vuestros animos para seguir hacia adelante.

Gracias a mi otra mitad, mi amiga, mi compañera y mi hermana Patricia, por todas las carcajadas que me ha sacado con sus ocurrencias y por aguantarme con

mis “ansias”. Te conozco desde antes de lo que pueda recordar y siempre has estado ahí, gracias por todo lo que hemos vivido juntas y por nuestra increíble conexión mental. Gracias también a mi “hermano” Rafa, mi primer verdadero amigo de la universidad, por todos los momentos que hemos vivido, por tus cancioncitas, por los almuerzos de “tupper” y simplemente por ser la persona maravillosa que eres.

Gracias también a mis amigos, compañeros, “colegas” y hermanos de facultad, Ramoni, Jose y Alex, gracias por ese pedazo de 5º curso que vivimos, por las risas, por las fiestas, por las anécdotas, por los viajes, por hacer hasta divertidos los exámenes tipo test, por los vídeos poco favorecedores, pero también por la ayuda que nos hemos brindado cuando alguno lo ha necesitado. Sin duda sois lo mejor que me llevo de esta etapa.

Gracias a mis amigas de toda la vida, mis “farrus” Virginia, María, Esther, Celia, Belén y Viki. Es un gustazo ver cómo hemos cambiado todas, y sin embargo seguimos igual de unidas, a pesar de la distancia.

Y por último, a todos los que en algún momento me brindaron su apoyo y en ningún momento dejaron de creer en que podría conseguirlo, GRACIAS de todo corazón.

Mirian Franco Maireles

Sevilla, 2015

Índice Abreviado

Índice

Índice de Figuras

Índice de Tablas

1 Introducción

En este primer capítulo, serán presentados los distintos objetivos y motivaciones que han llevado a la realización de este proyecto. Se expondrán las ventajas de tener instalada esta aplicación Android, en la que el usuario tendrá acceso a distintas rutinas que él ha elegido, sin la necesidad de una conexión a internet durante su utilización. Definiremos también los antecedentes de este proyecto y por último añadiremos la estructura que va a tener nuestra memoria.

1.1 Contexto

De todos es conocido el auge de la práctica deportiva de los últimos años. Este boom, unido al tecnológico ya existente y en concreto al uso de smartphones, ha contribuido a la proliferación de todo tipo de aplicaciones creadas con el fin de ayudar a los usuarios en su entrenamiento, facilitar el seguimiento del rendimiento y la monitorización de la actividad física.

Por otro lado, nos encontramos ante una sociedad con un ritmo cada vez más frenético en el que se dispone de poco tiempo para la realización de ejercicio o para acudir a un centro especializado. Tampoco hay que olvidarse de la crisis económica mundial de los últimos años, en los que para muchas personas, acudir a un gimnasio o centro de entrenamiento ha quedado relegado a segundo plano por motivos económicos. De aquí surgen aplicaciones deportivas de diferentes clases. Nosotros nos centraremos en las especializadas en vídeos de rutinas de ejercicios y fitness.

El auto-entrenamiento es un método para el aprendizaje de cualquier habilidad deportiva, en el que no disponemos de la figura de un entrenador que pueda guiarnos, motivarnos y corregirnos al ejecutar los ejercicios, sin embargo tiene la ventaja de la flexibilidad temporal y en la mayoría también de la económica. Cada persona elige el momento del entrenamiento y su duración, y está principalmente pensado para

aquellas personas que por alguna causa no pueden acudir a un centro profesional.

1.2 Presentación del problema

Actualmente, la gran mayoría de aplicaciones Android de entrenamiento personal existentes en el Play Store cuentan con una limitación en el número de rutinas y ejercicios impuesto por la propia aplicación y suelen ser muy generales con respecto al área de entrenamiento. Para mejorar este aspecto, va a ser el propio usuario el encargado de elegir y descargar las rutinas que desee, tanto en número como en área de entrenamiento. Puede elegir si sólo desea rutinas especializadas en una zona o en varias. El usuario podrá descargarse de una página web, las rutinas que él desee. La idea de esta página sería que cualquier usuario pueda ser productor y consumidor de rutinas, el límite en el número sólo lo imponen los propios usuarios. Sin embargo, el diseño de esta página se escapa del alcance de nuestro proyecto. Nuestro punto de partida será suponer que ya hemos realizado la descarga de estas rutinas y que se encuentran contenidas en la carpeta Download de nuestro dispositivo, contenidas en carpetas zip.

Casi la totalidad de estas aplicaciones son desasistidas, es decir, no son específicas para el estado de forma de cada usuario además de que el usuario no sabe si el ejercicio lo está realizando correctamente, hecho que influye en la efectividad de éste. Para intentar suplir de algún modo estas carencias, nuestra aplicación contará con una amplia descripción de cada ejercicio, los músculos que entran en juego, cuál es su ejecución correcta o en qué casos se debe abandonar su ejecución si aparece algún tipo de dolencia. Además se pondrá a disposición del usuario varias opciones de nivel de forma física, para que el usuario elija cuál es el adecuado para su caso.

Otra desventaja que se observa en la mayoría de las aplicaciones que se ofertan es que la mayoría son reproducidas desde un servidor externo y si no se dispone de un buena conexión a internet, se producirán cortes e incluso no se podrían llegar a reproducir. Nuestra aplicación reproducirá los vídeos contenidos directamente en la memoria de nuestro dispositivo. El objetivo es que la aplicación funcione siempre en tiempo real y en cualquier lugar donde se encuentre el usuario.

Con el mismo objetivo anterior, mantener siempre el funcionamiento de la aplicación en tiempo real, hemos usado SQLite para el almacenamiento tanto de información y datos del usuario como de los vídeos. Su utilización implicará más rapidez que otros sistemas gestores de bases de datos pues no necesita ninguna configuración ni administrador y toda la base de datos se almacena en un único archivo, esto se detallará en el Capítulo 2 apartado 2.5.

1.3 Antecedentes

En este apartado presentaremos los antecedentes que han servido de base para la realización de este proyecto.

1.3.1 Referencias

Recabando información, vamos a comentar las líneas de trabajo anteriores a este proyecto relacionadas con el entrenamiento personal, las cuales nos han servido de base.

Aplicación Android para la rehabilitación física del usuario usando MySQL, PHP y codificación JSON

Se trata de un proyecto desarrollado en la Escuela Técnica Superior de Ingenieros de Sevilla por el alumno Antonio José Díaz Lora a fecha de Junio del 2014. El objetivo principal de este trabajo es el de una aplicación de entrenamiento remoto, sin necesidad de que el entrenamiento sea asistido en tiempo real, sino que el médico o fisioterapia asigna los ejercicios previamente a un usuario determinado. Los vídeos se encontrarán alojados en un servidor externo a los usuarios.

Se trata de una aplicación muy completa, pero tiene la desventaja de que el usuario no dispone de los ejercicios en su dispositivo, y en un momento determinado que quisiera hacer uso de ellos, la conexión podría fallar. Además se trata de una aplicación menos autónoma que la nuestra, ya que depende de la asignación que realiza el profesional.

Quizás el uso de esta aplicación pueda parecer más ventajoso en el ámbito médico de rehabilitaciones físicas, sin embargo, con nuestra aplicación también podríamos cubrir estas necesidades. Al ser creadas las rutinas por los propios usuarios, podrían diseñar rutinas de ejercicios de cualquier tipo y naturaleza, como por ejemplo rutinas específicas para distintas dolencias.

En el ámbito deportivo, nuestra aplicación también ofrece muchas ventajas, el usuario quizás prefiera tener la opción de elegir sus rutinas de ejercicios, y así buscar su propia motivación. Con nuestra aplicación, el límite de rutinas lo impondrían nuestros propios usuarios, pues ellos pueden elegir y descargarse sólo las que deseen.

Monitorización mediante streaming de vídeo para entrenamiento personal

Este proyecto, realizado en la Escuela Técnica Superior de Ingenieros de Sevilla cuyo autor es Antonio Clavaín Mateo en el 2014 tiene como objetivo el diseño

de una infraestructura para que cualquier persona que disponga de un dispositivo Android y una conexión a internet pueda realizar sesiones de ejercicios mientras es monitorizado por un especialista a través de la cámara del dispositivo.

Al igual que en el anterior, el funcionamiento de la aplicación depende totalmente de una conexión a internet. Además el usuario, pierde autonomía con respecto a la elección del momento del entrenamiento, pues debe acordarlo previamente con el entrenador.

No hay que olvidarse de la gran cantidad de ofertas con respecto a aplicaciones de entrenamiento personal que hay actualmente disponibles en Play store, y que no pueden compararse con el trabajo realizado por una sola persona, sin embargo vamos a comentar algunas muy usuales.

Nike+ Training Club (gratuita)

Contiene más de 100 ejercicios disponibles al usuario y dispone soporte Chrome-Cast para emitir los videos en la TV. Además también tiene la opción de hacer un seguimiento de los entrenamientos.

Nuestra aplicación sin embargo, tiene la opción de que el catálogo de rutinas y ejercicios puede ser ampliado por los usuarios o empresas. Ellos son los productores y consumidores de rutinas.

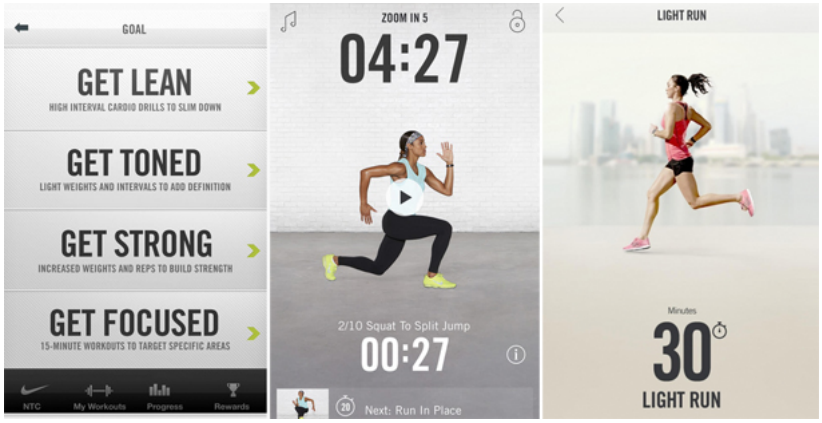

Figura 1.1 Interfaz de la app Nike+ Training Club.

SWORKIT lite (gratuita)

Se trata de una aplicación de entrenamiento que incorpora rutinas de fuerza, cardio, estiramientos y yoga, pudiendo programarlas de 5 a 60 minutos. También incorpora un creador de rutinas en las que el usuario puede diseñar sus rutinas, eligiendo entre un listado de más de 170 ejercicios.

De nuevo vemos que el catálogo de ejercicios es muy rígido, pues son la propia empresa quienes lo imponen. Además sólo cuenta con cuatro categorías de ejercicios.

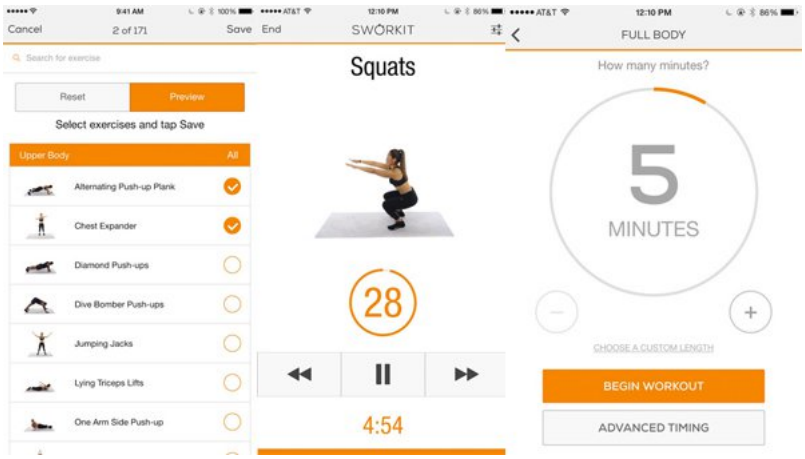

Figura 1.2 Interfaz de la app Sworkit lite.

1.3.2 Seven, 7 minutos de ejercicio (gratuita)

Esta aplicación propone un objetivo, realizar siete minutos de ejercicio al día durante siete meses, no se necesita ningún aparato de ejercicio, y viene con muchas imágenes explicativas. Además, como en un juego, te da tres vidas, en el momento que dejas de realizar ejercicio tres días en un mes, el progreso comenzará de nuevo.

Sin embargo esta aplicación tiene la desventaja de que no contiene vídeos para ver mejor la ejecución del ejercicio, y de que además debes ceñirte a su plan de entrenamiento.

Figura 1.3 Interfaz de la app Seven.

1.3.3 Inicio Entrenamientos (gratuita)

Aplicación para entrenar en cualquier lugar. Se pueden realizar varios tipos de circuitos. La aplicación indica el número de veces que hemos de repetir cada ejercicio y el tiempo que se tarda en realizarlo. Se trata de una aplicación muy sencilla, con tablas de ejercicios y dibujos que la acompañan.

A esta aplicación le falta quizás algún tipo de interactividad con el usuario y más dinamismo en forma de vídeos que lo hagan más atractivo al usuario.

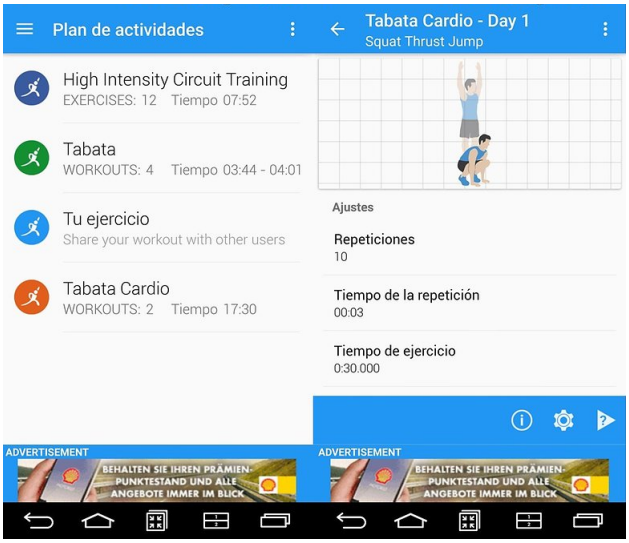

Figura 1.4 Interfaz de la app Inicio Entrenamientos.

1.3.4 Vientre Plano (gratuita)

Esta aplicación (Figura 1.5) es específica para esta zona del cuerpo. Cada rutina de ejercicios viene bien especificada, el tiempo que ha de durar cada ejercicio y el tiempo de descanso entre cada uno de ellos. También cuenta con vídeos explicativos.

Sin embargo es específica sólo para un área. Para ejercitar otras áreas deben de descargarse otro tipo de aplicaciones.

1.3.5 Adidas: correr y entrenamiento (gratuita)

Esta aplicación Adidas (Figura 1.6) tiene una interfaz de usuario muy trabajada y atractiva al usuario. Necesita la creación de una cuenta para acceder, para así almacenar la información propia del usuario. Tiene la opción de elegir el tipo de actividad, o bien correr o hacer ejercicios.

Figura 1.5 Interfaz de la app Vientre Plano.

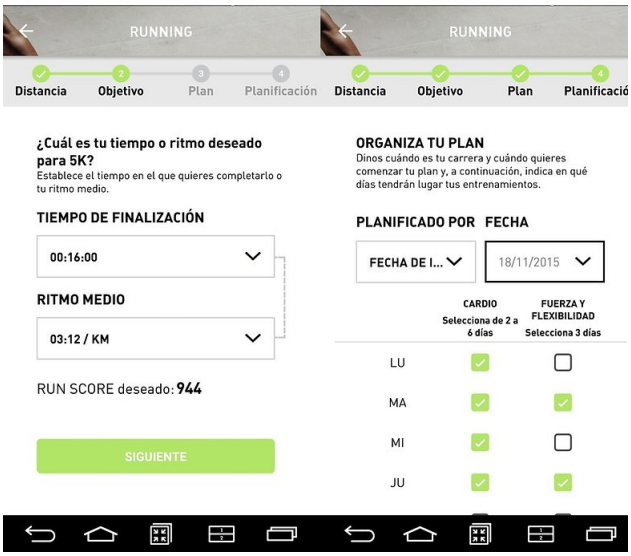

Figura 1.6 Interfaz de la app Adidas.

1.4 Descripción de la solución

En este apartado se presentarán las características principales de la solución que hemos adoptado.

1.4.1 Objetivos

El objetivo principal de nuestra solución es facilitar a los usuarios la realización de un auto-entrenamiento en el lugar y en el momento que se desee, con la mayor calidad y exactitud posible teniendo en cuenta la ausencia de un profesional que pueda supervisarlo, por tanto deberá disponer de toda la información necesaria para que el propio usuario pueda suplir de cierta forma el trabajo que haría un profesional especializado, tales como entrenador personal o fisioterapeuta.

El usuario podrá seleccionar al crear su perfil entre tres posibilidades de nivel de forma física que cree que se corresponde con su caso, a elegir entre “Bajo”, correspondiente con bajo estado de forma física, para aquellas personas que no hagan deporte con asiduidad y quieran comenzar a adquirir cierta rutina, “Medio” refiriéndose a estado de forma medio, para aquellas que ya llevan cierto tiempo realizando deporte un par de veces a la semana, o nivel “Alto”, para aquellas que disponen de un buen estado de forma y realizan deporte casi a diario.

Dependiendo de la elección que haya hecho el usuario, se le mostrarán solo los vídeos de cada rutina que tengan asignados ese nivel, el resto de ellos no serán visibles para ese usuario.

El funcionamiento de nuestra aplicación será totalmente independiente de la existencia de una red de datos en el lugar donde se encuentre el usuario, a continuación explicamos esto.

Al descargarse e instalarse esta app se incluirá en la carpeta raw del paquete de la aplicación una rutina de vídeos por defecto que incluirá nuestra aplicación, la rutina GAP que contiene ejercicios para la tonificación de glúteos, abdomen y piernas, por tanto, los vídeos se encontrarán en el dispositivo del usuario.

Podríamos adquirir nuevas rutinas desde una página web por ejemplo, y descargarlas en un .zip en nuestro teléfono móvil. Por tanto la aplicación podrá reconocer estos .zip que se encontrará en la carpeta Download del dispositivo y descomprimirlos en una carpeta que crea la aplicación, la carpeta storage/emulated/0/app_deportes/rutinas/.

Se necesitará para acceder a la aplicación un registro del usuario, pero este registro será local, quedando almacenada toda la información en la base de datos del propio dispositivo, gracias al sistema gestor SQLite. Esta característica permite que varios usuarios puedan acceder con sus datos propios, a la aplicación instalada en ese dispositivo, como por ejemplo varios miembros de una familia. También está pensado para una instalación de la aplicación en tabletas, donde no es tan extraño compartir dispositivos.

Con esta aplicación se permitirá al usuario hacer un seguimiento del tiempo dedicado a la semana a la práctica de ejercicio y se representará visualmente en una gráfica.

Cabe mencionar de nuevo que queda fuera del alcance de este proyecto la realización de la web de donde poder descargarse las rutinas en .zip, simplemente se partirá de que estas rutinas ya han sido descargadas en Download.

También podemos mencionar que cada cuenta de usuario dispone de un alto nivel de seguridad, pues la contraseña de usuario nunca se almacena en claro en la base de datos sino que se hace mediante la adición de una salt originada aleatoriamente,

posteriormente, esta pareja de contraseña + salt se pasará por una función hash. El salt y este hash es lo que verdaderamente almacenamos en la base de datos.

Dispondremos de una gestión del olvido de contraseña, en la que se generará una nueva contraseña, y se informará de su valor mediante el envío de un e-mail a la cuenta del usuario.

1.4.2 Funcionalidades

Cuando iniciamos la aplicación, en la actividad principal el usuario dispondrá de varias opciones.

- Autentificación de un usuario previamente registrado mediante la dupla (usuario, contraseña).
- Acceso a la actividad donde se registrará a un nuevo usuario.
- Acceso a la gestión del olvido de contraseña.

Una vez el usuario ha sido debidamente identificado, se accederá a su pantalla personalizada, en ella, aparecerá un saludo personalizado y las siguientes opciones.

- Acceso a la lista de rutinas que el usuario ya ha cargado con anterioridad (cabe recordar que la rutina GAP aparecerá siempre, pues es la que trae incorporada directamente la aplicación).
- Acceso a “Mi rendimiento”.
- Realizar cambios sobre la información de la cuenta del usuario
- Mensaje “acerca de” donde se informa al usuario de que la responsabilidad de la realización de los ejercicios recae sobre él.
- Opción de cargar nuevas rutinas que pudieran existir en el dispositivo.

Una vez se accede a una rutina determinada, aparecerá una lista con sus vídeos, pero sólo aquellos que se correspondan con el nivel de forma que ha indicado el usuario en su registro.

Si seleccionamos sobre uno de los elementos de la vista nos llevará a una nueva actividad con los siguientes elementos:

- Reproducción del vídeo del ejercicio concreto que se ha seleccionado.
- “Comenzar”, da por supuesto que el usuario va a proceder a realizar el ejercicio,

el cronómetro empezará a marchar y el vídeo comenzará y se repetirá tantas veces como tenga que repetirlo el usuario, de esta forma se tratará de hacer el ejercicio del usuario lo más interactivo posible con la aplicación.

- "Terminado", da por hecho que el usuario ya ha terminado las repeticiones y detiene el cronómetro.
- Información detallada sobre el ejercicio, que será de gran utilidad para su correcta realización:
  - Descripción de la correcta realización del ejercicio.
  - Finalidad o beneficios de este ejercicio.
  - Número de repeticiones.
  - Material necesario.
  - Músculos implicados.

1.4.3 Arquitectura

Veremos la arquitectura de nuestra solución que va a ser una arquitectura muy simple, de la que el usuario saldrá beneficiado en rapidez de ejecución.

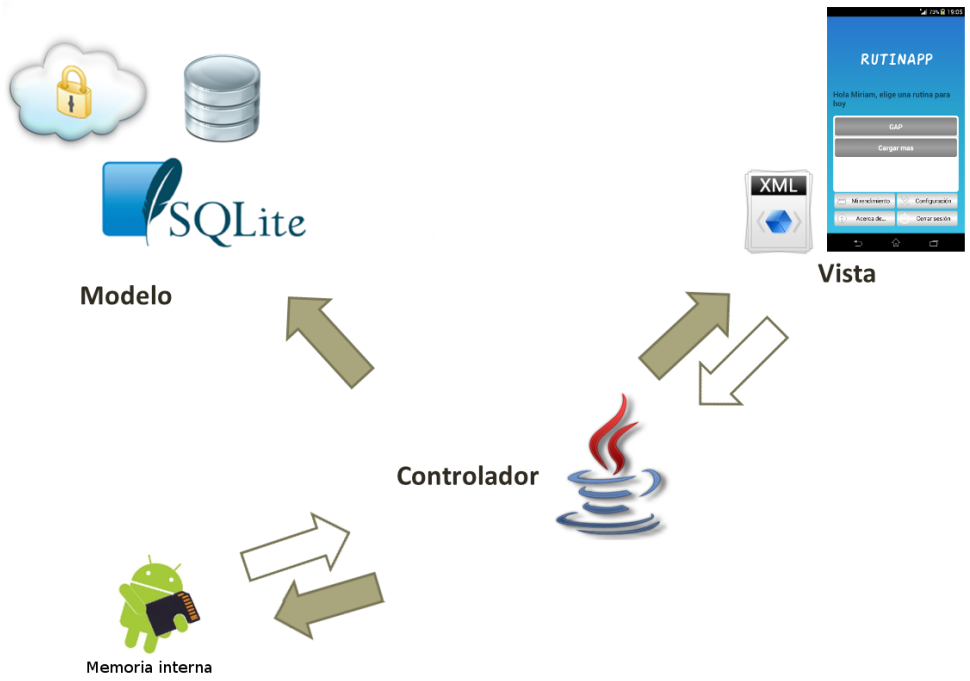

Figura 1.7 Arquitectura de nuestra solución.

Identificaremos los siguientes elementos, todos dentro de nuestro dispositivo móvil.

Vista : Interfaz de usuario de nuestra aplicación, desarrollada en xml.

Controlador : Esto sería la lógica Java de nuestra aplicación. Es quien realiza todos los cálculos y operaciones, y quien se comunica con el resto de componentes de la arquitectura. Realiza las consultas a la base de datos SQLite, accede a los elementos de la vista y lee y escribe sobre la memoria interna de nuestro dispositivo.

Base de datos SQLite : Base de datos que utiliza la aplicación para administrar toda la información tanto del usuario como de las rutinas y vídeos de la aplicación, o sobre los ejercicios que ha realizado un usuario. Recibirá las peticiones Android de los usuarios.

Memoria interna del dispositivo : En ella será donde tengamos almacenados todos los vídeos de las rutinas a los que nuestra aplicación puede acceder.

Cabe mencionar que las carpetas de rutinas tienen una estructura bien definida para que funcionen correctamente en la aplicación. Cada rutina contará con un archivo JSON, también con una organización interna específica, que consta de un array de dos objetos, uno para la descripción de la rutina, y el otro objeto contiene un array con la información de todos los vídeos. La carpeta de una rutina también deberá contener los vídeos en formato mp4. Hemos de tener en cuenta que el nombre que le pongamos al vídeo tiene que corresponderse con el campo Nombre que hayamos añadido en el archivo JSON para ese vídeo.

Cualquier usuario que construya una carpeta rutina de este tipo, será válida para su ejecución en la aplicación.

1.5 Estructura de la memoria

Vamos a presentar la estructura que tendrá la memoria de este proyecto.

En el primer capítulo se muestra una pequeña descripción del proyecto. Presentaremos los principales problemas que hemos encontrado y la solución que proponemos una vez los hemos analizado. Expondremos los antecedentes en los que nos hemos basado y las referencias. Finalizaremos con la estructura que va a tener nuestra memoria.

En el segundo capítulo, expondremos todas las tecnologías implicadas en la realización de este proyecto.

En el tercer capítulo realizaremos una descripción de la estructura que tendrá nuestra base de datos en el dispositivo, SQLite. Presentaremos el modelo entidad-relación e indicaremos el porqué de la elección de esta estructura.

En el cuarto capítulo, se hará uso del UML (Unified Modeling Language) para describir toda la estructura de nuestro software. Se trata de un lenguaje gráfico de modelado para visualizar y especificar métodos o procesos. Haremos uso de sus diferentes tipos de diagramas: diagramas de casos de uso, diagramas de clase y diagramas de secuencia.

En el quinto capítulo mostraremos con detalle la interfaz de usuario de nuestra aplicación y toda la funcionalidad a la que puede acceder el usuario.

En el último capítulo se comentarán las futuras líneas de continuación del proyecto. También se expondrán las conclusiones a las que ha llegado el autor, tras la realización del mismo.

A continuación dispondremos de varios Anexos para completar el contenido del trabajo.

En el anexo A, se expondrá paso a paso el proceso de instalación de todas las tecnologías de las que hemos hecho uso durante el transcurso de elaboración del proyecto, para facilitar al lector la puesta en marcha por si desea realizar pruebas. En concreto la instalación de Eclipse y la del apk de Android.

En el anexo B, hablaremos de la importancia de la seguridad de los datos sesibles del usuario en las aplicaciones, y como podemos dotar de mayor seguridad a las aplicaciones.

En el último anexo se mostrará todo el código de la aplicación, tanto el código java como el xml que conforman el conjunto de todo proyecto Android.

2 Tecnologías utilizadas

En el segundo capítulo, realizaremos una descripción de todo el entorno de desarrollo del que se ha hecho uso en la realización de nuestro proyecto, explicándolos uno a uno de forma detallada.

2.1 Android

A continuación hablaremos de qué es Android y sus principales características, su historia y evolución, su arquitectura, una comparativa con otras plataformas existentes y por último, su impacto en el mercado.

Figura 2.1 Logo de Android.

2.1.1 ¿Qué es Android?

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Firefox OS, Windows Phone, Symbian y Blackberry . Lo que lo hace

diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

El sistema permite programar aplicaciones en una variación de la maquina virtual de Java llamada Dalvik. El S.O. proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, la agenda de contactos…) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

2.1.2 Historia

Android Inc. era un sistema operativo prácticamente desconocido hasta que Google lo adquirió en el año 2005. Ese mismo año, comienzan a trabajar en la creación de una máquina virtual Java optimizada para móviles.

En el año 2007 se creó el consorcio Handset Alliance formado por varias empresas, entre ellas Google, con el objetivo de desarrollar estándares abiertos para móviles. Uno de sus objetivos era promover el diseño y la difusión de la plataforma Android.

En noviembre del 2007 se lanza una primera versión del Android SDK. Al año siguiente apareció el primer teléfono con Android (T-Mobile G1). En octubre Google libera el código fuente de Android bajo licencia de código abierto Apache. Ese mismo mes se abre Android Market para la descarga de aplicaciones. En 2009 se lanza la versión 1.5 del SDK que incorpora el teclado en pantalla. A finales del 2009 se lanza la versión 2.0 y durante el 2010 las versiones 2.1, 2.2 y 2.3.

En el año 2010 llegaría la consolidación de Android, como uno de los sistemas operativos móviles más usados.

En 2011 se lanzan las versiones 3.0, 3.1 y 3.2 específica para tablets y la 4.0 tanto para móviles como para tablets. Durante este año Android se consolida como la plataforma para móviles más importante, alcanzando cuotas de mercado del 50 %.

En 2012 Google reemplaza su Android Market por Google Play Store y se lanzan las versiones 4.1 y 4.2 del SDK. Android mantiene su espectacular crecimiento alcanzando cuotas de mercado del 70 %.

En 2013 se lanzan las versiones 4.3 y 4.4 y en 2015 se lanza la versión 5.0 (Lollipop). A finales de este año la cuota de mercado alcanza el 85 %.

2.1.3 Características de Android

Android presenta una serie de características que lo hacen diferente, combinando en una misma solución las siguientes cualidades:

- Plataforma realmente abierta.

  Es una plataforma de desarrollo libre basada en Linux y de código abierto. Una de sus grandes ventajas es que se puede usar , copiar e incluso modificar el sistema sin pagar derechos de autor.

- Adaptable a cualquier tipo de hardware.

  Android no ha sido diseñado para su uso exclusivo en móviles y tabletas. Hoy en día podemos encontrar relojes, cámaras... una gran variedad de sistemas empotrados que se basan en este sistema operativo. Este hecho proporciona grandes ventajas, pero también requiere un extra de esfuerzo por parte del programador. La aplicación debe funcionar correctamente en dispositivos con una amplia variedad de tipos de entrada, pantalla, memoria, etc. Esta característica contrasta por ejemplo con iOS, que tenemos que desarrollar una aplicación diferente para iPhone y otra diferente para iPad.

- Portabilidad asegurada.

  Las aplicaciones finales son desarrolladas en Java, lo que nos proporciona que podrán ser ejecutadas en cualquier tipo de CPU. Esto se consigue gracias al concepto de máquina virtual.

- Arquitectura basada en componentes inspirados en Internet.

  La interfaz de usuario se realiza en xml.

- Filosofía de dispositivo siempre conectado a Internet.

- Gran cantidad de servicios incorporados.

  Como localización basada en GPS o en redes, navegador, multimedia, etc.

- Aceptable nivel de seguridad.

  Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además cada aplicación dispone de una serie de permisos que limitan su rango de actuación.

- Optimizado para baja potencia y poca memoria.

  Android utiliza la máquina virtual Dalvik. Se trata de una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles.

- Alta calidad de gráficos y sonido.

Gráficos vectoriales suavizados, animaciones inspiradas en Flash, incorpora los codecs más comunes de audio y vídeo como H.264, mp4…

2.1.4 Arquitectura Android

Su arquitectura (Figura 2.2) está formada por 4 capas, y cada una de ellas están basadas en software libre.

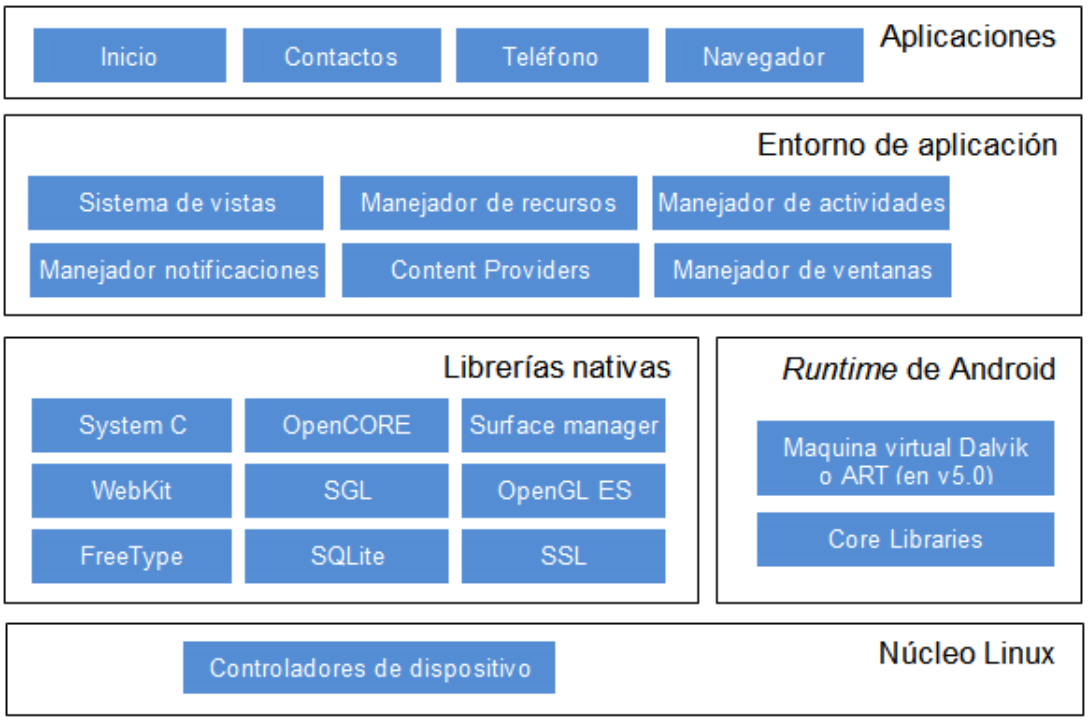

Figura 2.2 Arquitectura de Android.

- El núcleo Linux.

El núcleo de Android está formado por el sistema operativo Linux. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos o el soporte de drivers para dispositivos.

- Runtime de Android.

Está basado en el concepto de máquina virtual utilizado en Java, pero adaptada a las limitaciones de los dispositivos (poca memoria y procesador limitado), la máquina virtual Dalvik. Algunas de sus características son: ejecuta ficheros Dalvik ejecutables (.dex), para ahorrar memoria, cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik.

- Librerías nativas.

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Algunas de estas librerías son: System C library, Media Framework, Surface Manager, SQLite, Webkit

- Entorno de aplicación.

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios,...). Esta capa ha sido diseñada para facilitar la reutilización de componentes. Los servicios más importantes que incluye son: Views (extenso conjunto de vistas), Resource Manager (acceso a recursos que no son en código), Activity Manager (maneja el ciclo de vida de las aplicaciones), Notification Manager (permite a las aplicaciones mostrar alertas personalizadas en la barra de estado) y los Content Providers (mecanismo para acceder a datos de otras aplicaciones).

- Aplicaciones.

Nivel formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la Dalvik para garantizar la seguridad del sistema.

Normalmente las aplicaciones Android están escritas en Java, gracias al Android SDK. Existe también la opción de desarrollarlas en C/C++, pero para esta opción se utilizará el Android NDK

2.1.5 Comparativa con otras plataformas

En la Figura 2.3 se describen las principales características de las principales plataformas móviles. Puede observarse como el uso de Android ofrece a los usuarios multitud de ventajas.

2.1.6 Cuotas de mercado

Según un estudio realizado por la empresa Gartner Group sobre la evolución del mercado según el número de terminales vendidos (Figura 2.4) puede observarse el espectacular ascenso de la plataforma Android desde el 2009 al 2014.

En el segundo cuatrimestre del 2015, según datos de esta misma empresa, reveló que actualmente el 82.2 % de la cuota de mercado la posee Android, seguida de Apple con un 14.6 % y con Microsoft cerrando el podio (Figura 2.5).

En concreto, los datos en España alcanzan el 89.9 % en el primer cuatrimestre del 2015 según Kantar Worldpanel, creciendo 0.9 con respecto al mismo trimestre del

año anterior, mientras que su gran competidor Apple ha descendido en 0.2 puntos.

	Apple iOS 8	**Android 5.0**	**Windows Phone 8**	**BlackBerry 10**	**Firefox OS 2.2**
Compañía	Apple	Open Handset Alliance	Microsoft	BlackBerry	Mozilla Foundation
Núcleo del SO	Mac OS X	Linux	Windows NT	QNX	Linux
Licencia de *software*	Propietaria	Libre y abierto	Propietaria	Propietaria	Libre y abierto
Año de lanzamiento	2007	2008	2010	1999	2013
Fabricante único	Sí	No	No	Sí	No
Variedad de dispositivos	Modelo único	Muy alta	Media	Baja	Muy baja
Soporte memoria externa	No	Sí	Sí	Sí	Sí
Motor del navegador web	WebKit	WebKit/ Chromium (Blink)	Trident	WebKit	WebKit
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry World	Firefox Marketplace
Número de aplicaciones	800.000 (marzo 2013)	800.000 (marzo 2013)	130.000 (enero 2013)	100.000 (enero 2013)	¿?
Coste publicar	$99 / año	$25 una vez	$99 / año	Sin coste	Sin coste
Otras tiendas sin supervisión	No	Si	No	Si	Si
Familia CPU soportada	ARM	ARM, MIPS, x86	ARM	ARM	ARM, x86
Soporte 64 bits	Si	Si	No	No	No
Máquina virtual	No	Dalvik / ART	.net	No	Navegador Web
Lenguaje de programación	Objective-C, C++	Java, C++	C#, Visual Basic, C++	C, C++, Java	HTML5, CSS, JavaScript
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac	Windows, Mac, Linux
Multiusuario	No	Si	No	No	No
Modo invitado	Si	Si	No	No	No

Figura 2.3 Comparativa entre plataformas.

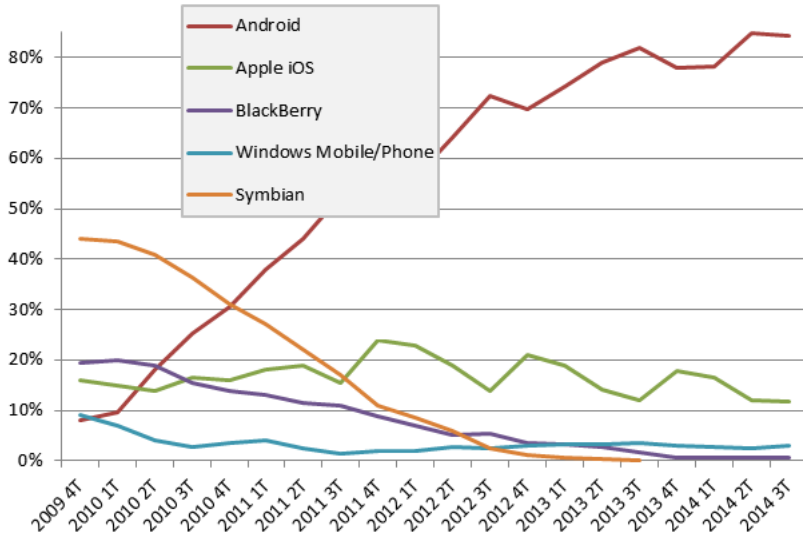

Figura 2.4 Evolución de la cuota de mercado del 2009 al 2014.

Worldwide Smartphone Sales to End Users by Operating System in 2Q15 (Thousands of Units)

Operating System	2Q15 Units	2Q15 Market Share (%)	2Q14 Units	2Q14 Market Share (%)
Android	271,010	82.2	243,484	83.8
iOS	48,086	14.6	35,345	12.2
Windows	8,198	2.5	8,095	2.8
BlackBerry	1,153	0.3	2,044	0.7
Others	1,229.0	0.4	1,416.8	0.5
Total	**329,676.4**	**100.0**	**290,384.4**	**100.0**

Source: Gartner (August 2015)

Figura 2.5 Cuota de mercado en el 2015.

2.2 Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y que fue diseñado para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es a partir del 2012, uno de los lenguajes de programación más usados, particularmente para aplicaciones de cliente-servidor web, con unos 10 millones de usuarios.

Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades a

bajo nivel. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Figura 2.6 Logo de Java.

Las características principales de Java son:

- Orientado a objetos. Diseño del software de forma que los distintos tipos de dato que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos).

- Independencia de la plataforma. Programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware.

- El recolector de basura. En Java el problema de fugas de memoria se evita en gran medida gracias al recolector de basura. El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas.

El diseño de Java y su robustez hace que Java sea muy utilizado en distintos ámbitos:

- En dispositivos móviles y sistemas embebidos.

  Desde la creación de la especificación J2ME (Java 2 Platform, Micro Edition), existe una versión del entorno de ejecución Java reducido y altamente optimizado, especialmente desarrollado para el mercado de dispositivos electrónicos de consumo.

- En el navegador web.

Desde la primera versión de java existe la posibilidad de desarrollar pequeñas aplicaciones (Applets) en Java que luego pueden ser incrustadas en una página HTML para que sean descargadas y ejecutadas por el navegador web. Estas mini-aplicaciones se ejecutan en una JVM que el navegador tiene configurada como extensión (plug-in) en un contexto de seguridad restringido configurable para impedir la ejecución local de código potencialmente malicioso.

- En sistemas de servidor.

  En la parte del servidor, Java es más popular que nunca, desde la aparición de la especificación de Servlets, componentes de la parte del servidor de Java, encargados de generar respuestas a las peticiones recibidas de los clientes.

- En aplicaciones de escritorio.

  Hoy en día existen multitud de aplicaciones gráficas de usuario basadas en Java.

2.3 Eclipse

Eclipse es un entorno de desarrollo integrado (IDE), de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones en Java. En él encontrarás todas las herramientas y funciones necesarias, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

Figura 2.7 Logo de Eclipse.

Fue desarrollado originariamente por IBM como el sucesor de su familia de herramientas para VisualAge. Actualmente es desarrollado por la fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El entorno de desarrollo integrado de Eclipse emplea módulos (plug-in) para proporcionar toda su funcionalidad al cliente a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos consigue una plataforma más ligera en cuando a software.

2.3.1 Uso de Eclipse en el proyecto

Para llevar el proyecto a cabo, se ha utilizado Eclipse, ya que resulta el entorno de desarrollo más recomendable para Android, proporcionando al programador muchas facilidades. En uno de los anexos especificaremos toda su instalación y puesta en marcha para la programación en Android. También se podría haber usado Android Studio, un entorno de desarrollo específico para la plataforma Android.

La estructura de ficheros de cualquier proyecto Android desarrollado en Eclipse es la que se muestra en la Figura

Está formado básicamente por un descriptor de la Aplicación (AndroidManifest.xml), el código fuente en Java y una serie de ficheros con resursos.

src : Carpeta que contiene el código fuente de la aplicación. Todos los ficheros Java se almacenarán en esta carpeta.

gen : Carpeta que contiene el código generado de forma automática por el SDK. Nunca hay que modificar de forma manual estos ficheros. Dentro encontramos.

- **BuildConfig.java** : Define la constante DEBUG para que desde Java puedas saber si tu aplicación está en fase de desarrollo.
- **R.java** : Define una clase que asocia los recursos de la aplicación con identificadores. De esta forma podremos acceder a los recursos desde el código Java.

assets : Carpeta que puede contener una serie arbitraria de ficheros carpetas que podrán ser usados por la aplicación. A diferencia de la carpeta res, nunca se modifica el contenido de esta carpeta, ni se les asociará un identificador.

bin : En esta carpeta se compila el código y se genera el .apk, fichero comprimido que contiene la aplicación final lista para instalar.

libs : Código jar con las librerías que quieras usar en tu proyecto.

res : Carpeta que contiene los resursos usados por la aplicación. Dentro encontraremos una serie de carpetas entre las que destacan:

- **drawable** : En esta carpeta se almacenan los ficheros de imágenes y descriptores de imágenes en xml.
- **layout** : Contiene los ficheros xml con las vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que componen la interfaz del usuario.

anim : Contiene ficheros xml con animaciones Tween. En nuestra aplicación, hemos añadido este tipo de animaciones para darle efectos a la pantalla Inicio, cuando el usuario inicia sesión.

raw : Ficheros adicionales que no se encuentran en formato xml. En nuestro caso se trata de la carpeta donde hemos introducido los vídeos y el archivo JSON de la rutina que incorpora el paquete de nuestra aplicación.

