Memoria TFG

Contador de repeticiones en ejercicios

Autor:
Adrián Montaña Álvarez

Tutores:
Agustín Trujillo Pino
Gustavo Medina del Rosario

Las Palmas, Diciembre 2014

íNDICE

ESTADO ACTUAL Y OBJETIVOS

La actividad física es un importante factor para llevar un estilo de vida saludable, pero mucha gente no siempre dispone del tiempo o medios para llevar acabo ciertas actividades físicas, ya sean deportes de equipo, en solitario, musculación, atletismo, etc. de forma regular. Es por esto que existe un auge en el desarrollo de rutinas de ejercicios simples y fáciles de realizar. Según se publica en el artículo "HIGH-INTENSITY CIRCUIT TRAINING USING BODY WEIGHT: Maximum Results With Minimal Investment - Klika, Brett C.S.C.S., B.S.; Jordan, Chris M.S., C.S.C.S., NSCA-CPT, ACSM HFS/APT" en el ACSM's Health&Fitness Journal [http://journals.lww.com/acsm-healthfitness/Fulltext/2013/05000/HIGH_INTENSITY_CIRCUIT_TRAINING_USING_BODY_WEIGHT_.5.aspx] es posible obtener una rutina de ejercicio físico adecuada para el día a día en un corto espacio de tiempo gracias a un incremento en la intensidad y número de repeticiones. A raíz de esto surgió el "7 minute workout" que incluye la realización de 12 ejercicios sin necesidad de equipamiento en solamente siete minutos.

Actualmente existe una gran variedad de apps para realizar ejercicios (basadas en su gran mayoría en la metodología "7 minute workout"), pero en su mayor parte contienen únicamente variaciones de la rutina de 12 ejercicios de forma predeterminada y no utilizan reconocimiento de movimiento para contabilizar las repeticiones, simplemente generan un ritmo a seguir como un metrónomo. La mayoría de estas aplicaciones no permiten la modificación de ejercicios o la creación de rutinas a medida para satisfacer necesidades adicionales de los diferentes usuarios. Un ejemplo de esto es la aplicación "7-min" (http://www.7-min.com/) disponible en formato aplicación Web y aplicación para iPhone/iPad, que, siguiendo una serie predeterminada de ejercicios y repeticiones permite al usuario realizar una rutina de trabajo físico en siete minutos.

Los objetivos que se pretenden cubrir con este trabajo es crear una aplicación que permite crear rutinas personalizadas para cada usuario; y reconocer las repeticiones de cada ejercicio mediante la detección de movimientos para llevar la cuenta. De esta forma no solo se puede realizar la rutina de "7 minute workout" sino que se permite una mayor flexibilidad a la hora de crear las rutinas de trabajo, desarrollando así una aplicación

JUSTIFICACIÓN DE LAS COMPETENCIAS ESPECÍFICAS CUBIERTAS

TFG01. Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas

El proceso de la creación de una solución software para el problema propuesto ha sido documentada para la creación de esta memoria para su posterior evaluación y defensa ante un tribunal. Para ello se han intentado integrar todas las competencias relevantes adquiridas en las enseñanzas de la titulación del Grado en Ingeniería Informática.

CII02. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

Durante las fases de análisis, diseño desarrollo y prueba del proyecto se ha intentado obtener la solución que satisfaga el mayor número de necesidades de los usuarios obtenidos mediante el análisis inicial de requisitos y el "feedback" de las pruebas; incrementando así el valor del producto.

T5. Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada. (G1, G2)

Se ha empleado la metodología tradicional de desarrollo de software a lo largo del proyecto, aplicando el enfoque en diferentes partes del proyecto en cada una de las fases dentro del marco de la metodología. La metodología en cascada (tradicional) se compone de cinco fases principales (Análisis, Diseño, Desarrollo, Prueba e Implementación), cada una de ellas cubiertos en la memoria del trabajo.

T8. Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.

Gracias a la alta modularidad del patrón de diseño y arquitectura empleada (Model-View-Controller) y la implementación del principio Abierto-Cerrado (Open/Closed principle) de programación orientada a objetos, se permite la extensión de las diferentes clases sin realizar modificaciones en el trabajo previo. Esto facilita la ampliación y futuro desarrollo de la aplicación manteniendo el código previo y asegurando su correcto funcionamiento, permitiendo así que se desarrollen nuevas funcionalidades sin necesidad de cambiar o rehacer código antiguo.