AndroidManifest.xml : Este fichero describe la aplicación Android. En él se indican las actividades, intenciones y proveedores de contenido de la aplicación. También se declara los permisos que requerirá nuestra aplicación. En nuestro caso, los permisos de acceso a Internet (para el envío del e-mail con la nueva contraseña), y el de escritura en la memoria del dispositivo, para crear las carpetas donde almacenar las rutinas.

2.4 JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos. Su principal ventaja es que puede ser leído por cualquier lenguaje de programación, por tanto es muy útil para el intercambio de información entre distintas tecnologías. Otro punto a favor es su facilidad de uso tanto para los seres humanos, ya que es fácil de leer o escribir, como para las máquinas por su facilidad de generación y análisis.

Se basa en un subconjunto del lenguaje de programación JavaScript. JSON describe los datos con una sintaxis y estructura específicas, para identificarlos y gestionarlos. JSON nació como una alternativa a XML.

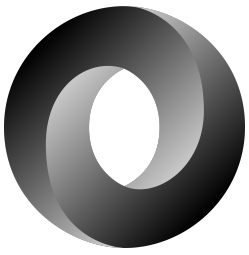

Figura 2.8 Logo de JSON.

Vamos a familiarizarnos con su sintaxis:

- JSON par elemento/valor.

Para asignar a un elemento un valor determinado, debemos separar el elemento de su valor, por medio del carácter ‘:’ .Ejemplo: “Nombre” : “valor”.

- Valores JSON.

  Los tipos de valores que podemos encontrarnos en un fichero JSON son, un número, un String, un booleno (true o false), un array (entre corchetes []), un objeto (representado entre llaves), o null.

- Objeto JSON.

  Los objetos JSON van encerrados entre llaves {}.

```
{
  "Nombreejercicio": "Extension de pierna",
  "Repeticiones": 25
}
```

- Array JSON.

  En JSON el contenido de un array tiene que ir entre corchetes []. Ejemplo:

```
[
  {
    "Nombre": "video01",
    "Titulo": "Extension de pierna",
    "Subtitulo": "zona: piernas",
    "Descripcion": "Descripcion extensa del ejercicio",
    "Estado de forma": "Bajo",
    "Repeticiones": 6
  },
  {
    "Nombre": "video02",
    "Titulo": "Contraccion abdomen",
    "Subtitulo": "zona: abdomen",
    "Descripcion": "Descripcion extensa y detallada del
       ejercicio2",
    "Estado de forma": "Medio",
    "Repeticiones": 4
  }
]
```

Mezclando todos estos elementos podremos crear cualquier estructura de almacenamiento que queramos, en la que los objetos y los arrays pueden contenerse los unos a los otros y formar estructuras tan complejas como queramos.

JSON es la mejor herramienta para compartir estructuras de datos simples, de ahí nuestra elección. Mantiene una alta rapidez de procesamiento, ya que transfiere menos datos, es muy flexible, la estructura es muy fácil de modificar y de añadir nuevos campos y la librería de Java incluye funciones para su tratamiento.

2.5 SQLite

SQLite es un sistema gestor de base de datos relacional. Lo que hace único a SQLite es que se considera una solución embebida y muy ligera, por lo que se trata de una solución ideal para dispositivos móviles.

Figura 2.9 Logo de SQLite.

La mayoría de los sistemas gestores de bases de datos son procesos de servidor autónomos que se ejecutan independientemente. SQLite es en realidad una librería que está enlazada dentro de las aplicaciones. Todas las operaciones de base de datos se manejan dentro de la aplicación mediante llamadas y funciones contenidas dentro de la librería SQLite. SQLite soporta las funciones de las bases de datos relacionales y sintaxis SQL.

2.5.1 Características

Las principales características de este sistema gestor de bases de datos son:

- Una base de datos completa se encuentra en un solo archivo. SQLite no tiene un proceso servidor independiente, lee y escribe directamente de archivos de disco ordinarios.
- Requiere poca memoria en tiempo de ejecución (250 kB), por lo que va a ser muy rápida.

- El formato de archivo de base de datos es multiplataforma. Puede copiar libremente una base de datos entre los sistemas de 32 bits y de 64 bits o entre arquitecturas big-endian y little-endian.

- Es una biblioteca muy compacta. Con todas las características habilitadas, el tamaño de la biblioteca puede ser inferior a 500KB. SQLite también se puede hacer para funcionar en un espacio mínimo de pila (4KB), ideal para dispositivos de memoria limitada, como móviles.

- Es totalmente autocontenida, sin dependencias externas.

- Cuenta con librerías de acceso para muchos lenguajes de programación, entre ellos Android.

- Soporta texto en formato UTF-8 y UTF-16, así como datos numéricos de 64 bits.

- Soporta funciones SQL definidas por el usuario (UDF).

- El código fuente es de dominio público y se encuentra muy bien documentado.

La base de datos que crea una determinada aplicación es sólo accesible por ella, otras aplicaciones no podrán acceder. Una vez creada, la base de datos se almacena en la carpeta /data/data/<nombre_del_paquete>/databases de un dispositivo Android.

Para SQLite trabajan un equipo internacional de desarrolladores que siguen ampliando las capacidades de SQLite y mejorando su fiabilidad y rendimiento.

2.5.2 ¿Dónde usar SQLite?

- En dispositivos embebidos y en “el internet de las cosas”.

  Debido a que SQLite es una base de datos que no requiere administración, funciona bien en dispositivos que deben operar sin apoyo humano experto. Es una buena opción para su uso en teléfonos móviles, decodificadores, televisores, consolas de juego, cámaras, relojes, aparatos de cocina, automóviles, etc.

- Sitios web.

  SQLite funciona como sistema gestor de base de datos para páginas web de tráfico bajo-medio. La cantidad de tráfico web que SQLite puede manejar depende de la frecuencia con que la página web utiliza su base de datos. En general, cualquier sitio que recibe menos de 100k visitas/día debe funcionar bien con SQLite, aunque este límite es muy conservador. La propia página web de SQLite https://www.sqlite.org/ utiliza SQLite y maneja sobre 400k-500k

peticiones HTTP por día, el cual el 15 %-20 % de estas peticiones tocan la base de datos.

- Análisis de datos.

  Los expertos en SQL pueden usar la línea de comandos (terminal) de SQLite para analizar grandes conjuntos de datos. Los análisis complejo se pueden realizar utilizando simples scripts escritos en Tcl o Python o en R. Los usos posibles incluyen por ejemplo análisis de sitios web, análisis de estadísticas deportivas o análisis de resultados experimentales.

- "Caché" de datos empresariales.

  Muchas empresas, utilizan SQLite como si fuera una “caché” que contiene los datos más utilizados en RDBMS empresariales. Esto reduce la latencia, ya que la mayoría de consultas irán a esta “caché”. También se reduce la carga de la red y en el servidor de datos central. Gracias a esto, en muchos casos la aplicación del lado del cliente puede continuar operando durante interrupciones de la red.

- Base de datos del lado del servidor.

  SQLite puede usarse como un almacén de datos en aplicaciones de servidor.

- Reemplazo de archivos especiales de disco.

  Muchos programas usan fopen(), fread() y fwrite() para crear y gestionar ficheros normales de datos. SQLite funciona bien como reemplazo de estos archivos.

- Bases de datos internas o temporales.

  Para los programas que tienen una gran cantidad de datos que tienen que ser ordenados de diversas formas.

- Sustituto de sistemas gestores de bases de datos durante demostraciones o pruebas.

- Educación y entrenamiento.

  Debido a que es fácil de configurar y usar, SQLite es un buen sistema gestor de base de datos para su uso en la enseñanza de SQL.

- Extensiones del lenguaje SQL experimentales.

  El diseño simple y modular de SQLite hace que sea una buena plataforma para la creación de nuevos prototipos.

2.5.3 Tipos de datos

Cada valor almacenado en una base de datos SQLite tiene una de las siguientes clases de almacenamiento:

- NULL: para valores null.
- TEXT: el valor es una cadena de texto, almacenada utilizando la codificación de la base de datos (UTEF-8, UTF-16).
- INTEGER: el valor es un entero con signo almacenado en 1, 2, 3, 4, 6 u 8 bytes dependiendo de la magnitud del valor.
- REAL: almacena un valor de punto flotante.
- BLOB: para datos en binario.

Como vemos, SQLite no tiene una clase de almacenamiento boolean, los valores booleanos se almacenan como enteros 0 (false) o 1 (true). Tampoco tiene una clase de almacenamiento reservado para recoger fechas o tiempo. En su lugar, SQLite los almacenará como valores TEXT, REAL o INTEGER. En el caso de nuestra aplicación, hemos elegido almacenarlos como TEXT.

La forma recomendada para crear una nueva base de datos SQLite, es instanciar un objeto de una clase que herede de la clase SQLiteOpenHelper y sobrescribir el método onCreate(), en el cuál se ejecutarán los comandos para crear las tablas necesarias para estructura de la base de datos. Para escribir y leer de la base de datos, se usan los métodos getWritableDatabase() y getReadableDatabase() respectivamente. Ambos métodos devuelven un objeto de la clase SQLiteDatabase que representa a la base de datos y proporciona métodos para su gestión como INSERT, DELETE, QUERY Y RAWQUERY (sentencia de consulta SQL que devuelve los resultados en forma de objeto Cursor). Este objeto Cursor que devuelve, es el mecanismo con el que se puede navegar por los resultados de una consulta de la base de datos, y leer filas y columnas.

2.6 Android Plot

AndroidPlot es una librería para la creación de gráficos dinámicos y estáticos dentro de una aplicación Android. Se pueden generar gráficas de todo tipo, tanto de líneas como de barras, circulares, etc. Además, su uso es libre bajo licencia Apache License, v. 2.0.

Figura 2.10 Logo de AndroidPlot.

Como cualquier librería, cuando la descargamos hay que meterla en eclipse en la carpeta “libs” del proyecto para poder usar sus funciones.

En nuestra aplicación en concreto, hacemos uso de ella para dibujar la gráfica de línea del tiempo de entrenamiento a la semana de un usuario. Para ello teníamos varias opciones de librerías similares a ésta, y también gratis, como GraphView, MPAndroidChart, AChartEngine, etc. Sin embargo, la elección de AndroidPlot se debe a que es bastante completa y personalizable, y existe bastante documentación, además la utilidad que vamos a darle no requiere de librerías demasiado especializadas, con las funciones que incorpora AndroidPlot nos es suficiente.

3 Modelado de la información

Este capítulo explica el modelado de la información que se ha realizado, se analizará las estructura de la base de datos, mostrando las tablas de las que se componen y una descripción de cada una, además de las relaciones entre ellas.

3.1 Modelo EE-RR

La Figura 3.1, muestra el modelo entidad-relación de nuestra base de datos. Como vemos se trata de un modelo muy sencillo en la que intervienen muy pocas tablas. SQLite sirve para esto, para la creación de pequeñas bases de datos.

3.2 Uso de las tablas

En este apartado se realizará una descripción de cada una de las tablas de nuestra base de datos SQLite "app_deportes", indicando los campos por los que está formada.

usuarios Tabla que almacena los datos principales de los usuarios, tales como nombre, apellidos, usuario, hash y salt (para encriptar la contraseña), e_mail y su estadodeforma. Como clave primaria está el campo "Id" para identificarlo de manera única en la tabla.

rutinas Contiene las distintas rutinas y sus datos, que han sido descomprimidas por el usuario desde la carpeta Download. Esta tabla, cuando se crea, se le introduce automáticamente la rutina "GAP", que está incorporada dentro del apk de la aplicación. Contiene también información sobre cada rutina: nombre, una descripción breve y la ruta que tendrá dentro del dispositivo una vez descomprimida, y que será del tipo: /storage/emulated/0/app_deportes/rutinas/…

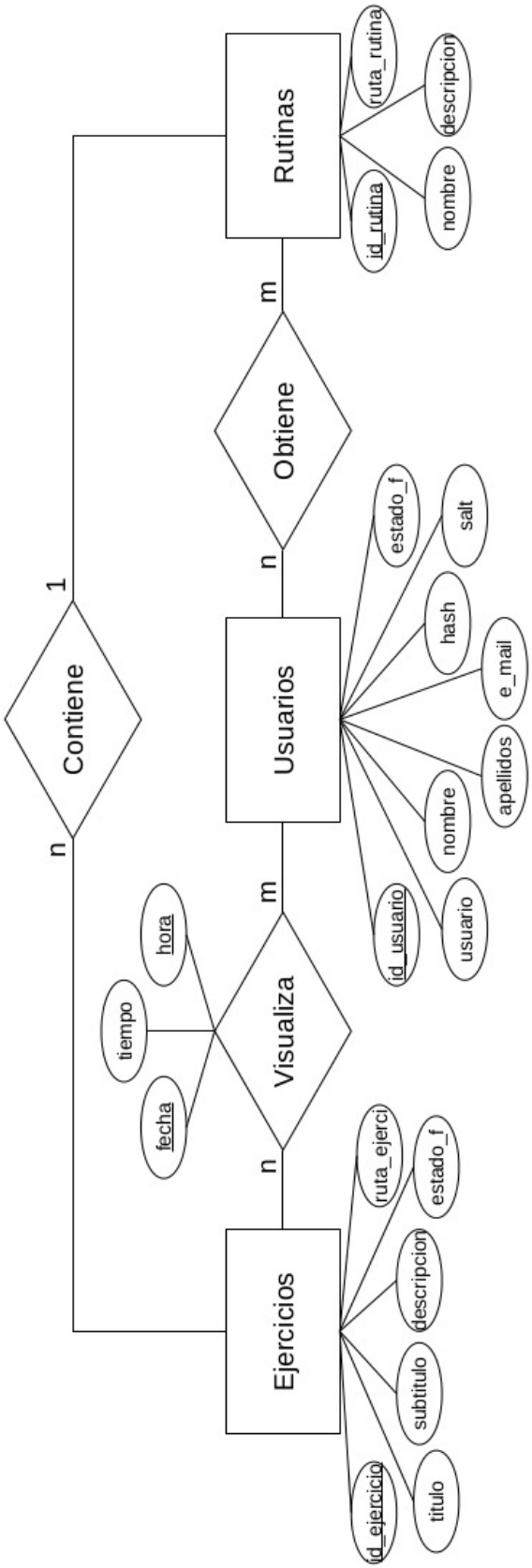

Figura 3.1 Modelo EE-RR de la base de datos "app_deportes".

Contiene también el campo id_rutina que es la clave primaria, para distinguir a cada una de las rutinas.

ejercicios Contiene cada uno de los ejercicios que contienen las rutinas, junto con toda su información y la ruta a la que pertenecen. Sus campos son ruta_ejercicio (la ruta del vídeo), título, subtítulo, descripción (descripción extensa que el usuario debería leer antes de ejecutar el ejercicio, pues incluye los músculos implicados en la realización, o cuál es la forma correcta de ejecutarlo), estadodeforma (el estado de forma física para el que resulta óptimo), repeticiones (número de repeticiones que deben de hacerse de este ejercicio para que resulte efectivo) y el id_rutina, identificador de la rutina a la que pertenece el ejercicio. Como clave primaria está el campo "id_ejercicio" para identificar a cada ejercicio de manera única en la tabla.

ejercicios_visualizados Contiene cada uno de los ejercicios (vídeos) que un usuario determinado ha realizado. Recordemos que el sistema considera que un ejercicio ha sido realizado por un usuario, cuando ha pulsado el botón "Comenzar", y posteriormente el botón "Terminado". Cada fila contendrá también los campos fecha y hora en la que lo realizó, y la duración del tiempo que empleó para ejecutarlo. Cada uno de los ejercicios puede realizarse varias veces por el mismo usuario, pero cada vez tendrá una fecha y hora distinta de realización, es por ello que las claves primarias de esta tabla son cuatro: id_usuario, id_ejercicio, fecha y hora.

usuario_rutina Contiene cada una de las rutinas que tiene en su lista, un usuario determinado. Contiene tan sólo dos campos, y ambos son claves primarias, los campos id_usuario e id_rutina.

Nos sirve tanto para mostrar la lista de rutinas personalizadas de cada usuario, como para llevar una cuenta de cuántos usuarios tiene una rutina en concreto. Si un usuario elimina una rutina de su lista, no se eliminará de la carpeta del dispositivo (…/app_deportes/rutinas/) a no ser que fuera el único usuario que poseía esa rutina en su lista.

4 Arquitectura y análisis

En este apartado haremos uso del UML (Lenguaje Unificado de Modelado) para analizar nuestro sistema de forma detallada. Lo examinaremos por medio de tres tipos de diagramas. Diagramas de casos de uso, diagramas de secuencia y el diagrama de clase.

4.1 Diagramas de Casos de uso

A continuación se expone una descripción de los casos de uso y sus diagramas para una mejor comprensión, posteriormente se especifican los actores que harán uso del sistema con el fin de identificar todos los casos de uso a desarrollar.

4.1.1 Identificación de actores

Usuario: Este actor es el usuario del sistema, el cual podrá acceder a las funciones del sistema.

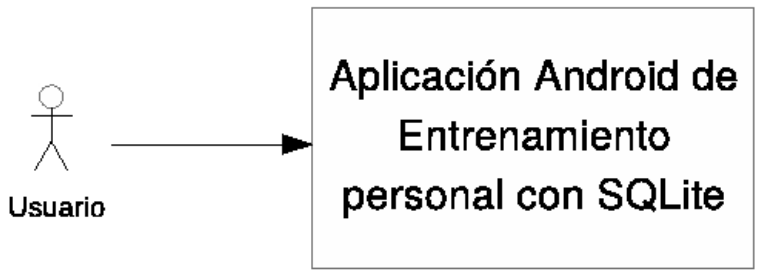

Figura 4.1 Identificación de actores.

4.1.2 Casos de uso

En el diagrama se muestran los casos de uso identificados que se desarrollan.

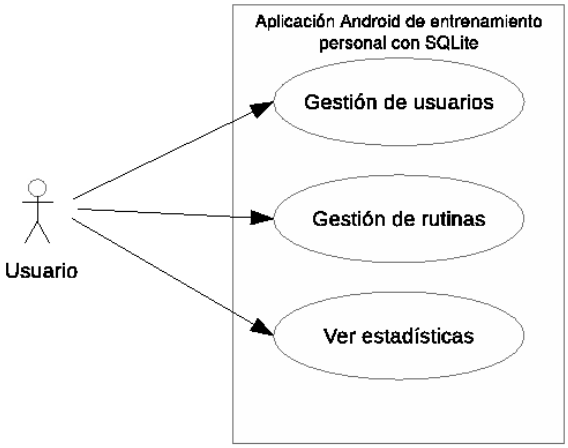

Figura 4.2 Diagrama de casos de uso.

Tabla 4.1 Especificación del caso de uso 1 - Gestión de usuarios.

Caso de uso	*Gestión de usuarios*
Objetivo	*Administrar los usuarios locales de la aplicación.*
Actores	*Usuario.*
Disparador	*Este caso de uso comienza cuando un usuario se registra, inicia sesión o selecciona olvido de contraseña.*
Precondiciones	*Ninguna.*
Descripción	*El actor accede a las distintas operaciones que se pueden realizar en la gestión de usuarios.*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
	Opciones para un Usuario que no se ha identificado: • *Identificarse en el sistema (iniciar sesión)* • *Registrarse en el sistema* • *Indicar olvido de contraseña* *Opciones para un Usuario identificado:* • *Modificar datos privados del usuario* • *Cerrar sesión* • *Eliminar cuenta de usuario* • *Modificar contraseña del usuario*
Cursos alternativos	
Ninguno.	

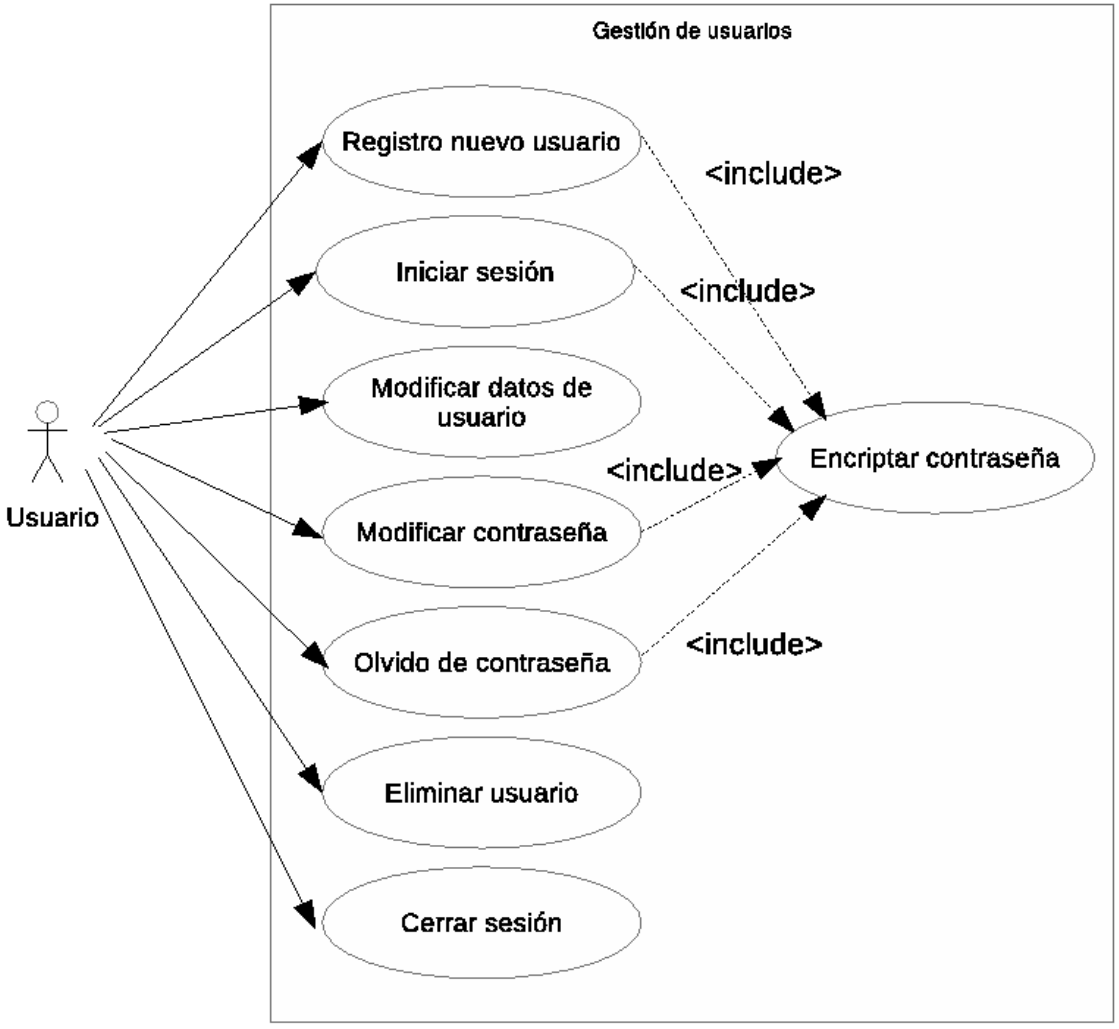

Figura 4.3 Caso de uso 1 - Gestión de usuarios.

Tabla 4.2 Especificación del caso de uso 2 - Gestión de rutinas.

Caso de uso	*Gestión de rutinas*
Objetivo	*Administrar las rutinas y ejercicios de la aplicación.*
Actores	*Usuario.*
Disparador	*Este caso de uso comienza cuando un usuario selecciona una rutina de su lista, o bien elige cargar rutinas de otros usuarios, o de la carpeta de descargas del dispositivo.*
Precondiciones	*El usuario debe haberse registrado previamente, e iniciado sesión.*
Descripción	*El actor accede a las distintas operaciones que se pueden realizar en la gestión de rutinas.*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
	Opciones para rutinas de su lista: • *Mostrar lista de ejercicios de la rutina acorde a su nivel de forma física.* • *Reproducción del ejercicio.* • *Parar la reproducción del ejercicio.* • *Pasar al siguiente ejercicio de la rutina.* • *Indicar que el usuario va a proceder a realizar el ejercicio.* • *Indicar que el usuario a finalizado el ejercicio.* *Opciones para cargar otras rutinas que no están en su lista.* • *Guardar en su lista, rutinas disponibles de otros usuarios.* • *Acceder a las rutinas de la carpeta download y descomprimirlas.*
Cursos alternativos	
Ninguno.	

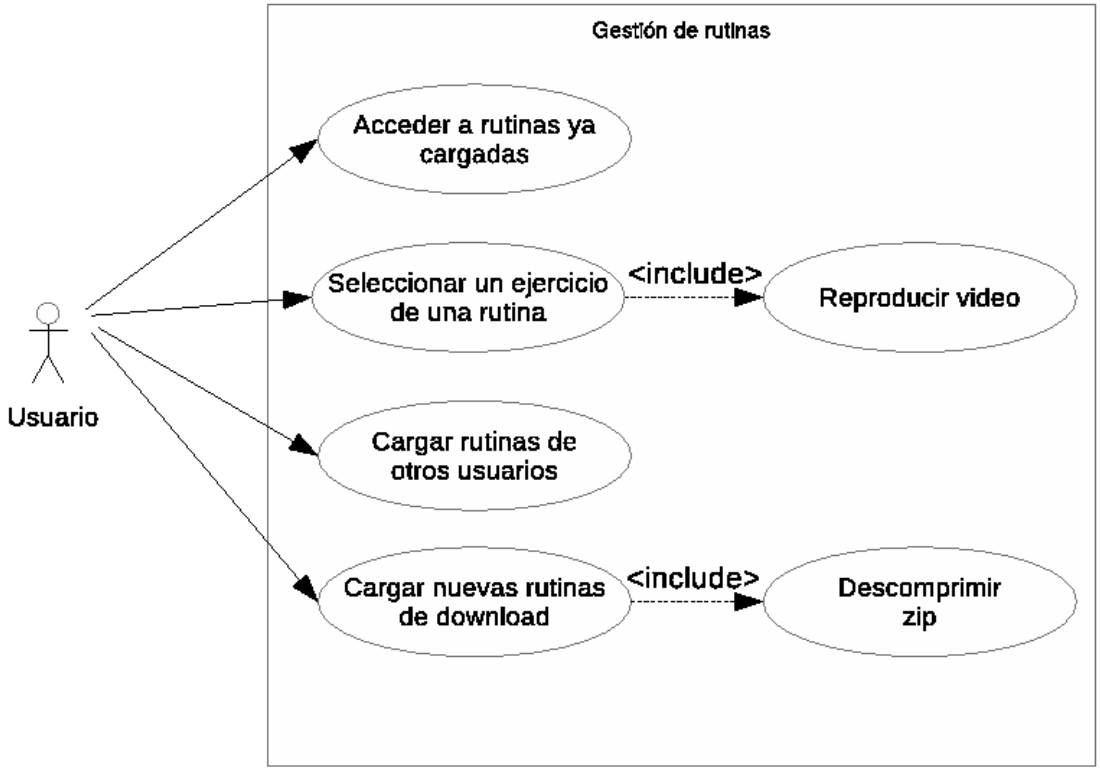

Figura 4.4 Caso de uso 2 - Gestión de rutinas.

4.2 Diagramas de secuencia

En estos diagramas veremos la interacción que se produce entre los objetos de la aplicación a través de una línea vertical que representa el tiempo. Veremos los diagramas con funcionalidad más importante en el sistema.

4.2.1 Registro de un usuario

En primer lugar analizaremos el diagrama del proceso de registro de un nuevo usuario en el sistema (Figura 4.5).

Una vez iniciada la aplicación, el usuario debe pulsar sobre el botón “Registrarse” para que pueda iniciar sesión posteriormente. Al pulsarlo, la actual actividad “MainActivity”, lanza un Intent que inicia una nueva pantalla, “NuevoUsuario”, en la que se mostrará al usuario un formulario que debe rellenar con sus datos. Los campos de este formulario son:

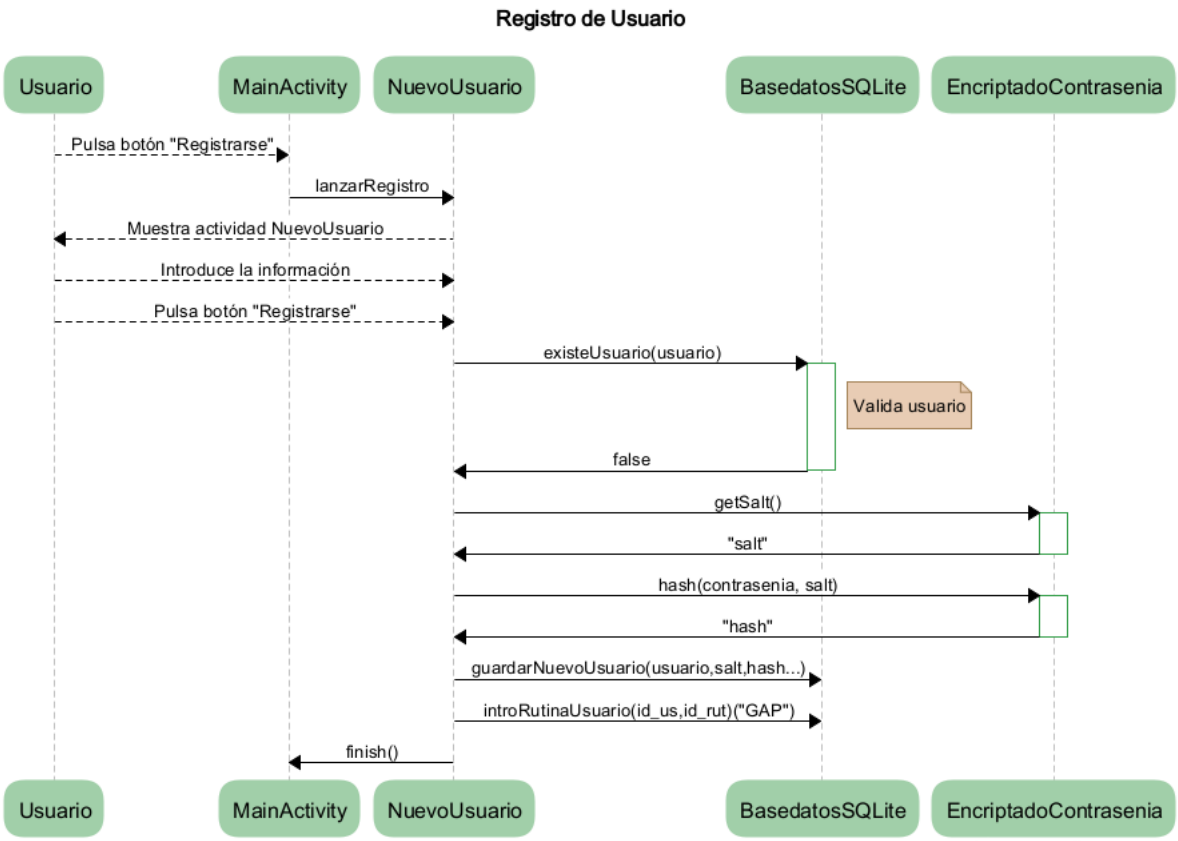

Figura 4.5 Diagrama de secuencia del registro de un usuario.

- Nombre.
- Apellidos.
- Usuario.
- Contraseña.
- E-mail.
- Estado de foma (Bajo, Medio, Alto).

Cuando el usuario pulse el botón registrarse, el sistema realiza varias acciones:

1. Comprueba que todos los campos estén rellenos. En caso contrario creará un objeto toast que muestre un mensaje por pantalla de unos segundos, informando al usuario.

2. Si todos los campos están rellenos, comprueba si el valor del campo usuario introducido ya existe en la tabla usuarios de la base de datos. Si esto ocurre se informa con un toast al usuario para que introduzca un valor diferente.

3. Si el usuario no existe, se genera un salt aleatorio. Se procede a encriptar la contraseña con una función hash(contraseña, salt).

4. Se procede a guardar este nuevo usuario en una nueva fila de la base de datos: guardarNuevoUsuario(usuario, hash, salt, nombre, apellidos, e-mail, estadoforma).

5. Introduce la rutina "GAP" en la tabla usuario_rutina, pues es la rutina que tendrán todos los usuarios por defecto: introRutinaUsuario(id_usuario, id_-rutina). A continuación ejecuta finish() para volver a MainActivity().

4.2.2 Inicio de sesión

Vamos a ver el proceso que se sigue cuando un usuario inicia sesión (Figura 4.6).

Una vez el usuario se ha registrado satisfactoriamente podrá iniciar sesión. Cuando el usuario pulsa sobre el botón "Iniciar" el sistema:

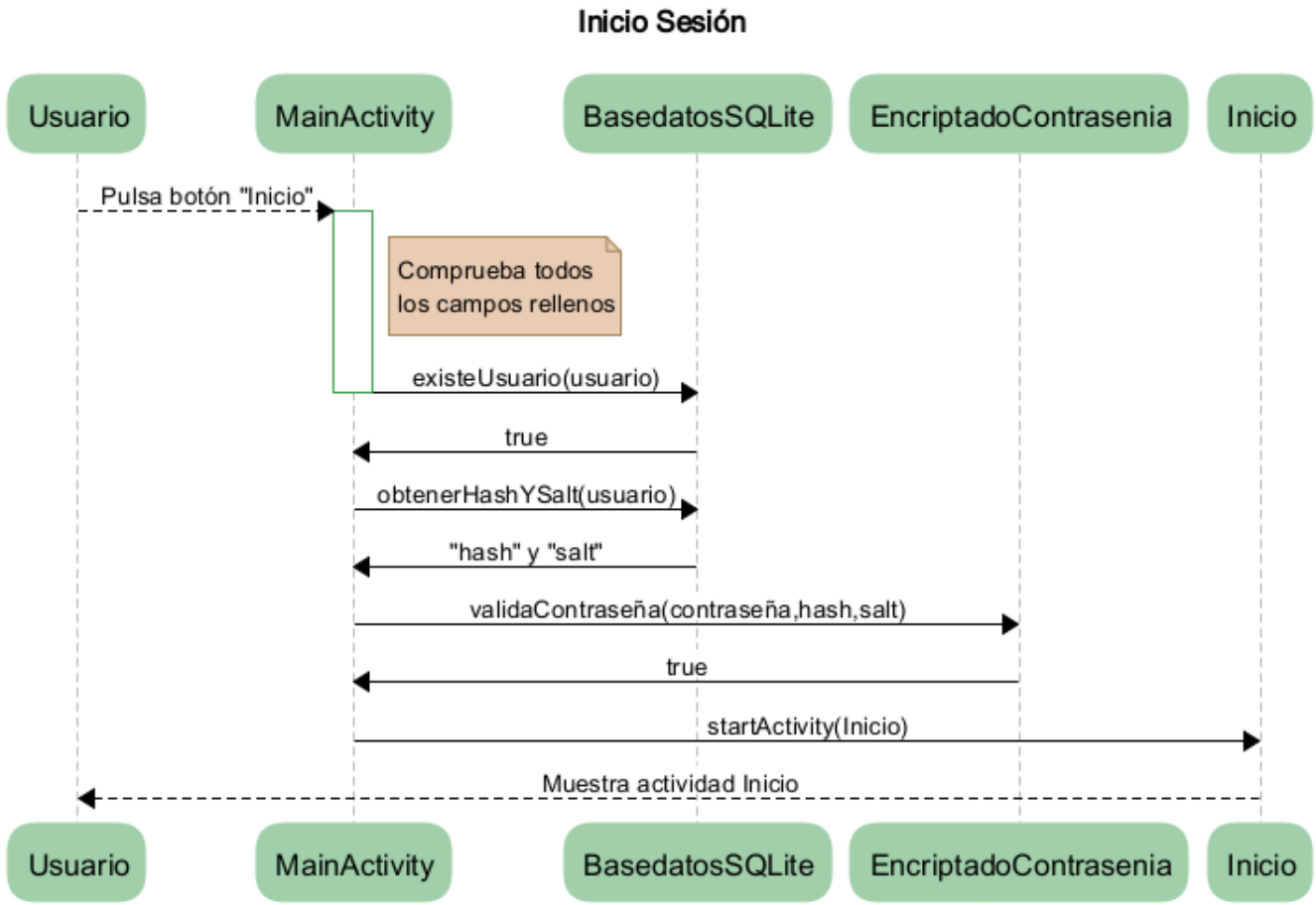

Figura 4.6 Diagrama de secuencia de un usuario que inicia sesión.

1. Comprueba que los campos usuario y contraseña no están vacíos. En cuyo caso se informa con un mensaje toast al usuario.

2. Si ambos campos contienen caracteres, comprobamos si el valor de usuario existe en la tabla usuarios: basedatos.existeUsuario(usuario). En caso de no existir, se informa del error con un toast.

3. Si existe, extraemos sus campos hash y salt de la base de datos: obtenerHashY-Salt(usuario) y validamos que la contraseña introducida, genera el mismo valor de hash que el extraído: validaContrasenia(contraseña, hash, salt).

4. Si ambas coinciden, el sistema procede a lanzar la actividad Inicio.

4.2.3 Acceso a los ejercicios de una rutina

Veamos cómo accede el sistema a los ejercicios de una rutina (Figura 4.7).

Una vez el usuario ha iniciado sesión, y se encuentra en la pantalla Inicio, se encontrará con varios elementos View, entre ellos un ListView cuyos ítems son los nombres de las rutinas que ha descargado en otra ocasión. La primera vez, esta lista sólo contendrá la rutina GAP, y el ítem de "Cargar más" que veremos más adelante.

Si el usuario pulsa sobre una de las rutinas, el sistema activará la pantalla ListEjercicios, pasándole como extras el identificador y el path de la rutina seleccionada. En esta actividad, el sistema:

1. Obtiene el nombre de la rutina y su descripción, para que sean visualizadas en la parte superior de la pantalla: obtenerNombreRutina(id_rutina), obtenerDescripcionRutina(id_rutina).

2. Obtiene el estado de forma del usuario, y obtiene sólo los ejercicios de la rutina que coinciden con el estado de forma del usuario: listaEjEstadoForma(ruta, estado)

3. Creamos un adaptador definido en la clase AdaptadorEjercicio con los datos de títulos, subtítulos y la lista de ejercicios que ya ha visualizado anteriormente el usuario: ListAdapter adaptador= new AdaptadorEjercicio(uris, títulos, subtitulos, id_us, visualizados).

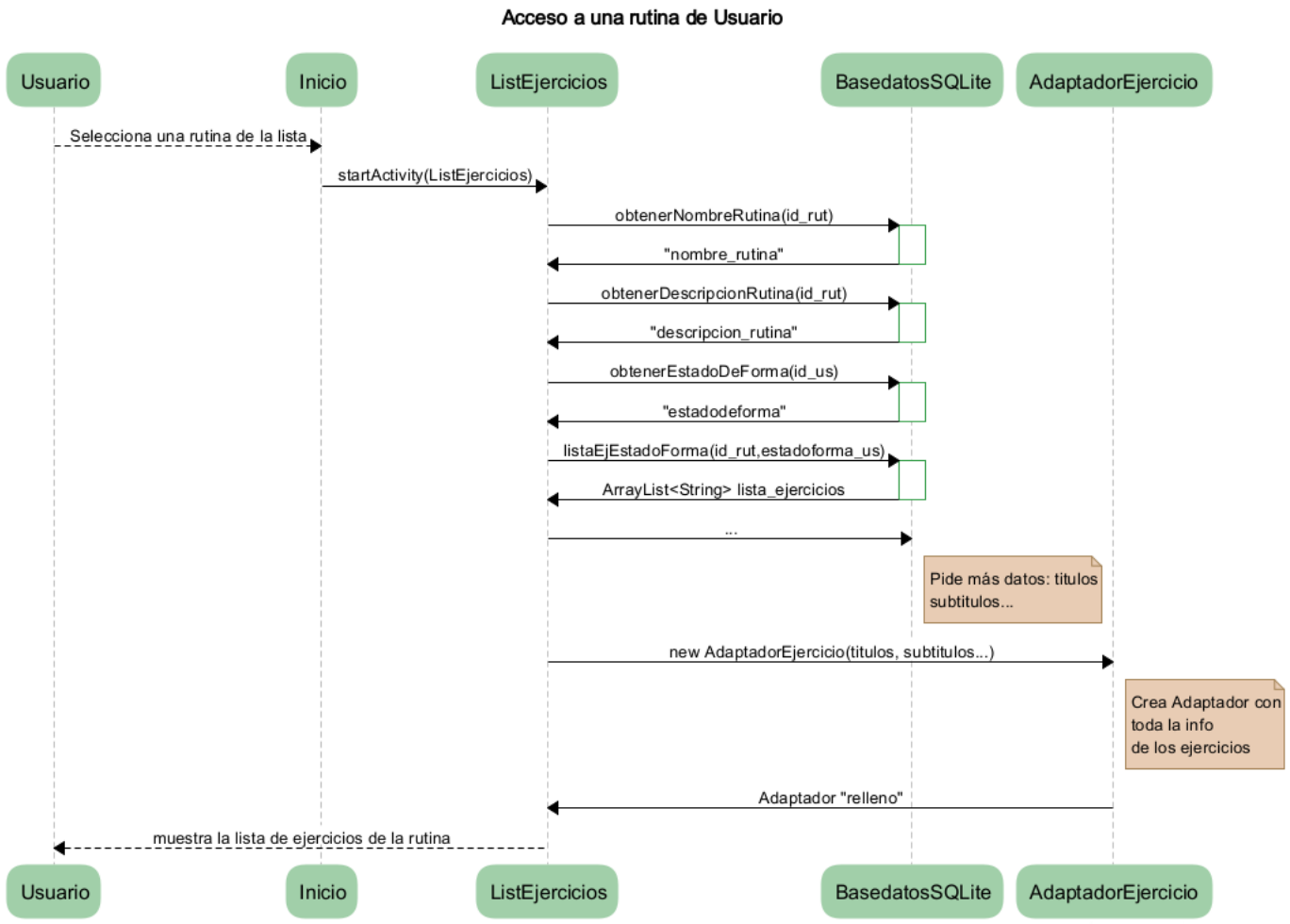

Figura 4.7 Diagrama de secuencia de acceso a la lista de ejercicios de una rutina.

4.2.4 Obtener rutinas de otros usuarios

El siguiente diagrama nos muestra el proceso de acceso a otras rutinas que ya han descomprimido otros usuarios en la carpeta /app_deportes/rutinas/, y que por tanto no será necesario repetir este proceso (Figura 4.8).

En la actividad Inicio, si el usuario pulsa sobre el ítem “Cargar más”, el sistema activará la pantalla ListMasRutinas. El sistema obtendrá la lista con los identificadores de rutinas que ya existen en la tabla usuario_rutina, pero de la que no se dispone de una fila con su identificador de usuario: obtenerMasRutinas(id_usuario). También obtendrá una lista con sus nombres.

A continuación el sistema añadirá al listView de la actividad ListMasRutinas un adaptador con la lista de nombres de estas rutinas.

Si el usuario hace click sobre una de ellas, y pulsa “Aceptar”, el sistema introducirá una nueva fila en la tabla usuario_rutina, correspondientes al id de la rutina seleccionada y al id del usuario: introRutinaUsuario(id_usuario, id_rutina). Tras esto, el sistema vuelve a la actividad Inicio.

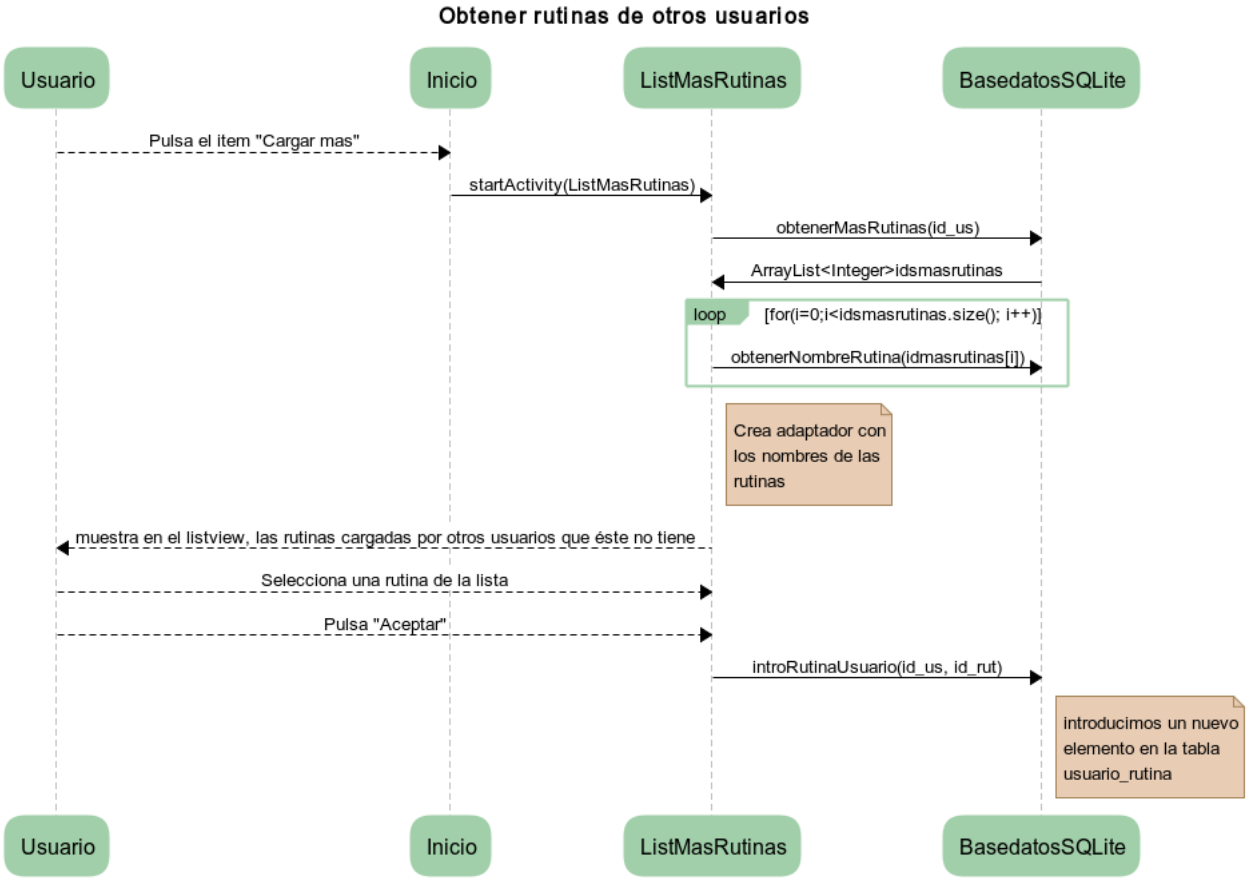

Figura 4.8 Diagrama de secuencia de acceso a la lista de otras rutinas de usuarios.

4.2.5 Cargar nuevas rutinas de la carpeta download del dispositivo

Vamos a ver el proceso de carga de nuevas rutinas comprimidas en archivos zip en la carpeta download (Figura 4.9).

Partimos de que el usuario se encuentra actualmente en la pantalla ListMasRtinas. Si pulsa el botón “Cargar más rutinas” el sistema accionará una nueva pantalla, ListNuevasRutinas, pero antes de lanzarla, realiza las siguientes acciones:

- Obtiene la lista de ficheros que contiene la carpeta download del dispositivo

```
1 File f= new File(Environment.getExternalStoragePublicDirectory(
    Environment.DIRECTORY_DOWNLOADS))
```

```
2 File[] ficheros= f.listFiles()
```

- Almacena sólo la ruta y el nombre de aquellos ficheros que son de extensión .zip

En la pantalla ListNuevasRutinas, se mostrará un listview cuyos ítems serán los nombres de estos ficheros zip que hemos almacenado en la anterior actividad.

Si el usuario pulsa uno de ellos, y después “Aceptar”, ejecutará un objeto de la clase CargaZip, que es de tipo Asynctask, ya que este proceso de descompresión requiere de cierto tiempo y se mostrará un mensaje de espera mientras se realizan todas las operaciones necesarias en un hilo secundario. Esto evitará que el usuario crea que la aplicación ha quedado bloqueada.

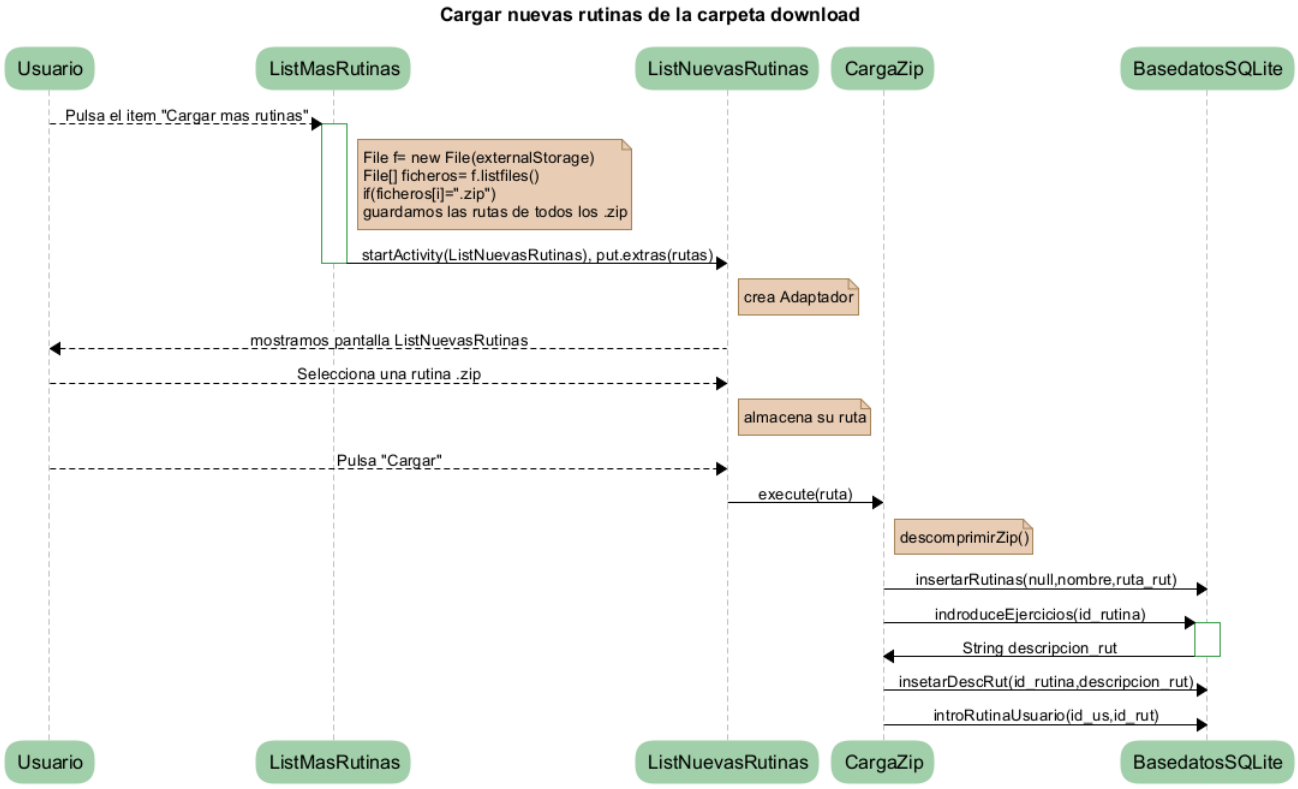

Figura 4.9 Diagrama de secuencia de acceso a nuevas rutinas descargadas por el usuario.

En la función del hilo secundario doInBackground el usuario hará una descompresión del archivo zip, a una carpeta que creará en la ruta: …/app_deportes/rutinas/nombre_rutina. Para ello llamará a su método descomprimirZip(path_rutina).

Una vez se han descomprimido todos los archivos correctamente, mete los vídeos

en la tabla ejercicios, la rutina en la tabla rutinas e introduce una nueva entrada en la tabla usuario_rutina.

4.2.6 Reproducir un ejercicio

Este diagrama (Figura 4.10) nos muestra la secuencia de acciones que realiza el sistema cuando el usuario selecciona un ejercicio de una rutina.

Partimos de que el usuario se encuentra en la clase ListEjercicios, donde ya comentamos que se mostraban los ejercicios de esa rutina correspondientes con el estado de forma del usuario. Si el usuario pulsa sobre uno de ellos, el sistema lanzará la actividad ReproductorEjercicio. Una vez activada, se ofrece varias posibilidades al usuario:

- Si pulsa el botón play/pause:
  - Si el vídeo está parado: videoView.isPlaying()==false

    Lo reproduce una vez: videoView.start()
  - Si el vídeo se está reproduciendo: videoView.isPlaying==true

    Pausa el vídeo: videoView.pause()
- Si el usuario pulsa Stop: detiene el ejercicio videoView.stopPlayback()
- Si presiona “Comenzar”:
  - Comienza a correr el objeto cronómetro de la clase Chronometer.
  - Comienza la reproducción del ejercicio: videoView.start(). Cuando el sistema detecte la finalización del vídeo, gracias al escuchador OnCompletionListener(), comprobará si se ha reproducido ya tantas veces como indique el campo “repeticiones” del ejercicios. Si es así, detiene las reproducciones, pero si no, comienza una nueva reproducción automática del ejercicio:

```
1    if (contador == num_rep){
2        mVideoView.stopPlayback();
3    }
4    else {
5        contador++;
6        mVideoView.start();
7    }
```

- Si toca “Terminado”:

- Detiene el objeto cronómetro: cronometro.stop().
- Detiene la reproducción del vídeo: videoView.stop().
- Obtiene la fecha y hora actual del sistema:

```
1    DateFormat formatodate= new SimpleDateFormat("yyyy/MM/dd");
2    String date= formatodate.format(new Date());
3    DateFormat formatotime= new SimpleDateFormat("HH:mm:ss");
4    String time= formatotime.format(new Date());
```

- Almacena en la tabla ejercicio_visualizados la fecha, hora y visualización.

- Por último, si pulsa “Siguiente”:

  El sistema llamará de nuevo a la actividad ReproductorEjercicios, pero del ejercicio siguiente de la lista.

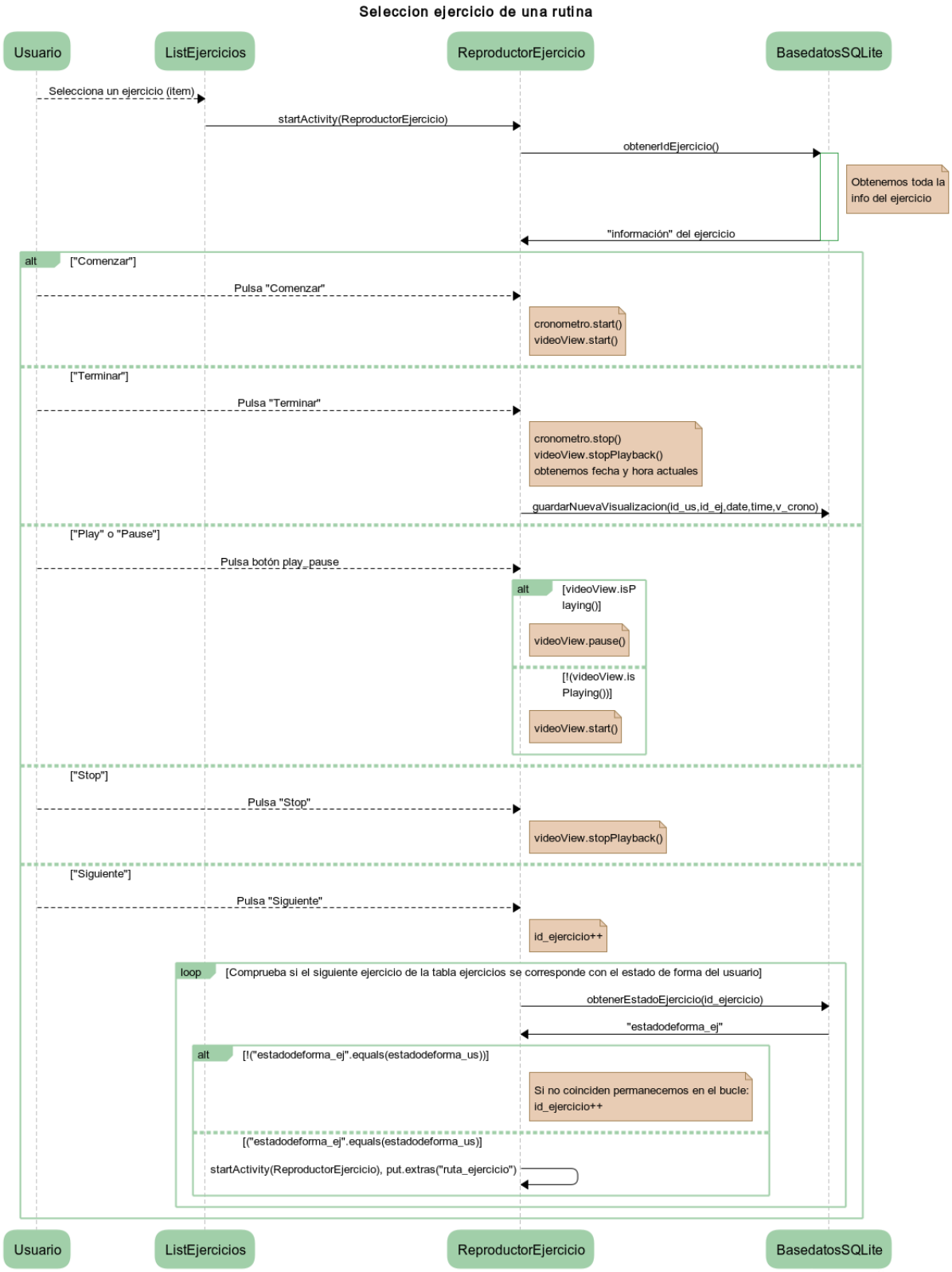

Figura 4.10 Diagrama de secuencia del reproductor de ejercicios.

4.3 Diagrama de clases

El diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando sus clases, atributos, métodos y las relaciones entre los objetos. En la Figura 4.11 mostramos el diagrama de clases de nuestra aplicación. Para una mejor visibilidad, sólo hemos añadido las clases principales.

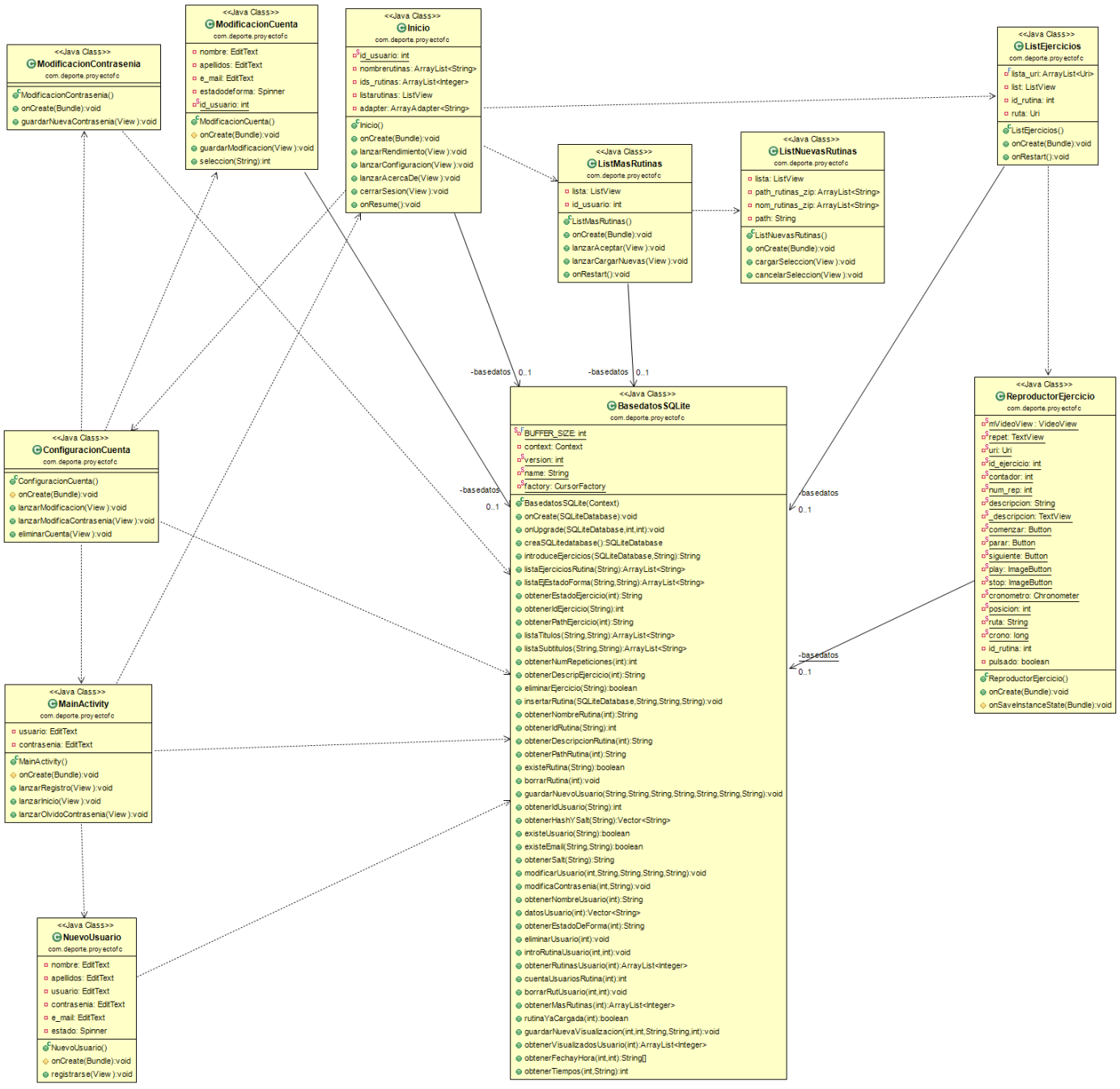

Figura 4.11 Diagrama de secuencia del reproductor de ejercicios.

Podemos ver cómo la clase BasedatosSQLite es la que dispone de más métodos, pues es la que se encarga de toda las gestiones y mantenimiento que se realiza en la base de datos. También observamos que prácticamente todas las clases están relacionadas con ella, debido a que en todas las clases deberemos de realizar algún tipo de operación sobre la base de datos, ya sea lectura o escritura.

5 Aplicación de entrenamiento. Interfaz de usuario y funcionalidad.

En este capítulo veremos el diseño de la interfaz de usuario de nuestra aplicación, de qué elementos consta dicha interfaz y de qué manera están distribuidos los distintos elementos que la componen, además de especificar qué funcionalidad tiene cada uno de ellos.

5.1 Introducción

Vamos a comentar de nuevo las funcionalidades de las que va a disponer nuestra aplicación cuando es iniciada.

- Autentificación del usuario gracias a la dupla usuario-contraseña.
- Registro de un nuevo usuario.
- Generación de nueva contraseña aleatoria de 8 dígitos y su envío mediante e-mail para gestionar el olvido de contraseña.

Una vez el usuario ha accedido correctamente con su usuario y contraseña, dispondrá de estas funcionalidades:

- Modificación de la información personal del usuario.
- Modificación de la contraseña del usuario.

- Eliminar la cuenta de usuario.
- Acceso al rendimiento semanal del usuario mediante una gráfica y una lista detallada.
- Acceso a la lista de rutinas disponibles del usuario.
- Acceso a la lista de ejercicios de la rutina seleccionada, pero solo a aquellos ejercicios correspondientes con el nivel de forma física que el usuario ha indicado en su registro.
- Acceso a las rutinas que otro usuario ha cargado en el dispositivo con anterioridad.
- Carga de nuevas rutinas que pudieran existir en la carpeta Download del dispositivo.

Una vez ha sido seleccionado un ejercicio de una rutina de su lista, el usuario podrá:

- Reproducir y pausar el vídeo del ejercicio.
- Parar el vídeo.
- Pasar al siguiente vídeo de la rutina.
- Pulsar el botón “Comenzar”. Con este botón, el sistema entiende que el usuario ha decidido realizar el ejercicio e inicia una serie de funciones:
  - Inicia un cronómetro para contabilizar el tiempo que el usuario tarda en realizar este ejercicio.
  - Reproduce el vídeo tantas veces como sea necesario, para que haya tantas repeticiones del ejercicio como las que debe realizar el usuario. De esta forma se consigue cierta interactividad entre la aplicación y el usuario.
- Pulsar el botón “Terminado”. Este botón no realiza acción alguna si el usuario no ha pulsado anteriormente el botón “Comenzar”, pero si lo ha pulsado, el sistema interpreta que el usuario ha terminado de realizar todas las repeticiones que se le pide del ejercicio, el sistema:
  - Detiene el cronómetro y almacena este tiempo para su utilización en la información del rendimiento semanal.
  - Si el usuario termina antes de que lo hagan las repeticiones de reproducción del vídeo, detiene también su reproducción.

- Ver el cronómetro y el número de repeticiones que se recomienda hacer del ejercicio.
- Ver información detallada del ejercicio.

5.2 Pantalla principal

La pantalla principal consta de los siguientes elementos o interfaces principales para la correcta comunicación entre la aplicación y el usuario (Figura 5.1).

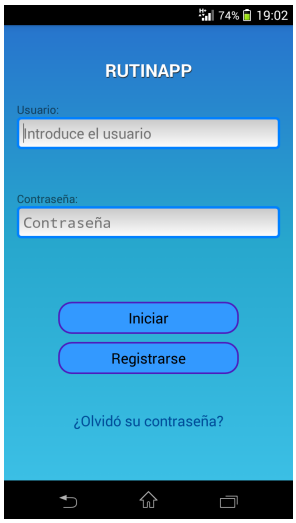

Figura 5.1 Pantalla principal.

Título : Situado en la parte superior de la pantalla, es necesario para identificar el Software y contendrá por tanto el nombre de la aplicación.

Zona de identificación : Posicionada debajo del título, nos permite introducir los datos para acceder a la funcionalidad del programa.

Zona de botones : Colocada justo debajo de la zona de identificación, consta de dos botones, el de “Iniciar”, y el de “Registrarse”.

En esta pantalla, el usuario podrá realizar las siguientes acciones:

1. Introducir sus datos para autentificarse.
2. Autentificarse.
3. Registrarse.
4. Indicar que ha olvidado su contraseña.

5.2.1 Botón de Registrarse

Si pulsamos el botón “Registrarse”, pasaremos a una nueva actividad donde podremos introducir los datos de información de usuario para realizar el registro en el sistema, como se muestra en la Figura 5.2.

El usuario podrá rellenar estos campos con los valores que desee, excepto la respuesta a la pregunta “¿Cuál es su estado de forma?” que deberá elegir una de las tres opciones. El sistema comprobará que no exista otro usuario con el mismo valor de campo usuario, en cuyo caso, notificaremos al usuario para que elija otro valor para este campo.

Figura 5.2 Pantalla de registro de un usuario.

Una vez el usuario haya introducido todos los valores, pulsará el botón “Registrarse”, y se comprobará que todos los campos hayan sido rellenados. Tras dicha comprobación, en caso de que alguno/s estén en blanco, se informará al usuario.

Figura 5.3 Aviso de campos sin rellenar.

Si por el contrario, se han rellenado correctamente, el usuario quedará registrado.

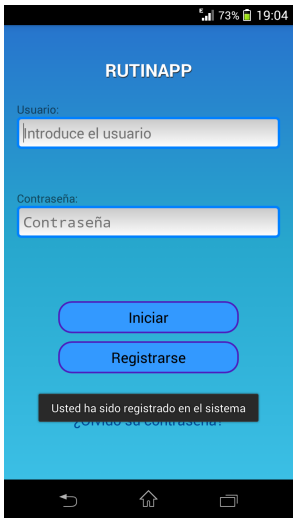

Figura 5.4 Aviso de que se ha registrado correctamente.

5.2.2 Botón Iniciar

Cuando se desea acceder a la aplicación, será necesario que el usuario se haya registrado antes en el sistema, e introducir correctamente la dupla usuario-contraseña. Después de introducir los parámetros y pulsar Iniciar, comenzará la pantalla Inicio que veremos más delante.

Si por el contrario, falta algún campo por rellenar, o el usuario no existe, o se ha introducido una contraseña incorrecta, se mostrará un aviso al usuario:

Figura 5.5 Aviso de contraseña incorrecta.

5.2.3 Olvido de contraseña

En el caso en que el usuario haya olvidado la contraseña que introdujo cuando se registró, podrá solicitar una nueva usando esta funcionalidad. Al pulsar sobre "¿Olvidó su contraseña?" se mostrará al usuario una nueva interfaz (Figura 5.6).

Figura 5.6 Pantalla de gestión del olvido de contraseña.

En ella, el usuario deberá introducir su usuario y e-mail con los que está registrado. Al pulsar "Enviar", si el e-mail introducido no se corresponde con el e-mail almacenado de este usuario, se informará del error (Figura 5.7).

Figura 5.7 Error: el e-mail introducido no es con el que el usuario está registrado.

Si por el contrario usuario y e-mail se corresponden, el usuario recibirá en su correo una nueva contraseña de ocho caracteres generada de forma aleatoria por la aplicación. Este proceso, como requiere de cierto tiempo para su realización, se ejecutará en un hilo secundario, mostrándole en el principal al usuario un aviso de espera para que no piense que el sistema no está respondiendo (Figura 5.8).

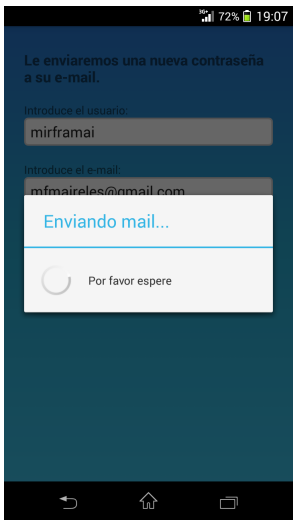

Figura 5.8 Mensaje de espera mientras el sistema realiza todas las acciones.

5.3 Pantalla Inicio

Esta actividad aparece una vez el usuario ha iniciado sesión correctamente, y consta de los siguientes componentes, que podemos ver en la imagen:

Figura 5.9 Pantalla inicial de una sesión.

Título : Situado en la parte superior de la pantalla, muestra de nuevo el título de la aplicación.

Saludo : Aparece debajo del título, nos muestra una frase de bienvenida personalizada con el nombre del usuario que ha iniciado sesión.

Lista de rutinas : Se encuentra inmediatamente después del saludo, y muestra al usuario una lista de las rutinas que él ha elegido almacenar. Su último elemento es el botón "cargar más", que le da la opción de acceder a nuevas rutinas. O bien rutinas que haya cargado antes otro usuario, o bien nuevas rutinas que haya descargado y se encuentren en la carpeta Download.

Zona de información : Consta de cuatro botones situados en la parte inferior de la pantalla, y que permiten el acceso a información del usuario y de la aplicación:

- El botón "Mi rendimiento".
- Configuración.
- Acerca De...
- Cerrar Sesión.

5.3.1 Mi rendimiento

Si el usuario pulsa sobre “Mi rendimiento”, se mostrará una nueva pantalla (Figura 5.10).

Zona gráfica : En la parte superior se sitúa una gráfica en la que el usuario puede ver su rendimiento semanal de una manera vistosa e intuitiva. En el eje de abscisas se representan los días de la semana: L, M, X, J, V, S, D y en el eje de ordenadas el tiempo indicado en minutos que el usuario ha dedicado cada día a entrenarse.

Rendimiento detallado : En esta zona se incluye la información que se recoge de la gráfica pero por escrito.

Figura 5.10 Pantalla GraficaRendimiento.

5.3.2 Configuración

En caso de que el usuario presione el botón “Configuración”, se presentará una nueva interfaz en la que podrá acceder a varias tareas de configuración de su cuenta:

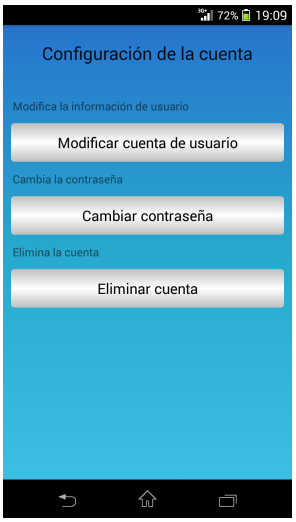

Figura 5.11 Pantalla ConfiguracionCuenta.

- Modificar la cuenta.

  Si el usuario lo pulsa, se muestra una pantalla en la que todos los campos están rellenos con los datos del usuario que actualmente dispone el sistema. Una vez el usuario haya introducido los cambios pertinentes y pulse "Guardar cambios", se mostrará un mensaje informando de que la operación ha sido realizada. En la figura 5.12 hemos cambiado el estado de forma de Bajo a Alto.

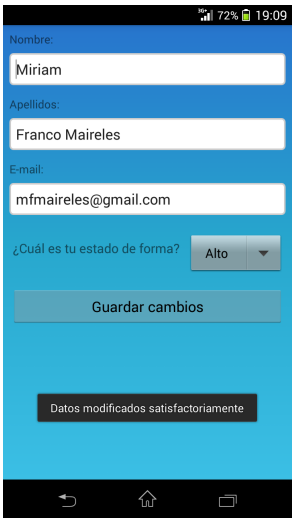

Figura 5.12 Pantalla ModificaciónCuenta.

- Cambiar la contraseña.

  Si presiona este botón, se muestra una pequeña pantalla (Figura 5.13) en la que el usuario deberá introducir la nueva contraseña dos veces. Se hace así para evitar despistes por parte del usuario y que no guarde una contraseña cuando creía que había introducido otra.

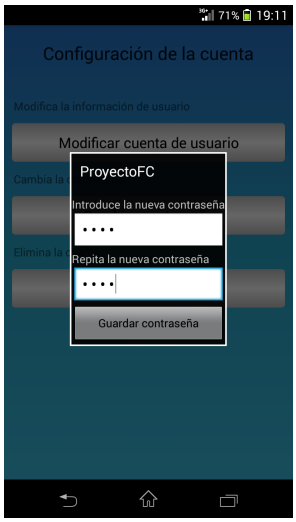

Figura 5.13 Pantalla de modificación de la contraseña.

Si tras pulsar Aceptar las contraseñas no coinciden, se muestra un aviso de error al usuario. Si coinciden, se informa al usuario de que la contraseña ha sido modificada satisfactoriamente (Figura 5.14).

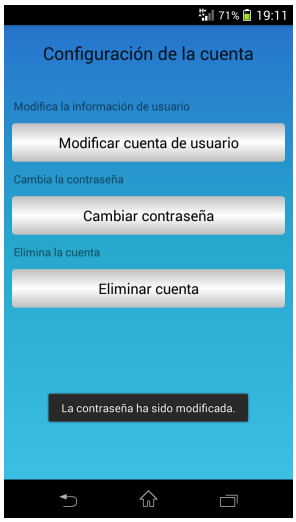

Figura 5.14 La contraseña ha sido modificada.

- Eliminar la cuenta.

  Si pulsa este botón, la cuenta será eliminada del sistema.

5.3.3 Acerca de

Cuando el usuario pulsa sobre el botón “Acerca De”, se muestra una pantalla con información de la aplicación que es de interés al usuario.

Figura 5.15 Pantalla AcercaDe.

5.3.4 Cerrar sesión

Cuando el usuario pulsa sobre “Cerrar sesión”, se saldrá del servicio automáticamente y volverá a la pantalla principal de la aplicación. Pulsando dicho botón será la única forma de volver a la pantalla principal ya que se ha bloqueado el funcionamiento del botón “atrás” del dispositivo en la actividad Inicio.

5.4 Eliminación de una rutina de la lista

Si el usuario presiona de forma prolongada sobre una de las rutinas de su lista, se le mostrará una pequeña ventana, preguntando si desea eliminar la rutina de su lista.

Si indica “Aceptar”, la rutina se eliminará de su lista. Si pulsa “Cancelar”, se volverá a la pantalla de Inicio sin realizarse cambio alguno.

Figura 5.16 Mensaje de confirmación de eliminación de una rutina.

5.5 Cargar más rutinas

En el caso en que el usuario pulse sobre el botón “Cargar más”, aparecerá una actividad (Figura 5.17) en la que podrá verse un listado de las rutinas que no posee el usuario en su lista, pero que han sido descomprimidas por otros usuarios de la aplicación en este dispositivo y que se encuentran en la carpeta /sdcard/app_-deportes/rutinas/ de la memoria del dispositivo. Si selecciona una y pulsa aceptar, se incluirá ya en su lista de rutinas.

Figura 5.17 Listado de rutinas disponibles de otros usuarios.

Si no existe ninguna rutina que no disponga ya el usuario, se mostrará en lugar de la lista, el mensaje: “NO HAY RUTINAS DISPONIBLES DE OTROS USUARIOS”.

5.5.1 Cargar nuevas rutinas

En caso de pulsar este botón, se transportará al usuario a una nueva pantalla en la que aparecerá una lista de nombres de rutinas (Figura 5.18). Los componentes de esta nueva lista se tratará de archivos de extensión .zip que se encuentran en la carpeta Download del dispositivo del usuario, y que no han sido ya descomprimidas en la carpeta /sdcard/app_deportes/rutinas/.

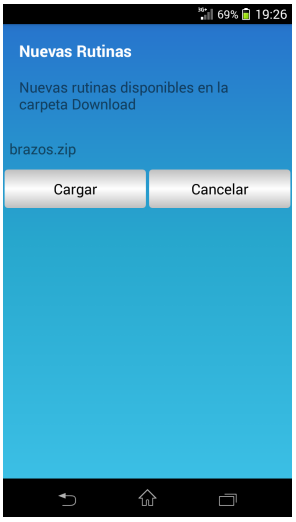

Figura 5.18 Rutinas disponibles en la carpeta del dispositivo.

Si selecciona un elemento y pulsa “Cargar”, realizará la tarea de descompresión del archivo y posterior inclusión en su lista de rutinas en un hilo secundario. El principal se encargará de mostrarle al usuario un aviso de que debe esperar.

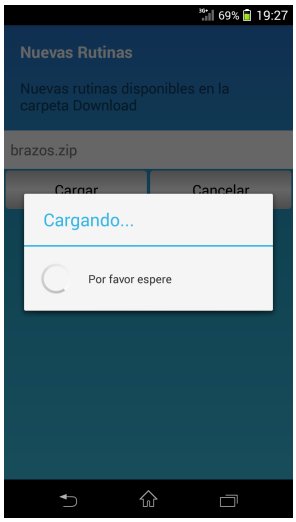

Figura 5.19 Mensaje de espera mientras se realiza la descompresión.

5.6 Acceso a los ejercicios de una rutina

Cuando el usuario pulse sobre una de las rutinas de su lista, se le mostrará una interfaz (Figura 5.20) con los siguientes componentes:

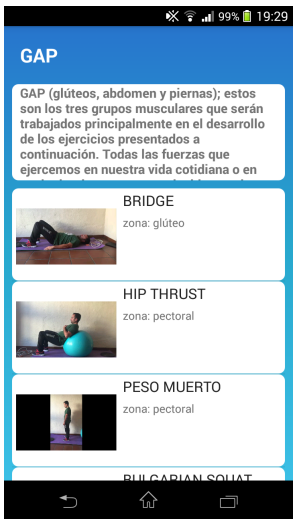

Figura 5.20 Listado de ejercicios de una rutina, según el estado de forma del usuario.

Nombre de la rutina : Situado en la parte superior izquierda, muestra el nombre de la rutina que se ha seleccionado en la anterior pantalla.

Descripción de la rutina : Situada debajo del nombre, nos describe de forma concisa y breve cuáles son los objetivos que se consiguen con esta rutina.

Lista de ejercicios : Muestra todos los ejercicios de la rutina diseñados para el nivel de forma física que posee el usuario.

Cada uno de los elementos ejercicio que contiene la lista contiene:

- Imagen de un instante del vídeo del ejercicio, para que el usuario pueda hacerse una idea de en qué consistirá.
- Nombre del ejercicio en cuestión.
- En caso de que el ejercicio ya lo haya realizado el usuario, se indicará con la palabra “HECHO” en color rojo, y al lado, la fecha y hora en que lo realizó.
- Breve descripción del ejercicio de a lo sumo dos líneas. En nuestra rutina GAP la usamos para mostrar la zona general que se ejercitará.

5.7 Acceso a un ejercicio

Si el usuario pulsa sobre uno de los ejercicios de la rutina, el sistema mostrará una nueva interfaz al usuario con estos elementos (Figura 5.21):

Figura 5.21 Pantalla de reproducción de un ejercicio.

Reproductor de vídeo : Situado en la parte superior de la pantalla. Es el encargado de visualizar la reproducción del ejercicio.

Controladores de reproducción : Botones situados justo debajo del reproductor, en la parte central de la pantalla. Presionándolos, el usuario puede realizar estas acciones.

- Play/Pause: Reproduce el vídeo una vez para que el usuario pueda ver en qué consiste. Si está reproduciéndose y lo pulsa otra vez, el vídeo se pondrá en pausa, y si pulsa de nuevo, continuará la reproducción por donde lo había dejado.

- Stop: Detiene la reproducción del vídeo. Si desea volver a reproducirlo deberá pulsar "play", en cuyo caso comenzará desde el principio.

- Siguiente: Muestra la pantalla de reproducción del siguiente vídeo disponible en la lista de la rutina.

- Comenzar: Si el usuario pulsa "Comenzar", estará indicando que va a comenzar a realizar el ejercicio. A partir de este momento el sistema contabilizará el tiempo que transcurre hasta que pulsa el botón "Terminado" y lo tomará como el tiempo que ha tardado el usuario en hacer el ejercicio. El ejercicio se reproducirá un número de veces seguidas, tantas como se necesite para que se visualicen las repeticiones que el usuario debe realizar del ejercicio.

- Terminado: Si el usuario presiona "Terminado", después de haber pulsado "Comenzar" anteriormente, estará indicando que ya ha realizado todas las repeticiones necesarias de este ejercicio. El sistema parará el cronómetro y almacenará el tiempo que ha trascurrido desde que se pulsó "Comenzar". Si el usuario tarda en realizar las repeticiones menos que el vídeo, o simplemente el usuario pulsa "Terminado" antes de que el reproductor realice todas las repeticiones asignadas, se parará tanto el cronómetro como las reproducciones, y se almacenará el tiempo del crono.

6 Conclusiones y líneas futuras del proyecto

6.1 Líneas de continuación del proyecto

Comencemos recordando resumidamente todo el ámbito de nuestro trabajo. Una vez el usuario ha iniciado sesión, accede a su lista de rutinas, creada por él mismo. Como dijimos, nuestra aplicación sólo trae la rutina "GAP", pero la aplicación tiene la función de poder incorporar nuevas rutinas comprimidas en la carpeta Dowloads, creadas por él mismo u otros usuarios, siempre que se mantenga la estructura que debe de tener una carpeta (archivo JSON correctamente estructurado y vídeos mp4) o bien rutinas que ya han descomprimido e incorporado a su lista otros usuarios locales de la aplicación. De esta forma, el usuario ingresa en su lista las rutinas que le interesen.

Cada rutina, incorpora vídeos según los tres niveles de forma que permite la aplicación y cada vídeo incluye una descripción detallada señalando aspectos como en qué consiste el ejercicio, los materiales, los aspectos a destacar o los músculos implicados.

El usuario también dispone de la opción de modificar sus propios datos personales y su contraseña, y modificar de forma manual su nivel de estado de forma física, accediendo a los vídeos de sus rutinas que tengan ese nuevo nivel de forma. Además se ha añadido una gráfica personal sobre el tiempo invertido cada día de la semana al entrenamiento y ejercicio, pudiendo ver el usuario su progresión semanal.

Para complementar la funcionalidad de esta aplicación, podría plantearse la creación de una página web repositorio donde cualquier usuario pueda crear, descargar e introducir nuevas rutinas, ajustándose a la estructura de carpeta que debe de tener una nueva rutina. En esta página podría darse la opción de que se generase

automáticamente el archivo JSON, con los datos introducidos por el usuario que la está creando en un formulario. Se debe dar la opción de que cualquier usuario pueda descargarse las rutinas creadas por cualquier otro usuario, e incluso la opción de puntuarlas, recomendarlas o realizar comentarios sobre ella. Podría considerarse la opción de sacar provecho económico, ya sea pagando por algunas rutinas o incorporando función publicitaria.

Con respecto a la aplicación, podríamos crear la opción de sincronizar varios dispositivos a la vez, y visualizar al mismo tiempo en cada uno de ellos las mismas rutinas, por ejemplo para la práctica de ejercicio en grupo con amigos o familiares.

Podríamos incorporar más funcionalidades para hacerla más completa. Se podría incluir más posibilidades de estados de forma física y más concretos, en función de la edad, peso, etc. Incluir algún tipo de evaluación inicial, tanto de estado de salud previo (una anamnesis general previa), como una evaluación de forma física.

Podría incluirse una percepción subjetiva del esfuerzo, de forma que el usuario se evaluara así mismo su nivel de "cansancio" después del entrenamiento, con el fin de poder progresar al grado de intensidad más adecuado de cada sujeto. Así mismo, podría incorporarse gráficos sobre esta progresión.

Podría incorporarse nuevas funcionalidades a la aplicación para hacerla más motivante, decir quién usa la aplicación cerca de ti, un chat interno para los usuarios, compartir tus gráficas de progresión para que pueda verlas todo el mundo, e incluso podría darse la opción de incluir algún tipo de retos (en función de los objetivos, pérdida de peso, o mejora de la fuerza y la flexibilidad).

6.2 Conclusiones finales

Una vez concluidas todas las fases de desarrollo de la aplicación, podemos sacar las conclusiones de nuestro trabajo.

Hemos construido una aplicación que ofrece al usuario libertad en muchos aspectos. Libertad para escoger el momento de entrenamiento y libertad en el sentido en que ofrece al usuario la posibilidad de personalizar el contenido de su aplicación. El usuario puede escoger sólo las rutinas que le interese y además puede crear sus propias rutinas y compartirlas con otros usuarios. En este sentido, la aplicación gana en utilidad, pues puede moverse en muchos entornos, desde rutinas de fitness y ejercicios de rehabilitación a deportes más especializados como pilates, yoga, etc, el límite lo imponen los propios usuarios. En definitiva, hemos diseñado una aplicación horizontal, pues sirve para cualquier tipo de rutinas deportivas.

Me gustaría dar mi valoración de todo lo que me ha supuesto la realización de este trabajo.

En primer lugar, resaltar el gran esfuerzo que supone la realización de un proyecto fin de carrera, desde la primera idea inicial hasta que ya le vas dando forma y van surgiendo y resolviéndose los problemas que van apareciendo. Me quedo con eso también, con los errores, pues gracias a ellos ha sido como verdaderamente he aprendido, y la enorme satisfacción que supone el ser capaz de resolverlos.

He aprendido multitud de nuevos conceptos, destacando todo lo relacionado con Android, pues era algo totalmente nuevo para mí que no había visto a lo largo de mi formación en la escuela, sus librerías, su estructura, el xml y recordar Java, que llevaba varios años sin usar. También conceptos de los que ni siquiera había escuchado hablar como SQLite o JSON y que han sido protagonistas en mi proyecto.

Este proyecto me ha dado la oportunidad de ampliar mis conocimientos sobre programación y estoy segura de que todo lo aprendido podré ponerlo muy pronto en práctica a lo largo de mi vida profesional. Muchas gracias a mi tutora María Teresa Ariza Gómez por toda su ayuda, sus correcciones, sus consejos y su dedicación que tanto me han ayudado a que este proyecto haya llegado a buen puerto.

Anexo A

Seguridad en los datos sensibles de las aplicaciones

Cuando desarrollamos una aplicación de cualquier tipo y en cualquier plataforma, es muy importante tener en cuenta el manejo seguro de datos sensibles del usuario. En este anexo vamos a hablar de la seguridad a nivel de aplicación y en concreto del mecanismo hashing, muy importante en el almacenamiento de datos en el dispositivo.

A.1 Introducción

Android no es, ni mucho menos, una plataforma totalmente segura. Se ha convertido en la más atacada por parte de los cibercriminales. Por eso se debe realizar un diseño que identifique y proteja adecuadamente los datos de carácter sensible y que maneje de forma segura las credenciales del usuario.