G4. Transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

Este proyecto se ha realizado con la cotutorización de la empresa Singular Factory SLU en el aspecto técnico del proyecto; y la ayuda de los entrenadores personales del Gimnasio en Forma S.L. en el Paseo de San Antonio durante el desarrollo de los ejercicios y rutinas de ejemplo presentes en la aplicación. Para ello, ha sido necesaria la correcta comunicación de detalles, tanto técnicos como del negocio, tanto con público especializado como no especializado.

APORTACIONES

Como se explica en los objetivos del proyecto, se busca obtener una aplicación que permita al usuario llevar a cabo una rutina de ejercicios, tanto la rutina del "7 minute workout" como rutinas personalizadas. De esta forma se pueden satisfacer necesidades que otras aplicaciones de este tipo no puede en este momento, ya sea por la imposibilidad de realizar alguno de los ejercicios propuestos en la rutina; o un cambio en la intensidad y número de repeticiones a realizar.

Esta aplicación no solo ayuda al usuario a llevar a cabo ejercicio físico en sesiones cortas y cómodas una, o varias veces, al día a su medida; sino que permita la implementación de nuevas rutinas de ejercicios que se desarrollen en un futuro, ya sean similares al "7 minute workout" con su metodología enfocada a una alta intensidad en los ejercicios; o otras rutinas basadas en otro tipo de metodologías.

Se pretende aportar una aplicación para dispositivos Android de planificación de rutinas y ejercicios usando un algoritmo de reconocimiento de movimientos a grandes rasgos mediante el uso de puntos de control de valores de vectores de aceleración absolutos.

HERRAMIENTAS DE DESARROLLO

Para llevar a cabo este proyecto se han utilizado las siguientes herramientas de desarrollo, entornos de programación y aplicaciones comerciales:

Entorno de programación:

- Eclipse: - http://www.eclipse.org/
- Plugin para Eclipse Android SDK - Android Development Tools (ADT): - http://developer.android.com/sdk/installing/installing-adt.html

Editor de ficheros de texto:

- Notepad++: - http://notepad-plus-plus.org/

Editor de imágenes:

- Adobe Photoshop CS5.

Procesador de datos y gráficos:

- Microsoft Excel.

Desarrollo de prototipos:

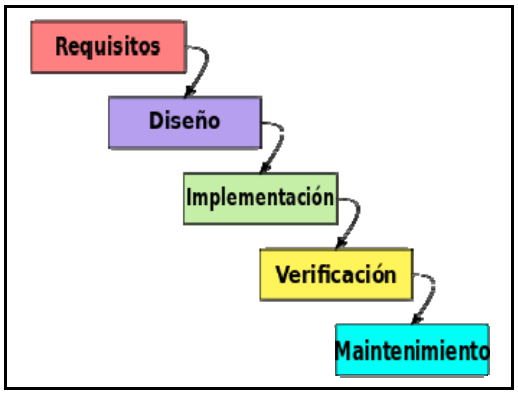
- Pencil: - http://pencil.evolus.vn/

METODOLOGÍA

Para el desarrollo de este proyecto se decidió emplear la metodología clásica basada en el modelo en cascada, ya que funciona particularmente bien con equipos reducidos y productos, o ideas, bien definidas al requerir menor capital y herramientas que otras metodologías para funcionar de manera optima; es fácil de entender y de implementar fase por fase de manera secuencial.

Modelo en cascada:

El modelo en cascada es el enfoque metodológico que ordena las fases del proceso de desarrollo, de tal forma que el inicio de cada fase debe esperar a la finalización de la fase anterior. Al final de cada fase se comprueba el estado de la misma y se determina si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero que se originó y es la base de todos los demás modelos de ciclo de vida. La versión original fue propuesta por Winston W. Royce en 1970 y posteriormente revisada por Barry Boehm en 1980 e Ian Sommerville en 1985



ANÁLISIS

Descripción inicial:

" Título: Contador de repeticiones en ejercicios

Resumen:

Aplicación móvil que, ayudada por acelerómetros y giroscopios, de soporte al desarrollo de actividades físicas que necesiten un plan de trabajo basado en repeticiones y series. Al APP ayudaría a contabilizar los ejercicios, planificar entrenamientos y gestionar el ritmo de cada serie mediante sonidos."

Teniendo en cuenta la descripción inicial del proyecto, investigar el ámbito del problema, posibles necesidades y diversas reuniones se alcanzó una lista definitiva de requisitos necesarios para completar la solución. Estos requisitos no solo se ven influidos por el ámbito del problema, sino por las necesidades y restricciones del ámbito tecnológico.

Requisitos:

Lista de requisitos, tanto funcionales como no funcionales, esenciales para la elaboración de la solución:

Requisitos funcionales:

- Acceder a una lista de ejercicios, organizados por tipos.
- Acceder a la información detallada de cada ejercicio.
- Hacer sesiones contabilizadas de repeticiones de ejercicios (repeticiones libres o con un número objetivo de repeticiones).
- Usar señales sonoras al contabilizar las repeticiones para mantener un ritmo.
- Crear rutinas de ejercicios (una serie ordenada de repeticiones de ejercicios)
- Eliminar rutinas.
- Editar rutinas de ejercicios existentes (ordenar, añadir y eliminar ejercicios y repeticiones).
- Realizar rutinas de ejercicios completas.
- Acceder a la información detallada de cada rutina.

Requisitos no funcionales:

- La solución ha de ser una aplicación para dispositivos móviles (App).
- Solo compatible con dispositivos Android (API level 11 en adelante)

Una vez determinados todos los requisitos del sistema es necesario analizar y determinar quienes son las figuras involucradas en la interacción con la aplicación tanto de manera directa (Actores Principales), como de manera indirecta (Actores Secundarios).

Actores Principales:

- Ejercitador - usuario que utiliza la aplicación para ejercicios y rutinas.

Definición de casos de uso:

Una vez aprobados todos los casos de uso se procede al refinamiento de los requisitos funcionales en casos de uso. Los casos de uso describen los pasos necesarios para llevar a cabo una tarea incluyendo el estado inicial del sistema (precondición) necesario para realizar la tarea y el estado en el que queda el sistema tras realizar la tarea (poscondición). De esta forma se logra describir la secuencia de interacciones que se han de llevar a cabo entre el actor principal y el sistema.

Descripción formal de los casos de uso:

Cada caso de uso ha de ser definido en detalle, evitando el uso de lenguaje técnico ya que han de ser comprendidos y confirmados por el cliente. Cada caso de uso incluye un flujo de actividad principal que describe lo describe y un número de posibles flujos de actividad alternativos.

Especificaciones de los casos de uso de la aplicación:

Sección del caso de uso	**Comentario**
Nombre	Realizar ejercicio
Actor primario	Ejercitador
Otros actores	-
Precondición	Existe el ejercicio.
Poscondición	Repeticiones del ejercicio contabilizadas.
Escenario de éxito principal	1. Se selecciona el ejercicio. 2a. Se ajusta el número de repeticiones objetivo. 3a. Se contabilizan repeticiones hasta alcanzar el número objetivo de repeticiones para el ejercicio.
Flujo alternativo	2b. Se selecciona el modo de repeticiones libres. 3b. Se contabilizan repeticiones hasta que el usuario decida parar.
Otros requisitos	-

Sección del caso de uso	Comentario
Nombre	Realizar rutina
Actor primario	Ejercitador
Otros actores	-
Precondición	Existe la rutina y los ejercicios que la componen.
Poscondición	Repeticiones de cada ejercicio de la rutina contabilizadas.
Escenario de éxito principal	1. Se selecciona la rutina. 2. Se carga el ejercicio de la rutina con sus repeticiones objetivo. 3. Se contabilizan repeticiones hasta alcanzar el número objetivo de repeticiones para el ejercicio. 4a. Si no quedan más ejercicios en la rutina, se finaliza la rutina.
Flujo alternativo	4b. Si quedan más ejercicios se repite desde el paso 2 con el siguiente ejercicio.
Otros requisitos	-

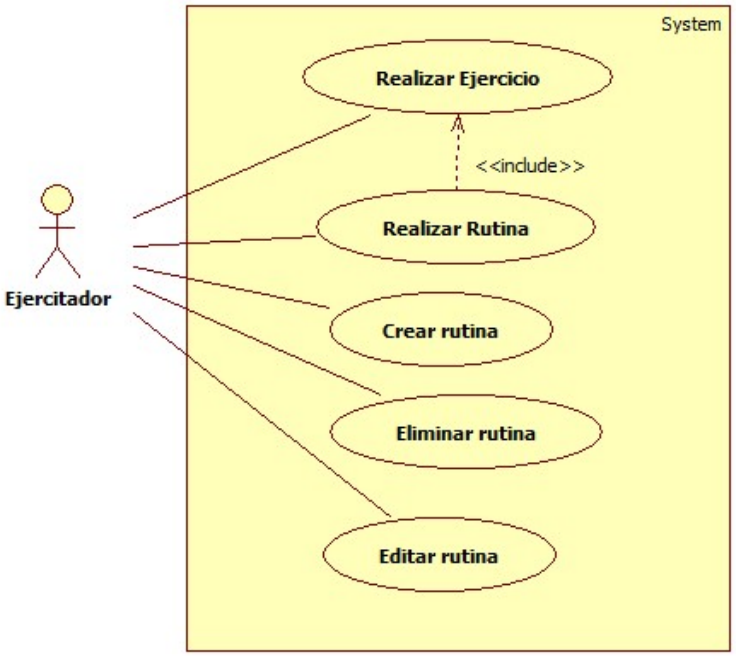
Sección del caso de uso	Comentario
Nombre	Crear rutina
Actor primario	Ejercitador
Otros actores	-
Precondición	Existen los ejercicios.
Poscondición	Rutina creada
Escenario de éxito principal	1. Se introduce el nombre de la nueva rutina. 2a. Se añaden ejercicios. 3. Se selecciona el número de repeticiones
Flujo alternativo	2b. Se elimina un ejercicio previamente añadido. 2c. Se reordenan los ejercicios
Otros requisitos	-