Es preferible almacenar la información sensible en el lado del servidor, al ser los dispositivos móviles elementos susceptibles de ser robados o extraviados, pero en el caso de almacenar dicha información en el terminal, será necesaria cifrarla, y si es posible, habilitar los mecanismos necesarios para poder eliminarla remotamente, en caso de que sea necesario. Existen diversas metodologías cuya máxima es el "security by design" y que permiten evaluar la seguridad de las aplicaciones, como es el caso de OWASP u OASAM, e incluso desde Android establecen una serie de consejos a la hora de desarrollar una aplicación. Dichas metodologías tienen como objetivo identificar los peligros correspondientes a la seguridad de dispositivos móviles y proporcionar los controles necesarios en el desarrollo para reducir su

impacto y la probabilidad de explotación de los mismos.

Vamos a ver mecanismos para proteger las credenciales y datos del usuario, en el caso de que no quede más remedio que almacenarlos en el dispositivo, como es el caso de nuestra aplicación.

A.2 ¿Qué son y para qué sirven los “hash”?

Los hash o funciones de resumen son algoritmos que consiguen crear a partir de una entrada (ya sea un texto, una contraseña, un archivo, etc.) una salida alfanumérica de longitud normalmente fija que representa un resumen de toda la información que se le ha dado, es decir, a partir de los datos de la entrada crea una cadena que sólo puede volverse a crear con esos mismos datos. Tiene varios objetivos, como asegurar que no se ha modificado un archivo en una transmisión, hacer ilegible una contraseña (que es para lo que la utilizamos en nuestra aplicación), o firmar digitalmente un documento.

A.3 Características de los “hash”

Las funciones hash se encargan de representar de forma compacta un archivo o conjunto de datos que normalmente es de mayor tamaño que el hash.

Este algoritmo de criptografía usa algoritmos que asegura que con la respuesta (o hash) nunca se podrá saber cuáles han sido los datos de entrada, no son reversibles, lo que indica que es una función unidireccional y que comúnmente se les llama algoritmos de cifrado de una sola vía. Sabiendo que los resúmenes son de longitud fija, y normalmente menor que la longitud de los datos de entrada podríamos preguntarnos si podrían repetirse estos hashs. La respuesta teóricamente es que sí, ya que no es posible tener una función hash perfecta (que no repita la respuesta), pero esto no supone ningún problema ya que si se consiguieran dos hash iguales los contenidos serían totalmente distintos.

Hay muchos algoritmos de hash al alcance de los programadores y en algunos casos forman parte de las librerías de cifrado de los entornos de programación como por ejemplo Java. Algunos de estos algoritmos son MD5 (Message Digest Algorithm 5), SHA-1 (Secure Hashing Algorithm 1), Whirlpool, etc. Actualmente se están imponiendo nuevas funciones como SHA-2 Y Bcrypt.

A.4 Ejemplos y formas de uso

A.4.1 Protección de contraseñas

Las funciones hash son muy utilizadas, una de las utilidades que tiene es la de proteger la confidencialidad de una contraseña, ya que si estuviera en texto plano podría ser accesible por cualquiera y aun así no podrían ser capaces de resolverla.

Para saber si una contraseña que está guardada en una base de datos, es igual a la que hemos introducido, no se descifra el hash (ya que es imposible), sino que se aplicará la misma función hashing de resumen a la contraseña que hemos introducido, y se comparará con el resultado que teníamos guardado.

Sin embargo hay que tener en cuenta que no necesariamente el usuario está seguro usando alguno de estos algoritmos de hash, ya que también existe lo que se denominan "Ataques por Diccionario" (Dictionary Attack), que consiste en utilizar bases de datos con los hash resultantes de cifrar los tipos de claves de acceso más comunes, incluso puede conseguirse este tipo de bases de datos en línea. Entonces, ¿cuál es la solución para proteger las contraseñas de los usuarios de este tipo de ataques? En estos diccionarios se encuentran claves deficientes o pobres, por lo que el problema se reduciría si los usuarios escogieran claves menos evidentes. Lo mejor para evitar este tipo de ataques es lo que se conoce como "Salt key", que no es más que una cadena de texto que se agrega al texto original antes de ser cifrado. De esta forma el atacante no puede comparar los hash ya que necesitaría un nuevo diccionario, es más, incluso si el atacante llegara a conocer el Salt, no le serviría de nada, ya que los diccionarios existentes no tendrían las cadenas resultantes de utilizarlo. El programador sólo tiene que tener en cuenta agregarlo de igual forma cada vez que compare el hash guardado con el producido a la hora de validar la clave.

A la hora de escoger uno de los algoritmos de hashing, el programador debe de tener en cuenta la velocidad de ejecución de la función, y al contrario de lo que pueda parecer, escoger la más lenta. Al usuario sólo le supondrá unas décimas más de segundo, pero para un atacante esta pequeña diferencia puede hacer que tarde más tiempo en generar tablas rainbow, horas, meses e incluso años.

Todo este proceso puede mejorarse aún más y complicarse tanto como se desee agregando más funciones hashing, tantas iteraciones como se desee o cálculos recursivos de hash, pero deberá antes pensarse el tiempo y recursos que podría tomar un cálculo demasiado complejo en relación a la cantidad de usuarios.

A.4.2 Asegurar la integridad de la información

Otro uso que tiene esta función es la de asegurar la integridad de los datos, por ejemplo en páginas webs que proporcionan descargas de archivos grandes, dan también el resumen del archivo y la función usada. Esto es un método para saber si un documento está íntegro tras su recepción.

A.4.3 Firma digital

La firma digital nos ayuda a verificar la identidad del emisor de un mensaje. El método más simple de firma digital es crear un hash de la información enviada y cifrarlo con nuestra clave privada (de nuestro par de claves de la criptografía asimétrica) para que cualquiera con nuestra clave pública pueda ver el hash real y verificar que el contenido del archivo es el que hemos enviado nosotros.

A.5 Uso en nuestra aplicación

Nuestra aplicación guarda el password de forma muy segura ya que no sólo realiza el hash, sino que además hemos añadido una “Salt key” de 16 bytes, generado aleatoriamente para cada nuevo usuario registrado con una función SecureRandom. En nuestra base de datos hemos almacenado este salt, y también el resultado de realizar el hash a la concatenación de contraseña + salt con el algoritmo SHA-1.

Tal vez el uso de estos métodos de encriptado esté más justificado en aplicaciones que “salen” a internet. En una ejecución normal de nuestra aplicación, no accederá a internet en ningún momento, sin embargo, si hacemos uso de la función de olvido de contraseña, sí que lo hará. Además, no hay que olvidar que SQLite guarda los datos en un único archivo y en texto plano, por lo que será conveniente para casos como extravíos o robos de los dispositivos móviles o tabletas, que evitará que puedan acceder a nuestros datos personales.

Anexo B

Manual de instalación

En este anexo incorporaremos un manual de instalación de todas las herramientas que han sido utilizadas para la realización de este proyecto final de carrera. A continuación presentaremos las herramientas que hemos usado:

- Eclipse Mars
- Java (JDK)
- Android (SDK)
- Sony PC Companion 2.1 (dependiendo del móvil, en mi caso un Sony Xperia SP)

B.1 Instalación de Eclipse

Para la instalación de Eclipse, escribimos en el navegador la siguiente dirección web: *http://www.eclipse.org/downloads/*, y seleccionamos la versión compatible con nuestro sistema operativo, ya sea de 32 bits o 64 bits.

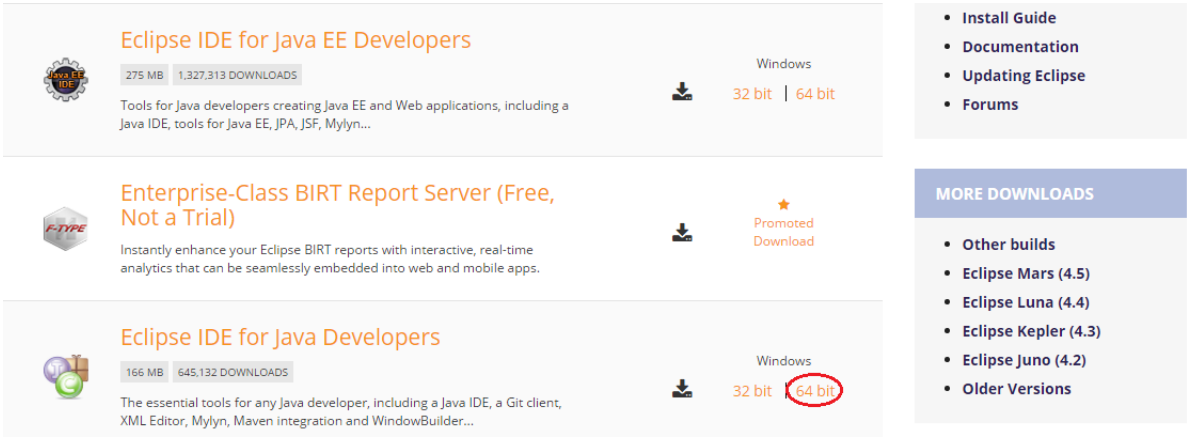

Figura B.1 Selección de la versión correcta de Eclipse.

En nuestro caso, seleccionamos la opción de 64 bits y esperamos que nos dejen descargar desde el enlace de descarga disponible.

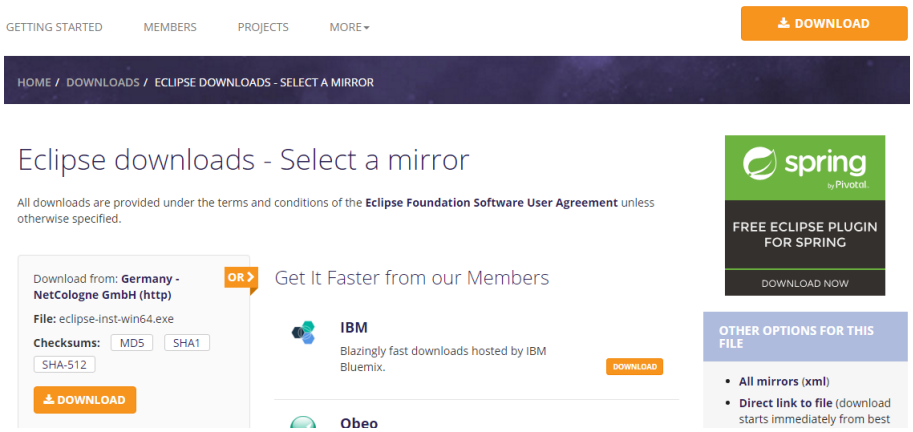

Figura B.2 Selección de la descarga.

Pulsamos sobre “Donwload” y esperamos que comience la descarga.

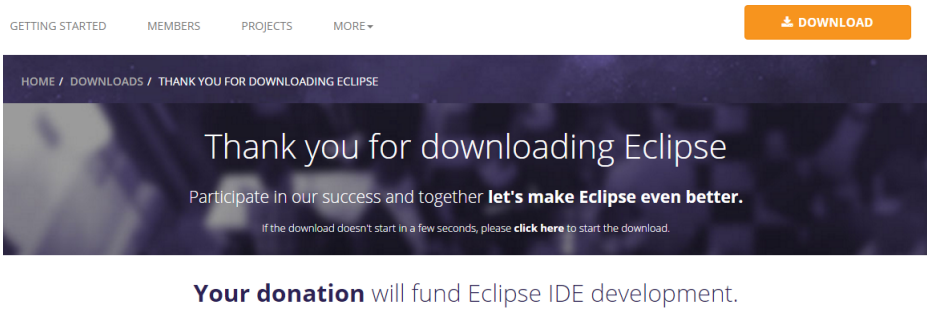

Figura B.3 Inicio de la descarga Eclipse.

Una vez descargado el archivo .zip lo descomprimimos.

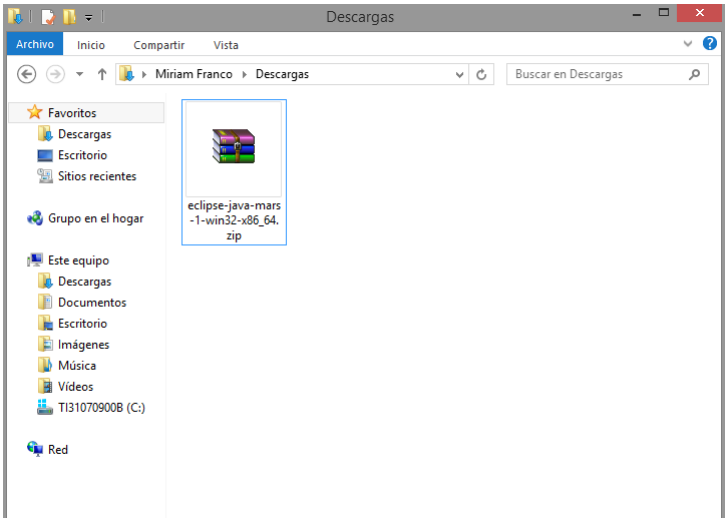

Figura B.4 Finalización de la descarga de Eclipse.

Una vez esté descomprimido en el directorio C:/, en mi caso, encontramos en el interior todos los archivos necesarios para la ejecución del programa.

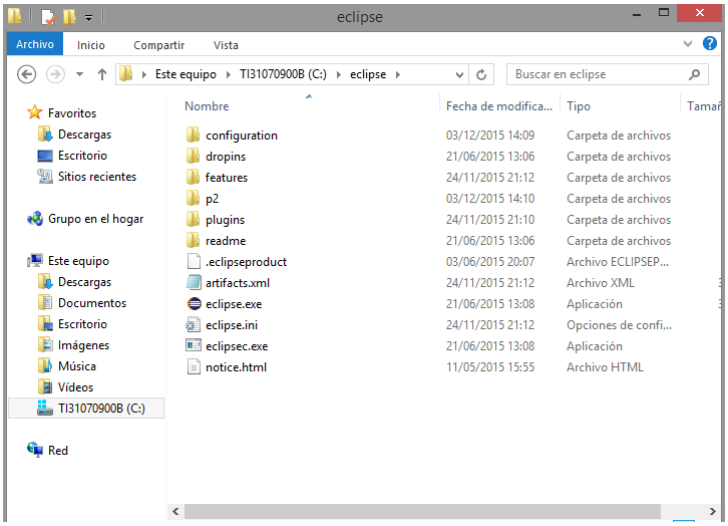

Figura B.5 Directorio de Eclipse.

Para más comodidad, podemos crear un acceso directo del ejecutable pulsando botón derecho sobre la aplicación “eclipse”->”Enviar a”->”Escritorio (crear un acceso directo)”.

Ahora, ejecutamos el programa. Si no tenemos instalado el JDK, nos aparecerá el siguiente error.

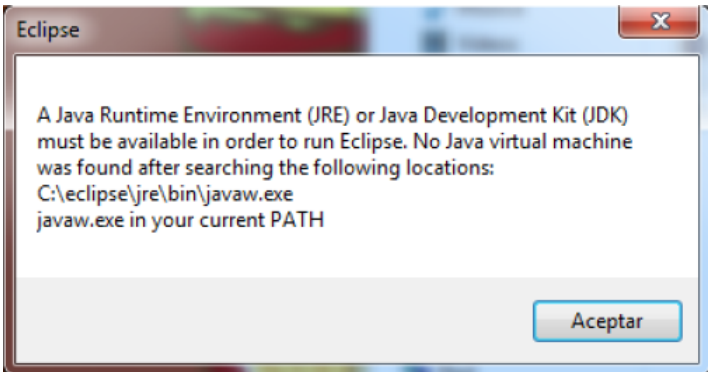

Figura B.6 Error en Eclipse al no tener el JDK instalado.

Este error lo solucionaremos en el siguiente punto.

B.2 Instalación de JDK de Java

En caso de no tener instalado el JDK, bastaría con ir a la siguiente dirección web: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Figura B.7 Instalación del JDK.

Seleccionamos “Download” sobre la opción de JDK.

Java SE Development Kit 8u45

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	146.89 MB	jdk-8u45-linux-i586.rpm
Linux x86	166.88 MB	jdk-8u45-linux-i586.tar.gz
Linux x64	145.19 MB	jdk-8u45-linux-x64.rpm
Linux x64	165.24 MB	jdk-8u45-linux-x64.tar.gz
Mac OS X x64	221.98 MB	jdk-8u45-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	131.73 MB	jdk-8u45-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	92.9 MB	jdk-8u45-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	139.51 MB	jdk-8u45-solaris-x64.tar.Z
Solaris x64	95.88 MB	jdk-8u45-solaris-x64.tar.gz
Windows x86	175.98 MB	jdk-8u45-windows-i586.exe
Windows x64	180.44 MB	jdk-8u45-windows-x64.exe

Back to top

Figura B.8 Selección de la versión del JDK.

Aceptamos el acuerdo de licencia pulsando sobre “Accept License Agreement” y seleccionamos la versión apropiada en función de la versión del sistema operativo que tengamos instalado. En mi caso, selecciono la versión de 64 bits.

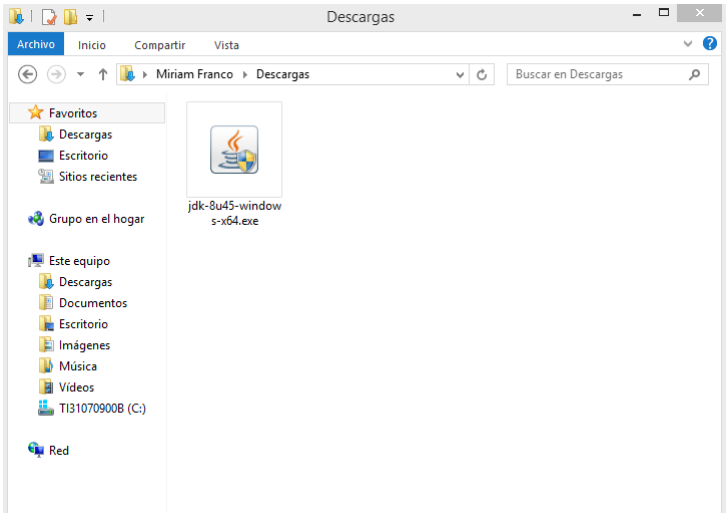

Figura B.9 JDK descargado.

Ejecutamos el fichero descargado y comenzamos la instalación.

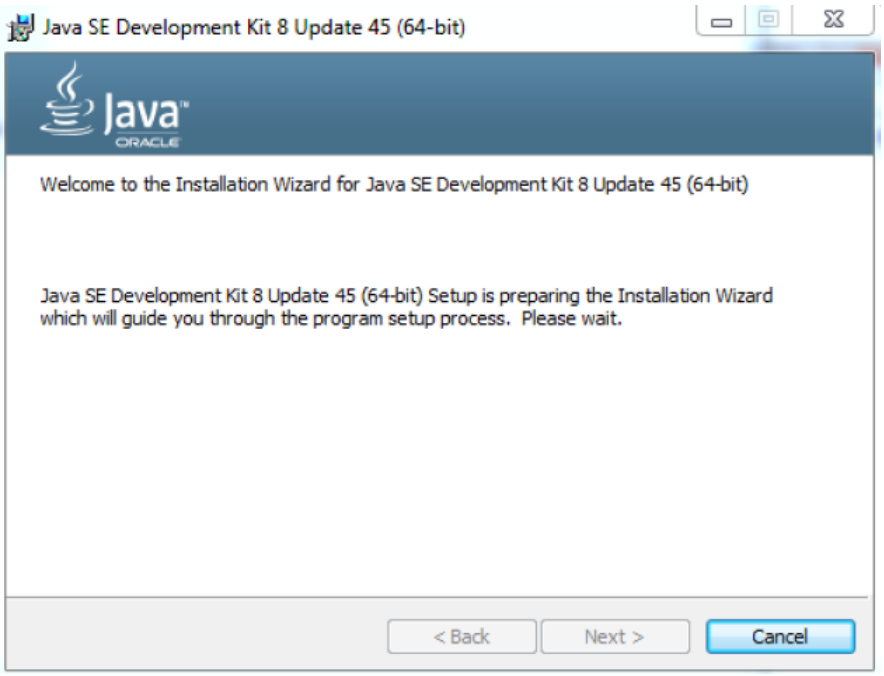

Figura B.10 Asistente de instalación.

Pulsamos "Next".

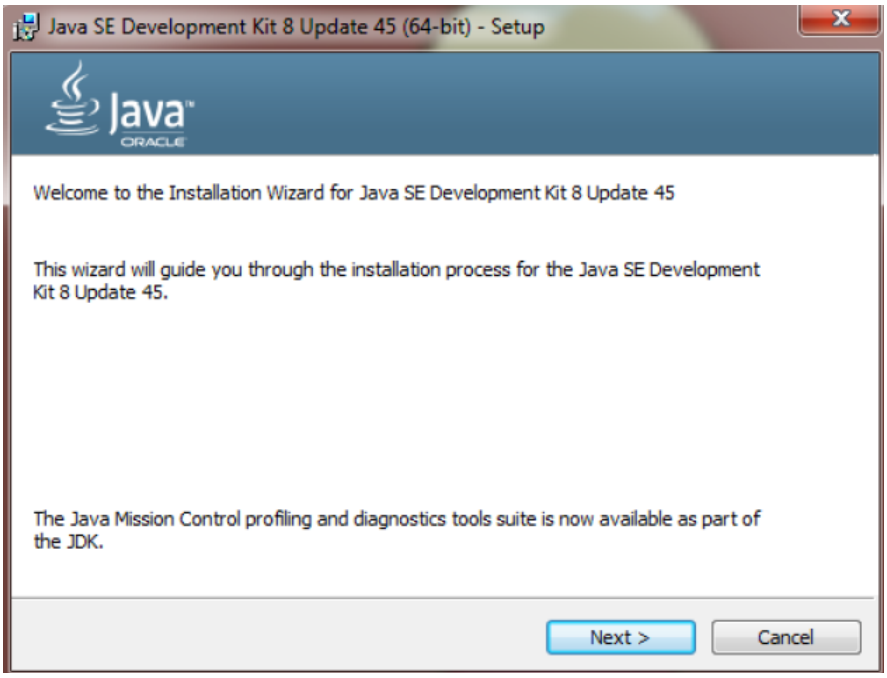

Figura B.11 Asistente de instalación. Paso 1.

Pulsamos "Next"

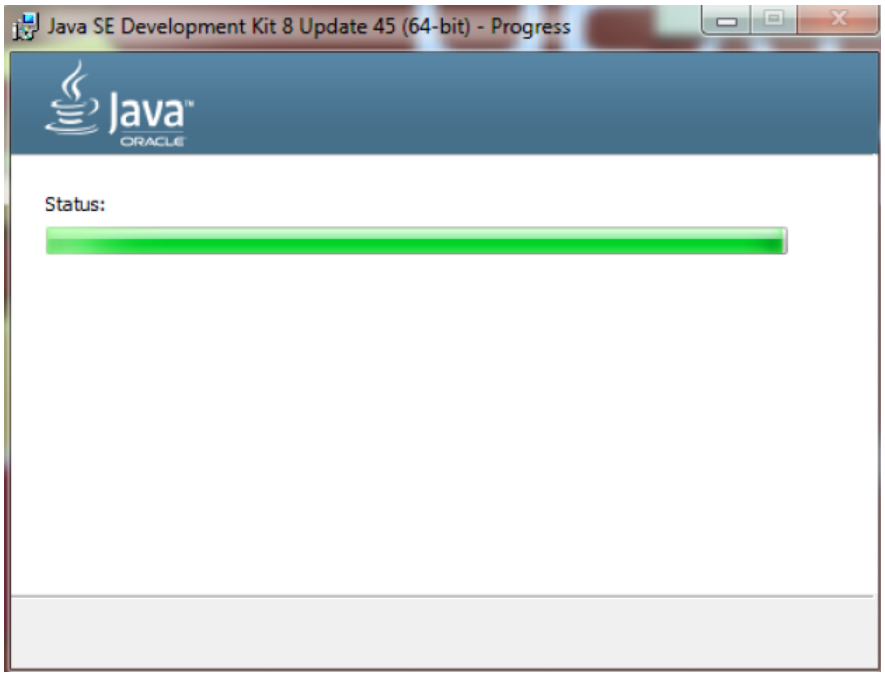

Figura B.12 Asistente de instalación. Paso 2.

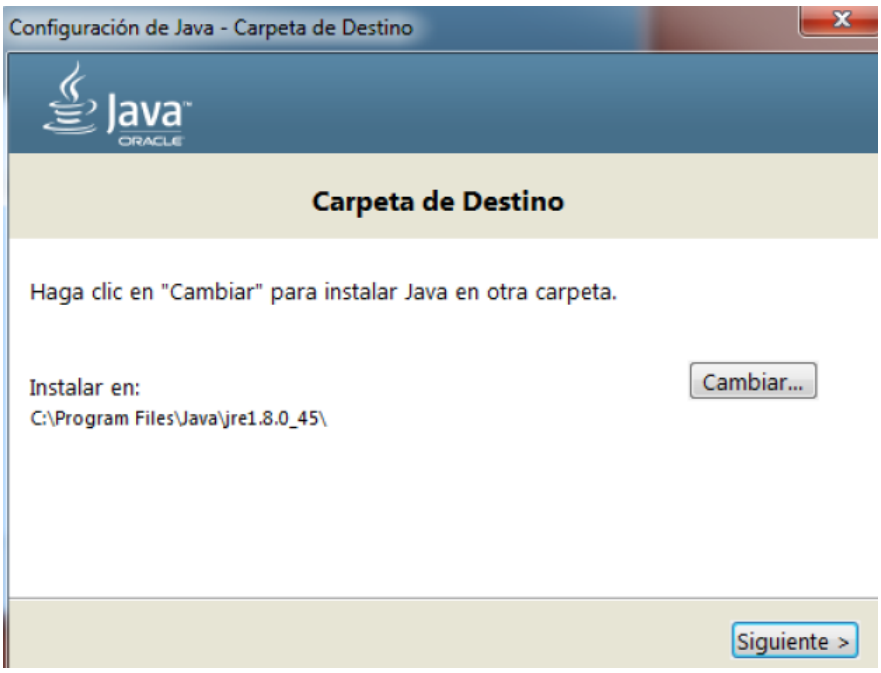

Figura B.13 Asistente de instalación. Paso 3.

Elegimos el directorio de instalación, pulsamos “Siguiente” y la instalación comenzará.

Figura B.14 Asistente de instalación. Paso 4.

Esperamos a que la instalación termine.

B.3 Instalación del Android SDK

Si queremos desarrollar una aplicación Android, nos hará falta una herramienta de desarrollo para realizar esta tarea. Ésta es el Android SDK. Para su descarga, iremos a la siguiente dirección web: https://developer.android.com/sdk/installing/index.html

En este caso, nos descargaremos la versión Stand-Alone, que posteriormente necesitaremos vincular con Eclipse para poder usar éste como entorno de desarrollo de nuestra aplicación Android.

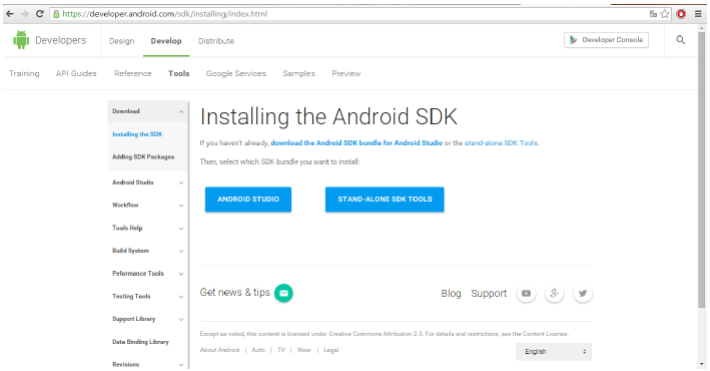

Figura B.15 Android SDK.

Tras la elección de la versión Stand-Alone, descargamos la versión compatible con nuestro sistema operativo.

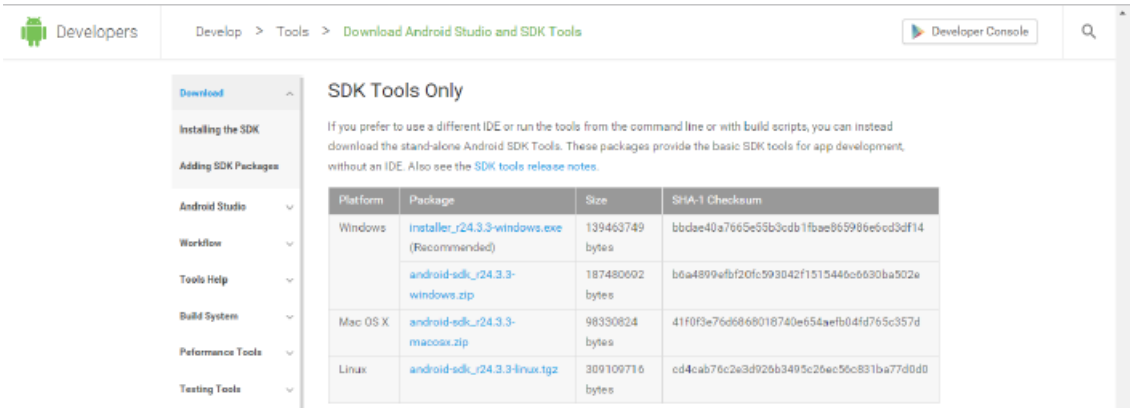

Figura B.16 Elección de la plataforma correspondiente.

En nuestro caso, elegimos la versión "installer_r24.3.3-windows.exe" para Windows. Tras dicha elección, sólo nos quedará aceptar los términos y condiciones para poder realizar la descarga.

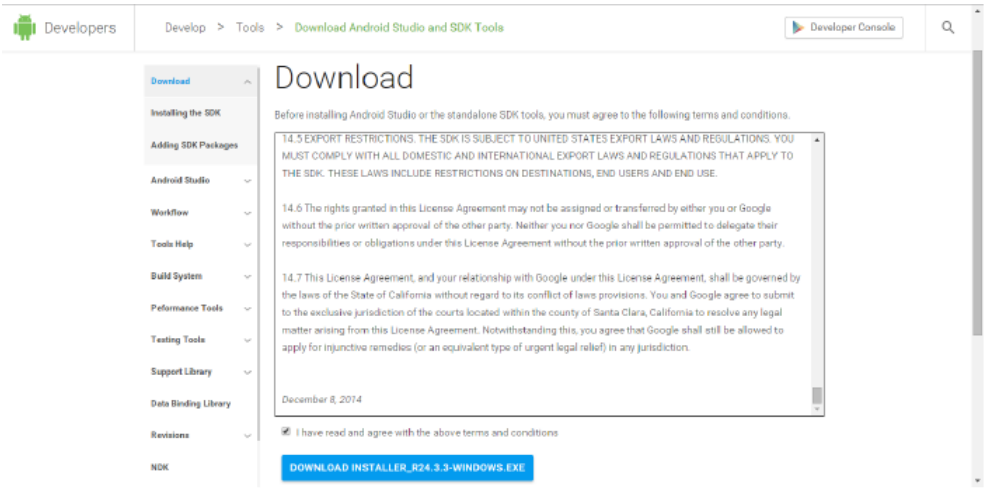

Figura B.17 Términos y condiciones de la descarga.

Una vez realizada la descarga, procedemos a la instalación, ejecutando el asistente de instalación con permisos de administrador.

Figura B.18 Asistente de instalación Android SDK. Paso 1.

Pulsamos “Next”.

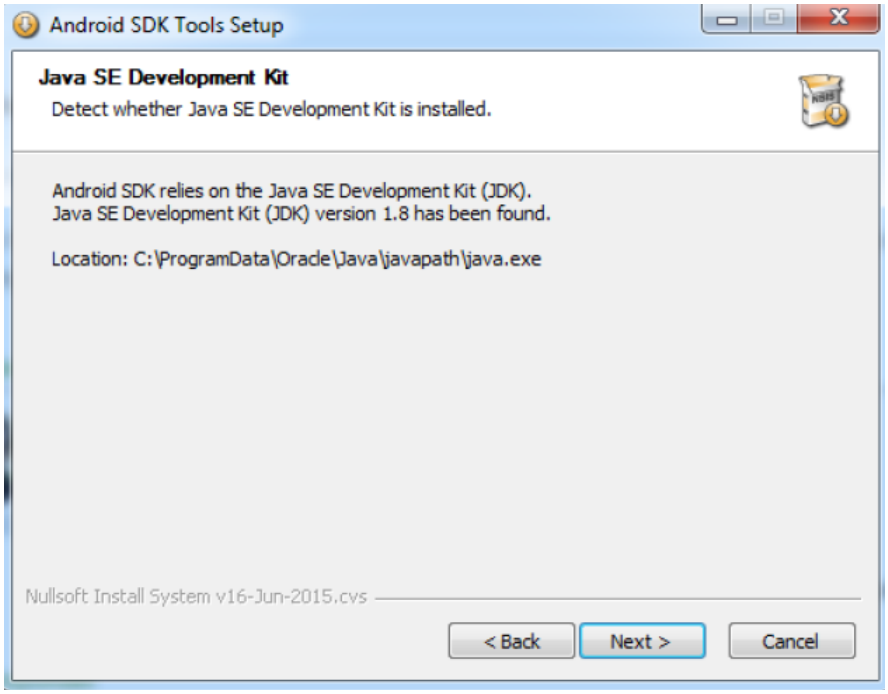

Figura B.19 Asistente de instalación Android SDK. Paso 2.

Pulsamos “Next”.

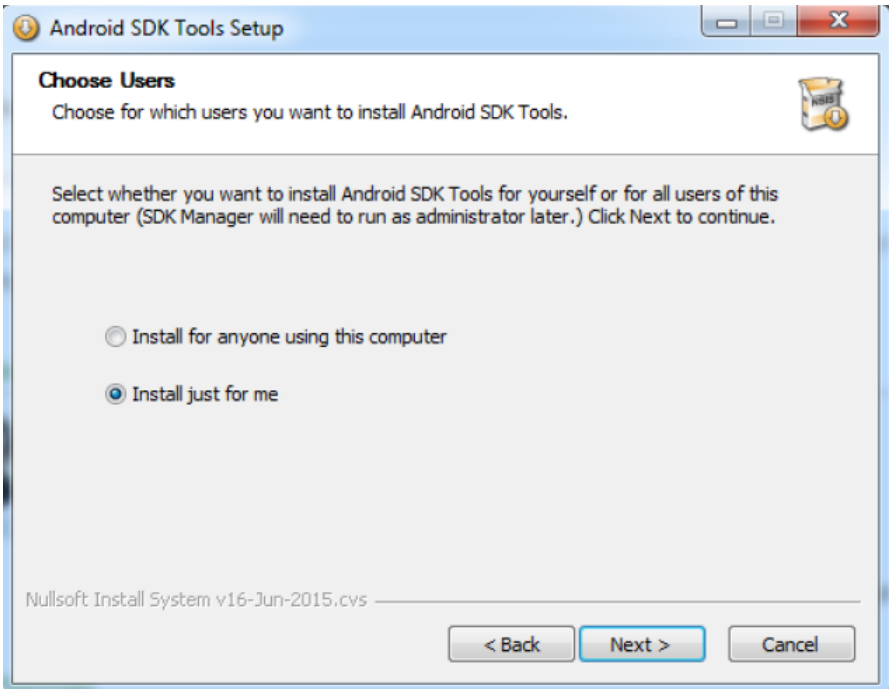

Figura B.20 Asistente de instalación Android SDK. Paso 3.

Nos aparecen dos opciones donde podemos elegir si queremos instalar el programa para todos los usuarios del ordenador o no. Elegimos una de las dos y pulsamos "Next".

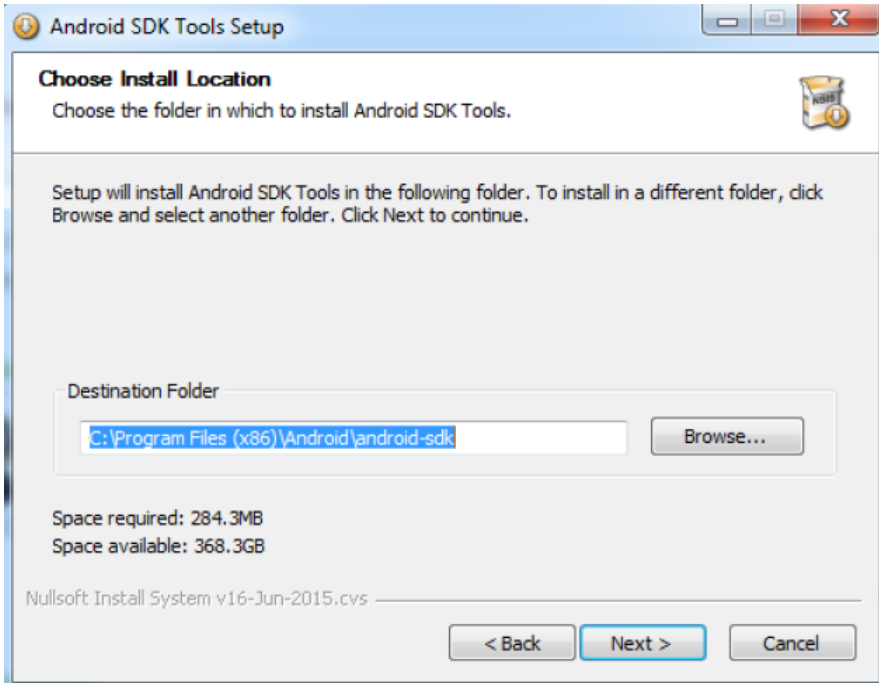

Figura B.21 Asistente de instalación Android SDK. Paso 4.

Elegimos el directorio de instalación y pulsamos “Next”.

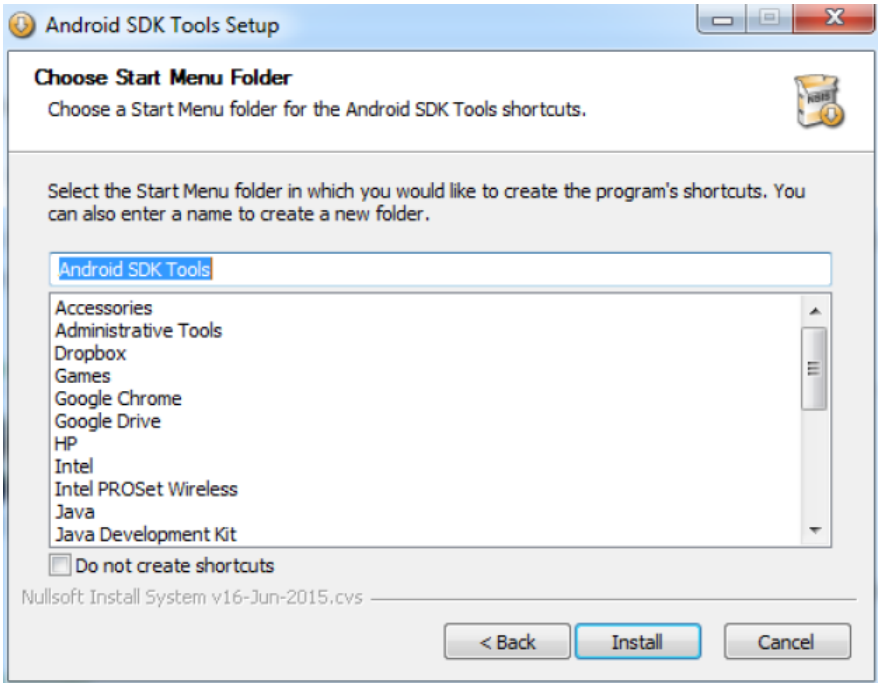

Figura B.22 Asistente de instalación Android SDK. Paso 5.

En este punto, el asistente nos da la opción de crear un acceso directo. Tomamos una decisión y pulsamos “Install”.

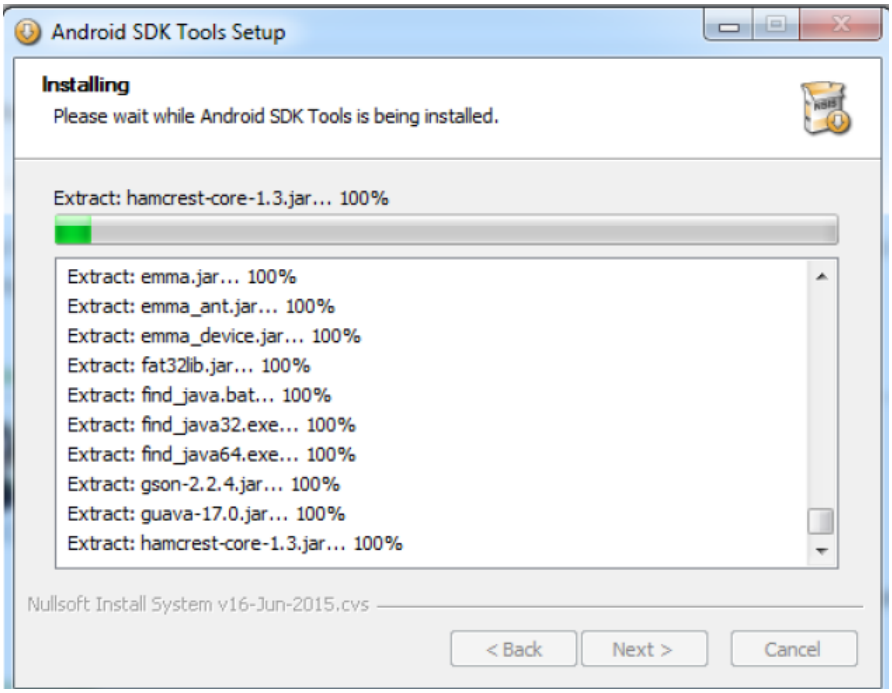

Figura B.23 Asistente de instalación Android SDK. Paso 6.

La instalación comenzará y, solo nos queda esperar que se complete.

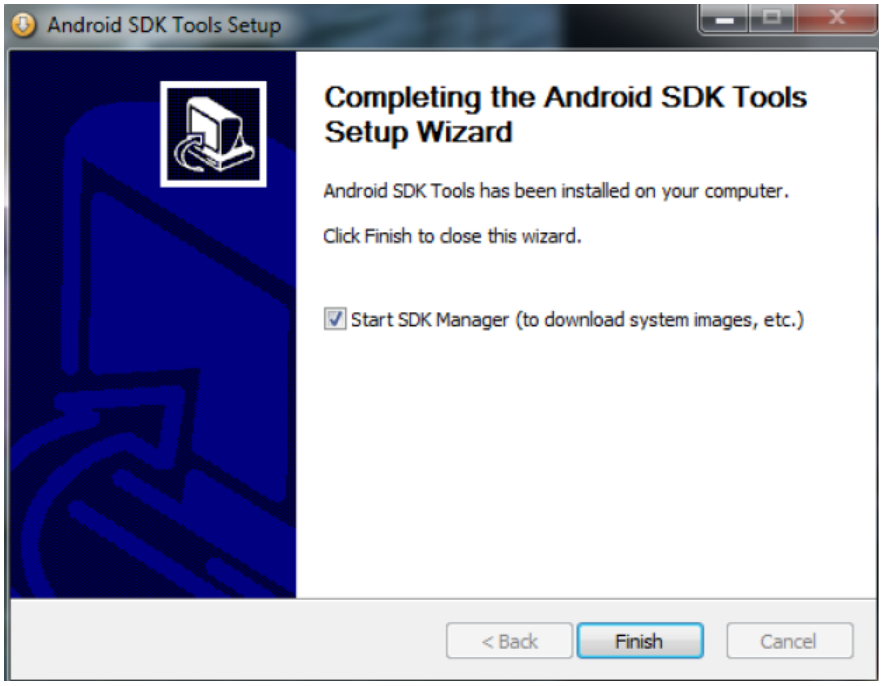

Figura B.24 Asistente de instalación Android SDK. Paso 7.

Llegados a este punto la instalación se da por finalizada la instalación. Ahora tenemos que vincular las dos herramientas para poder desarrollar nuestra aplicación Android en eclipse.

B.3.1 Vinculación de Android SDK y Eclipse

Partimos de que ya tenemos instalados correctamente Eclipse y Android SDK. Ahora nos falta vincularlos, y eso lo haremos de la siguiente manera. Abriendo Eclipse nos situamos sobre la barra de herramientas y buscamos la siguiente ruta: “Help > Install New Software”.

Pulsamos sobre la opción “Add” y modificamos ambos campos, como aparece en la siguiente imagen.

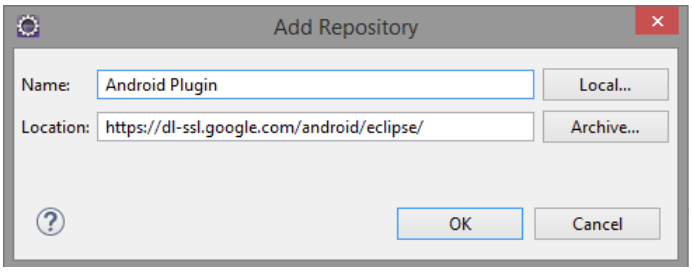

Figura B.25 Vinculación Android SDK y Eclipse. Paso 1.

La dirección web proviene de la página web para desarrolladores de Android: http://developer.android.com/sdk/installing/installing-adt.html. Tras completar ambos campos nos aparecerá una opción a instalar que hará mención a las herramientas de desarrollador (Developer Tools) como se muestra a continuación.

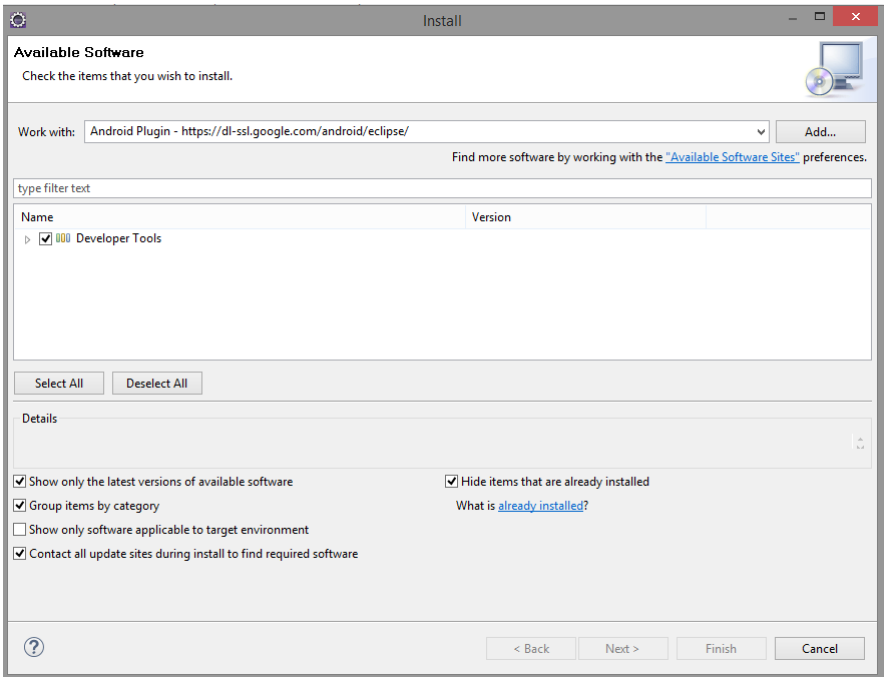

Figura B.26 Vinculación Android SDK y Eclipse. Paso 2.

Tras esto, hemos finalizado la vinculación de las dos herramientas necesarias para el desarrollo de nuestra aplicación Android en Eclipse. Reiniciamos Eclipse.

Anexo C

Códigos Java y xml de la aplicación Android

En este anexo se encuentran todos los códigos Java y xml que hemos desarrollado para crear la aplicación. Como ya hemos comentado, el lenguaje Android está estructurado en dos partes, el xml, donde se diseña la interfaz que verá el usuario (las distintas actividades) y el código Java que aporta la lógica para poder relizar todas las operaciones de la actividad en la que estemos.

Vamos a ver cada uno de estos archivos separados por familias, que corresponden a cada actividad de la que dispone la aplicación. Adjuntaremos también clase que se encarga de mantener y de realizar todas las operaciones sobre la base de datos, la clase que gestiona el fichero preferencias, donde almacenaremos el identificador de usuario de una determinada sesión y la clase que se encarga de leer un fichero JSON. También mostraremos el archivo de configuración AndroidManifest.xml y el fichero de recursos string.xml.

C.1 Familia MainActivity

C.1.1 MainActivity.java

```
1
2 package com.deporte.proyectofc;
3
4 /*
5  * Clase principal de la aplicacion. En ella el usuario podra
6  * registrarse, logearse o indicar que ha olvidado la contraseña.
```

```
7  */
8
9  import java.util.ArrayList;
10 import android.app.Activity;
11 import android.content.Intent;
12 import android.os.Bundle;
13 import android.view.View;
14 import android.widget.EditText;
15 import android.widget.Toast;
16
17 public class MainActivity extends Activity {
18
19     private EditText usuario;
20     private EditText contrasenia;
21     private BasedatosSQLite basedatos;
22
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         //Para que la primera vez se cree la base de datos antes de
               mostrar la pantalla
28         basedatos= new BasedatosSQLite(this);
29         basedatos.creaSQLitedatabase();
30         setContentView(R.layout.activity_main);
31         usuario= (EditText) findViewById(R.id._usuario);
32         contrasenia= (EditText) findViewById(R.id._contrasenia);
33     }
34
35
36     /*Metodo llamado cuando se pulsa sobre el boton registrar.
37      *Lanzamos la actividad para dar de alta a un nuevo usuario*/
38     public void lanzarRegistro(View view){
39         Intent i = new Intent(this, NuevoUsuario.class);
40         startActivity(i);
41     }
42
43
44     /*Metodo llamado cuando se pulsa sobre el boton Iniciar.
45      *Comprobamos si el usuario y la contraseña existen y accedemos
46      *a la actividad Inicio*/
47     public void lanzarInicio(View view){
48
49         String v_usuario= usuario.getText().toString();
50         String v_contrasenia= contrasenia.getText().toString();
51         //Comprobamos que en el campo usuario se haya introducido
               caracteres.
52         if(v_usuario.isEmpty()){
53             if(v_contrasenia.isEmpty()){
54                 Toast toast= Toast.makeText(getApplicationContext(), "Los
                       campos Usuario y Contraseña están vacíos",
55                     Toast.LENGTH_SHORT);
```

```
56                 toast.show();
57             }
58             else{
59                 Toast toast= Toast.makeText(getApplicationContext(), "El
                       campo Usuario está vacío.", Toast.LENGTH_SHORT);
60                 toast.show();
61             }
62         }
63         //Comprobamos que en el campo contraseña se han introducido
               caracteres, en caso contrario se muestra mensaje.
64         else  if(v_contrasenia.isEmpty()){
65             Toast toast= Toast.makeText(getApplicationContext(), "El campo
                    Contraseña está vacío ", Toast.LENGTH_SHORT);
66             toast.show();
67         }
68         //Si los campos usuario y contraseña contienen caracteres,
               comprobamos si existe el usuario
69         //y si escorrecta la contraseña.
70         else{
71             if(basedatos.existeUsuario(v_usuario)){
72                 //El usuario existe.
73                 //Obtenemos el hash y el salt almacenados de este usuario
74                 ArrayList<String> hashysalt= basedatos.obtenerHashYSalt(
                       v_usuario);
75                 //Validamos la contraseña introducida
76                 boolean correcta= EncriptadoContraseña.validaContrasenia(
                       v_contrasenia, hashysalt.get(0),
77                           EncriptadoContraseña.fromHexString(hashysalt.get(1)
                                ));
78                 if(correcta){
79                     int id_us= basedatos.obtenerIdUsuario(v_usuario);
80                     //Obtenemos el nombre del usuario para el saludo
                           inicial en la siguiente pantalla
81                     String nombre_usuario= basedatos.obtenerNombreUsuario(
                           id_us);
82                     //Guardamos de forma permanente durante la sesion el
                           id del usuario en fichero preferencias.
83                     AlmacenId valor_id= new AlmacenId(this);
84                     Intent i = new Intent(this, Inicio.class);
85                     valor_id.guardarId(id_us);
86                     i.putExtra("nombre_usuario", nombre_usuario);
87                     //Lanzamos actividad Inicio
88                     startActivity(i);
89                 }
90                 else{
91                     Toast toast= Toast.makeText(getApplicationContext(),
92                             "Contraseña incorrecta.", Toast.LENGTH_SHORT);
93                     toast.show();
94                 }
95             }
96             //No existe el usuario
97             else{
```

```
98                  Toast toast= Toast.makeText(getApplicationContext(),
99                          "El usuario no existe.", Toast.LENGTH_SHORT);
100                 toast.show();
101             }
102         }
103     }
104
105
106     /*Metodo llamado cuando se pulsa sobre el boton olvido de contraseña
            */
107     public void lanzarOlvidoContrasenia(View view){
108         Intent i= new Intent(this, ContraseniaOlvidada.class);
109         startActivity(i);
110     }
111
112 }
```

C.1.2 activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:id="@+id/LinearLayout1"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      android:padding="16dp"
9      tools:context="com.deporte.proyectofc.MainActivity" >
10
11     <TextView
12         android:id="@+id/titulo"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:layout_gravity="center"
16         android:layout_margin="8dp"
17         android:layout_weight="1"
18         android:gravity="center"
19         android:shadowColor="#222"
20         android:shadowDx="1"
21         android:shadowDy="2"
22         android:shadowRadius="1.5"
23         android:text="@string/app_name"
24         android:textAppearance="?android:attr/textAppearanceLarge"
25         android:textColor="#FFF"
26         android:textStyle="bold" />
27
28     <LinearLayout
29         android:layout_width="match_parent"
30         android:layout_height="wrap_content"
31         android:layout_weight="1"
```

```
32          android:orientation="vertical" >
33
34          <TextView
35              android:id="@+id/usuario"
36              android:layout_width="wrap_content"
37              android:layout_height="wrap_content"
38              android:text="@string/usuario" />
39
40          <EditText
41              android:id="@+id/_usuario"
42              android:layout_width="match_parent"
43              android:layout_height="40dp"
44              android:background="@drawable/edittextstyle"
45              android:ems="10"
46              android:hint="@string/introduce_usuario"
47              android:paddingLeft="8dp"
48              android:paddingRight="8dp" >
49
50              <requestFocus android:layout_width="wrap_content" />
51          </EditText>
52      </LinearLayout>
53
54      <LinearLayout
55          android:layout_width="match_parent"
56          android:layout_height="wrap_content"
57          android:layout_weight="1"
58          android:orientation="vertical" >
59
60          <TextView
61              android:id="@+id/contrasenia"
62              android:layout_width="match_parent"
63              android:layout_height="wrap_content"
64              android:layout_marginTop="18sp"
65              android:text="@string/contrasenia_usuario" />
66
67          <EditText
68              android:id="@+id/_contrasenia"
69              android:layout_width="match_parent"
70              android:layout_height="40dp"
71              android:background="@drawable/edittextstyle"
72              android:ems="10"
73              android:hint="@string/introduce_contrasenia"
74              android:inputType="textPassword"
75              android:paddingLeft="8dp"
76              android:paddingRight="8dp" />
77      </LinearLayout>
78
79      <LinearLayout
80          android:layout_width="match_parent"
81          android:layout_height="wrap_content"
82          android:layout_marginTop="35sp"
83          android:layout_weight="1"
```

```
84          android:orientation="vertical" >
85
86          <Button
87              android:id="@+id/iniciar"
88              android:layout_width="225dp"
89              android:layout_height="40sp"
90              android:layout_gravity="center"
91              android:layout_margin="10dp"
92              android:background="@drawable/buttonstyle"
93              android:onClick="lanzarInicio"
94              android:text="@string/login" />
95
96          <Button
97              android:id="@+id/registrarse"
98              android:layout_width="225dp"
99              android:layout_height="40sp"
100             android:layout_gravity="center"
101             android:background="@drawable/buttonstyle"
102             android:onClick="lanzarRegistro"
103             android:text="@string/registro" />
104     </LinearLayout>
105
106     <LinearLayout
107         android:layout_width="match_parent"
108         android:layout_height="wrap_content"
109         android:layout_weight="1"
110         android:orientation="vertical" >
111
112         <Button
113             android:id="@+id/olvido_pass"
114             android:layout_width="225dp"
115             android:layout_height="wrap_content"
116             android:layout_gravity="center"
117             android:background="@android:color/transparent"
118             android:onClick="lanzarOlvidoContrasenia"
119             android:text="@string/olvido_contrasenia"
120             android:textColor="#004C99" />
121     </LinearLayout>
122
123 </LinearLayout>
```

C.2 Familia Nuevo Usuario

C.2.1 NuevoUsuario.java

```
1
2 package com.deporte.proyectofc;
```

```
3
4  /* Clase que registra a un nuevo usuario en la base de datos. */
5
6  import java.security.MessageDigest;
7  import java.security.SecureRandom;
8  import android.app.Activity;
9  import android.database.sqlite.SQLiteDatabase;
10 import android.os.Bundle;
11 import android.view.View;
12 import android.widget.ArrayAdapter;
13 import android.widget.EditText;
14 import android.widget.Spinner;
15 import android.widget.Toast;
16
17 public class NuevoUsuario extends Activity {
18
19     private EditText nombre;
20     private EditText apellidos;
21     private EditText usuario;
22     private EditText contrasenia;
23     private EditText e_mail;
24     private Spinner estadodeforma;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.nuevo_usuario);
31         //Accedemos a los views que vamos a necesitar
32         nombre= (EditText) findViewById(R.id._nombre);
33         apellidos= (EditText) findViewById(R.id._apellidos);
34         usuario= (EditText) findViewById(R.id._usuario);
35         contrasenia= (EditText) findViewById(R.id._contrasenia);
36         e_mail= (EditText) findViewById(R.id._e_mail);
37         estadodeforma= (Spinner) findViewById(R.id.estado);
38         String[] valores = {"Bajo","Medio","Alto"};
39         estadodeforma.setAdapter(new ArrayAdapter<String>(this,
40                 android.R.layout.simple_spinner_item, valores));
41     }
42
43     //Al pulsar el boton, nos registramos como nuevo usuario en la base
           de datos
44     public void registrarse(View view){
45
46         String v_nombre= nombre.getText().toString();
47         String v_apellidos= apellidos.getText().toString();
48         String v_usuario= usuario.getText().toString();
49         String v_contrasenia= contrasenia.getText().toString();
50         String v_e_mail= e_mail.getText().toString();
51         String v_estado= estadodeforma.getSelectedItem().toString();
52
53         //Comprobamos que todos los campos estén rellenos
```

```
54          if(v_nombre.isEmpty() || v_apellidos.isEmpty() || v_usuario.
                isEmpty() || v_contrasenia.isEmpty() ||
55                  v_e_mail.isEmpty() || v_estado.isEmpty()){
56              Toast toast= Toast.makeText(getApplicationContext(), "Rellene
                    todos los campos.", Toast.LENGTH_LONG);
57              toast.show();
58          }
59          else{
60              BasedatosSQLite basedatos= new BasedatosSQLite(this);
61              //Si el usuario ya existe
62              if(basedatos.existeUsuario(v_usuario)){
63                  Toast toast= Toast.makeText(getApplicationContext(),
64                          "El usuario ya existe, pruebe con otro", Toast.
                                LENGTH_SHORT);
65                  toast.show();
66              }
67              //Si no existe, crea una nueva entrada en la tabla usuario,
                    haciendo uso de los datos introducidos.
68              else{
69                  //Generamos salt aleatorio.
70                  byte[] salt= EncriptadoContraseña.getSalt();
71
72                  //Realiza el hash de la pareja contraseña + salt.
73                  String hash= EncriptadoContraseña.hash(v_contrasenia, salt
                        );
74                  //Almacena los datos introducidos, excepto la contraseña
                        que en su lugar se
75                  //almacena el salt y el hash calculado.
76                  basedatos.guardarNuevoUsuario(v_usuario, hash,
                        EncriptadoContraseña.byteArrayToHexString(salt),
77                            v_nombre, v_apellidos, v_e_mail, v_estado);
78
79                  //Añadimos la rutina que tendra cada usuario nuevo
                        registrado, en la tabla rutina_usuario
80                  //la rutina GAP.
81                  int id_usuario= basedatos.obtenerIdUsuario(v_usuario);
82                  String nombre_rutina= "GAP";
83                  SQLiteDatabase bd= basedatos.creaSQLitedatabase();
84                  int id_rutina= basedatos.obtenerIdRutina(bd, nombre_rutina)
                        ;
85                  basedatos.introRutinaUsuario(id_usuario, id_rutina);
86                  Toast toast= Toast.makeText(getApplicationContext(),
87                          "Usted ha sido registrado en el sistema", Toast.
                                LENGTH_SHORT);
88                  toast.show();
89                  finish();
90
91              }
92          }
93      }//registrarse
94  }//NuevoUsuario
95
```

```
96
97 /*
98  * Clase que se encarga de gestionar la encriptacion de la contraseña
        para el almacenamiento en la base de datos, y de validar si la
        contraseña introducida es correcta.
99  */
100 class EncriptadoContraseña {
101
102     private static final char digits[] = { '0', '1', '2', '3', '4', '5', '6'
        , '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
103
104     //Devuelve una salt aleatorio, que será aplicada junto a la contraseñ
        a para el hashing.
105     public static byte[] getSalt(){
106
107         SecureRandom sr= new SecureRandom();
108         byte[] salt= new byte[16];
109         sr.nextBytes(salt);
110
111         return salt;
112     }
113
114     //Devuelve el hash del conjunto contraseña + salt
115     public static String hash(String contrasenia, byte[] salt){
116
117         try{
118             MessageDigest md = MessageDigest.getInstance("SHA-1");
119             md.update(salt);
120             md.update(contrasenia.getBytes("UTF-8"));
121             byte[] mdBytes = md.digest();
122             String hashString = byteArrayToHexString(mdBytes);
123             return hashString;
124         }catch(Exception e){
125             throw new RuntimeException(e);
126         }
127     }
128
129     //Valida si la contraseña introducida por el usuario es correcta.
130     public static boolean validaContrasenia(String contrasenia, String
        correctohash, byte[] salt){
131
132         String hashcalculado= EncriptadoContraseña.hash(contrasenia, salt
            );
133         //Devuelve true si es correcta.
134         return (hashcalculado.equals(correctohash));
135
136     }
137
138     //Metodo que pasa de hexadecimal a String
139     public static String byteArrayToHexString(byte[] b){
140
141         StringBuffer hexString = new StringBuffer(b.length);
```

```
142         for(int i = 0; i < b.length; i++){
143             hexString.append(digits[(b[i] & 0xF0) >> 4]);
144             hexString.append(digits[b[i] & 0x0F]);
145         }
146 
147         return hexString.toString();
148     }
149 
150     //Metodo que pasa de String a hexadecimal.
151     public static byte[] fromHexString(String s) {
152 
153         int length = (s.length() / 2);
154         byte[] bytes = new byte[length];
155         for(int i=0 ; i<length ; i++) {
156             bytes[i] = (byte) ((Character.digit(s.charAt(i * 2), 16) << 4)
                    | Character
157                    .digit(s.charAt((i * 2) + 1), 16));
158         }
159 
160         return bytes;
161     }
162 
163 }
```

C.2.2 nuevo_usuario.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent" >
5  
6      <LinearLayout
7          android:layout_width="match_parent"
8          android:layout_height="wrap_content"
9          android:orientation="vertical" >
10 
11         <TextView
12             android:id="@+id/nombre"
13             android:layout_width="wrap_content"
14             android:layout_height="wrap_content"
15             android:layout_margin="8dp"
16             android:text="@string/nombre"
17             android:textColor="#FFF" />
18 
19         <EditText
20             android:id="@+id/_nombre"
21             android:layout_width="match_parent"
22             android:layout_height="40dp"
23             android:layout_marginLeft="8dp"
24             android:layout_marginRight="8dp"
```

```
25              android:paddingLeft="8dp"
26              android:paddingRight="8dp"
27              android:background="@drawable/edittextgen"
28              android:ems="10"
29              android:hint="@string/intro_nombre">
30
31              <requestFocus />
32          </EditText>
33
34          <TextView
35              android:id="@+id/apellidos"
36              android:layout_width="wrap_content"
37              android:layout_height="wrap_content"
38              android:layout_margin="8dp"
39              android:text="@string/apellidos"
40              android:textColor="#FFF" />
41
42          <EditText
43              android:id="@+id/_apellidos"
44              android:layout_width="match_parent"
45              android:layout_height="40dp"
46              android:layout_marginLeft="8dp"
47              android:layout_marginRight="8dp"
48              android:paddingLeft="8dp"
49              android:paddingRight="8dp"
50              android:background="@drawable/edittextgen"
51              android:ems="10"
52              android:hint="@string/intro_apellidos" />
53
54          <TextView
55              android:id="@+id/usuario"
56              android:layout_width="wrap_content"
57              android:layout_height="wrap_content"
58              android:layout_margin="8dp"
59              android:text="@string/usuario"
60              android:textColor="#FFF" />
61
62          <EditText
63              android:id="@+id/_usuario"
64              android:layout_width="match_parent"
65              android:layout_height="40dp"
66              android:layout_marginLeft="8dp"
67              android:layout_marginRight="8dp"
68              android:paddingLeft="8dp"
69              android:paddingRight="8dp"
70              android:background="@drawable/edittextgen"
71              android:ems="10"
72              android:hint="@string/intro_usuario"/>
73
74          <TextView
75              android:id="@+id/contrasenia"
76              android:layout_width="wrap_content"
```

```
77              android:layout_height="wrap_content"
78              android:layout_margin="8dp"
79              android:text="@string/contrasenia_usuario"
80              android:textColor="#FFF" />
81
82          <EditText
83              android:id="@+id/_contrasenia"
84              android:layout_width="match_parent"
85              android:layout_height="40dp"
86              android:layout_marginLeft="8dp"
87              android:layout_marginRight="8dp"
88              android:paddingLeft="8dp"
89              android:paddingRight="8dp"
90              android:background="@drawable/edittextgen"
91              android:ems="10"
92              android:hint="@string/intro_contrasenia"
93              android:inputType="textPassword" />
94
95          <TextView
96              android:id="@+id/e_mail"
97              android:layout_width="wrap_content"
98              android:layout_height="wrap_content"
99              android:layout_margin="8dp"
100             android:text="@string/e_mail"
101             android:textColor="#FFF" />
102
103         <EditText
104             android:id="@+id/_e_mail"
105             android:layout_width="match_parent"
106             android:layout_height="40dp"
107             android:layout_marginLeft="8dp"
108             android:layout_marginRight="8dp"
109             android:paddingLeft="8dp"
110             android:paddingRight="8dp"
111             android:background="@drawable/edittextgen"
112             android:ems="10"
113             android:hint="@string/intro_e_mail"
114             android:inputType="textEmailAddress" />
115
116         <LinearLayout
117             android:layout_width="match_parent"
118             android:layout_height="wrap_content"
119             android:layout_margin="6dp"
120             android:orientation="horizontal" >
121
122             <TextView
123                 android:id="@+id/textView1"
124                 android:layout_width="wrap_content"
125                 android:layout_height="wrap_content"
126                 android:layout_margin="6dp"
127                 android:text="@string/nivel"
128                 android:textSize="16sp" />
```

```
129
130             <Spinner
131                 android:id="@+id/estado"
132                 android:layout_width="0dp"
133                 android:layout_height="wrap_content"
134                 android:layout_marginLeft="6dp"
135                 android:layout_marginRight="6dp"
136                 android:layout_marginTop="12dp"
137                 android:layout_weight="1"
138                 android:background="@android:drawable/btn_dropdown" />
139         </LinearLayout>
140
141         <Button
142             android:id="@+id/button1"
143             android:layout_width="match_parent"
144             android:layout_height="wrap_content"
145             android:layout_margin="14dp"
146             android:background="@drawable/buttongen"
147             android:onClick="registrarse"
148             android:text="@string/registro" />
149     </LinearLayout>
150
151 </ScrollView>
```

C.3 Familia Inicio de sesión

C.3.1 Inicio.java

```
1
2  package com.deporte.proyectofc;
3
4  /*
5   * Clase encargada de visualizar la pantalla Inicio, una vez el usuario
        ha sido autentificado. Podra elegir entre las rutinas que ya ha
        cargado en su lista, o cargar nuevas rutinas. El usuario también
        podrá modificar sus datos de usuario y su contraseña, o bien ver su
        rendimiento semanal.
6   */
7  import java.util.ArrayList;
8
9  import android.app.ListActivity;
10 import android.content.Intent;
11 import android.database.sqlite.SQLiteDatabase;
12 import android.graphics.Typeface;
13 import android.os.Bundle;
14 import android.view.KeyEvent;
15 import android.view.View;
```

```
16 import android.view.animation.Animation;
17 import android.view.animation.AnimationUtils;
18 import android.widget.AdapterView;
19 import android.widget.AdapterView.OnItemClickListener;
20 import android.widget.AdapterView.OnItemLongClickListener;
21 import android.widget.ArrayAdapter;
22 import android.widget.Button;
23 import android.widget.ListView;
24 import android.widget.TextView;
25
26 public class Inicio extends ListActivity{
27
28     private int id_usuario;
29     private BasedatosSQLite basedatos;
30     private ArrayList<String> nombrerutinas= new ArrayList<String>();
31     private ArrayList<Integer> ids_rutinas;
32     private ListView listarutinas;
33     private ArrayAdapter<String> adapter;
34
35     @Override
36     public void onCreate(Bundle savedInstanceState) {
37
38         super.onCreate(savedInstanceState);
39         setContentView(R.layout.inicio);
40
41         TextView titulo= (TextView) findViewById(R.id.titulo_app);
42         TextView pregunta= (TextView) findViewById(R.id.pregunta);
43         Bundle extras = getIntent().getExtras();
44         String nombre_usuario = extras.getString("nombre_usuario");
45         Typeface font = Typeface.createFromAsset(getAssets(),"ActionMan.
               ttf");
46         titulo.setTypeface(font);
47         //Asignamos animacion tween al titulo.
48         Animation animacionT= AnimationUtils.loadAnimation(this, R.anim.
               animacion);
49         titulo.startAnimation(animacionT);
50
51         AlmacenId id_us= new AlmacenId(this);
52         id_usuario= id_us.obtenerId();
53         basedatos= new BasedatosSQLite(this);
54         //Visualizamos la lista de rutinas de las que dispone el usuario.
55         ids_rutinas= basedatos.obtenerRutinasUsuario(id_usuario);
56         for(int i=0 ; i<ids_rutinas.size() ; i++){
57             nombrerutinas.add(basedatos.obtenerNombreRutina(ids_rutinas.
                   get(i)));
58         }
59
60         //Añadimos el item a la lista encargado de cargar más rutinas.
61         nombrerutinas.add("Cargar mas");
62         //Accedemos al elemento ListView.
63         listarutinas= (ListView) findViewById(android.R.id.list);
64         //Le asignamos un adaptador a este ListView.
```

```
65          adapter= new ArrayAdapter<String>(this, R.layout.entrada_rutina, R.
                id.nombrerutina, nombrerutinas);
66          adapter.notifyDataSetChanged();
67          listarutinas.setAdapter(adapter);
68          listarutinas.setItemsCanFocus(false);
69
70          Button b_miestadistica= (Button) findViewById(R.id.
                misestadisticas);
71          Button b_configuracion= (Button) findViewById(R.id.configuracion);

72          Button b_acercade= (Button) findViewById(R.id.acerca_de);
73          Button b_cerrarsesion= (Button) findViewById(R.id.cerrar);
74
75          //Asignamos animacion tween a los 4 botones de abajo.
76          Animation animacionB1= AnimationUtils.loadAnimation(this, R.anim.
                animacion_b1);
77          b_miestadistica.startAnimation(animacionB1);
78          Animation animacionB2= AnimationUtils.loadAnimation(this, R.anim.
                animacion_b2);
79          b_configuracion.startAnimation(animacionB2);
80          Animation animacionB3= AnimationUtils.loadAnimation(this, R.anim.
                animacion_b3);
81          b_acercade.startAnimation(animacionB3);
82          Animation animacionB4= AnimationUtils.loadAnimation(this, R.anim.
                animacion_b4);
83          b_cerrarsesion.startAnimation(animacionB4);
84
85          //Personalizamos el saludo con el nombre del usuario
86          pregunta.setText("Hola " + nombre_usuario + ", elige una rutina
                para hoy");
87
88          //Añadimos un escuchador a los item para que nos muestre la
                correspondiente
89          //lista de videos de la rutina elegida.
90          listarutinas.setOnItemClickListener(new OnItemClickListener() {
91              @Override
92              public void onItemClick(AdapterView<?> a, View view, int
                    position, long id){
93
94                  /* El item "cargar mas" es especial. Nos conduce a otra
                        actividad donde podremos seleccionar otras rutinas no
                        disponibles en la lista del usuario. */
95                  if ((nombrerutinas.get((int)id)).compareTo("Cargar mas")==0)
                        {
96                      Intent i= new Intent(Inicio.this, ListMasRutinas.class);
97                      startActivity(i);
98                  }
99                  else{
100                     Intent i= new Intent(Inicio.this, ListEjercicios.class);
101                     SQLiteDatabase bd= basedatos.creaSQLitedatabase();
102                     int id_rutina= basedatos.obtenerIdRutina(bd,
                            nombrerutinas.get((int)id));
```

```
103                 bd.close();
104                 i.putExtra("id_rutina", id_rutina);
105                 startActivity(i);
106             }
107         }
108     });
109
110     //Escuchador de click largo para borrar una rutina de nuestra
        lista.
111     //Este click no hará nada en caso de que los item sea el de
        cargar mas o el de
112     //la rutina GAP que no está permitido borrarla.
113     listarutinas.setOnItemLongClickListener(new
        OnItemLongClickListener() {
114         @Override
115         public boolean onItemLongClick(AdapterView<?> arg0, View arg1,
            int pos, long id) {
116             String nombre_rutina= nombrerutinas.get((int)id);
117             if(nombre_rutina.compareTo("Cargar mas")!=0 &&
                nombre_rutina.compareTo("GAP")!=0){
118                 SQLiteDatabase bd= basedatos.creaSQLitedatabase();
119                 int id_rutina= basedatos.obtenerIdRutina(bd,
                    nombre_rutina);
120                 bd.close();
121                 //Obtenemos el numero de usuarios que tienen
                    actualmente en su lista esta rutina.
122                 int cuenta= basedatos.cuentaUsuariosRutina(id_rutina);
123                 //Nueva actividad que pide confirmación
124                 Intent i= new Intent(Inicio.this, EliminacionRutina.
                    class);
125                 i.putExtra("id_rutina", id_rutina);
126                 i.putExtra("cuentarutina", cuenta);
127                 startActivity(i);
128             }
129             return true;
130         }
131     });
132 }//onCreate
133
134
135 /* Metodo que lanza la actividad que muestra la grafica de su
    rendimiento semanal. */
136 public void lanzarRendimiento(View view){
137     Intent i= new Intent(this, GraficasRendimiento.class);
138     startActivity(i);
139 }
140
141
142 /* Metodo llamado cuando se pulsa sobre el boton Configuracion.
    Lanzamos la actividad donde podremos configurar la cuenta de
    usuario. */
143 public void lanzarConfiguracion(View view){
```

```
144         Intent i= new Intent(this, ConfiguracionCuenta.class);
145         startActivity(i);
146     }
147
148
149     /* Metodo que lanza la actividad acerca de encargado de informar al
            usuario sobre el uso responsable de la app. */
150     public void lanzarAcercaDe(View view){
151         Intent i= new Intent(this, AcercaDe.class);
152         startActivity(i);
153     }
154
155
156     //Bloqueamos el boton atrás del dispositivo
157     @Override
158     public boolean onKeyDown(int keycode, KeyEvent event){
159         if (keycode==KeyEvent.KEYCODE_BACK && event.getRepeatCount() == 0){
160             return true;
161         }
162         return super.onKeyDown(keycode, event);
163     }
164
165
166     /* Metodo activado cuando se pulsa sobre el boton Cerrar sesion
167      * Cerramos la sesion del usuario, volviendo a la pantalla de inicio
             de la app. */
168     public void cerrarSesion(View view){
169         finish();
170     }
171
172
173     /* Metodo que recarga la actividad despues de ser detenida, pero al
            volver lo hara sin animaciones. */
174     @Override
175     public void onResume(){
176     super.onResume();
177         adapter.clear();
178         ids_rutinas= basedatos.obtenerRutinasUsuario(id_usuario);
179         for(int i=0 ; i<ids_rutinas.size() ; i++){
180             nombrerutinas.add(basedatos.obtenerNombreRutina(ids_rutinas.
                    get(i)));
181         }
182         //Añadimos el elemento a la lista para cargar más rutinas.
183         nombrerutinas.add("Cargar mas");
184         listarutinas= (ListView) findViewById(android.R.id.list);
185         //Le asignamos un adaptador a este ListView.
186         adapter= new ArrayAdapter<String>(this, R.layout.entrada_rutina, R.
                id.nombrerutina, nombrerutinas);
187         listarutinas.setAdapter(adapter);
188         listarutinas.setItemsCanFocus(false);
189     }
```

```
190
191 }
```

C.3.2 listadorutinas.xml

La clase Inicio contiene un elemento ListView que muestra las rutinas que tiene almacenadas el usuario, por ello, esta clase se compone dos XML:

- inicio.xml: carga la pantalla inicio. Uno de sus elementos se trata de un ListView, por lo que cargará el fondo sobre el que irá la lista, la cual será rellenada con los elementos entrada_rutina.xml.
- entrada_rutina.xml: será cada uno de los elementos de los que se compone la lista

inicio.xml

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:id="@+id/LinearLayout1"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical"
7     android:padding="16dp"
8     tools:context="com.deporte.proyectofc.MainActivity" >
9
10    <TextView
11        android:id="@+id/titulo_app"
12        android:layout_width="match_parent"
13        android:layout_height="wrap_content"
14        android:layout_gravity="center"
15        android:layout_marginTop="10dp"
16        android:layout_weight="3"
17        android:gravity="center"
18        android:text="@string/app_name"
19        android:textColor="#FFF"
20        android:textSize="45sp"
21        android:textStyle="bold" />
22
23    <TextView
24        android:id="@+id/pregunta"
25        android:layout_width="wrap_content"
26        android:layout_height="wrap_content"
27        android:layout_gravity="center"
28        android:layout_weight="0.51"
29        android:text="@string/pregunta"
30        android:textSize="20sp"
31        android:textStyle="bold" />
```

```
32
33      <FrameLayout
34          android:layout_width="match_parent"
35          android:layout_height="200dp"
36          android:background="@drawable/fondolistrutinas"
37          android:focusable="false" >
38
39          <ListView
40              android:id="@android:id/list"
41              android:layout_width="match_parent"
42              android:layout_height="wrap_content"
43              android:drawSelectorOnTop="false"
44              android:focusable="false"
45              android:focusableInTouchMode="false" >
46          </ListView>
47
48          <TextView
49              android:id="@android:id/empty"
50              android:layout_width="match_parent"
51              android:layout_height="wrap_content"
52              android:focusable="false"
53              android:text="@string/error_list_r" />
54      </FrameLayout>
55
56      <LinearLayout
57          android:layout_width="match_parent"
58          android:layout_height="wrap_content"
59          android:orientation="horizontal" >
60
61          <Button
62              android:id="@+id/misestadisticas"
63              android:layout_width="match_parent"
64              android:layout_height="40dp"
65              android:layout_margin="3dp"
66              android:layout_weight="1"
67              android:background="@drawable/buttongen"
68              android:drawableLeft="@android:drawable/ic_menu_week"
69              android:drawablePadding="0dp"
70              android:drawableStart="@android:drawable/ic_menu_week"
71              android:singleLine="true"
72              android:onClick="lanzarRendimiento"
73              android:paddingLeft="4dp"
74              android:paddingRight="4dp"
75              android:text="@string/mis_estadisticas"
76              android:textSize="12sp" />
77
78          <Button
79              android:id="@+id/configuracion"
80              android:layout_width="match_parent"
81              android:layout_height="40dp"
82              android:layout_margin="3dp"
83              android:layout_weight="1"
```

```
84          android:background="@drawable/buttongen"
85          android:drawableLeft="@android:drawable/ic_menu_manage"
86          android:drawablePadding="0dp"
87          android:drawableStart="@android:drawable/ic_menu_manage"
88          android:singleLine="true"
89          android:onClick="lanzarConfiguracion"
90          android:paddingLeft="4dp"
91          android:paddingRight="4dp"
92          android:text="@string/configuracion"
93          android:textSize="12sp" />
94      </LinearLayout>
95
96      <LinearLayout
97          android:layout_width="match_parent"
98          android:layout_height="wrap_content"
99          android:orientation="horizontal" >
100
101         <Button
102             android:id="@+id/acerca_de"
103             android:layout_width="match_parent"
104             android:layout_height="40dp"
105             android:layout_margin="3dp"
106             android:layout_weight="1"
107             android:background="@drawable/buttongen"
108             android:drawableLeft="@android:drawable/ic_menu_info_details"
109             android:drawablePadding="0dp"
110             android:drawableStart="@android:drawable/ic_menu_info_details"
111             android:onClick="lanzarAcercaDe"
112             android:singleLine="true"
113             android:paddingLeft="4dp"
114             android:paddingRight="4dp"
115             android:text="@string/acerca_de"
116             android:textSize="12sp" />
117
118         <Button
119             android:id="@+id/cerrar"
120             android:layout_width="match_parent"
121             android:layout_height="40dp"
122             android:layout_margin="3dp"
123             android:layout_weight="1"
124             android:background="@drawable/buttongen"
125             android:drawableLeft="@android:drawable/ic_lock_power_off"
126             android:drawablePadding="0dp"
127             android:drawableStart="@android:drawable/ic_lock_power_off"
128             android:onClick="cerrarSesion"
129             android:singleLine="true"
130             android:paddingLeft="4dp"
131             android:paddingRight="4dp"
132             android:text="@string/cerrar"
133             android:textSize="12sp" />
134     </LinearLayout>
135
```

```
</LinearLayout>
```

entrada_rutina.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingLeft="4dp"
    android:paddingRight="4dp"
    android:paddingTop="4dp" >

    <Button
        android:id="@+id/nombrerutina"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="1dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:background="@drawable/itemlistrutinas"
        android:clickable="false"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:textColor="#FFF"
        android:textStyle="bold" />

</LinearLayout>
```

C.4 Familia Olvido de contraseña

C.4.1 ContraseniaOlvidada.java

```

package com.deporte.proyectofc;

/* Clase que se encarga de gestionar el olvido de contrasenia cuando un
    usuario lo ha solicitado.*/

import java.util.Properties;
import java.util.Random;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
```

```
13 import javax.mail.Transport;
14 import javax.mail.internet.InternetAddress;
15 import javax.mail.internet.MimeMessage;
16
17 import android.app.Activity;
18 import android.app.ProgressDialog;
19 import android.content.Context;
20 import android.os.AsyncTask;
21 import android.os.Bundle;
22 import android.view.View;
23 import android.view.View.OnClickListener;
24 import android.widget.Button;
25 import android.widget.EditText;
26 import android.widget.Toast;
27
28 public class ContraseniaOlvidada extends Activity {
29
30     private final int LONGITUD= 8;
31     private BasedatosSQLite basedatos;
32
33     @Override
34     protected void onCreate(Bundle savedInstanceState){
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.contrasenia_olvidada);
37         basedatos= new BasedatosSQLite(this);
38
39         Button bsend= (Button) findViewById(R.id.enviar);
40         bsend.setOnClickListener(new OnClickListener(){
41             @Override public void onClick(View view){
42                 EditText e_mail= (EditText) findViewById(R.id._email);
43                 EditText usuario= (EditText) findViewById(R.id._usuario);
44                 String v_e_mail= e_mail.getText().toString();
45                 String v_usuario= usuario.getText().toString();
46                 //Comprueba si existe una entrada en la tabla usuarios con
                       el usuario y
47                 //e_mail introducidos
48                 if(basedatos.existeEmail(v_usuario, v_e_mail)){
49                     //Generamos una nueva contraseña aleatoria, la cual
                           enviaremos al
50                     //correo del usurio
51                     String cadenaAl= getCadenaAleatoria(LONGITUD);
52                     //Le calculamos el hash con la salt del usuario
53                     String salt= basedatos.obtenerSalt(v_usuario);
54                     String hashNuevo= EncriptadoContraseña.hash(cadenaAl,
                           EncriptadoContraseña.fromHexString(salt));
55                     //Lo guardamos en la base de datos
56                     AlmacenId valor_id= new AlmacenId(
                           getApplicationContext());
57                     int id= valor_id.obtenerId();
58                     basedatos.modificaContrasenia(id, hashNuevo);
59
60                     //Procedemos al envio del correo
```

```
61                  new EnvioMail(ContraseniaOlvidada.this).execute(
                        v_e_mail, cadenaAl);
62
63              }
64              else{
65                  Toast toast= Toast.makeText(getApplicationContext(),
66                          "El e-mail no se corresponde con el usuario",
                            Toast.LENGTH_SHORT);
67                  toast.show();
68              }
69          }
70      });
71  }
72
73  //Generador aleatorio de contraseñas
74  protected String getCadenaAleatoria(int longitud){
75      String cadena="";
76      long milis= new java.util.GregorianCalendar().getTimeInMillis();
77      Random random= new Random(milis);
78      int i=0;
79      while(i<longitud){
80          char c= (char)random.nextInt();
81          //Solo caracteres de 0 a 9, de A a Z y de a a z
82          if( (c<='9' && c>='0') || (c>='a' && c<='z') || (c>='A' && c<=
                'Z')){
83              cadena+= c;
84              i++;
85          }
86      }
87      return cadena;
88  }
89
90 }
91
92
93 /* Clase que se encarga de realizar el envio del e-mail con la nueva
       contrasenia a traves de un hilo secundario, mientras en el hilo
       primario se muestra el mensaje Enviando e-mail, */
94 class EnvioMail extends AsyncTask<String , Void, Void>{
95      private Context contexto;
96      private ProgressDialog dialog;
97
98      public EnvioMail(Context contexto){
99          this.contexto= contexto;
100     }
101
102     @Override protected void onPreExecute(){
103         dialog= ProgressDialog.show(contexto, "Enviando mail...", "Por
                favor espere", true);
104     }
105
106     @Override protected Void doInBackground(String... to){
```

```
107         sendMail(to[0], to[1]);
108         return null;
109     }
110 
111     @Override protected void onPostExecute(Void v) {
112         dialog.dismiss();
113     }
114 
115 
116     //Metodo que manda un e-mail desde la cuenta gmail de la app al
        correo del usuario, haciendo uso del protocolo SMTP
117     public void sendMail(String to, String newpassword){
118 
119         final String username = "app.deporte.us@gmail.com";
120         final String password = "appdeporte1234";
121 
122         //Configuracion del protocolo
123         Properties props = new Properties();
124         props.setProperty("mail.transport.protocol", "smtp");
125         props.setProperty("mail.host", "smtp.gmail.com");
126         props.put("mail.smtp.auth", "true");
127         props.put("mail.smtp.port", "465");
128         props.put("mail.smtp.socketFactory.port", "465");
129         props.put("mail.smtp.socketFactory.class",
130                 "javax.net.ssl.SSLSocketFactory");
131         props.put("mail.smtp.socketFactory.fallback", "false");
132         props.setProperty("mail.smtp.quitwait", "false");
133 
134         Session session = Session.getInstance(props,
135           new javax.mail.Authenticator() {
136             protected PasswordAuthentication getPasswordAuthentication() {
137                 return new PasswordAuthentication(username, password);
138             }
139           });
140 
141         try {
142 
143             //Introduce los campos del correo
144             Message message = new MimeMessage(session);
145             message.setFrom(new InternetAddress("app.deporte.us@gmail.com
                    "));
146             message.setRecipients(Message.RecipientType.TO,
147                 InternetAddress.parse(to));
148             message.setSubject("Nueva contraseña app_deporte");
149             message.setText("Su nueva contraseña es: " + newpassword + "."
                    );
150 
151             Transport.send(message);
152 
153         } catch (MessagingException e) {
154             throw new RuntimeException(e);
155         }
```

```
156     }
157 }
```

C.4.2 contrasenia_olvidada.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/textView1"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_marginBottom="20sp"
12         android:layout_marginLeft="25sp"
13         android:layout_marginRight="25sp"
14         android:layout_marginTop="30sp"
15         android:text="@string/info_olvido"
16         android:textStyle="bold"
17         android:textAppearance="?android:attr/textAppearanceMedium" />
18
19     <TextView
20         android:id="@+id/usuario"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_marginLeft="25sp"
24         android:layout_marginStart="25sp"
25         android:text="@string/usuario"
26         android:textAppearance="?android:attr/textAppearanceSmall" />
27
28     <EditText
29         android:id="@+id/_usuario"
30         android:layout_width="match_parent"
31         android:layout_height="35dp"
32         android:layout_marginLeft="25sp"
33         android:layout_marginRight="25sp"
34         android:layout_marginBottom="20sp"
35         android:layout_marginStart="25sp"
36         android:paddingLeft="8dp"
37         android:paddingRight="8dp"
38         android:hint="@string/introduce_usuario"
39         android:background="@drawable/edittextgen"
40         android:ems="10"/>
41
42     <TextView
43         android:id="@+id/email"
44         android:layout_width="wrap_content"
45         android:layout_height="wrap_content"
```