Sección del caso de uso	Comentario
Nombre	Eliminar rutina
Actor primario	Ejercitador
Otros actores	-
Precondición	Existe la rutina.
Poscondición	Rutina eliminada.
Escenario de éxito principal	1. Se selecciona la rutina a eliminar. 2. Se elimina la rutina.
Flujo alternativo	
Otros requisitos	-

Sección del caso de uso	Comentario
Nombre	Editar rutina
Actor primario	Ejercitador
Otros actores	-
Precondición	Existe la rutina.
Poscondición	Rutina editada
Escenario de éxito principal	1. Se edita el nombre de la rutina. 2a. Se añaden ejercicios. 3. Se selecciona el número de repeticiones
Flujo alternativo	2b. Se elimina un ejercicio previamente añadido. 2c. Se reordenan los ejercicios
Otros requisitos	-

Diagramas de casos de uso:

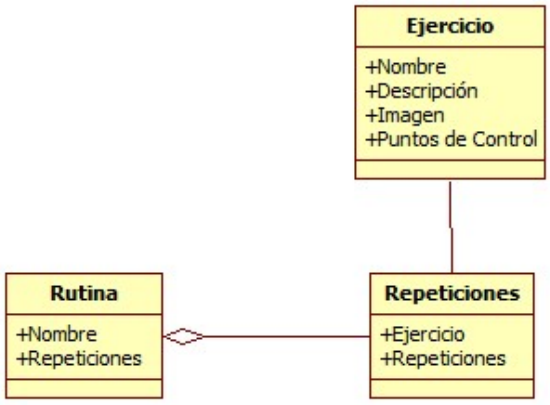
Diagrama de interacción entre actores y casos de uso dentro del límite del sistema:



Entidades:

Un ámbito de negocio existen una serie de entidades que se definen por sus características y su relación con otras entidades. Estas entidades definen como funciona un negocio y como los actores interactúan con ellos.

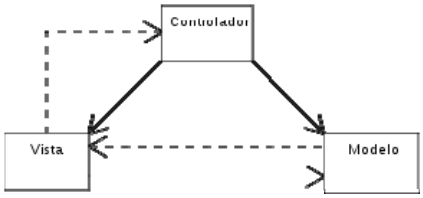
La lógica del problema está compuesta por tres clases principales: rutinas compuestas por grupo de repeticiones, a su vez compuestas por un ejercicio y el número objetivo de repeticiones.



DISEÑO

Arquitectura de la solución:

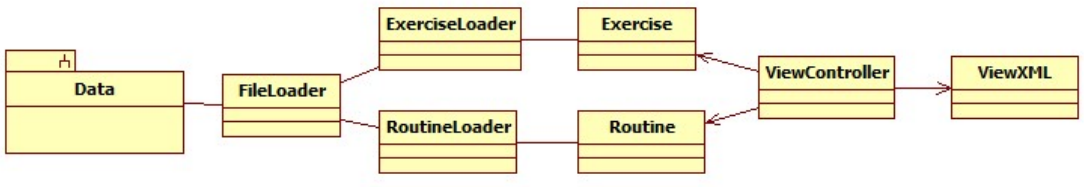
La arquitectura de la aplicación se implementará usando el patrón de arquitectura de software MVC (Model-View-Controller) para separar los datos, la lógica de negocio y la interfaz gráfica de usuario.



Esta separación de responsabilidades permite el diseño modular de las distintas partes que componen el sistema. De esta forma cada parte es totalmente independiente y puede ser sustituida por otro módulo que realiza el mismo propósito de una manera distinta pero que usa la misma interfaz sin afectar a los demás componentes de dependen de este. De esta forma se asegura la longevidad de la solución al permitir una manera fácil para poder expandir los servicios que provee la aplicación reaccionando a cambio en el mercado o ámbito de negocio.