```
46          android:layout_marginLeft="25sp"
47          android:layout_marginStart="25sp"
48          android:text="@string/e_mail"
49          android:textAppearance="?android:attr/textAppearanceSmall" />
50
51      <EditText
52          android:id="@+id/_email"
53          android:layout_width="match_parent"
54          android:layout_height="35dp"
55          android:layout_marginLeft="25sp"
56          android:layout_marginRight="25sp"
57          android:layout_marginStart="25sp"
58          android:paddingLeft="8dp"
59          android:paddingRight="8dp"
60          android:background="@drawable/edittextgen"
61          android:ems="10"
62          android:inputType="textEmailAddress" />
63
64      <Button
65          android:id="@+id/enviar"
66          android:layout_width="wrap_content"
67          android:layout_height="wrap_content"
68          android:layout_marginLeft="25sp"
69          android:layout_marginRight="25sp"
70          android:layout_marginTop="30sp"
71          android:background="@drawable/buttongen"
72          android:text="@string/enviar" />
73
74  </LinearLayout>
```

C.5 Familia ListaEjercicios de una rutina

C.5.1 Listadeejercicios.java

Esta familia estará formada por dos clases: ListEjercicios.java y AdaptadorEjercicio.java

- ListEjercicios.java: Se encarga de mostrar el elemento ListView y asignarle un adaptador con el contenido.
- AdaptadorEjercicio.java: Es el adaptador del elemento ListView de la clase anterior. Se encarga de personalizar el contenido de cada item de la lista.

ListEjercicios.java

```java
package com.deporte.proyectofc;

/*
 * Clase que muestra una lista con los videos contenidos en una rutina,
     con sus items representados con una determinada estructura indicada
     por la clase Adaptador.
 */

import java.util.ArrayList;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.TextView;

public  class ListEjercicios extends ListActivity{

    private ArrayList<String> lista_rut_ejer= new ArrayList<String>();
    private ListView list;

    @Override public void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_ejercicios);

        //Extraemos el id de rutina de la anterior actividad.
        Bundle extras = getIntent().getExtras();
        int id_rutina= extras.getInt("id_rutina");
        TextView nombre_rutina= (TextView) findViewById(R.id.nombrerutina
            );
        TextView descripcion_rutina= (TextView) findViewById(R.id.
            descripcionrutina);
        BasedatosSQLite basedatos= new BasedatosSQLite(this);
        String nom_rutina= basedatos.obtenerNombreRutina(id_rutina);
        nombre_rutina.setText(nom_rutina);
        String descrip_rutina= basedatos.obtenerDescripcionRutina(
            id_rutina);
        descripcion_rutina.setText(descrip_rutina);

        AlmacenId id_usuario= new AlmacenId(this);
        int id_us= id_usuario.obtenerId();
        //Obtenemos el estado de forma del usuario
        String estadodeforma_us= basedatos.obtenerEstadoDeForma(id_us);
        //Obtenemos de la bbdd la lista de ejercicios que incluye esa
            rutina con el nivel de forma
        //del usuario.
```

```
46          ArrayList<String> lista_ejercicios= basedatos.listaEjEstadoForma(
                id_rutina, estadodeforma_us);
47          //Obtenemos el uri de cada ejercicio.
48          for(int i=0 ; i< lista_ejercicios.size() ; i++){
49              lista_rut_ejer.add(lista_ejercicios.get(i));
50          }
51
52          ArrayList<String> lista_titulos= basedatos.listaTitulos(id_rutina,
                 estadodeforma_us);
53          ArrayList<String> lista_subtitulos= basedatos.listaSubtitulos(
                id_rutina, estadodeforma_us);
54
55          list= (ListView) findViewById(android.R.id.list);
56          list.setItemsCanFocus(false);
57
58          //Obtenemos la lista de ejercicios que han sido realizados ya por
                 el usuario.
59          ArrayList<Integer> visualizados= basedatos.
                obtenerVisualizadosUsuario(id_us);
60          ListAdapter adaptador_ejercicio= new AdaptadorEjercicio(this,
                lista_rut_ejer, lista_titulos, lista_subtitulos, id_us ,
                visualizados);
61          list.setAdapter(adaptador_ejercicio);
62
63          //Escuchador del item que se ha pulsado para la reproduccion de
                su video correspondiente.
64          list.setOnItemClickListener(new OnItemClickListener() {
65              @Override
66              public void onItemClick(AdapterView<?> a, View view, int
                    position, long id){
67                  Intent i= new Intent(ListEjercicios.this,
                        ReproductorEjercicio.class);
68                  i.putExtra("ruta", lista_rut_ejer.get(position).toString()
                        );
69                  startActivity(i);
70              }
71          });
72      }//onCreate
73
74      //Metodo que recarga la actividad cuando se vuelve a ella.
75      @Override public void onRestart(){
76          super.onRestart();
77          finish();
78          startActivity(getIntent());
79      }
80  }
```

AdaptadorEjercicio.java

```
1  package com.deporte.proyectofc;
2
```

```
3  /*
4   * Clase encargada de definir el adaptador para mostrar la lista de
        ejercicios de una rutina. Los elementos (item) del ListView serán
        una pequeña imagen del video del ejercicio junto con un titulo, un
        pequeño subtitulo y la fecha y hora si ya se ha realizado antes el
        ejercicio.
5   */
6
7  import java.util.ArrayList;
8  import android.app.Activity;
9  import android.content.Context;
10 import android.graphics.Bitmap;
11 import android.media.ThumbnailUtils;
12 import android.provider.MediaStore.Images.Thumbnails;
13 import android.view.LayoutInflater;
14 import android.view.View;
15 import android.view.ViewGroup;
16 import android.widget.BaseAdapter;
17 import android.widget.ImageView;
18 import android.widget.TextView;
19
20 public class AdaptadorEjercicio extends BaseAdapter{
21
22     private final Activity actividad;
23     private final ArrayList<String> lista_ejercicios;
24     private final ArrayList<String> lista_titulos;
25     private final ArrayList<String> lista_subtitulos;
26     private final ArrayList<Integer> visualizados;
27     private final int id_usuario;
28
29     public AdaptadorEjercicio(Activity actividad, ArrayList<String>
           lista_ejercicios, ArrayList<String> lista_titulos,
30             ArrayList<String> lista_subtitulos, int id_usuario, ArrayList<
                   Integer> visualizados) {
31
32           super();
33           this.actividad = actividad;
34           this.lista_ejercicios = lista_ejercicios;
35           this.lista_titulos= lista_titulos;
36           this.lista_subtitulos= lista_subtitulos;
37           this.id_usuario= id_usuario;
38           this.visualizados= visualizados;
39          notifyDataSetChanged();
40     }
41
42
43     //Metodo necesario para adaptar el contenido de cada uno de los item.
44     public View getView(int position, View convertView, ViewGroup parent)
           {
45
46         //En ViewHolder se almacenan las vistas por primera vez para no
               tenerlas que cargar siempre
```

```
47          View row= convertView;
48          ViewHolder holder= null;
49          row= null;
50          if (row==null){
51              LayoutInflater inflater = actividad.getLayoutInflater();
52              row= inflater.inflate(R.layout.entrada_ejercicio, null, true);
53
54              holder= new ViewHolder();
55              holder.imagen_ej= (ImageView)row.findViewById(R.id.ic_video);
56              holder.imagen_ej.setFocusable(false);
57              holder.titulo= (TextView)row.findViewById(R.id.titulo);
58              holder.titulo.setFocusable(false);
59              holder.subtitulo= (TextView) row.findViewById(R.id.descripcion
                    );
60              holder.subtitulo.setFocusable(false);
61              holder.hecho= (TextView) row.findViewById(R.id.visto);
62              holder.fecha= (TextView) row.findViewById(R.id.fecha);
63              holder.hora= (TextView) row.findViewById(R.id.hora);
64
65              row.setTag(holder);
66          }
67
68          //Creamos una imagen a partir de la ruta del video
69          Bitmap imag= ThumbnailUtils.createVideoThumbnail(lista_ejercicios.
                get(position), Thumbnails.MINI_KIND);
70          holder.imagen_ej.setImageBitmap(imag);
71          holder.titulo.setText(lista_titulos.get(position));
72          holder.subtitulo.setText(lista_subtitulos.get(position));
73          Context contexto= actividad.getApplicationContext();
74          BasedatosSQLite basedatos= new BasedatosSQLite(contexto);
75          //Obtenemos id del ejercicio de este item
76          int id_ejercicio= basedatos.obtenerIdEjercicio(lista_ejercicios.
                get(position));
77
78          if (visualizados.contains(id_ejercicio)){
79
80              holder.hecho.setVisibility(View.VISIBLE);
81              //Obtenemor la fecha y la hora a la que ha sido visto
82              String[] fechayhora= basedatos.obtenerFechayHora(id_usuario,
                    id_ejercicio);
83              holder.fecha.setText(fechayhora[0]);
84              holder.fecha.setVisibility(View.VISIBLE);
85              holder.hora.setText(fechayhora[1]);
86              holder.hora.setVisibility(View.VISIBLE);
87          }
88
89          return row;
90      }
91
92      public int getCount() {
93          return lista_ejercicios.size();
94      }
```

```
95
96      public Object getItem(int arg0) {
97          return lista_ejercicios.get(arg0);
98      }
99
100     public long getItemId(int position) {
101         return position;
102     }
103
104 }
105
106 //Clase que almacena las vistas
107 class ViewHolder {
108
109         ImageView imagen_ej;
110         TextView titulo;
111         TextView subtitulo;
112         TextView hecho;
113         TextView fecha;
114         TextView hora;
115
116 }
```

C.5.2 Listadeejercicios.xml

De nuevo, al tratarse de una lista de elementos, necesitamos dos XML:

- list_ejercicios.xml: su función es cargar el fondo sobre el que irá la lista, que será rellenada con los elementos entrada_ejercicio.xml.
- entrada_ejercicio.xml: se encarga de definir la vista de cada uno de los elementos de los que se compone el ListView de lista_ejercicios.xml.

list_ejercicios.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/nombrerutina"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_marginLeft="20sp"
12         android:layout_marginTop="20sp"
13         android:layout_marginRight="20sp"
```

```
14          android:layout_marginBottom="10sp"
15          android:contentDescription="@string/nom_rutina"
16          android:textAppearance="?android:attr/textAppearanceMedium"
17          android:textColor="#FFF"
18          android:textSize="10pt"
19          android:textStyle="bold" />
20
21      <ScrollView
22          android:id="@+id/scrolldescripcion"
23          android:layout_width="match_parent"
24          android:layout_height="120dp"
25          android:layout_margin="10sp"
26          android:background="@drawable/fondolistrutinas">
27
28          <TextView
29          android:id="@+id/descripcionrutina"
30          android:layout_width="match_parent"
31          android:layout_height="wrap_content"
32          android:padding="10sp"
33          android:contentDescription="@string/desc_rutina"
34          android:textStyle="bold"
35          android:textSize="16sp"/>
36
37      </ScrollView>
38
39      <FrameLayout
40          android:layout_width="match_parent"
41          android:layout_height="0dip"
42          android:layout_marginLeft="10dp"
43          android:layout_marginRight="10dp"
44          android:layout_weight="1"
45          android:focusable="false" >
46
47          <ListView
48              android:id="@android:id/list"
49              android:layout_width="match_parent"
50              android:layout_height="match_parent"
51              android:drawSelectorOnTop="false"
52              android:focusable="false"
53              android:focusableInTouchMode="false" >
54          </ListView>
55
56          <TextView
57              android:id="@android:id/empty"
58              android:layout_width="match_parent"
59              android:layout_height="match_parent"
60              android:focusable="false"
61              android:text="@string/error_list_e" />
62      </FrameLayout>
63
64  </LinearLayout>
```

entrada_ejercicio.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:layout_marginBottom="10dp"
    android:background="@drawable/fondolistrutinas"
    android:focusable="false"
    android:focusableInTouchMode="false" >

    <ImageView
        android:id="@+id/ic_video"
        android:layout_width="125dp"
        android:layout_height="110dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_margin="5dp"
        android:clickable="false"
        android:contentDescription="@string/imag"
        android:focusable="false"
        android:focusableInTouchMode="false" />

    <TextView
        android:id="@+id/titulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_margin="3dp"
        android:layout_toEndOf="@id/ic_video"
        android:layout_toRightOf="@id/ic_video"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:singleLine="false"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="18sp" />

    <LinearLayout
        android:id="@+id/layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/titulo"
        android:layout_toEndOf="@id/ic_video"
        android:layout_toRightOf="@id/ic_video"
        android:focusable="false"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/visto"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:focusable="false"
            android:text="@string/hecho"
            android:textColor="#F00"
            android:textSize="12sp"
            android:visibility="gone" />

        <TextView
            android:id="@+id/fecha"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:focusable="false"
            android:textSize="12sp"
            android:visibility="gone" />

        <TextView
            android:id="@+id/hora"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:focusable="false"
            android:textSize="12sp"
            android:visibility="gone" />
    </LinearLayout>

    <TextView
        android:id="@+id/descripcion"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/layout"
        android:layout_margin="3dp"
        android:layout_toEndOf="@id/ic_video"
        android:layout_toRightOf="@id/ic_video"
        android:focusable="false"
        android:focusableInTouchMode="false" />

</RelativeLayout>
```

C.6 Familia Eliminacionrutinas

C.6.1 EliminacionRutina.java

```
package com.deporte.proyectofc;

/*
```

```
4  * Clase que se encarga de eliminar una rutina de la lista de rutinas del
       usuario. Si es el unico usuario que dispone de la rutina
      seleccionada, se eliminara también de la carpeta /app_deportes/
      rutinas/ del dispositivo.
5  */
6
7  import java.io.File;
8  import java.util.ArrayList;
9  import android.app.Activity;
10 import android.database.sqlite.SQLiteDatabase;
11 import android.os.Bundle;
12 import android.view.View;
13
14 public class EliminacionRutina extends Activity{
15
16     private int id_rutina;
17     private int cuenta;
18
19     @Override
20     public void onCreate(Bundle savedInstanceState){
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.eliminacion_rutina);
23         Bundle extras= getIntent().getExtras();
24         id_rutina= extras.getInt("id_rutina");
25         cuenta= extras.getInt("cuentarutina");
26
27     }
28
29     public void lanzarAceptar(View view){
30         BasedatosSQLite basedatos= new BasedatosSQLite(this);
31         AlmacenId id= new AlmacenId(this);
32         int _id_usuario= id.obtenerId();
33         //Si otros usuarios tienen en su lista esta rutina, solo borramos
               la entrada
34         //de este usuario en la tabla usuario_rutina
35         if(cuenta>1){
36             basedatos.borrarRutUsuario(_id_usuario, id_rutina);
37             finish();
38         }
39         //Si es el unico que la tenia
40         else{
41             basedatos.borrarRutUsuario(_id_usuario, id_rutina);
42             //Eliminamos los ejercicios de la rutina de la tabla
                   ejercicios
43             ArrayList<String> ejercicios= basedatos.listaEjerciciosRutina(
                   id_rutina);
44             for(int i=0 ; i<ejercicios.size() ; i++){
45                 basedatos.eliminarEjercicio(ejercicios.get(i));
46             }
47             //borramos la carpeta de esta rutina de la carpeta app_rutinas
48             SQLiteDatabase bd= basedatos.creaSQLitedatabase();
```

```
49              String path_rutina= basedatos.obtenerPathRutina(bd, id_rutina)
                    ;
50              bd.close();
51              if(eliminaCarpeta(path_rutina)){
52                  //borramos de la tabla rutina
53                  basedatos.borrarRutina(id_rutina);
54              }
55              finish();
56          }
57      }
58
59      //Elimina una carpeta, pero para ello debe eliminar todo su contenido
60      public boolean eliminaCarpeta(String path_rutina){
61          File file= new File(path_rutina);
62          //Si existe lo borramos
63          if(file.exists()){
64              borrarfic(file);
65              if(file.delete()){
66                  return true;
67              }else return false;
68          }else return false;
69      }
70
71      //Borra el contenido de un directorio
72      public void borrarfic(File directorio){
73          File[] ficheros= directorio.listFiles();
74          for(int i=0 ; i<ficheros.length ; i++){
75              if(ficheros[i].isDirectory()){
76                  borrarfic(ficheros[i]);
77              }
78              ficheros[i].delete();
79          }
80      }
81
82      //Cancelamos la operacion
83      public void lanzarCancelar(View view){
84          finish();
85      }
86  }
```

C.6.2 eliminacion_rutina.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/eliminarut"
```

```
9           android:layout_width="match_parent"
10          android:layout_height="wrap_content"
11          android:gravity="center"
12          android:text="@string/confir_eliminacion" />
13
14      <LinearLayout
15          android:layout_width="match_parent"
16          android:layout_height="wrap_content"
17          android:orientation="horizontal" >
18
19          <Button
20              android:id="@+id/aceptar"
21              android:layout_width="match_parent"
22              android:layout_height="40dp"
23              android:layout_weight="1"
24              android:onClick="lanzarAceptar"
25              android:text="@string/aceptar" />
26
27          <Button
28              android:id="@+id/cancelar"
29              android:layout_width="match_parent"
30              android:layout_height="40dp"
31              android:layout_weight="1"
32              android:onClick="lanzarCancelar"
33              android:text="@string/cancelar" />
34      </LinearLayout>
35
36  </LinearLayout>
```

C.7 Familia ReproductorEjercicio

C.7.1 ReproductorEjercicio.java

```
1  package com.deporte.proyectofc;
2
3  /* Clase que se encarga de reproducir el video correspondiente a un
       ejercicios seleccionado. */
4
5  import java.text.DateFormat;
6  import java.text.SimpleDateFormat;
7  import java.util.Date;
8  import android.app.Activity;
9  import android.content.Intent;
10 import android.media.MediaPlayer;
11 import android.net.Uri;
12 import android.os.Bundle;
13 import android.os.SystemClock;
```

```
14 import android.view.View;
15 import android.view.View.OnClickListener;
16 import android.widget.Button;
17 import android.widget.Chronometer;
18 import android.widget.ImageButton;
19 import android.widget.TextView;
20 import android.widget.Toast;
21 import android.widget.VideoView;
22
23 public class ReproductorEjercicio extends Activity{
24
25     private static VideoView mVideoView;
26     private static TextView repeticiones;
27     private static Uri uri_ejercicio;
28     private static int id_ejercicio;
29     private static int contador_repet= 0;
30     private static int num_repeticiones= 0;
31     private static String descripcion="";
32     private static TextView _descripcion;
33     private static Button comenzar, terminado, siguiente;
34     private static ImageButton play_pause, stop;
35     private static Chronometer cronometro;
36     private static BasedatosSQLite basedatos;
37     private static long crono;
38     private int id_rutina;
39     //Almcenan los estados de los botones comenzar y play, por si
           cambiamos orientacion
40     private boolean estado_bcomenzar= false;
41     private boolean estado_bplay= false;
42
43
44     @Override public void onCreate(Bundle savedInstanceState){
45
46         super.onCreate(savedInstanceState);
47         setContentView(R.layout.reproductor_ejercicio);
48         Bundle extras= getIntent().getExtras();
49         uri_ejercicio= Uri.parse(extras.getString("ruta"));
50
51         basedatos= new BasedatosSQLite(getApplicationContext());
52         id_ejercicio= basedatos.obtenerIdEjercicio(uri_ejercicio.toString
               ());
53         id_rutina= basedatos.obtenerIdRutEjercicio(id_ejercicio);
54         num_repeticiones= basedatos.obtenerNumRepeticiones(id_ejercicio);
55         mVideoView= (VideoView) findViewById(R.id.surface_view);
56         repeticiones= (TextView) findViewById(R.id.repeticiones);
57         mVideoView.setZOrderOnTop(true);
58         mVideoView.setVideoURI(uri_ejercicio);
59         comenzar= (Button) findViewById(R.id.comenzar);
60         cronometro= (Chronometer) findViewById(R.id.cronometro);
61
62         /* El siguiente bloque sirve por si la posición del móvil cambia
               de vertical a horizontal o viceversa, pues cada vez que esto
```

```
        ocurre la actividad se reinicia. Queremos que cuando se ponga
        en horizontal toda la pantalla muestre el video para una mejor
         visualizacion. Ademas queremos que no se detenga la
        reproduccion del video o del cronommetro. */
63
64      //Si la actividad se muestra por primera vez
65      if (savedInstanceState == null){
66          mVideoView.seekTo(500);
67      }
68      else {
69          //Obtenemos los datos obtenidos antes de cerrar la actividad
                en la variable savedInstanceState.
70          estado_bcomenzar= savedInstanceState.getBoolean("
                pulsado_comenzar");
71          estado_bplay= savedInstanceState.getBoolean("pulsado_play");
72          //Si se habia pulsado "Comenzar".
73          if (estado_bcomenzar){
74              //Obtenemos los valores que se almacenaron antes de
                    detener la actividad
75              contador_repet= savedInstanceState.getInt("contador");
76              int pos= savedInstanceState.getInt("posicion");
77              //reproducimos el video por la posicion por donde iba
78              mVideoView.seekTo(pos);
79              mVideoView.start();
80              crono = savedInstanceState.getLong("crono");
81              //Continuamos el cronometro por donde iba antes de cambiar
                     la posicion del dispositivo.
82              cronometro.setBase(SystemClock.elapsedRealtime() - crono);
83              cronometro.start();
84
85              mVideoView.setOnCompletionListener(new MediaPlayer.
                    OnCompletionListener(){
86                  public void onCompletion(MediaPlayer media){
87                      //Si hemos llegado a la ultima repeticion pasamos a
                             considerarlo como visualizado, es decir, el
                            ejercicio ha sido realizado.
88                      if (contador_repet == num_repeticiones){
89                          mVideoView.stopPlayback();
90                      }
91                      else {
92                          //Lo reproduce de nuevo.
93                          mVideoView.start();
94                          contador_repet++;
95                      }
96                  }
97              });
98          }
99          //Si se habia pulsado play
100         else if (estado_bplay){
101             contador_repet= savedInstanceState.getInt("contador");
102             int pos= savedInstanceState.getInt("posicion");
103             mVideoView.seekTo(pos);
```

```
104                 mVideoView.start();
105             }
106             else{
107                 mVideoView.seekTo(500);
108             }
109         }
110 
111         mVideoView.requestFocus();
112         repeticiones.setText(num_repeticiones + " repeticiones");
113         descripcion= basedatos.obtenerDescripEjercicio(id_ejercicio);
114         _descripcion= (TextView) findViewById(R.id._descripcion);
115         _descripcion.setText(descripcion);
116 
117         //Comenzamos la reproduccion del ejercicio al pulsar "COMENZAR" e
                iniciamos el cronometro
118         comenzar.setOnClickListener(new OnClickListener(){
119 
120             public void onClick(View view){
121                 mVideoView.seekTo(0);
122                 mVideoView.start();
123                 estado_bcomenzar= true;
124                 cronometro.setBase(SystemClock.elapsedRealtime());
125                 cronometro.start();
126                 //Estamos reproduciendo la primera repeticion
127                 contador_repet= 1;
128                 mVideoView.setOnCompletionListener(new MediaPlayer.
                        OnCompletionListener(){
129                     public void onCompletion(MediaPlayer media){
130                         //Si hemos llegado a la ultima repeticion pasamos a
                                considerarlo como visualizado, es decir, el
                                ejercicion ha sido realizado.
131                         if(contador_repet == num_repeticiones){
132                             mVideoView.stopPlayback();
133                         }
134                         else{
135                             //Volvemos a reproducirlo.
136                             contador_repet++;
137                             mVideoView.start();
138                         }
139                     }
140                 });
141             }
142         });
143 
144         //Pulsamos "Terminado".
145         //Paramos el crono y el video porque el usuario ha terminado de
                realizar el ejercicio.
146         terminado= (Button) findViewById(R.id.parar);
147         terminado.setOnClickListener(new OnClickListener(){
148             public void onClick(View view){
149                 cronometro.stop();
150                 mVideoView.stopPlayback();
```

```
151                 mVideoView.setVideoURI(uri_ejercicio);
152                 /*Hemos acabado el ejercicio, por tanto lo guardamos con
                        la fecha y la hora
153                  *en que lo hemos realizado.*/
154                 //Obtenemos id del usuario.
155                 AlmacenId _id= new AlmacenId(getApplicationContext());
156                 int id= _id.obtenerId();
157                 //Obtenemos fecha y hora
158                 DateFormat formatodate= new SimpleDateFormat("yyyy/MM/dd")
                        ;
159                 String date= formatodate.format(new Date());
160                 DateFormat formatotime= new SimpleDateFormat("HH:mm:ss");
161                 String time= formatotime.format(new Date());
162                 long milis= SystemClock.elapsedRealtime()-cronometro.
                        getBase();
163                 int tiempo= (int)(milis/1000);
164                 basedatos.guardarNuevaVisualizacion(id, id_ejercicio, date,
                         time, tiempo);
165             }
166         });
167
168         //Usuario pulsa play/pause.
169         //Comenzamos la reproduccion del video
170         play_pause= (ImageButton) findViewById(R.id.play);
171         play_pause.setOnClickListener(new OnClickListener(){
172             public void onClick(View view){
173                 estado_bplay= true;
174                 //Drawable fondo;
175                 if(mVideoView.isPlaying()){
176                     System.out.println("reproduciendo play");
177                     mVideoView.pause();
178                     play_pause.setImageResource(R.drawable.pause);
179                 }
180                 else{
181                     System.out.println("pausa");
182                     mVideoView.start();
183                     play_pause.setImageResource(R.drawable.play);
184                 }
185             }
186         });
187
188         //Usuario pulsa stop.
189         //Paramos el video del ejercicio.
190         stop= (ImageButton) findViewById(R.id.stop);
191         stop.setOnClickListener(new OnClickListener(){
192             public void onClick(View view){
193                 if(mVideoView.isPlaying()){
194                     mVideoView.stopPlayback();
195                     mVideoView.setVideoURI(uri_ejercicio);
196                 }
197             }
198         });
```

```
199
200         //Usuario pulsa "Siguiente".
201         //Reproducimos el siguiente ejercicio.
202         siguiente= (Button) findViewById(R.id.siguiente);
203         siguiente.setOnClickListener(new OnClickListener(){
204             public void onClick(View view){
205                 AlmacenId id_usuario= new AlmacenId(getApplicationContext
                        ());
206                 int id_us= id_usuario.obtenerId();
207                 String estadoformausuario= basedatos.obtenerEstadoDeForma(
                        id_us);
208                 do{
209                     id_ejercicio++;
210                 }while(basedatos.existeEjercicio(id_ejercicio) && !(
                        basedatos.obtenerEstadoEjercicio(id_ejercicio).equals(
                        estadoformausuario)));
211
212                 if(basedatos.existeEjercicio(id_ejercicio)){
213                     uri_ejercicio= Uri.parse(basedatos.
                            obtenerPathEjercicio(id_ejercicio));
214                     String _uri= uri_ejercicio.toString();
215                     //Comprobamos que el video pertenezca a la rutina
                            actual
216                     if(id_rutina==basedatos.obtenerIdRutEjercicio(
                            id_ejercicio)){
217                         finish();
218                         Intent i= new Intent(getApplicationContext(),
                                ReproductorEjercicio.class);
219                         i.putExtra("uri", _uri);
220                         startActivity(i);
221                     }
222                     else{
223                         Toast toast= Toast.makeText(getApplicationContext()
                                ,
224                                 "Ya no hay más videos en esta rutina.",
                                    Toast.LENGTH_SHORT);
225                         toast.show();
226                     }
227                 }
228                 else{
229                     Toast toast= Toast.makeText(getApplicationContext(),
230                             "Ya no hay más videos en esta rutina.", Toast.
                                LENGTH_SHORT);
231                     toast.show();
232                 }
233             }
234         });
235
236     }//onCreate
237
238     //Metodo que guarda el estado de las variables en un Bundle antes de
            detener la actividad.
```

```
239     @Override
240     protected void onSaveInstanceState(Bundle outState){
241         super.onSaveInstanceState(outState);
242         //Almacenamos el tiempo del cronometro.
243         long elapsedMillis = (SystemClock.elapsedRealtime() - cronometro.
                getBase());
244         //Almacenamos el estado del boton "Comenzar".
245         boolean pul_comenzar= estado_bcomenzar;
246         boolean pul_play= estado_bplay;
247         //Almacenamos la posicion de reproduccion del ejercicio.
248         int posicion= mVideoView.getCurrentPosition();
249         //Almacenamos el valor del contador de reproducciones.
250         int cont= contador_repet;
251         outState.putLong("crono", elapsedMillis);
252         outState.putInt("posicion", posicion);
253         outState.putInt("contador", cont);
254         outState.putBoolean("pulsado_comenzar", pul_comenzar);
255         outState.putBoolean("pulsado_play", pul_play);
256     }
257
258 }
```

C.7.2 reproductor_ejercicio.xml

```
1  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
2      android:id="@+id/RelativeLayout1"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent" >
5
6      <VideoView
7              android:id="@+id/surface_view"
8              android:layout_width="match_parent"
9              android:layout_height="match_parent"
10             android:layout_alignParentTop="true" />
11
12     <LinearLayout
13         android:id="@+id/botones1"
14         android:layout_width="match_parent"
15         android:layout_height="50dp"
16         android:layout_below="@id/surface_view"
17         android:background="#FFF"
18         android:orientation="horizontal" >
19
20         <ImageButton
21             android:id="@+id/play"
22             android:layout_width="wrap_content"
23             android:layout_height="match_parent"
24             android:layout_weight="1"
25             android:src="@drawable/play" />
```

```
26
27          <ImageButton
28              android:id="@+id/stop"
29              android:layout_width="wrap_content"
30              android:layout_height="match_parent"
31              android:layout_weight="1"
32              android:src="@drawable/stop" />
33
34          <Button
35              android:id="@+id/siguiente"
36              android:layout_width="wrap_content"
37              android:layout_height="match_parent"
38              android:layout_weight="2"
39              android:text="@string/sig"
40              android:textColor="#00F"
41              android:textStyle="bold" />
42      </LinearLayout>
43
44      <LinearLayout
45          android:id="@+id/botones2"
46          android:layout_width="match_parent"
47          android:layout_height="wrap_content"
48          android:layout_below="@id/botones1"
49          android:background="#FFF"
50          android:orientation="horizontal" >
51
52          <Button
53              android:id="@+id/comenzar"
54              android:layout_width="wrap_content"
55              android:layout_height="wrap_content"
56              android:layout_weight="1"
57              android:background="#0F0"
58              android:text="@string/comenzar"
59              android:textStyle="bold" />
60
61          <Button
62              android:id="@+id/parar"
63              android:layout_width="wrap_content"
64              android:layout_height="wrap_content"
65              android:layout_weight="1"
66              android:background="#F00"
67              android:text="@string/terminado"
68              android:textStyle="bold" />
69      </LinearLayout>
70
71      <LinearLayout
72          android:id="@+id/crono"
73          android:layout_width="match_parent"
74          android:layout_height="wrap_content"
75          android:layout_below="@id/botones2"
76          android:layout_margin="4dp"
77          android:background="@drawable/fondolistrutinas"
```

```
78          android:orientation="horizontal" >
79
80          <Chronometer
81              android:id="@+id/cronometro"
82              android:layout_width="wrap_content"
83              android:layout_height="wrap_content"
84              android:layout_margin="10sp"
85              android:layout_weight="3"
86              android:format="%s"
87              android:textSize="25sp" />
88
89          <TextView
90              android:id="@+id/repeticiones"
91              android:layout_width="wrap_content"
92              android:layout_height="wrap_content"
93              android:layout_margin="10sp"
94              android:layout_weight="1"
95              android:text="@string/repet"
96              android:textColor="#F00"
97              android:textSize="20sp"
98              android:textStyle="bold" />
99      </LinearLayout>
100
101     <LinearLayout
102         android:layout_width="match_parent"
103         android:layout_height="wrap_content"
104         android:layout_alignParentLeft="true"
105         android:layout_below="@id/crono"
106         android:layout_marginBottom="4dp"
107         android:layout_marginLeft="4dp"
108         android:layout_marginRight="4dp"
109         android:background="@drawable/fondolistrutinas"
110         android:orientation="vertical" >
111
112         <TextView
113             android:id="@+id/descripcion"
114             android:layout_width="match_parent"
115             android:layout_height="wrap_content"
116             android:layout_marginBottom="10sp"
117             android:layout_marginLeft="10sp"
118             android:layout_marginRight="10sp"
119             android:text="@string/descripcion"
120             android:textStyle="bold"
121             android:textSize="18sp" />
122
123         <ScrollView
124             android:layout_width="wrap_content"
125             android:layout_height="wrap_content" >
126
127             <TextView
128                 android:id="@+id/_descripcion"
129                 android:layout_width="wrap_content"
```

```
130                 android:layout_height="wrap_content"
131                 android:layout_marginBottom="10sp"
132                 android:layout_marginLeft="10sp"
133                 android:layout_marginRight="10sp" />
134         </ScrollView>
135     </LinearLayout>
136
137 </RelativeLayout>
```

C.8 Familia AcercaDe...

C.8.1 AcercaDe.java

```
1  package com.deporte.proyectofc;
2
3  /*
4   * Clase que muestra la actividad AcercaDe, que muestra informacion de
        interes para el usuario sobre la aplicacion.
5   */
6
7  import android.app.Activity;
8  import android.os.Bundle;
9
10 public class AcercaDe extends Activity{
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState){
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.acerca_de);
16     }
17 }
```

C.8.2 acerca_de

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/textView1"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_margin="25sp"
```

```
12          android:text="@string/acerca_de"
13          android:textAppearance="?android:attr/textAppearanceMedium" />
14
15      <TextView
16          android:id="@+id/textView2"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:layout_marginBottom="20sp"
20          android:layout_marginLeft="20sp"
21          android:layout_marginRight="20sp"
22          android:text="@string/acercade_mensaje" />
23
24  </LinearLayout>
```

C.9 Familia GraficasRemdimiento

C.9.1 GraficasRendimiento.java

El código Java de esta familia está compuesto por dos clases:

- GraficasRendimiento.java: Se encarga de realizar todos los cáculos para pintar correctamente la gráfica, y ademas incluye el elemento ListView.
- AdaptadorListRendimiento.java: Adaptador del ListView de la clase anterior, que adapta el contenido de cada item.

GraficasRendimiento.java

```
1  package com.deporte.proyectofc;
2
3  /*
4   * Clase encargada de representar la gráfica sobre el tiempo dedicado a
        hacer ejercicio durante la semana, medido en minutos.
5   */
6
7  import java.text.DateFormat;
8  import java.text.FieldPosition;
9  import java.text.Format;
10 import java.text.ParsePosition;
11 import java.text.SimpleDateFormat;
12 import java.util.Arrays;
13 import java.util.Calendar;
14 import java.util.Date;
15 import java.util.GregorianCalendar;
16
17 import com.androidplot.xy.LineAndPointFormatter;
```

```
18 import com.androidplot.xy.SimpleXYSeries;
19 import com.androidplot.xy.XYPlot;
20 import com.androidplot.xy.XYSeries;
21 import com.androidplot.xy.XYStepMode;
22
23 import android.app.ListActivity;
24 import android.graphics.Color;
25 import android.os.Bundle;
26 import android.widget.ListAdapter;
27 import android.widget.ListView;
28
29 public class GraficasRendimiento extends ListActivity{
30
31     private XYPlot migrafica;
32     private Date dia;
33     private ListView list;
34
35     @Override
36     protected void onCreate(Bundle savedInstanceState) {
37         //Mostramos el layout
38         super.onCreate(savedInstanceState);
39         setContentView(R.layout.graficas_rendimiento);
40         migrafica= (XYPlot) findViewById(R.id.migrafica);
41
42         //nombre de los ejes
43         migrafica.setDomainLabel("Dia de la semana");
44         migrafica.setRangeLabel("Minutos de entreno");
45
46         //Le asignamos al eje horizontal las etiquetas de los dias de la
               semana
47         migrafica.getGraphWidget().setDomainValueFormat(new
               GraphXLabelFormat());
48         //Ponemos el color fondo blanco
49         migrafica.getGraphWidget().getGridBackgroundPaint().setColor(
               Color.WHITE);
50         //Subtividimos el eje horizontal en 7 partes correspondientes a
               los dias de la semana
51         migrafica.setDomainStep(XYStepMode.SUBDIVIDE, 7);
52         LineAndPointFormatter series1Format = new LineAndPointFormatter(
53                 Color.rgb(0, 128, 255),             // Color de la línea
54                 Color.rgb(0, 76, 153),              // Color del punto
55                 null, null);                  // Relleno
56
57
58         XYSeries serie;
59         BasedatosSQLite basedatos= new BasedatosSQLite(this);
60         //id del usuario
61         AlmacenId id= new AlmacenId(this);
62         int id_usuario= id.obtenerId();
63         //fecha de hoy
64         dia= new Date();
65         GregorianCalendar cal= new GregorianCalendar();
```

```
66          cal.setTime(dia);
67
68          //dia de la semana de hoy
69          int dia_semana= cal.get(Calendar.DAY_OF_WEEK);
70
71          //fecha de hoy en String
72          DateFormat formatodate= new SimpleDateFormat("yyyy/MM/dd");
73          String date= formatodate.format(dia);
74
75          //Inicializamos los datos de los ejes
76          Number[] diasemana= {1.1, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1};
77          Number[] datos_semana_ej= {0, 0, 0, 0, 0, 0, 0};
78
79          Number[] minutos= {0,0,0,0,0,0,0};
80          Number[] segundos= {0,0,0,0,0,0,0};
81
82          int duracion_entreno= 0;
83          int pos= 0;
84          //Bucle que va asignando los valores a cada dia de la semana
                desde el dia de hoy hacia atras. Cuando llega al domingo
                termina de representar.
85          do{
86              //Obtenemos el tiempo de entrenamiento del dia date
87              duracion_entreno= basedatos.obtenerTiempos(id_usuario, date);
88              pos= diaPosicion(dia_semana);
89              datos_semana_ej[pos]= tiempo_transcurrido(duracion_entreno);
90              minutos[pos]= tiempo_min_seg(duracion_entreno)[0];
91              segundos[pos]= tiempo_min_seg(duracion_entreno)[1];
92              //Asignamos a cal un dia menos
93              cal.add(Calendar.DATE, -1);
94              dia_semana= cal.get(Calendar.DAY_OF_WEEK);
95              dia= cal.getTime();
96              date= formatodate.format(dia);
97          }while(dia_semana!=Calendar.SUNDAY);
98
99          //Asigna los valores a representar en la grafica
100         serie= new SimpleXYSeries(Arrays.asList(diasemana),
101                 Arrays.asList(datos_semana_ej), "Rendimiento de la semana"
                        );
102
103         //Asigna a la grafica los valores y el formato anteriormente
                definido
104         migrafica.addSeries(serie, series1Format);
105         dia= new Date();
106         cal.setTime(dia);
107         dia_semana= cal.get(Calendar.DAY_OF_WEEK);
108         //fecha de hoy
109         date= formatodate.format(dia);
110         migrafica.redraw();
111
112         migrafica.calculateMinMaxVals();
113
```

```
114         //Debajode la grafica incluimos el listView con los datos de la
                grafica por escrito
115         list= (ListView) findViewById(android.R.id.list);
116         list.setItemsCanFocus(false);
117         String[] dias_semana= {"Lunes","Martes","Miércoles","Jueves","
                Viernes","Sabado","Domingo"};
118         ListAdapter adaptador= new AdaptadorListRendimiento(this,
                dias_semana, minutos, segundos);
119         list.setAdapter(adaptador);
120     }
121
122
123     //Pasa el tiempo de segundos a minutos con decimales.
124     public double tiempo_transcurrido(int segundos){
125         int seg=0, añadir=0;
126         double min= 0;
127         if(segundos>=60){
128             min= segundos/60;
129             seg= segundos%60;
130             añadir= seg/60;
131             min= (double)(min+añadir);
132         }
133         else{
134             min= (double)segundos/60;
135         }
136         System.out.println("min " + min);
137         return min;
138     }
139
140
141     //Pasa se segundos a minutos y segundos
142     public int[] tiempo_min_seg(int segundos){
143         int[] tiempo= new int[2];
144         if(segundos>=60){
145             tiempo[0]= segundos/60;
146             tiempo[1]= segundos%60;
147         }
148         else{
149             tiempo[0]= 0;
150             tiempo[1]= segundos;
151         }
152         return tiempo;
153     }
154
155     //Asigna una posicion numerica para guardarlo en el array
156     public int diaPosicion(int dia){
157         int posicion=0;
158         switch (dia){
159             case Calendar.MONDAY: posicion=0;
160             break;
161             case Calendar.TUESDAY: posicion=1;
162             break;
```

```
163             case Calendar.WEDNESDAY: posicion=2;
164             break;
165             case Calendar.THURSDAY: posicion=3;
166             break;
167             case Calendar.FRIDAY: posicion=4;
168             break;
169             case Calendar.SATURDAY: posicion=5;
170             break;
171             default : posicion=6;
172             break;
173         }
174         return posicion;
175     }
176
177
178     //Para represantar en la abcisa los dias de la semana.
179     private class GraphXLabelFormat extends Format{
180         private final String[] LABELS = {"L", "M", "X", "J", "V", "S", "D"
                };
181
182         @Override
183         public StringBuffer format(Object object, StringBuffer buffer,
                FieldPosition field) {
184             int parsedInt = Math.round(Float.parseFloat(object.toString())
                    );
185             String labelString = LABELS[parsedInt-1];
186
187             buffer.append(labelString);
188             return buffer;
189         }
190
191         @Override
192         public Object parseObject(String string, ParsePosition position) {
193             return java.util.Arrays.asList(LABELS).indexOf(string);
194         }
195     }
196 }
```

AdaptadorListRendimiento.java

```
1  package com.deporte.proyectofc;
2
3  /* Clase encargada de definir el adaptador para mostrar los minutos de
       cada dia de la semana que realiza ejercicio un usuario. */
4
5  import android.app.Activity;
6  import android.view.LayoutInflater;
7  import android.view.View;
8  import android.view.ViewGroup;
9  import android.widget.BaseAdapter;
10 import android.widget.TextView;
```

```
11
12 public class AdaptadorListRendimiento extends BaseAdapter{
13
14     private Activity actividad;
15     private String[] dias_semana;
16     private Number[] minutos;
17     private Number[] segundos;
18     private TextView dia_sem;
19     private TextView min;
20
21     public AdaptadorListRendimiento(Activity actividad, String[]
           dias_semana, Number[] minutos, Number[] segundos){
22         super();
23         this.actividad= actividad;
24         this.dias_semana= dias_semana;
25         this.minutos= minutos;
26         this.segundos= segundos;
27     }
28
29     //Genera la vista de cada item
30     public View getView(int position, View convertView, ViewGroup parent){
31
32         if(convertView==null){
33             LayoutInflater inflater = actividad.getLayoutInflater();
34             convertView= inflater.inflate(R.layout.entrada_grafica_rend,
                   null, true);
35         }
36
37         dia_sem= (TextView) convertView.findViewById(R.id.diasemana);
38         dia_sem.setFocusable(false);
39         min= (TextView) convertView.findViewById(R.id.minutos);
40         min.setFocusable(false);
41         dia_sem.setText(dias_semana[position]);
42         //le asignamos los minutos de ejercicio de ese dia de la semana.
43         min.setText("Duración del entranamiento:" + "\n" + minutos[
               position] + " min ," + segundos[position] + " seg.");
44
45         return convertView;
46     }
47
48     public int getCount() {
49         return dias_semana.length;
50     }
51
52     public Object getItem(int arg0) {
53         return dias_semana[arg0];
54     }
55
56     public long getItemId(int position) {
57         return position;
58     }
59 }
```

C.9.2 listarendimiento.xml

De nuevo, como la vista incorpora una lista, precisaremos de dos XML, graficas_-rendimiento.xml y entrada_grafica_rend.xml.

graficas_rendimiento.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:layout_gravity="left"
        android:textSize="18sp"
        android:text="@string/mis_estadisticas" />

    <com.androidplot.xy.XYPlot
        android:id="@+id/migrafica"
        android:layout_width="match_parent"
        android:layout_height="260dp"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:layout_marginTop="5dp"
        title="Rendimiento semanal"
        android:background="@android:color/transparent">
    </com.androidplot.xy.XYPlot>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textStyle="bold"
        android:text="@string/t_entreno"
        android:gravity="center"/>
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:focusable="false"
        android:background="@drawable/fondolistrutinas"
        android:layout_margin="6dp"
        android:padding="6dp">
        <ListView
```

```
44            android:id="@android:id/list"
45            android:layout_width="match_parent"
46            android:layout_height="wrap_content"
47            android:drawSelectorOnTop="false">
48
49        </ListView>
50        <TextView
51            android:id="@android:id/empty"
52            android:layout_width="match_parent"
53            android:layout_height="50dp"
54            android:textSize="16sp"
55            android:textStyle="bold"
56            android:text="@string/error_list_g" />
57
58    </FrameLayout>
59 </LinearLayout>
```

entrada_grafica_rend.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/diasemana"
9          android:layout_width="match_parent"
10         android:layout_height="wrap_content"
11         android:layout_margin="6dp"
12         android:text="@string/dia_sem"
13         android:textStyle="bold" />
14
15     <LinearLayout
16         android:layout_width="match_parent"
17         android:layout_height="wrap_content"
18         android:orientation="horizontal" >
19
20         <TextView
21             android:id="@+id/minutos"
22             android:layout_width="match_parent"
23             android:layout_height="wrap_content"
24             android:layout_margin="3dp"
25             android:layout_marginTop="6dp"
26             android:text="@string/tiempo"
27             android:textSize="20sp" />
28     </LinearLayout>
29
30 </LinearLayout>
```

C.10 Familia ConfiguracionCuenta

Pantalla que muestra las opciones que hay sobre configuración de la cuenta.

C.10.1 ConfiguracionCuenta.java