Diseño de la arquitectura para la aplicación:

Siguiendo la arquitectura MVC se diseña la estructura arquitectónica del sistema a desarrollar siguiendo la división de clases en el módulo que le corresponde, las actividades forman parte del módulo del Controlador, las interfaces en XML forman parte del módulo de la Vista; y las entidades forman parte del módulo del Modelo.



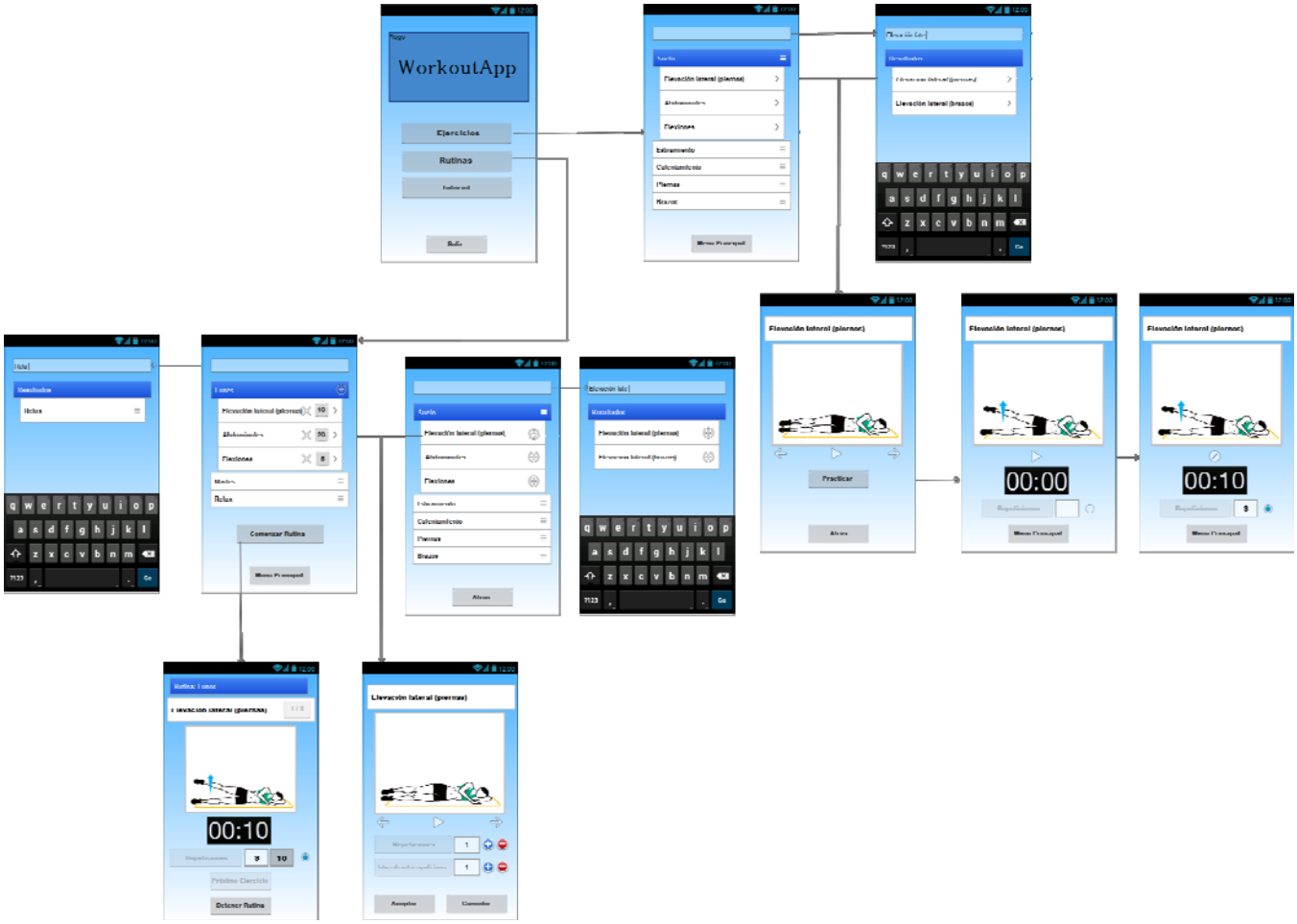
Vistas de la aplicación:

Basándose en los casos de uso se diseñaron las siguientes vistas. Las vista se diseñan inicialmente como parte del "mockup" de la aplicación en Pencil para obtener un prototipo de la aplicación y las transiciones entre vistas.