```
1  package com.deporte.proyectofc;
2
3  /*
4   * Clase encargada de visualizar la actividad de configuracion. Se podra
        modificar varios aspectos relacionados con los datos de usuario, que
         se habian introducido en su registro.
5   */
6
7  import android.app.Activity;
8  import android.content.Intent;
9  import android.os.Bundle;
10 import android.view.View;
11
12 public  class  ConfiguracionCuenta extends Activity{
13
14     @Override
15     protected  void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.configuracion_cuenta);
18     }
19
20     /*Lanzamos la actividad para modificar algun dato del usuario.*/
21     public  void lanzarModificacion(View view){
22         Intent i = new Intent(this, ModificacionCuenta.class);
23         startActivity(i);
24     }
25
26     /*Metodo activado cuando se detecta pulsacion sobre el boton cambiar
           contraseña. Lanzamos ventana para la modificación de la contraseñ
           a.*/
27     public  void lanzarModificaContrasenia(View view){
28         Intent i= new Intent(this, ModificacionContrasenia.class);
29         startActivity(i);
30     }
31
32     /*Metodo lanzado cuando se pulsa sobre el boton eliminar cuenta.
           Elimina de forma permanente la actual cuenta de usuario.*/
33     public  void eliminarCuenta(View view){
34
35         BasedatosSQLite basedatos= new BasedatosSQLite(this);
36         AlmacenId valor_id= new AlmacenId(this);
37         //Eliminamos el usuario cuyo id es el guardado en el archivo
               preferencias
```

```
38          basedatos.eliminarUsuario(valor_id.obtenerId());
39          //Una vez eliminado el usuario, volvemos a la actividad principal,

40          //vaciando la pila de actividades
41          Intent intent= new Intent(this, MainActivity.class);
42          intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
43          startActivity(intent);
44      }
45
46  }
```

C.10.2 configuracion_cuenta.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/textView1"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_gravity="center"
12         android:layout_marginBottom="30sp"
13         android:layout_marginTop="20sp"
14         android:text="@string/config_cuenta"
15         android:textAppearance="?android:attr/textAppearanceLarge" />
16
17     <TextView
18         android:id="@+id/textView2"
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content"
21         android:layout_margin="12sp"
22         android:text="@string/modifica_info"
23         android:textAppearance="?android:attr/textAppearanceSmall" />
24
25     <Button
26         android:id="@+id/button1"
27         android:layout_width="match_parent"
28         android:layout_height="wrap_content"
29         android:layout_gravity="center"
30         android:layout_marginLeft="10dp"
31         android:layout_marginRight="10dp"
32         android:background="@drawable/buttongen"
33         android:onClick="lanzarModificacion"
34         android:text="@string/bmodificar_info" />
35
36     <TextView
37         android:id="@+id/textView3"
```

```
38          android:layout_width="wrap_content"
39          android:layout_height="wrap_content"
40          android:layout_margin="12sp"
41          android:text="@string/cambia_pass"
42          android:textAppearance="?android:attr/textAppearanceSmall" />
43
44      <Button
45          android:id="@+id/button3"
46          android:layout_width="match_parent"
47          android:layout_height="wrap_content"
48          android:layout_marginLeft="10dp"
49          android:layout_marginRight="10dp"
50          android:background="@drawable/buttongen"
51          android:onClick="lanzarModificaContrasenia"
52          android:text="@string/bcambiar_pass" />
53
54      <TextView
55          android:id="@+id/textView4"
56          android:layout_width="wrap_content"
57          android:layout_height="wrap_content"
58          android:layout_margin="12sp"
59          android:text="@string/elimina_cuenta"
60          android:textAppearance="?android:attr/textAppearanceSmall" />
61
62      <Button
63          android:id="@+id/button2"
64          android:layout_width="match_parent"
65          android:layout_height="wrap_content"
66          android:layout_gravity="center"
67          android:layout_marginLeft="10dp"
68          android:layout_marginRight="10dp"
69          android:background="@drawable/buttongen"
70          android:onClick="eliminarCuenta"
71          android:text="@string/beliminar_cuenta" />
72
73  </LinearLayout>
```

C.11 Familia ModificacionCuenta

C.11.1 ModificacionCuenta.java

```
1  package com.deporte.proyectofc;
2
3  /* Modifica los datos personales de la cuenta del usuario. */
4
5  import java.util.ArrayList;
6  import android.app.Activity;
```

```
7  import android.os.Bundle;
8  import android.view.View;
9  import android.widget.ArrayAdapter;
10 import android.widget.EditText;
11 import android.widget.Spinner;
12 import android.widget.Toast;
13
14 public class ModificacionCuenta extends Activity{
15
16     private EditText nombre;
17     private EditText apellidos;
18     private EditText e_mail;
19     private Spinner estadodeforma;
20     private static int id_usuario= 0;
21     private BasedatosSQLite basedatos;
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25
26         super.onCreate(savedInstanceState);
27         //Visualizamos
28         setContentView(R.layout.modificacion_usuario);
29
30         nombre= (EditText) findViewById(R.id._nombre);
31         apellidos= (EditText) findViewById(R.id._apellidos);
32         e_mail= (EditText) findViewById(R.id._e_mail);
33         estadodeforma= (Spinner) findViewById(R.id._estado);
34         basedatos= new BasedatosSQLite(this);
35         AlmacenId id_u= new AlmacenId(this);
36         id_usuario= id_u.obtenerId();
37         //Obtenemos toda la infromación del usuario
38         ArrayList<String> datos= basedatos.datosUsuario(id_usuario);
39         //Ponemos los datos que tenemos actualmente
40         nombre.setText(datos.get(4));
41         apellidos.setText(datos.get(5));
42         e_mail.setText(datos.get(6));
43         String[] valores = {"Bajo","Medio","Alto"};
44         estadodeforma.setAdapter(new ArrayAdapter<String>(this,
45                 android.R.layout.simple_spinner_item, valores));
46         estadodeforma.setSelection(seleccion(datos.get(7)));
47     }
48
49     public void guardarModificacion(View view){
50
51         String v_nombre= nombre.getText().toString();
52         String v_apellidos= apellidos.getText().toString();
53         String v_e_mail= e_mail.getText().toString();
54         String v_estadodeforma= estadodeforma.getSelectedItem().toString
               ();
55         if(v_nombre.isEmpty() || v_apellidos.isEmpty() || v_e_mail.
               isEmpty()){
```

```
56              Toast toast= Toast.makeText(getApplicationContext(), "Rellene
                    todos los campos.", Toast.LENGTH_SHORT);
57              toast.show();
58          }
59          else{
60              //Guardamos la modificación.
61              basedatos.modificarUsuario(id_usuario, v_nombre, v_apellidos,
                    v_e_mail, v_estadodeforma);
62              Toast toast= Toast.makeText(getApplicationContext(),
63                      "Datos modificados satisfactoriamente", Toast.
                            LENGTH_SHORT);
64              toast.show();
65          }
66      }
67
68      //Asigna una posicion del spinner a cada String
69      public int seleccion(String estado){
70          int resultado;
71          if((estado.compareTo("Bajo"))==0) resultado= 0;
72          else if((estado.compareTo("Medio"))==0) resultado= 1;
73          else if((estado.compareTo("Alto"))==0) resultado= 2;
74          else resultado= 0;
75
76          return resultado;
77      }
78
79  }
```

C.11.2 modificacion_usuario.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/nombre"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_margin="8dp"
12         android:text="@string/nombre" />
13
14     <EditText
15         android:id="@+id/_nombre"
16         android:layout_width="match_parent"
17         android:layout_height="40dp"
18         android:layout_marginBottom="6dp"
19         android:layout_marginLeft="8dp"
20         android:layout_marginRight="8dp"
```

```
21          android:background="@drawable/edittextgen"
22          android:ems="10"
23          android:paddingLeft="8dp"
24          android:paddingRight="8dp" >
25
26          <requestFocus />
27      </EditText>
28
29      <TextView
30          android:id="@+id/apellidos"
31          android:layout_width="wrap_content"
32          android:layout_height="wrap_content"
33          android:layout_margin="8dp"
34          android:text="@string/apellidos" />
35
36      <EditText
37          android:id="@+id/_apellidos"
38          android:layout_width="match_parent"
39          android:layout_height="40dp"
40          android:layout_marginBottom="6dp"
41          android:layout_marginLeft="8dp"
42          android:layout_marginRight="8dp"
43          android:background="@drawable/edittextgen"
44          android:ems="10"
45          android:paddingLeft="8dp"
46          android:paddingRight="8dp" />
47
48      <TextView
49          android:id="@+id/e_mail"
50          android:layout_width="wrap_content"
51          android:layout_height="wrap_content"
52          android:layout_margin="8dp"
53          android:text="@string/e_mail" />
54
55      <EditText
56          android:id="@+id/_e_mail"
57          android:layout_width="match_parent"
58          android:layout_height="40dp"
59          android:layout_marginBottom="6dp"
60          android:layout_marginLeft="8dp"
61          android:layout_marginRight="8dp"
62          android:background="@drawable/edittextgen"
63          android:ems="10"
64          android:inputType="textEmailAddress"
65          android:paddingLeft="8dp"
66          android:paddingRight="8dp" />
67
68      <LinearLayout
69          android:layout_width="match_parent"
70          android:layout_height="wrap_content"
71          android:layout_margin="6dp"
72          android:orientation="horizontal" >
```

```
73
74          <TextView
75              android:id="@+id/estado"
76              android:layout_width="wrap_content"
77              android:layout_height="wrap_content"
78              android:layout_margin="6dp"
79              android:text="@string/nivel"
80              android:textSize="16sp" />
81
82          <Spinner
83              android:id="@+id/_estado"
84              android:layout_width="0dp"
85              android:layout_height="wrap_content"
86              android:layout_marginLeft="6dp"
87              android:layout_marginRight="6dp"
88              android:layout_marginTop="12dp"
89              android:layout_weight="1"
90              android:background="@android:drawable/btn_dropdown" />
91      </LinearLayout>
92
93      <Button
94          android:id="@+id/button1"
95          android:layout_width="match_parent"
96          android:layout_height="wrap_content"
97          android:layout_margin="10dp"
98          android:onClick="guardarModificacion"
99          android:text="@string/modificar" />
100
101 </LinearLayout>
```

C.12 Familia ModificaciónContraseña

C.12.1 ModificacionContrasenia.java

```
1  package com.deporte.proyectofc;
2
3  /* Modifica la contraseña del usuario. */
4
5  import java.util.ArrayList;
6  import android.app.Activity;
7  import android.os.Bundle;
8  import android.view.View;
9  import android.widget.EditText;
10 import android.widget.Toast;
11
12 public class ModificacionContrasenia extends Activity{
13
```

```
14    @Override public void onCreate(Bundle savedInstanceState){
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.modificacion_contrasenia);
17    }
18
19    //Si pulsamos sobre el boton guardar
20    public void guardarNuevaContrasenia(View view){
21
22        BasedatosSQLite basedatos= new BasedatosSQLite(this);
23        AlmacenId id_u= new AlmacenId(this);
24        ArrayList<String> datos= basedatos.datosUsuario(id_u.obtenerId());

25        EditText contrasenia1= (EditText) findViewById(R.id.
              _NContrasenia1);
26        EditText contrasenia2= (EditText) findViewById(R.id.
              _NContrasenia2);
27        //Guardamos ambos String indroducidos en los EditText
28        String vcontrasenia1= contrasenia1.getText().toString();
29        String vcontrasenia2= contrasenia2.getText().toString();
30
31        //Comprueba que ambas cadenas introducidas son identicas
32        if(vcontrasenia1.compareTo(vcontrasenia2)==0){
33            byte[] salt= EncriptadoContraseña.fromHexString(datos.get(3));
34            //Calcula el nuevo hash con los valores salt(que es el mismo)
                  y la nueva contraseña
35            String nuevohash= EncriptadoContraseña.hash(vcontrasenia1,
                  salt);
36            //Comprobamos que la nueva contraseña es distinta a la que hay
                   actualmente
37            if(!datos.get(2).equals(nuevohash)){
38                System.out.println("es distinta");
39                //Procedemos a guardar la nueva contraseña en el lugar de
                      la antigua
40                basedatos.modificaContrasenia(id_u.obtenerId(), nuevohash)
                      ;
41                Toast toast= Toast.makeText(getApplicationContext(),
42                        "La contraseña ha sido modificada.", Toast.
                              LENGTH_SHORT);
43                toast.show();
44            }
45            //Volvemos a la pantalla anterior
46            finish();
47        }
48        else{
49            //Si ambas contraseñas son distintas, no se puede proceder al
                  cambio y mostramos un mensaje informando de la situación
50            Toast toast= Toast.makeText(getApplicationContext(),
51                    "Error.Las contraseñas no coinciden.", Toast.
                          LENGTH_SHORT);
52            toast.show();
53        }
54    }
```

```
55 }
```

C.12.2 modificacion_contrasenia.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/NContrasenia1"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="@string/intro_nueva_c" />
12
13     <EditText
14         android:id="@+id/_NContrasenia1"
15         android:layout_width="match_parent"
16         android:layout_height="wrap_content"
17         android:ems="10"
18         android:inputType="textPassword" >
19
20         <requestFocus />
21     </EditText>
22
23     <TextView
24         android:id="@+id/NContrasenia2"
25         android:layout_width="wrap_content"
26         android:layout_height="wrap_content"
27         android:text="@string/repita_c" />
28
29     <EditText
30         android:id="@+id/_NContrasenia2"
31         android:layout_width="match_parent"
32         android:layout_height="wrap_content"
33         android:ems="10"
34         android:inputType="textPassword" />
35
36     <Button
37         android:id="@+id/guardar"
38         android:layout_width="match_parent"
39         android:layout_height="wrap_content"
40         android:onClick="guardarNuevaContrasenia"
41         android:text="@string/modificar_contrasenia" />
42
43 </LinearLayout>
```

C.13 Familia ListaMasRutinas de otros usuarios locales

En esta caso, aunque se trate de una lista, solo tendremos un código java y otro XML, porque utilizaremos un Adaptador ya definido por el sistema, creando un objeto ArrayAdapter<String>.

C.13.1 ListMasRutinas.java

```
1  package com.deporte.proyectofc;
2
3  /*
4   * Clase que se encarga de mostrar la lista de rutinas que el usuario aun
        no ha cargado pero que otro/s sí lo han hecho. Tambien se podra
       buscar nuevas rutinas que aun no han sido cargadas por nadie.
5   */
6
7  import java.io.File;
8  import java.util.ArrayList;
9  import android.app.ListActivity;
10 import android.content.Intent;
11 import android.database.sqlite.SQLiteDatabase;
12 import android.os.Bundle;
13 import android.os.Environment;
14 import android.view.View;
15 import android.widget.AdapterView;
16 import android.widget.ArrayAdapter;
17 import android.widget.ListView;
18
19 public  class  ListMasRutinas extends ListActivity{
20
21     private ListView lista;
22     private  int id_usuario;
23     private BasedatosSQLite basedatos;
24
25
26     @Override
27     public  void onCreate(Bundle savedInstanceState){
28
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.list_mas_rutinas);
31
32         //Mostramos las opciones de rutinas que no ha cargado este
               usuario pero que si lo han hecho otros usuarios locales
33         basedatos= new BasedatosSQLite(this);
34         AlmacenId _id= new AlmacenId(this);
35         id_usuario= _id.obtenerId();
36         ArrayList<Integer> idsmasrutinas= basedatos.obtenerMasRutinas(
               id_usuario);
37         ArrayList<String> nombresmasrutinas= new ArrayList<String>();
```

```
38          for(int i=0 ; i<idsmasrutinas.size() ; i++){
39              nombresmasrutinas.add(basedatos.obtenerNombreRutina(
                    idsmasrutinas.get(i)));
40          }
41          //Las visualizamos en el ListView
42          lista= (ListView) findViewById(android.R.id.list);
43          lista.setAdapter(new ArrayAdapter<String>(this, android.R.layout.
                simple_list_item_single_choice, nombresmasrutinas));
44          lista.setItemsCanFocus(false);
45          lista.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
46
47      }
48
49
50      //Metodo que incluye en la lista del usuario la nueva rutina
            seleccionada
51      public void lanzarAceptar(View view){
52
53          int position= lista.getCheckedItemPosition();
54          if( position != AdapterView.INVALID_POSITION){
55              String nombre_rutina= lista.getItemAtPosition(position).
                    toString();
56
57              //Obtenemos el identificador de rutina a partir del nombre
58              SQLiteDatabase bd= basedatos.creaSQLitedatabase();
59              int id_rutina= basedatos.obtenerIdRutina(bd, nombre_rutina);
60              bd.close();
61              //Lo introducimos en la tabla usuario_rutina
62              basedatos.introRutinaUsuario(id_usuario, id_rutina);
63              finish();
64          }
65
66      }
67
68
69      //Metodo que busca en la carpeta dowload nuevas rutinas descargadas
70      public void lanzarCargarNuevas(View view){
71
72          ArrayList<String> path_rutinas_zip= new ArrayList<String>();
73          ArrayList<String> nom_rutinas_zip= new ArrayList<String>();
74          String path_dowload= (Environment.
                getExternalStoragePublicDirectory(Environment.
                DIRECTORY_DOWNLOADS).
                    getAbsolutePath()).toString();
75          File f= new File(path_dowload);
76          File[] ficheros= f.listFiles();
77
78          //Recorremos toda la lista de ficheros que contiene dowload
79          for(int cont=0 ; cont<ficheros.length ; cont++){
80              //Si el fichero acaba en .zip, lo añadimos a la lista de
                    elecciones
81              //posibles
82
```

```
83              if(ficheros[cont].getName().endsWith(".zip")){
84                  int pos= ficheros[cont].getName().indexOf(".zip");
85                  String cargada= ficheros[cont].getName().substring(0, pos)
                        ;
86                  //Comprobamos que esta rutina no esta cargada ya
87                  if(!basedatos.existeRutina(cargada)){
88                      path_rutinas_zip.add(ficheros[cont].getAbsolutePath());

89                      nom_rutinas_zip.add(ficheros[cont].getName());
90                  }
91              }
92          }
93          Intent i= new Intent(this, ListNuevasRutinas.class);
94          //Comunicamos a la siguiente actividad los nombre y rutas de los .
                zip
95          i.putStringArrayListExtra("pathrutinas", path_rutinas_zip);
96          i.putStringArrayListExtra("nombresrutinas", nom_rutinas_zip);
97          startActivity(i);
98
99      }
100
101     //Metodo para recargar la actividad
102     @Override public void onRestart(){
103         super.onRestart();
104         finish();
105         startActivity(getIntent());
106     }
107
108 }
```

C.13.2 list_mas_rutinas.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:layout_width="match_parent"
9          android:layout_height="wrap_content"
10         android:layout_margin="20sp"
11         android:text="@string/otras"
12         android:textSize="20sp"
13         android:textStyle="bold"
14         android:textColor="#FFF" />
15
16     <TextView
17         android:layout_width="match_parent"
18         android:layout_height="wrap_content"
```

```
19          android:layout_marginLeft="20sp"
20          android:layout_marginRight="20sp"
21          android:layout_marginBottom="20sp"
22          android:text="@string/otras_rutinas"
23          android:textSize="18sp"/>
24
25      <FrameLayout
26          android:layout_width="match_parent"
27          android:layout_height="wrap_content"
28          android:layout_margin="6dp"
29          android:background="@drawable/fondolistrutinas"
30          android:focusable="false"
31          android:padding="6dp" >
32
33          <ListView
34              android:id="@android:id/list"
35              android:layout_width="match_parent"
36              android:layout_height="wrap_content"
37              android:drawSelectorOnTop="false" >
38          </ListView>
39
40          <TextView
41              android:id="@android:id/empty"
42              android:layout_width="match_parent"
43              android:layout_height="50dp"
44              android:text="@string/error_list_o"
45              android:textSize="16sp"
46              android:textStyle="bold" />
47      </FrameLayout>
48
49      <Button
50          android:id="@+id/aceptar"
51          android:layout_width="match_parent"
52          android:layout_height="wrap_content"
53          android:layout_margin="6dp"
54          android:background="@drawable/buttongen"
55          android:onClick="lanzarAceptar"
56          android:text="@string/aceptar" />
57
58      <Button
59          android:id="@+id/cargarmas"
60          android:layout_width="match_parent"
61          android:layout_height="wrap_content"
62          android:layout_margin="6dp"
63          android:background="@drawable/buttongen"
64          android:onClick="lanzarCargarNuevas"
65          android:text="@string/bcargar_nuevas" />
66
67  </LinearLayout>
```

C.14 Familia ListaNuevasRutinas de Download

De nuevo, al igual que en la familia anterior, utilizaremos un adaptador definido por Android.

C.14.1 ListNuevasRutinas.java

```
1  package com.deporte.proyectofc;
2
3  /* Clase que muestra una lista de nuevas rutinas comprimidas en la
       carpeta Download. */
4
5  import java.io.File;
6  import java.io.FileInputStream;
7  import java.io.FileNotFoundException;
8  import java.io.FileOutputStream;
9  import java.io.IOException;
10 import java.util.ArrayList;
11 import java.util.zip.ZipEntry;
12 import java.util.zip.ZipInputStream;
13
14 import android.app.Activity;
15 import android.app.ListActivity;
16 import android.app.ProgressDialog;
17 import android.content.Context;
18 import android.database.sqlite.SQLiteDatabase;
19 import android.os.AsyncTask;
20 import android.os.Bundle;
21 import android.os.Environment;
22 import android.view.View;
23 import android.widget.AdapterView;
24 import android.widget.AdapterView.OnItemClickListener;
25 import android.widget.ArrayAdapter;
26 import android.widget.ListView;
27 import android.widget.Toast;
28
29 public class ListNuevasRutinas extends ListActivity{
30
31     private ListView lista;
32     private ArrayList<String> path_rutinas_zip;
33     private ArrayList<String> nom_rutinas_zip;
34     private String path;
35
36     //Metodo que muestra una lista con los .zip que podemos incluir en
          nuestra app
37     @Override
38     public void onCreate(Bundle savedInstanceState){
39
40         super.onCreate(savedInstanceState);
```

```
41          setContentView(R.layout.list_nuevas_rutinas);
42          Bundle extras= getIntent().getExtras();
43          path_rutinas_zip= extras.getStringArrayList("pathrutinas");
44          nom_rutinas_zip= extras.getStringArrayList("nombresrutinas");
45
46          lista= (ListView) findViewById(android.R.id.list);
47          lista.setAdapter(new ArrayAdapter<String>(this, android.R.layout.
                simple_list_item_activated_1, nom_rutinas_zip));
48          lista.setItemsCanFocus(false);
49          lista.setSelector(android.R.color.background_light);
50
51          lista.setOnItemClickListener(new OnItemClickListener() {
52              @Override
53              public void onItemClick(AdapterView<?> a, View view, int
                    position, long id){
54                  //ruta del .zip seleccionado
55                  path= path_rutinas_zip.get(position);
56              }
57          });
58      }
59
60
61      /*Al aceptar una rutina comprimida, pasaremos a descomprimirla en
            nuestra carpeta de rutinas*/
62      public void cargarSeleccion(View view){
63          if(path!=null){
64              new CargaZip(this).execute(path);
65          }
66      }
67
68
69      public void cancelarSeleccion(View view){
70          //Al cancelar volvemos a la anterior actividad.
71          finish();
72      }
73  }
74
75
76  /*Clase que se encarga de descomprimir la carpeta zip cuya ruta se pasa
        como parametro. */
77  class CargaZip extends AsyncTask<String, Integer, String>{
78      private Context contexto;
79      private ProgressDialog progreso;
80      private BasedatosSQLite basedatos;
81      private String path_rutina_desc;
82
83
84      public CargaZip(Context contexto){
85          this.contexto= contexto;
86      }
87
88
```

```
89      //Metodo que contiene lo que se ejecuta en el hilo principal.
90      @Override protected void onPreExecute(){
91
92          progreso= new ProgressDialog(contexto);
93          progreso= ProgressDialog.show(contexto, "Cargando...", "Por favor
                espere", true);
94
95      }
96
97
98      //Metodo que contiene las acciones a realizar por el hilo secundario.
99      @Override protected String doInBackground(String... n){
100
101         String resultado= "correcto";
102         descomprimirZip(n[0]);
103
104         basedatos= new BasedatosSQLite(contexto);
105         SQLiteDatabase bd= basedatos.creaSQLitedatabase();
106         String nombre_rutina= path_rutina_desc.substring((Environment.
                getExternalStorageDirectory().getPath().toString() +
107                 "/app_deportes/rutinas/").length());
108         //Introducimos la nueva rutina en la tabla rutinas
109         basedatos.insertarRutina(bd, null, nombre_rutina, path_rutina_desc
                );
110         int id_rutina= basedatos.obtenerIdRutina(bd, nombre_rutina);
111         //Metemos los videos que contiene en la tabla video.
112         String descripcion_rutina= basedatos.introduceEjercicios(bd,
                id_rutina);
113         if(!descripcion_rutina.equals("Rutinainvalida")){
114
115             //Insertamos la descripcion que hemos obtenido de la rutina
                    tras leer el Json.
116             basedatos.insertarDescRut(bd, id_rutina, descripcion_rutina);
117             bd.close();
118             //Lo metemos en la tabla usuario_rutina
119             AlmacenId _id= new AlmacenId(contexto);
120             int _id_usuario= _id.obtenerId();
121             basedatos.introRutinaUsuario(_id_usuario, id_rutina);
122         }
123         else{
124             basedatos.borrarRutina(id_rutina);
125             eliminaCarpeta(path_rutina_desc);
126             resultado= "incorrecto";
127         }
128         return resultado;
129     }
130
131
132     @Override protected void onPostExecute(String v){
133         progreso.dismiss();
134         if(v.equals("incorrecto")){
```

```
135             Toast toast= Toast.makeText(contexto, "Rutina con formato invá
                    lido. Debe contener archivos mp4 y un archivo JSON.",
                    Toast.LENGTH_SHORT);
136             toast.show();
137         }
138         ((Activity)contexto).finish();
139     }
140
141
142     //Metodo que descomprime un archivo .zip que se encuentra en la ruta
            que se introduce como parametro, a un destino cuya ruta es el
            String "destino".
143     public void descomprimirZip(String path){
144
145         //Donde se va a descomprimir el zip
146         String destino= Environment.getExternalStorageDirectory().
                getAbsolutePath().toString() + "/app_deportes/rutinas/";
147         try{
148             ZipInputStream inputzip= new ZipInputStream(new
                    FileInputStream(path));
149             ZipEntry entrada= inputzip.getNextEntry();
150             while(entrada!=null){
151                 String filepath= destino + entrada.getName();
152                 if(!entrada.isDirectory()){
153                     //Si se trata de un archivo, lo extrae en destino
154                     extraefichero(inputzip, filepath);
155                     }
156                 else{
157                     File dir= new File(filepath);
158                     dir.mkdir();
159                 }
160                 inputzip.closeEntry();
161                 entrada= inputzip.getNextEntry();
162             }
163             inputzip.close();
164         }catch(FileNotFoundException e){
165             System.err.println(e);
166         }catch(IOException e){
167             System.err.println(e);
168         }
169
170     }
171
172
173     //Extrae un fichero del .zip a el destino indicado por el String path.
174     private void extraefichero(ZipInputStream inputzip, String path) throws
             IOException{
175
176         File newfile= new File(path);
177         if(!newfile.getParentFile().exists()){
178             new File(newfile.getParent()).mkdirs();
```

```
179         }
180         path_rutina_desc= newfile.getParentFile().getAbsolutePath();
181         FileOutputStream fos = new FileOutputStream(newfile);
182         byte[] bytesIn = new byte[1024];
183         int len;
184         while ((len = inputzip.read(bytesIn)) > 0) {
185             fos.write(bytesIn, 0, len);
186         }
187         fos.close();
188 
189     }
190 
191     public boolean eliminaCarpeta(String path_rutina){
192         File file= new File(path_rutina);
193         //Si existe lo borramos
194         if(file.exists()){
195             borrarfic(file);
196             if(file.delete()){
197                 return true;
198             }else return false;
199         }else return false;
200     }
201 
202     public void borrarfic(File directorio){
203         File[] ficheros= directorio.listFiles();
204         for(int i=0 ; i<ficheros.length ; i++){
205             if(ficheros[i].isDirectory()){
206                 borrarfic(ficheros[i]);
207             }
208             ficheros[i].delete();
209         }
210     }
211 
212 }
```

C.14.2 list_nuevas_rutinas.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6 
7      <TextView
8          android:layout_width="match_parent"
9          android:layout_height="wrap_content"
10         android:layout_margin="20dp"
11         android:text="@string/nuevas"
12         android:textSize="20sp"
13         android:textColor="#FFF"
```

```
14          android:textStyle="bold"/>
15
16      <TextView
17          android:layout_width="match_parent"
18          android:layout_height="wrap_content"
19          android:layout_marginLeft="20dp"
20          android:layout_marginRight="20dp"
21          android:layout_marginBottom="20dp"
22          android:textSize="18sp"
23          android:text="@string/nuevas_rutinas"/>
24
25      <FrameLayout
26          android:layout_width="match_parent"
27          android:layout_height="wrap_content"
28          android:focusable="false" >
29
30          <ListView
31              android:id="@android:id/list"
32              android:layout_width="match_parent"
33              android:layout_height="wrap_content" >
34          </ListView>
35
36          <TextView
37              android:id="@android:id/empty"
38              android:layout_width="match_parent"
39              android:layout_height="wrap_content"
40              android:text="@string/error_list_n" />
41      </FrameLayout>
42
43      <LinearLayout
44          android:layout_width="match_parent"
45          android:layout_height="wrap_content"
46          android:orientation="horizontal" >
47
48          <Button
49              android:id="@+id/cargar"
50              android:layout_width="match_parent"
51              android:layout_height="wrap_content"
52              android:layout_margin="2dp"
53              android:layout_weight="1"
54              android:background="@drawable/buttongen"
55              android:onClick="cargarSeleccion"
56              android:text="@string/bcargar" />
57
58          <Button
59              android:id="@+id/cancelar"
60              android:layout_width="match_parent"
61              android:layout_height="wrap_content"
62              android:layout_margin="2dp"
63              android:layout_weight="1"
64              android:background="@drawable/buttongen"
65              android:onClick="cancelarSeleccion"
```

```
66              android:text="@string/cancelar" />
67      </LinearLayout>
68
69  </LinearLayout>
```

C.15 Clase AlmacenId

Esta clase define un fichero de preferencias donde se almacenará el identificador de usuario durante una sesión de la aplicación. Hacemos esto porque el id del usuario es un valor al que accedemos continuamente y en cada una de las pantallas.

C.15.1 AlmacenId.java

```
1  package com.deporte.proyectofc;
2
3  /*
4   * Clase que guarda de manera permanente el valor id del usuario que ha
        iniciado sesion en un archivo de preferencias. Este valor es muy
        usado durante toda la sesion de la aplicacion. Cuando se cierre
        sesion, se eliminara el archivo que lo contiene.
5   */
6
7  import android.content.Context;
8  import android.content.SharedPreferences;
9
10 public class AlmacenId {
11
12     private Context contexto;
13
14     public AlmacenId(Context contexto) {
15           this.contexto = contexto;
16     }
17
18     //Guardamos de forma permanente el id del usuario durante la sesion,
           para acceder a el desde cualquier clase.
19     public void guardarId(int id){
20         SharedPreferences preferencias= contexto.getSharedPreferences("
               almacenamiento_id", Context.MODE_PRIVATE);
21         SharedPreferences.Editor editor = preferencias.edit();
22         editor.putInt("id", id);
23         editor.commit();
24     }
25
26     //Obtenemos el id del usuario que esta almacenado.
27     public int obtenerId(){
28         int id=0;
```

```
29          SharedPreferences preferencias= contexto.getSharedPreferences("
                almacenamiento_id", Context.MODE_PRIVATE);
30          id= preferencias.getInt("id", 0);
31
32          return id;
33      }
34
35  }
```

C.16 Clase que gestiona la base de datos SQLite

C.16.1 BasedatosSQLite.java

```
1  package com.deporte.proyectofc;
2
3  /* Clase que se encarga del manejo de toda la base de datos de la
       aplicacion. Esta BBDD se llamara app_deporte.*/
4
5  import java.io.File;
6  import java.io.FileInputStream;
7  import java.io.FileNotFoundException;
8  import java.io.FileOutputStream;
9  import java.io.IOException;
10 import java.io.InputStream;
11 import java.lang.reflect.Field;
12 import java.util.ArrayList;
13 import java.util.Collections;
14 import android.content.ContentValues;
15 import android.content.Context;
16 import android.database.Cursor;
17 import android.database.sqlite.SQLiteDatabase;
18 import android.database.sqlite.SQLiteDatabase.CursorFactory;
19 import android.database.sqlite.SQLiteOpenHelper;
20 import android.os.Environment;
21
22 public class BasedatosSQLite extends SQLiteOpenHelper{
23
24      private static final int BUFFER_SIZE = 2048;
25      private Context context;
26      private static int version = 1;
27      private static String name = "app_deporte" ;
28      private static CursorFactory factory = null;
29
30      public BasedatosSQLite(Context context) {
31          super(context, name, factory, version);
32          this.context= context;
33      }
```

```
34
35      /* Metodo llamado automaticamente cuando se crea el primer objeto de
            esta clase. Se encarga de crear todas las tablas necesarias para
            el manejo de nuestra app, y de copiar la rutina que viene
            incluida en el paquete de la aplicacion (en la carpeta raw) a la
            carpeta app_deportes de nuestro dispositivo, para asi tener todas
             las rutinas en un solo sitio. */
36      @Override public void onCreate(SQLiteDatabase bd){
37
38          //Tabla que guarda los distintos usuarios
39          bd.execSQL("CREATE TABLE usuarios (" +
40                  "id_usuario INTEGER PRIMARY KEY AUTOINCREMENT, " +
41                  "usuario TEXT, " + "hash TEXT, " +
42                  "salt TEXT, " + "nombre TEXT, "
43                  + "apellidos TEXT, " + "e_mail TEXT, " +
44                  "estadodeforma TEXT);");
45
46          //Tabla que guarda los distintos ejercicios.
47          bd.execSQL("CREATE TABLE ejercicios (" +
48                  "id_ejercicio INTEGER PRIMARY KEY AUTOINCREMENT, " + "
                        ruta_ejercicio TEXT, " +
49                  "titulo TEXT, " + "subtitulo TEXT, " + "descripcion TEXT,
                        " +
50                  "estadodeforma TEXT, " + "repeticiones INTEGER, " + "
                        id_rutina INTEGER, " +
51                  "FOREIGN KEY(id_rutina) REFERENCES rutinas(id_rutina) ON
                        DELETE CASCADE);");
52
53          //Tabla que guarda las distintas rutinas
54          bd.execSQL("CREATE TABLE rutinas (" +
55                  "id_rutina INTEGER PRIMARY KEY AUTOINCREMENT, " + "
                        descripcion_rutina TEXT, " +
56                  "nombre_rutina TEXT UNIQUE, ruta_rutina TEXT);");
57
58          //Tabla que relaciona a los usuarios, con los ejercicios que ha
                realizados.
59          bd.execSQL("CREATE TABLE ejercicios_visualizados (" +
60                  "id_usuario INTEGER, id_ejercicio INTEGER, " +
61                  "fecha TEXT, " + "hora TEXT ," + "tiempo INTEGER ," +
62                  "FOREIGN KEY(id_usuario) REFERENCES usuarios(id_usuario)
                        ON DELETE CASCADE, " +
63                  "FOREIGN KEY(id_ejercicio) REFERENCES ejercicios(
                        id_ejercicio) ON DELETE CASCADE, " +
64                  "PRIMARY KEY(id_usuario, id_ejercicio, fecha, hora));");
65
66          //Tabla para guardar las rutinas que han sido ya cargadas por el
                usuario.
67          bd.execSQL("CREATE TABLE usuario_rutina (" +
68                  "id_usuario INTEGER, id_rutina INTEGER, "+
69                  "FOREIGN KEY(id_usuario) REFERENCES usuarios(id_usuario)
                        ON DELETE CASCADE, " +
```

```
70              "FOREIGN KEY (id_rutina) REFERENCES rutinas(id_rutina) ON
                    DELETE CASCADE, " +
71              "PRIMARY KEY(id_usuario, id_rutina));");
72
73          //Copiamos en la carpeta app_deportes del dispositivo, el
                contenido de raw.
74          InputStream inputstream= null;
75          FileOutputStream dest= null;
76          String nombre= "";
77          byte[] data= new byte[BUFFER_SIZE];
78          int len;
79
80          String destino= Environment.getExternalStorageDirectory().
                getAbsolutePath().toString() + "/app_deportes/rutinas/gap";
81          File dir= new File(destino);
82          if(dir.mkdirs() || dir.isDirectory()){
83              if(dir.list().length==0){
84                  Field[] campos= R.raw.class.getFields();
85                  for(int count=0 ; count<campos.length ; count++){
86                      nombre= campos[count].getName();
87                      int id= context.getResources().getIdentifier(nombre, "
                            raw", context.getPackageName());
88                      inputstream= context.getResources().openRawResource(id)
                            ;
89                      if(nombre.compareTo("inforutina")==0){
90                          nombre= nombre+".json";
91                      }
92                      else{
93                          nombre= nombre+".mp4";
94                      }
95                      try{
96                          dest= new FileOutputStream(new File(destino,
                                nombre));
97                          try{
98                              while ((len = inputstream.read(data)) != -1) {
99                                  dest.write(data, 0, len);
100                             }
101                         }catch(IOException e){
102
103                         }
104                     }catch(FileNotFoundException e){
105                         System.err.println(e);
106                     }
107
108                 }
109                 try{
110                     inputstream.close();
111                     dest.close();
112                 }catch(IOException e){
113                     System.err.println(e);
114                 }
115             }
```

```
116         }
117         else {
118             System.err.println("No se ha podido crear el directorio /
                    app_deportes/rutinas/gap/");
119         }
120
121         String path_rutina= Environment.getExternalStorageDirectory().
                getAbsolutePath().toString() + "/app_deportes/rutinas/gap/";
122         insertarRutina(bd, null, "GAP", path_rutina);
123         int id_rutina= obtenerIdRutina(bd, "GAP");
124         //Rellenamos la tabla ejercicios, con los ejercicios que venian
                en la app en raw folder, los de la rutina GAP.
125         String descripcion_rutina= introduceEjercicios(bd, id_rutina);
126         insertarDescRut(bd, id_rutina, descripcion_rutina);
127
128     }/*Fin onCreate*/
129
130
131
132     @Override public void onUpgrade(SQLiteDatabase db, int oldVersion, int
            newVersion) {
133         // En caso de una nueva version habra que actualizar las tablas.
134         //Se elimina la version anterior de la tabla
135         db.execSQL("DROP TABLE IF EXISTS usuarios");
136         db.execSQL("DROP TABLE IF EXISTS ejercicios");
137         db.execSQL("DROP TABLE IF EXISTS rutinas");
138         db.execSQL("DROP TABLE IF EXISTS ejercicios_visualizados");
139         db.execSQL("DROP TABLE IF EXISTS usuario_rutina");
140
141         //Se crea la nueva version de la tabla
142         onCreate(db);
143     }
144
145     //Crea un objeto SQLiteDatabase.
146     public SQLiteDatabase creaSQLitedatabase(){
147         SQLiteDatabase bd= getWritableDatabase();
148         return bd;
149     }
150
151     /* METODOS PARA TABLA EJERCICIOS */
152
153     //Introducimos en la tabla ejercicios los ejercicios de una rutina
154     public String introduceEjercicios(SQLiteDatabase bd, int id_rutina){
155
156         String descripcion_rutina="";
157         String titulo="", subtitulo="", descripcion= "", estadodeforma= "
                ";
158         int repeticiones= 0;
159         boolean existeJson= false;
160         boolean existeMp4= false;
161         ArrayList<Ejercicio> infoejercicios= null;
162
```

```
163         String pathrutina= obtenerPathRutina(bd, id_rutina);
164         File fic= new File(pathrutina);
165         if(fic.exists()){
166             File[] ficheros= fic.listFiles();
167             for(int j=0 ; j<ficheros.length ; j++){
168                 if(ficheros[j].getName().endsWith(".json")){
169                     existeJson= true;
170                 }
171                 else if(ficheros[j].getName().endsWith(".mp4")){
172                     existeMp4= true;
173                 }
174             }
175             if(!existeJson || !existeMp4){
176                 descripcion_rutina= "Rutinainvalida";
177                 return descripcion_rutina;
178             }
179 
180             for(int i=0 ; i<ficheros.length ; i++){
181                 if(ficheros[i].getName().endsWith(".json")){
182                     existeJson=true;
183                     try{
184                         System.out.println("entra a leer el json");
185                         InputStream in= new FileInputStream(ficheros[i]);
186                         LecturaJson leerjson= new LecturaJson(in);
187                         infoejercicios= leerjson.readJsonStream();
188                         descripcion_rutina= leerjson.getDescripcion_rutina
                                ();
189                     }catch(FileNotFoundException e){
190                         System.err.println(e);
191                     }
192                 }
193             }
194 
195             for(int i=0; i<ficheros.length ; i++){
196                 if(ficheros[i].getName().endsWith(".mp4")){
197                     String identificador= ficheros[i].getName();
198                     for(Ejercicio v : infoejercicios){
199                         String nombre_ejercicio= identificador.replace(".
                                mp4", "");
200                         if(nombre_ejercicio.equals(v.getNombre())){
201                             titulo= v.getTitulo();
202                             subtitulo= v.getSubtitulo();
203                             descripcion= v.getDescripcion();
204                             estadodeforma= v.getEstado();
205                             repeticiones= v.getRepeticiones();
206                             System.out.println("path " + ficheros[i].
                                    getAbsolutePath());
207                             System.out.println("titulo " + titulo);
208                             System.out.println("subtitulo " + subtitulo);
209                             System.out.println("descripcion " + descripcion
                                    );
210                             System.out.println("estado " + estadodeforma);
```

```
211                         System.out.println("repeticiones " +
                                repeticiones);
212                         bd.execSQL("INSERT INTO ejercicios VALUES (null,
                                \"" + ficheros[i].getAbsolutePath() + "\",
                                \""+
213                         titulo + "\", \"" + subtitulo + "\", \"" +
                                descripcion + "\", \"" + estadodeforma +
214                         "\", " + repeticiones + ", " + id_rutina + ");"
                                );
215                     }
216                 }
217             }
218         }
219     }
220     return descripcion_rutina;
221
222 }
223
224 //Obtenemos las rutas de todos los ejercicios que pertenecen a una
        rutina.
225     public ArrayList<String> listaEjerciciosRutina(int id_rutina){
226
227         ArrayList<String> ejercicios= new ArrayList<String>();
228         SQLiteDatabase bd= getReadableDatabase();
229         Cursor cursor= bd.rawQuery("SELECT ruta_ejercicio FROM
                ejercicios WHERE id_rutina= " + id_rutina, null);
230         while(cursor.moveToNext()){
231             ejercicios.add(cursor.getString(0));
232         }
233         cursor.close();
234         bd.close();
235
236         return ejercicios;
237     }
238
239 //Obtenemos las rutas de todos los ejercicios pertenecientes a una
        rutina y a un estado de forma.
240     public ArrayList<String> listaEjEstadoForma(int id_rutina, String
            estadodeforma){
241
242         ArrayList<String> ejercicios= new ArrayList<String>();
243         SQLiteDatabase bd= getReadableDatabase();
244         Cursor cursor= bd.rawQuery("SELECT ruta_ejercicio FROM
                ejercicios WHERE id_rutina=" + id_rutina + " AND
                estadodeforma=\"" + estadodeforma + "\"", null);
245         while(cursor.moveToNext()){
246             ejercicios.add(cursor.getString(0));
247         }
248         cursor.close();
249         bd.close();
250
251         return ejercicios;
```

```
252         }
253
254     //Obtener el estado de forma de un ejercicio
255     public String obtenerEstadoEjercicio(int id_ejercicio){
256
257         String estadodeforma= "";
258         SQLiteDatabase bd= getReadableDatabase();
259         Cursor cursor= bd.rawQuery("SELECT estadodeforma FROM ejercicios
                WHERE id_ejercicio=" + id_ejercicio, null);
260         if(cursor.moveToFirst()){
261             estadodeforma= cursor.getString(0);
262         }
263         cursor.close();
264         bd.close();
265
266         return estadodeforma;
267     }
268
269     //Obtenemos el id de un ejercicio a partir de su path
270     public int obtenerIdEjercicio(String ruta_ejercicio){
271
272         int id_ejercicio= 0;
273         SQLiteDatabase bd= getReadableDatabase();
274         Cursor cursor= bd.rawQuery("SELECT id_ejercicio FROM ejercicios
                WHERE ruta_ejercicio=\"" +
275         ruta_ejercicio + "\";", null);
276         if(cursor.moveToFirst()){
277             id_ejercicio= cursor.getInt(0);
278         }
279         cursor.close();
280         bd.close();
281
282         return id_ejercicio;
283     }
284
285     //Obtenemos el path de un ejercicio a partir de su id
286     public String obtenerPathEjercicio(int id_ejercicio){
287
288         String path_ejercicio="";
289         SQLiteDatabase bd= getReadableDatabase();
290         Cursor cursor= bd.rawQuery("SELECT ruta_ejercicio FROM ejercicios
                WHERE id_ejercicio=" + id_ejercicio , null);
291         while(cursor.moveToNext()){
292             path_ejercicio= cursor.getString(0);
293         }
294         cursor.close();
295         bd.close();
296
297         return path_ejercicio;
298     }
299
300     //Obtenemos el id de rutina al que pertenece un ejercicio
```

```
301     public int obtenerIdRutEjercicio(int id_ejercicio){
302
303         int id_rutina= 0;
304         SQLiteDatabase bd= getReadableDatabase();
305         Cursor cursor= bd.rawQuery("SELECT id_rutina FROM ejercicios
                WHERE id_ejercicio=" + id_ejercicio, null);
306         if(cursor!=null && cursor.getCount()>0){
307             cursor.moveToFirst();
308             id_rutina= cursor.getInt(0);
309         }
310         cursor.close();
311         bd.close();
312
313         return id_rutina;
314     }
315
316     //Obtenemos los titulos de los ejercicios correspondientes a una
            rutina y a un estado de forma
317     public ArrayList<String> listaTitulos(int id_rutina, String
            estadodeforma){
318
319         ArrayList<String> titulos= new ArrayList<String>();
320         SQLiteDatabase bd= getReadableDatabase();
321         Cursor cursor= bd.rawQuery("SELECT titulo FROM ejercicios WHERE
                id_rutina=" + id_rutina + " AND estadodeforma=\"" +
                estadodeforma + "\"" , null);
322         while(cursor.moveToNext()){
323             titulos.add(cursor.getString(0));
324         }
325         cursor.close();
326         bd.close();
327
328         return titulos;
329     }
330
331     //Obtenemos los subtitulos de los ejercicios correspondientes a una
            rutina y a un estado de forma
332     public ArrayList<String> listaSubtitulos(int id_rutina, String
            estadodeforma){
333
334         ArrayList<String> subtitulos= new ArrayList<String>();
335         SQLiteDatabase bd= getReadableDatabase();
336         Cursor cursor= bd.rawQuery("SELECT subtitulo FROM ejercicios
                WHERE id_rutina=" + id_rutina + " AND estadodeforma=\"" +
                estadodeforma + "\"", null);
337         while(cursor.moveToNext()){
338             subtitulos.add(cursor.getString(0));
339         }
340         cursor.close();
341         bd.close();
342
343         return subtitulos;
```

```
344     }
345
346     //Obtenemos el numero de repeticiones de un ejercicio
347     public int obtenerNumRepeticiones(int id_ejercicio){
348
349         SQLiteDatabase bd= getReadableDatabase();
350         Cursor cursor= bd.rawQuery("SELECT repeticiones FROM ejercicios
                WHERE id_ejercicio= " +
351         id_ejercicio + ";", null);
352         int repeticiones= 0;
353         if(cursor.moveToFirst()){
354             repeticiones= cursor.getInt(0);
355         }
356         cursor.close();
357         bd.close();
358
359         return repeticiones;
360     }
361
362     //Obtenemos la descripcion del ejercicio
363     public String obtenerDescripEjercicio(int id_ejercicio){
364
365         SQLiteDatabase bd= getReadableDatabase();
366         Cursor cursor= bd.rawQuery("SELECT descripcion FROM ejercicios
                WHERE id_ejercicio= " +
367         id_ejercicio + ";", null);
368         String descripcion="";
369         if(cursor.moveToFirst()){
370             descripcion= cursor.getString(0);
371         }
372         cursor.close();
373         bd.close();
374
375         return descripcion;
376     }
377
378     //Existe un video con ese id
379     public boolean existeEjercicio(int id_ejercicio){
380
381         boolean resultado;
382         SQLiteDatabase bd= getReadableDatabase();
383         Cursor cursor= bd.rawQuery("SELECT id_ejercicio FROM ejercicios
                WHERE id_ejercicio=" + id_ejercicio, null);
384         if(cursor!=null && cursor.getCount()>0){
385             resultado= true;
386         }
387         else{
388             resultado= false;
389         }
390         cursor.close();
391         bd.close();
392
```

```
393         return resultado;
394     }
395
396     //Eliminamos el ejercicio de la base de datos
397     public boolean eliminarEjercicio(String path_ejercicio){
398
399         boolean resultado= false;
400         SQLiteDatabase bd= getWritableDatabase();
401         Cursor cursor= bd.rawQuery("DELETE FROM ejercicios WHERE
                ruta_ejercicio=\"" + path_ejercicio + "\";", null);
402         if(cursor.moveToFirst()){
403             resultado= true;
404         }
405         cursor.close();
406         bd.close();
407
408         return resultado;
409     }
410
411
412     /*METODOS PARA LA TABLA RUTINAS*/
413
414     //Insertamos una nueva rutina.
415     public void insertarRutina(SQLiteDatabase bd, String
            descripcion_rutina, String nombre_rutina, String path_rutina){
416
417         bd.execSQL("INSERT INTO rutinas VALUES (null, \"" +
                descripcion_rutina + "\", \"" + nombre_rutina +
418                 "\", \"" + path_rutina +"\");");
419     }
420
421     //Obtenemos nombre de la rutina a partir del id
422     public String obtenerNombreRutina(int id_rutina){
423
424         String nombrerutina="";
425         SQLiteDatabase bd= getReadableDatabase();
426         Cursor cursor= bd.rawQuery("SELECT nombre_rutina FROM rutinas
                WHERE id_rutina=" + id_rutina, null);
427         if(cursor!=null && cursor.getCount()>0){
428             cursor.moveToFirst();
429             nombrerutina=cursor.getString(0);
430         }
431         cursor.close();
432         bd.close();
433
434         return nombrerutina;
435     }
436
437     //Obtener el identificador de la rutina a partir del nombre
438     public int obtenerIdRutina(SQLiteDatabase bd, String nombre_rutina){
439
440         int id_rutina=0;
```

```
441         Cursor cursor= bd.rawQuery("SELECT id_rutina FROM rutinas WHERE
                nombre_rutina=\"" +
442         nombre_rutina + "\";", null);
443         if(cursor!=null && cursor.getCount()>0){
444             cursor.moveToFirst();
445             id_rutina= cursor.getInt(0);
446         }
447         cursor.close();
448 
449         return id_rutina;
450     }
451 
452     //Obtener descripcion de una rutina dada su identificador
453     public String obtenerDescripcionRutina(int id_rutina){
454 
455         String descripcion_rutina="";
456         SQLiteDatabase bd= getReadableDatabase();
457         Cursor cursor= bd.rawQuery("SELECT descripcion_rutina FROM
                rutinas WHERE id_rutina=" + id_rutina, null);
458         if(cursor!=null && cursor.getCount()>0){
459             cursor.moveToFirst();
460             descripcion_rutina= cursor.getString(0);
461         }
462         cursor.close();
463         bd.close();
464 
465         return descripcion_rutina;
466     }
467 
468     //Modificar la descripcion de una rutina
469     public void insertarDescRut(SQLiteDatabase bd, int id_rutina, String
            descripcion_rutina){
470 
471         bd.beginTransaction();
472         ContentValues valores = new ContentValues();
473         valores.put("descripcion_rutina", descripcion_rutina);
474         bd.update("rutinas", valores, "id_rutina=" + id_rutina, null);
475         bd.setTransactionSuccessful();
476         bd.endTransaction();
477     }
478 
479     //Obtenemos la ruta de la rutina a partir del identificador de la
            rutina
480         public String obtenerPathRutina(SQLiteDatabase bd, int id_rutina){
481 
482             Cursor cursor= bd.rawQuery("SELECT ruta_rutina FROM rutinas
                    WHERE id_rutina=" + id_rutina, null);
483             String path= null;
484             while(cursor.moveToNext()){
485                 path= cursor.getString(0);
486             }
487             cursor.close();
```

```
488
489             return path;
490         }
491
492     //Compruba si existe una rutina dado el nombre
493     public boolean existeRutina(String nombre_rutina){
494
495         SQLiteDatabase bd= getReadableDatabase();
496         Cursor cursor= bd.rawQuery("SELECT id_rutina FROM rutinas WHERE
                nombre_rutina=\"" +
497         nombre_rutina + "\";", null);
498         if(cursor!=null && cursor.getCount() > 0 && cursor.moveToFirst()){
499             return true;
500         }
501         else{
502             return false;
503         }
504     }
505
506     //Borrar elemento de la tabla rutinas
507     public void borrarRutina(int id_rutina){
508
509         SQLiteDatabase bd= getWritableDatabase();
510         bd.beginTransaction();
511         bd.delete("rutinas", "id_rutina=" + id_rutina, null);
512         bd.setTransactionSuccessful();
513         bd.endTransaction();
514         bd.close();
515     }
516
517     //METODOS TABLA USUARIOS
518
519     //Registramos a un nuevo usuario
520     public void guardarNuevoUsuario(String usuario, String hash, String
            salt, String nombre,
521             String apellidos, String e_mail, String estadodeforma){
522
523         SQLiteDatabase bd= getWritableDatabase();
524         bd.execSQL("INSERT INTO usuarios VALUES (null, \"" + usuario + "
                \", \"" + hash + "\", \"" +
525                 salt + "\", \"" + nombre + "\", \"" + apellidos + "\", \""
                        + e_mail
526                 + "\", \"" + estadodeforma + "\");");
527         bd.close();
528     }
529
530     //Obtener id de un usuario a partir del usuario
531     public int obtenerIdUsuario(String usuario){
532
533         int id_usuario= 0;
534         SQLiteDatabase bd= getReadableDatabase();
```

```
535         Cursor cursor= bd.rawQuery("SELECT id_usuario FROM usuarios WHERE
                " + "usuario= \"" + usuario + "\";", null);
536         if(cursor!=null && cursor.getCount()>0){
537             cursor.moveToFirst();
538             id_usuario= cursor.getInt(cursor.getColumnIndex("id_usuario"))
                    ;
539         }
540         cursor.close();
541         bd.close();
542 
543         return id_usuario;
544     }
545 
546     //Obtenemos el hash y la salt de un determinado usuario
547     public ArrayList<String> obtenerHashYSalt(String usuario){
548 
549         SQLiteDatabase bd= getReadableDatabase();
550         Cursor cursor= bd.rawQuery("SELECT u.hash, u.salt FROM usuarios u
                WHERE u.usuario= \""
551         + usuario + "\";", null);
552         ArrayList<String> hashysalt= new ArrayList<String>();
553         if(cursor.moveToFirst()){
554             hashysalt.add(cursor.getString(0));
555             hashysalt.add(cursor.getString(1));
556         }
557         cursor.close();
558         bd.close();
559 
560         return hashysalt;
561     }
562 
563     //Comprueba si existe un determinado usuario
564     public boolean existeUsuario(String usuario){
565 
566         SQLiteDatabase bd= getReadableDatabase();
567         Cursor cursor= bd.rawQuery("SELECT id_usuario FROM usuarios u
                WHERE " +
568         "usuario=\"" + usuario + "\";", null);
569         if(cursor!=null && cursor.getCount() > 0 && cursor.moveToFirst()){
570             return true;
571         }
572         else{
573             return false;
574         }
575     }
576 
577     //Existe una dupla usuario-e_mail
578     public boolean existeEmail(String usuario, String e_mail){
579 
580         int i=0;
581         SQLiteDatabase bd= getReadableDatabase();
```

```
582         Cursor cursor= bd.rawQuery("SELECT e_mail, usuario FROM usuarios",
                null);
583         if (cursor.moveToFirst()){
584             do{
585                 if (usuario.compareTo(cursor.getString(cursor.
                        getColumnIndex("usuario")))==0 &&
586                         e_mail.compareTo(cursor.getString(cursor.
                                getColumnIndex("e_mail")))==0){
587                     i++;
588                 }
589             }while(cursor.moveToNext());
590 
591             if (i>0){
592                 return  true;
593             }else{
594                 return  false ;
595             }
596         }
597         else{
598             return  false ;
599         }
600     }
601 
602     //Obtnemos la salt de un usuario
603     public  String obtenerSalt(String usuario){
604 
605         String salt="";
606         SQLiteDatabase bd= getReadableDatabase();
607         Cursor cursor= bd.rawQuery("SELECT salt FROM usuarios WHERE
                usuario=\""
608                 + usuario + "\";", null);
609         if (cursor.moveToFirst()){
610             salt= cursor.getString(cursor.getColumnIndex("salt"));
611         }
612         cursor.close();
613         bd.close();
614 
615         return  salt;
616     }
617 
618     //Modificamos los datos de un usuario
619     public  void  modificarUsuario(int id_usuario, String nombre, String
            apellidos, String e_mail, String estadodeforma){
620 
621         SQLiteDatabase bd= getWritableDatabase();
622         bd.beginTransaction();
623         ContentValues valores = new ContentValues();
624         valores.put("nombre", nombre);
625         valores.put("apellidos", apellidos);
626         valores.put("e_mail", e_mail);
627         valores.put("estadodeforma", estadodeforma);
628         bd.update("usuarios", valores, "id_usuario=" + id_usuario, null);
```

```
629         bd.setTransactionSuccessful();
630         bd.endTransaction();
631         bd.close();
632     }
633 
634     //Modificamos la contraseña
635     public void modificaContrasenia(int id_usuario, String hash){
636 
637         SQLiteDatabase bd= getWritableDatabase();
638         bd.beginTransaction();
639         ContentValues valores= new ContentValues();
640         valores.put("hash", hash);
641         bd.update("usuarios", valores, "id_usuario=" + id_usuario, null);
642         bd.setTransactionSuccessful();
643         bd.endTransaction();
644         bd.close();
645     }
646 
647     //Obtenemos el nombre para el saludo inicial al usuario
648     public String obtenerNombreUsuario(int id_usuario){
649 
650         String nombre="";
651         SQLiteDatabase bd= getReadableDatabase();
652         Cursor cursor= bd.rawQuery("SELECT nombre FROM usuarios WHERE " +
653         "id_usuario=" + id_usuario, null);
654         if(cursor!=null && cursor.moveToFirst()){
655             nombre = cursor.getString(cursor.getColumnIndex("nombre"));
656         }
657         cursor.close();
658         bd.close();
659 
660         return nombre;
661     }
662 
663     //Obtenemos todos los datos del usuario
664     public ArrayList<String> datosUsuario (int id_usuario){
665 
666         ArrayList<String> datos= new ArrayList<String>();
667         SQLiteDatabase bd= getReadableDatabase();
668         Cursor cursor= bd.rawQuery("SELECT * FROM usuarios WHERE
                id_usuario=" + id_usuario, null);
669         if(cursor.moveToFirst()){
670             //Existe al menos un registro
671             int indicecol = cursor.getColumnIndex("id_usuario");
672             datos.add(cursor.getString(indicecol));
673             indicecol = cursor.getColumnIndex("usuario");
674             datos.add(cursor.getString(indicecol));
675             indicecol = cursor.getColumnIndex("hash");
676             datos.add(cursor.getString(indicecol));
677             indicecol = cursor.getColumnIndex("salt");
678             datos.add(cursor.getString(indicecol));
679             indicecol = cursor.getColumnIndex("nombre");
```

```
680             datos.add(cursor.getString(indicecol));
681             indicecol = cursor.getColumnIndex("apellidos");
682             datos.add(cursor.getString(indicecol));
683             indicecol = cursor.getColumnIndex("e_mail");
684             datos.add(cursor.getString(indicecol));
685             indicecol = cursor.getColumnIndex("estadodeforma");
686             datos.add(cursor.getString(indicecol));
687             cursor.close();
688             bd.close();
689 
690             return datos;
691         }
692         else{
693             return null;
694         }
695     }
696 
697     //Obtenemos el estado de forma de un usuario
698     public String obtenerEstadoDeForma(int id_usuario){
699 
700         String estadodeforma="";
701         SQLiteDatabase bd= getReadableDatabase();
702         Cursor cursor= bd.rawQuery("SELECT estadodeforma FROM usuarios
                WHERE id_usuario=" + id_usuario, null);
703         if(cursor.moveToFirst()){
704             estadodeforma= cursor.getString(0);
705         }
706         cursor.close();
707         bd.close();
708 
709         return estadodeforma;
710     }
711 
712     //Eliminar una cuenta de usuario
713     public void eliminarUsuario(int id_usuario){
714 
715         SQLiteDatabase bd= getWritableDatabase();
716         bd.beginTransaction();
717         bd.delete("usuarios", "id_usuario=" + id_usuario, null);
718         bd.setTransactionSuccessful();
719         bd.endTransaction();
720         bd.close();
721     }
722 
723     //METODOS TABLA USUARIO_RUTINA
724 
725     //Introducimos un nuevo elemento en la tabla.El usuario ha cargado
            una nueva rutina de ejercicios(excepto la de gap que viene
            cargada por defecto)
726     public void introRutinaUsuario(int id_usuario, int id_rutina){
727 
728         SQLiteDatabase bd= getWritableDatabase();
```

```
729         bd.execSQL("INSERT INTO usuario_rutina VALUES (" + id_usuario + ",
                " + id_rutina + ");" );
730         bd.close();
731     }
732
733     //Obtenemos la lista de rutinas cargadas de las que actualmente
            dispone el usuario
734     public ArrayList<Integer> obtenerRutinasUsuario(int id_usuario){
735
736         ArrayList<Integer> ids= new ArrayList<Integer>();
737         SQLiteDatabase bd= getReadableDatabase();
738         Cursor cursor= bd.rawQuery("SELECT id_rutina FROM usuario_rutina
                WHERE id_usuario=" + id_usuario, null);
739         while(cursor.moveToNext()){
740             ids.add(cursor.getInt(0));
741         }
742         cursor.close();
743         bd.close();
744
745         return ids;
746     }
747
748     //Cuenta cuantos usuarios tienen una rutina
749     public int cuentaUsuariosRutina (int id_rutina){
750
751         SQLiteDatabase bd= getReadableDatabase();
752         int resultado= 0;
753         Cursor cursor= bd.rawQuery("SELECT id_rutina FROM usuario_rutina
                WHERE id_rutina=" + id_rutina, null);
754         while(cursor.moveToNext()){
755             resultado++;
756         }
757         cursor.close();
758         bd.close();
759
760         return resultado;
761     }
762     //El usuario desea eliminar una rutina
763     public void borrarRutUsuario(int _id_usuario, int id_rutina){
764
765         SQLiteDatabase bd= getWritableDatabase();
766         bd.beginTransaction();
767         bd.delete("usuario_rutina", "id_usuario=" + _id_usuario + " AND
                id_rutina=" + id_rutina, null);
768         bd.setTransactionSuccessful();
769         bd.endTransaction();
770         bd.close();
771     }
772
773     //Obtenemos las rutinas que el usuario aun no ha cargado
774     public ArrayList<Integer> obtenerMasRutinas(int id_usuario){
775
```

```
776         ArrayList<Integer> masrutinas= new ArrayList<Integer>();
777         SQLiteDatabase bd= getReadableDatabase();
778         Cursor cursor= bd.rawQuery("SELECT DISTINCT id_rutina FROM
                usuario_rutina WHERE id_rutina NOT IN (" +
779         "SELECT id_rutina FROM usuario_rutina WHERE id_usuario=" +
                id_usuario + ")", null);
780         while(cursor.moveToNext()){
781             masrutinas.add(cursor.getInt(0));
782         }
783         cursor.close();
784         bd.close();
785 
786         return masrutinas;
787     }
788 
789     //Compobamos si una rutina ha sido cargada por alguien
790     public boolean rutinaYaCargada(int id_rutina){
791 
792         SQLiteDatabase bd= getReadableDatabase();
793         Cursor cursor= bd.rawQuery("SELECT id_rutina FROM usuario_rutina
                WHERE " +
794         "id_rutina=" + id_rutina, null);
795         if(cursor!=null && cursor.getCount() > 0 && cursor.moveToFirst()){
796             return true;
797         }
798         else{
799             return false;
800         }
801     }
802 
803     //METODOS TABLA EJERCICIOS_VISUALIZADOS
804 
805     //Registramos una visulizacion del usuario
806     public void guardarNuevaVisualizacion(int id_usuario, int id_video,
            String fecha, String hora, int tiempo){
807 
808         SQLiteDatabase bd= getWritableDatabase();
809         bd.execSQL("INSERT INTO ejercicios_visualizados VALUES (" +
                id_usuario + ", "+ id_video + ", \"" + fecha +
810                     "\", \"" + hora + "\", " + tiempo +")");
811         bd.close();
812     }
813 
814     //Obtenemos todos los videos visualizados por un determinado usuario
815     public ArrayList<Integer> obtenerVisualizadosUsuario (int id_usuario){
816 
817         ArrayList<Integer> visualizados= new ArrayList<Integer>();
818         SQLiteDatabase bd= getReadableDatabase();
819         Cursor cursor= bd.rawQuery("SELECT DISTINCT id_ejercicio FROM
                ejercicios_visualizados " +
820         "WHERE id_usuario="+ id_usuario, null);
821         while(cursor.moveToNext()){
```

```
822             visualizados.add(cursor.getInt(0));
823         }
824         cursor.close();
825         bd.close();
826         Collections.sort(visualizados);
827 
828         return visualizados;
829     }
830     //Obtenemos la fecha y la hora más reciente de un video visto por un
        usuario
831     public String[] obtenerFechayHora (int id_usuario, int id_ejercicio){
832 
833         SQLiteDatabase bd= getReadableDatabase();
834         Cursor cursor= bd.rawQuery("SELECT fecha, hora FROM
            ejercicios_visualizados WHERE " +
835         "id_usuario= " + id_usuario + " AND id_ejercicio= " +
            id_ejercicio +
836         " ORDER BY fecha, hora DESC", null);
837         String[] fechayhora= new String[2];
838         if(cursor.moveToFirst()){
839             fechayhora[0]= cursor.getString(0);
840             fechayhora[1]= cursor.getString(1);
841         }
842         cursor.close();
843         bd.close();
844 
845         return fechayhora;
846     }
847 
848     //Obtenemos el tiempo en segundos que hemos dedicado en un día a cada
         ejercicio
849     public int obtenerTiempos(int id_usuario, String fecha){
850 
851         SQLiteDatabase bd= getReadableDatabase();
852         Cursor cursor= bd.rawQuery("SELECT tiempo FROM
            ejercicios_visualizados WHERE "+
853         "id_usuario= " + id_usuario + " AND fecha= \"" + fecha + "\"",
            null);
854         int tiempo= 0;
855         while(cursor.moveToNext()){
856             tiempo += cursor.getInt(0);
857         }
858         cursor.close();
859         bd.close();
860 
861         return tiempo;
862     }
863 }
```

C.17 Clase que lee el fichero JSON

C.17.1 LecturaJson.java

```
1  package com.deporte.proyectofc;
2
3  /* Clase que se encarga de leer el archivo json que incluirán todas las
       carpetas de rutinas.*/
4
5  import java.io.IOException;
6  import java.io.InputStream;
7  import java.io.InputStreamReader;
8  import java.io.UnsupportedEncodingException;
9  import java.util.ArrayList;
10
11 import android.util.JsonReader;
12
13 public class LecturaJson {
14
15     private InputStream input;
16     private String descripcion_rutina;
17
18     public LecturaJson(InputStream input){
19         this.input= input;
20     }
21
22     //Metodo encargado de leer correctamente el archivo JSON, que tiene
           una determinada estructura.Devuelve una lista de objetos de clase
            Ejercicio que contienen la informaacion leida.
23     public ArrayList<Ejercicio> readJsonStream(){
24         try{
25             JsonReader reader= new JsonReader(new InputStreamReader(input,
                    "UTF-8"));
26             ArrayList<Ejercicio> infoejercicios= new ArrayList<Ejercicio
                   >();
27             try{
28                 //Lo leemos atendiendo a la estructura que tendran los
                       ficheros json.
29                 //Nos situamos al inicio de array.
30                 reader.beginArray();
31                 while(reader.hasNext()){
32                     reader.beginObject();
33                     String elemento= reader.nextName();
34                     //El primer objeto del archivo es descripcion rutina
35                     if(elemento.equals("descripcion rutina")){
36                         descripcion_rutina= reader.nextString();
37                     }
38                     //El segundo objeto del archivo es un array de objetos
                           con la informacion de los ejercicios
39                     else if(elemento.equals("informacion ejercicios")){
```

```
40                      reader.beginArray();
41                      while(reader.hasNext()){
42                          infoejercicios.add(leer_info_ejercicios(reader)
                                );
43                      }
44                      reader.endArray();
45                  }
46
47                  reader.endObject();
48              }
49              reader.endArray();
50              reader.close();
51              return infoejercicios;
52
53          }catch(IOException e){
54              return null;
55          }
56      }catch(UnsupportedEncodingException e){
57          return null;
58      }
59  }
60
61  //Metodo encargado de leer cada uno de los objetos del array que
        contiene el objeto "informacion ejercicios" del archivo JSON y
        crear un objeto de la clase Ejercicio para cada uno de ellos.
62  public Ejercicio leer_info_ejercicios(JsonReader reader) throws
        IOException{
63      String nombre= null;
64      String titulo= null;
65      String subtitulo= null;
66      String descripcion= null;
67      String estadodeforma= null;
68      int repeticiones= 0;
69
70      reader.beginObject();
71      while(reader.hasNext()){
72          String campo= reader.nextName();
73          if(campo.equals("Nombre")){
74              nombre= reader.nextString();
75          }
76          else if(campo.equals("Titulo")){
77              titulo= reader.nextString();
78          }
79          else if(campo.equals("Subtitulo")){
80              subtitulo= reader.nextString();
81          }
82          else if(campo.equals("Descripcion")){
83              descripcion= reader.nextString();
84          }
85          else if(campo.equals("Estado de forma")){
86              estadodeforma= reader.nextString();
87          }
```

```
88              else if(campo.equals("Repeticiones")){
89                  repeticiones= reader.nextInt();
90              }
91          }
92          reader.endObject();
93
94          //Rellenamos con los datos un objeto de la clase Video
95          return new Ejercicio(nombre, titulo, subtitulo, descripcion,
                estadodeforma, repeticiones);
96      }
97
98      //Obtenemos la descripcion de la rutina que incluye el archivo JSON.
99      public String getDescripcion_rutina() {
100         return descripcion_rutina;
101     }
102
103 }
```

C.17.2 Ejercicio.java

```
1  package com.deporte.proyectofc;
2
3  /* Clase que guarda la informacion de un elemento de video leido del
       archivo infovideos.json. */
4
5  public class Ejercicio {
6
7      private String nombre;
8      private String titulo;
9      private String subtitulo;
10     private String descripcion;
11     private String estadodeforma;
12     private int repeticiones;
13
14     public Ejercicio(String nombre, String titulo, String subtitulo,
           String descripcion,
15             String estadodeforma, int repeticiones){
16
17         super();
18         this.nombre= nombre;
19         this.titulo= titulo;
20         this.subtitulo= subtitulo;
21         this.descripcion= descripcion;
22         this.estadodeforma= estadodeforma;
23         this.repeticiones= repeticiones;
24     }
25
26     public String getNombre() {
27         return nombre;
28     }
```

```
29
30      public String getTitulo() {
31          return titulo;
32      }
33
34      public String getSubtitulo() {
35          return subtitulo;
36      }
37
38      public String getDescripcion() {
39          return descripcion;
40      }
41      public String getEstado() {
42          return estadodeforma;
43      }
44
45      public int getRepeticiones() {
46          return repeticiones;
47      }
48
49  }
```

C.18 AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.deporte.proyectofc"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk
8          android:minSdkVersion="16"
9          android:targetSdkVersion="21" />
10
11     <uses-permission android:name="android.permission.
           WRITE_EXTERNAL_STORAGE" />
12     <uses-permission android:name="android.permission.INTERNET"/>
13
14     <application
15         android:allowBackup="true"
16         android:icon="@drawable/ic_launcher"
17         android:label="@string/app_name"
18         android:theme="@style/EstiloApp" >
19         <activity
20             android:name=".MainActivity"
21             android:label="@string/app_name"
22             android:screenOrientation="portrait" >
23             <intent-filter>
24                 <action android:name="android.intent.action.MAIN" />
```

```
25
26                <category android:name="android.intent.category.LAUNCHER"
                      />
27            </intent-filter>
28        </activity>
29        <activity android:name="Inicio"
30            android:screenOrientation="portrait">
31        </activity>
32        <activity android:name="NuevoUsuario"
33            android:screenOrientation="portrait">
34        </activity>
35        <activity android:name="ConfiguracionCuenta" >
36        </activity>
37        <activity android:name="ModificacionCuenta" >
38        </activity>
39        <activity android:name="ReproductorEjercicio" >
40        </activity>
41        <activity
42            android:name="ModificacionContrasenia"
43            android:theme="@android:style/Theme.Dialog" >
44        </activity>
45        <activity android:name="ContraseniaOlvidada" >
46        </activity>
47        <activity android:name="ListEjercicios" >
48        </activity>
49        <activity
50            android:name="AcercaDe"
51            android:theme="@android:style/Theme.Dialog" >
52        </activity>
53        <activity android:name="GraficasRendimiento" >
54        </activity>
55        <activity android:name="ListMasRutinas" >
56        </activity>
57        <activity android:name="ListNuevasRutinas" >
58        </activity>
59        <activity
60            android:name="EliminacionRutina"
61            android:theme="@android:style/Theme.Dialog" >
62        </activity>
63
64    </application>
65
66 </manifest>
```

C.19 String.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <!-- Nombre de la aplicación -->
```

```
4     <string name="app_name">RUTINAPP</string>
5
6     <!-- ACEPTAR Y CANCELAR -->
7     <string name="aceptar">Aceptar</string>
8     <string name="cancelar">Cancelar</string>
9
10    <!-- Datos de inicio sesion -->
11    <string name="usuario">Usuario:</string>
12    <string name="contrasenia_usuario">Contraseña:</string>
13
14    <!-- PANTALLA PRINCIPAL -->
15    <string name="introduce_usuario"> Introduce el usuario</string>
16    <string name="introduce_contrasenia"> Contraseña</string>
17    <string name="login">Iniciar</string>
18    <string name="registro">Registrarse</string>
19    <string name="olvido_contrasenia">¿Olvidó su contraseña?</string>
20
21    <!-- PANTALLA REGISTRO DE USUARIO y MODIFICACION CUENTA-->
22    <string name="nombre">Nombre:</string>
23    <string name="intro_nombre">Introduce el nombre</string>
24    <string name="apellidos">Apellidos:</string>
25    <string name="intro_apellidos">Introduce tus apellidos</string>
26    <string name="intro_usuario">Introduce un usuario</string>
27    <string name="intro_contrasenia">Introduce una contraseña</string>
28    <string name="e_mail">E-mail:</string>
29    <string name="intro_e_mail">Introduce un e-mail</string>
30    <string name="nivel">¿Cuál es tu estado de forma?</string>
31    <string name="modificar">Guardar cambios</string>
32
33    <!-- PANTALLA DE INICIO -->
34    <string name="mis_estadisticas">Mi rendimiento</string>
35    <string name="configuracion">Configuración</string>
36    <string name="acerca_de">Acerca de&#8230;</string>
37    <string name="cerrar">Cerrar sesión</string>
38    <string name="pregunta">Elige el ejercicio para hoy</string>
39    <string name="error_list_r">NO HAY RUTINAS</string>
40
41    <!-- ACERCA DE -->
42    <string name="acercade_mensaje">No nos hacemos responsables del daño
          que se pudiera producir por la realización de estos ejercicios.
43 En cualquier caso, consulte antes a un profesional.</string>
44
45    <!-- PANTALLA CONFIGURACION CUENTA -->
46    <string name="config_cuenta">Configuración de la cuenta</string>
47    <string name="modifica_info">Modifica la información de usuario</
          string>
48    <string name="bmodificar_info">Modificar cuenta de usuario</string>
49    <string name="cambia_pass">Cambia la contraseña</string>
50    <string name="bcambiar_pass">Cambiar contraseña</string>
51    <string name="elimina_cuenta">Elimina la cuenta</string>
52    <string name="beliminar_cuenta">Eliminar cuenta</string>
53
```

```
54      <!-- PANTALLA MODIFICACION CONTRASEA -->
55      <string name="intro_nueva_c">Introduce la nueva contraseña</string>
56      <string name="repita_c">Repita la nueva contraseña</string>
57      <string name="modificar_contrasenia">Guardar contraseña</string>
58
59      <!-- PANTALLA CONTRASENIA OLVIDADA -->
60      <string name="info_olvido">Le enviaremos una nueva contraseña a su e-
            mail.</string>
61      <string name="enviar">Enviar</string>
62
63      <!-- ELIMINACION RUTINA -->
64      <string name="confir_eliminacion">¿Desea eliminar de su lista?</
            string>
65
66      <!-- PANTALLA GRAFICA RENDIMIENTO -->
67      <string name="t_entreno">Tiempos de entrenamiento</string>
68      <string name="error_list_g">NO HAY DATOS DE LA SEMANA</string>
69      <string name="dia_sem">Dia de la semana</string>
70      <string name="tiempo">Tiempo en minutos: </string>
71
72      <!-- PANTALLA LISTA DE EJERCICIOS -->
73      <string name="nom_rutina">Nombre de la rutina</string>
74      <string name="desc_rutina">Descripcion de la rutina</string>
75      <string name="error_list_e">NO HAY VIDEO</string>
76      <string name="hecho">HECHO</string>
77      <string name="imag">Imagen del vídeo</string>
78
79      <!-- PANTALLA MAS RUTINAS de otros usuarios -->
80      <string name="otras">Otras rutinas</string>
81      <string name="otras_rutinas">Rutinas disponibles de otros usuarios</
            string>
82      <string name="error_list_o">NO HAY RUTINAS DISPONIBLES DE OTROS
            USUARIOS</string>
83      <string name="bcargar_nuevas">Cargar nuevas rutinas</string>
84
85      <!-- PANTALLA NUEVAS RUTINAS descargadas en Download -->
86      <string name="nuevas">Nuevas Rutinas</string>
87      <string name="nuevas_rutinas">Nuevas rutinas disponibles en la
            carpeta Download</string>
88      <string name="error_list_n">NO HAY NUEVAS RUTINAS EN EL DISPOSITIVO</
            string>
89      <string name="bcargar">Cargar nueva rutina</string>
90
91      <!-- PANTALLA REPRODUCCION -->
92      <string name="sig">Siguiente</string>
93      <string name="comenzar">Comenzar</string>
94      <string name="terminado">Terminado</string>
95      <string name="repet">nºrepeticiones</string>
96      <string name="descripcion">Descripcion</string>
97
98  </resources>
```

Bibliografía

[1] F.Ableson,R.Sen,C.King. Android. Guía para desarrolladores, 2011.

[2] J.R.Lequerica. Desarrollo de aplicaciones para Android, 2015.

[3] Apuntes de la asignatura bases de datos 5º Ingeniería de telecomunicación, 2011-2012.

[4] V.S.Rao,T.M.Krishna. «A Design of Mobile Health for Android Applications», American Journal of Engineering Research (AJER)e-ISSN : 2320-0847 p-ISSN : 2320-0936Volume-03, Issue-06, pp-20-29, 2014.

[5] S.Lee. «Creating and Using Databases for Android Applications», International Journal of Database Theory and ApplicationVol. 5,No. 2, June, 2012.

[6] «Java esencial - Diploma de Especialización en desarrollo de aplicaciones para Android» [En línea]. Available: http://www.androidcurso.com/index.php/tutoriales-java-esencial. [Último acceso: 2015].

[7] «Tutoriales: Android Fundamentos - Diploma de Especialización en desarrollo de aplicaciones para Android» [En línea]. Available: http://www.androidcurso.com/index.php/tutoriales-android-fundamentos. [Último acceso: 2015].

[8] «Android: Programación de aplicaciones» [En línea]. Available: http://androideric.blogspot.com.es/. [Último acceso: 2015].

[9] «AprendeAndroid.com - Crear una Base de Batos SQLite Android» [En línea]. Available: http://www.aprendeandroid.com/l5/sql1.htm. [Último acceso: 2015].

[10] «Bases de Datos en Android (I): Primeros pasos | sgoliver.net» [En línea].

Available: http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/. [Último acceso: 2015].

[11] «Bases de Datos en Android (II): Insertar/Actualizar/Eliminar | sgoliver.net» [En línea]. Available: http://www.sgoliver.net/blog/bases-de-datos-en-android-ii-insertaractualizareliminar/. [Último acceso: 2015].

[12] «Newest Questions - Stack Overflow» [En línea]. Available: http://stackoverflow.com/questions/. [Último acceso: 2015].

[13] «Consultas Técnicas Programación Android - sgoliver.net foro» [En línea]. Available: http://www.sgoliver.net/foro/viewforum.php?f=5&sid=dc63856a1b431c1326f9797c1aee1e5 [Último acceso: 2015].

[14] «Programmatically extract a zip file using Java» [En línea]. Available: http://www.codejava.net/java-se/file-io/programmatically-extract-a-zip-file-using-java. [Último acceso: 2015].

[15] «Package Index | Android Developers» [En línea]. Available: http://developer.android.com/intl/es/reference/packages.html. [Último acceso: 2015].

[16] «¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales» [En línea]. Available: http://www.genbetadev.com/seguridad-informatica/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales. [Último acceso: 2015].

[17] «Java (lenguaje de programación) - Wikipedia, la enciclopedia libre» [En línea]. Available: https://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29. [Último acceso: 2015].

[18] «Androidplot | A Simple XY Plot | » [En línea]. Available: http://androidplot.com/docs/a-simple-xy-plot/. [Último acceso: 2015].

[19] «Android Chart using AndroidPlot - Java Tutorial Blog» [En línea]. Available: http://javapapers.com/android/android-chart-using-androidplot/. [Último acceso: 2015].

[20] «JSON Syntax» [En línea]. Available: http://www.w3schools.com/json/json_syntax.asp. [Último acceso: 2015].

[21] «JSON» [En línea]. Available: http://www.json.org/json-es.html. [Último acceso: 2015].

[22] «API de Java para procesamiento JSON: Introducción a JSON» [En línea]. Available: http://www.oracle.com/technetwork/es/articles/java/api-java-para-

json-2251318-esa.html. [Último acceso: 2015].

[23] «Lenguaje unificado de modelado - Wikipedia, la enciclopedia libre» [En línea]. Available: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado. [Último acceso: 2015].

[24] «Tutorial de UML» [En línea]. Available: http://users.dcc.uchile.cl/ psalinas/uml/introduccion.html. [Último acceso: 2015].

[25] «websequencediagrams» [En línea]. Available: https://www.websequencediagrams.com/. [Último acceso: 2015].

[26] «ObjectAid UML Explorer - Home» [En línea]. Available: http://www.objectaid.com/. [Último acceso: 2015].

[27] «Desarrollo y Seguridad de Aplicaciones Web y Móviles: ¿Cómo mejorar el "hashing" de las claves de acceso sin molestar al usuario?» [En línea]. Available: http://tecnologiasweb.blogspot.com.es/2010/12/como-mejorar-el-hashing-de-las-claves.html. [Último acceso: 2015].

[28] «Encriptar y guardar contraseñas en base de datos - Blog - Arume» [En línea]. Available: http://www.arumeinformatica.es/blog/encriptar-y-guardar-contrasenas-en-base-de-datos/. [Último acceso: 2015].

[29] «Manejo de bases de datos en Android I | Androideity» [En línea]. Available: http://androideity.com/2011/10/12/manejo-de-bases-de-datos-en-android-i/. [Último acceso: 2015].

[30] «Bases de datos en aplicaciones móviles | Rodo Bautista - Academia.edu» [En línea]. Available: http://www.academia.edu/10488182/Bases_de_datos_en_-aplicaciones_m%C3%B3viles. [Último acceso: 2015].

[31] «Desarrollo seguro de aplicaciones para dispositivos móviles» [En línea]. Available: https://www.incibe.es/blogs/post/Seguridad/BlogSeguridad/Articulo_-y_comentarios/desarrollo_seguro_de_aplicaciones_para_dispositivos_moviles. [Último acceso: 2015].

[32] «¿Qué es Android?» [En línea]. Available: http://www.xatakandroid.com/sistema-operativo/que-es-android. [Último acceso: 2015].

[33] «SQLite: La Base de Datos Embebida | SG» [En línea]. Available: http://sg.com.mx/revista/17/sqlite-la-base-datos-embebida#.VmR5q4T8s8q. [Último acceso: 2015].

[34] «Las 10 mejores aplicaciones de fitness - AndroidPIT» [En línea]. Available: http://www.androidpit.es/mejores-aplicaciones-fitness. [Último acceso: 2015].

[35] «18 best Android fitness apps and workout apps» [En línea]. Available: http://www.androidauthority.com/best-android-fitness-apps-and-workout-apps-567999/. [Último acceso: 2015].

[36] «SQLite Home Page» [En línea]. Available: https://www.sqlite.org/. [Último acceso: 2015].

[37] «Tutorial de Android SQLite para Principiantes» [En línea]. Available: https://blog.udemy.com/tutorial-de-android-sqlite-para-principiantes/. [Último acceso: 2015].

[38] «Leer json desde directorio raw en android | Mis Snippets» [En línea]. Available: https://carmazone.wordpress.com/2014/09/07/leer-json-desde-directorio-raw-en-android/. [Último acceso: 2015].

[39] «Parsear Datos JSON En Android Con JsonReader Y Gson» [En línea]. Available: http://www.hermosaprogramacion.com/2015/01/android-json-parsing/. [Último acceso: 2015].

[40] «Java - Sending Email» [En línea]. Available: http://www.tutorialspoint.com/java/java_sending_email.htm. [Último acceso: 2015].

[41] «Michael Pratt - Seguridad en el almacenamiento de Passwords/Contraseñas» [En línea]. Available: http://www.michael-pratt.com/blog/8/Seguridad-en-el-almacenamiento-de-PasswordsContrasenas/. [Último acceso: 2015].

[42] «Android: Loading files from the Assets and Raw folders | 41 Post» [En línea]. Available: http://www.41post.com/3985/programming/android-loading-files-from-the-assets-and-raw-folders. [Último acceso: 2015].