- Menú principal
- Menú de Ejercicios
- Información de Ejercicio
- Contador de repeticiones de Ejercicio
- Menú de Rutinas
- Información de Rutina
- Contador de repeticiones de Rutina
- Creación/Edición de Rutina

Diseño de la interfaz de usuario:

Diseño inicial de la interfaz de usuario para usar en prototipo:



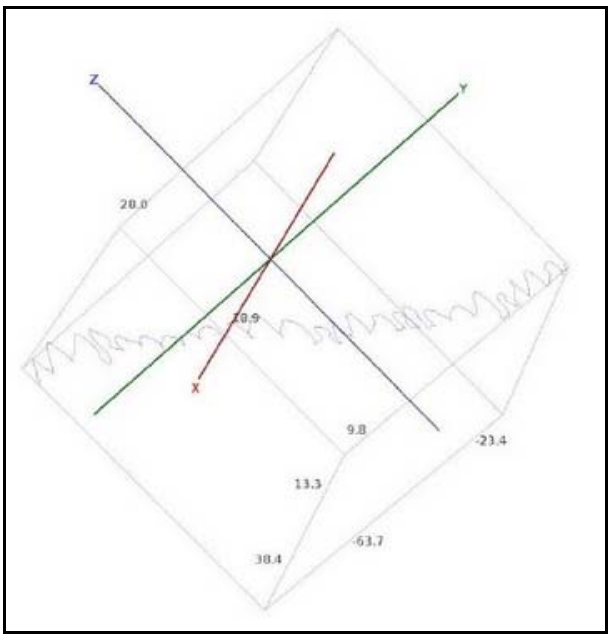
El diseño de la interfaz de usuario se rediseñó después de recibir el "feedback" del prototipo para obtener el diseño que finalmente se desarrollará en XML para la aplicación.

Diseño de algoritmo de reconocimiento de movimientos:

Se plantean dos posibles soluciones para el algoritmo de reconocimiento de movimientos mediante el uso del acelerómetro y giroscopio para la contabilización de repeticiones en los ejercicios:

1. Aceleración lineal tridimensional (3D linear acceleration signal):

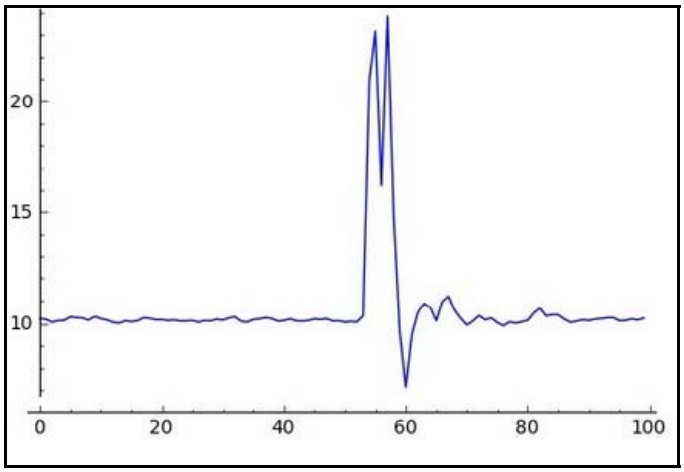
Esta solución se basa en la creación de patrones de movimientos 3D a base de la orientación del dispositivo y los valores de aceleración.



Es necesario el uso de gráficas de señales poco intuitivas y la compensación de la orientación de los valores de aceleración dependiendo de la orientación original del dispositivo, para reconocer cada movimiento se procesan y comparan las señales recibidas por los sensores con el patrón de referencia.

2. Valor absoluto de aceleración:

Esta solución se basa en calcular y utilizar el valor absoluto del vector de aceleración ignorado la dirección y eliminando la contribución de la gravedad al vector. Es particularmente útil en situaciones en la que se desconoce la posición y orientación inicial del dispositivo.



De esta manera, aunque no se controla la dirección del movimiento y por tanto resulta imposible confirmar el movimiento, se simplifica en gran medida el proceso de contabilización de movimientos mediante el registro de una consecución de diferentes tipos de movimientos (bruscos, suaves, quieto, etc). Esta solución no asegura que los movimientos se realizan perfectamente, sino que se sigue un patrón.

Elección del algoritmo a implementar:

Debido a la naturaleza del problema, se decidió que el algoritmo más apropiado es control mediante puntos de control de valores absolutos de aceleración en cada ejercicio. Aunque esto no permite un control fiable del movimiento, la alternativa no solo sería más computacionalmente costosa sino que además requeriría de un entrenamiento previo por parte del usuario, o tener unos altos niveles de tolerancia en los patrones.

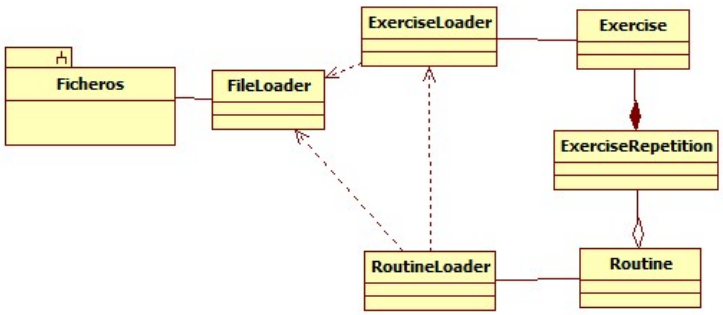
A su vez, el método basado en aceleración absoluta no requiere de información previa del usuario y su entorno, por lo que se puede utilizar ignorando ciertos factores exteriores (ángulos, inclinación, posición, etc).

DESARROLLO

En esta fase se implementan las clases y vistas diseñadas en la fase anterior. Las clases de entidades y controladores están implementadas en el lenguaje de programación orientada a objetos JAVA soportada por el Android SDK; y las vistas están implementadas usando lenguaje XML generados por el IDE Eclipse.

Desarrollo de clases de la lógica del negocio - Entidades:

Las entidades representan el modelo del problema y serán instanciadas y usadas por las clases de Actividades (Controladores). Dado que los datos de los ejercicios y las rutinas han de ser almacenadas en la memoria interna del dispositivo y recuperadas durante el tiempo de ejecución, es necesario crear una serie de clases estáticas encargadas de ello. Existen dos niveles de carga necesarios, la carga inicial de los ficheros de datos de ejercicios desde los assets compilados dentro de la aplicación a ficheros en la memoria interna del dispositivo, donde pueden ser accedidos y manipulados por la aplicación. Y, el segundo nivel de carga desde estos ficheros en la memoria interna a instancias de los objetos que representan dentro de la aplicación.



La clase FileLoader (estática) se encarga del primer nivel de carga durante la inicialización de la aplicación, teniendo en cuenta no sobrescribir rutinas nuevas, editadas o eliminadas.

Las clases ExerciseLoader y RoutineLoader (estáticas) se encargan del segundo nivel de carga, creando instancias de los objetos de ejercicios y rutinas respectivamente y guardando los datos de las rutinas a petición de los controladores.

Las clases Exercise y Routine representan los objetos de ejercicios y rutinas respectivamente. Además, existe una clase auxiliar - ExerciseRepetition - usada por la clase Routine para representar conjuntos de ejercicios y su número de repeticiones.

Interfaces de Clase:

Siguiendo la metodología de programación orientada a objetos, cada clase está formada por un conjunto de atributos, en este caso privados accesibles mediante métodos get() y set() de la clase al seguir la convención de programación en JAVA; y un conjunto de métodos privados (accesibles únicamente por los objetos de la propia clase) y métodos públicos.

Las clases previamente descritas están compuestas por los siguientes conjuntos de atributos privados y métodos públicos:

1. Exercise:

Atributos:

- **String** name
- **String** checkpoingNum
- **float[]** checkpoints
- **String** description
- **String** img

Métodos:

- **public** Exercise(String name,float[] checkpoints, String checkpointNum, String description, String img) - *Constructor.*
- **public String** getName() - *Devuelve el nombre del ejercicio.*
- **public String** getCheckpoingNum() - *Devuelve el número de puntos de control.*
- **public float[]** getCheckpoints() - *Devuelve el vector de puntos de control.*
- **public String** getDescription() - *Devuelve la descripción.*
- **public String** getImg() - *Devuelve la ruta de la imagen del ejercicio.*

2. ExerciseRepetition:

Atributos:

- **Exercise** exercise
- **int** repetitions

Métodos:

- **public** ExerciseRepetition(Exercise exercise, int repetitions) - *Constructor.*
- **public Exercise** getExercise() - *Devuelve el ejercicio.*
- **public int** getRepetitions() - *Devuelve el número de repeticiones.*

3. Routine:

Atributos:

- **String** name
- **ArrayList<ExerciseRepetition>** routine

Métodos:

- **public** Routine(String name) - *Constructor.*
- **public void** addExerciseRepetition(Exercise exercise, int repetitions) - *Añade el ejercicio y las repeticiones a la rutina,*
- **public void** deleteExercise(int index) - *Elimina el ejercicio en esa posición.*
- **public ArrayList<ExerciseRepetition>** getExerciseRepetitions() - *Devuelve el vector de repeticiones de ejercicios.*
- **public String** getName() - *Devuelve el nombre de la rutina.*
- **public void** moveExerciseUp(int i) - *Mueve el ejercicio una posición hacia arriba.*
- **public void** moveExerciseDown(int i) - *Mueve el ejercicio una posición hacia abajo.*
- **public void** repetitionUp(int i) - *Aumenta las repeticiones del ejercicio.*
- **public void** repetitionDown(int i) - *Disminuye las repeticiones del ejercicio.*

4. FileLoader:

Atributos:

- **final static String** listExercises= "listaEjercicios";
- **final static String** listRoutines= "listaRutinas";

Métodos:

- **public static void** loadFiles(Context context) - *Carga los ficheros.*

5. ExerciseLoader:

Métodos:

- **public static LinkedList<String>** loadExerciseList(Context context) - *Carga la lista de ejercicios.*
- **public static Exercise** loadExercise(String file, Context context) - *Carga el ejercicio.*

6. RoutineLoader:

Métodos:

- **public static LinkedList<String>** loadRoutineList(Context context) - *Carga la lista de rutinas.*
- **public static Routine** loadRoutine(String file, Context context) - *Carga la rutina.*
- **public static void** saveRoutine(Routine routine, Context context) - *Guarda la rutina en su fichero de datos.*
- **public static void** deleteRoutine(String oldRoutineName,Context context) - *Elimina la rutina.*

Ficheros de datos:

Los datos de ejercicios y rutinas han de ser almacenados en la memoria interna del dispositivo para poder ser preservados, para ello se contemplaron dos posibles opciones: crear y administrar una base de datos, o utilizar ficheros de texto plano.

Debido al bajo nivel de complexión de datos y relaciones a mantener se optó por usar fichero de texto plano para almacenar los datos, aunque se puede plantear el uso de una base de datos para almacenar los datos si su volumen y complejidad aumenta considerablemente en futuras iteraciones de la aplicación; en ese caso, las clases del módulo "Loader" serán remplazadas.

Los ficheros de ejercicios se identifican por el nombre del ejercicio que representan y contienen los datos necesario para instanciar un objeto de tipo Exercise completo.

Los ficheros de rutinas se identifican por el nombre la de rutina que representan y contienen los datos necesario para instanciar un objeto de tipo Routine y generar el vector de ExerciseRepetitions asociados al mismo.

Además, existen dos ficheros de índice que contienen las rutas de los ficheros de ejercicios y rutinas respectivamente. El fichero de la lista de ejercicios únicamente se actualiza cuando se añaden ejercicios nuevos a la aplicación, mientras que el fichero de la lista de rutinas ha de actualizarse cuando se crea, edita o elimina una rutina para mantener la coherencia de los datos de la aplicación.

Estructura de los ficheros de datos:

La estructura de los ficheros de datos es la siguiente. Nótese que los elementos marcado con un asterisco (*) o un par de paréntesis cuadrados ([ ]) son estructuras que se pueden repetir cuantas veces sean necesarias.

1. Ejercicio:

1	Nombre
2	Nº de puntos de control
3	[Punto de control]
4	Descripción
5	Ruta de imagen

2. Lista de Ejercicios:

*	Grupo
	[Ejercicio]

3. Rutinas:

*	Ejercicio
	Repeticiones

4. Lista de Rutinas:

*	Rutina

Reconocimiento de movimientos:

Para poder reconocer los movimientos detectados por el dispositivo como repeticiones de un ejercicio es necesario calcular primero los valores necesario para realizar la comparación de patrones y como se obtiene dicho patrón a partir de un movimiento.

Cálculos para la obtención del valor absoluto de la aceleración:

El acelerómetro registra valores de aceleración de los tres ejes X, Y, Z. Para poder calcular el valor absoluto del vector es necesario combinar los tres vectores y restar la contribución de la gravedad (g).

$$a = \sqrt{x^2 + y^2 + z^2} - g$$

No obstante, esta formula depende de que la orientación del dispositivo sea constante. Para evitar esto es necesario eliminar la contribución de la gravedad en cada eje antes de combinarlos. Para ello se ha de calcular la contribución de la gravedad de forma dinámica usando la siguiente formula:

gx= (alpha*gx) + ((1-alpha)*X)

siendo alpha = 0.8

La constante alpha se calcula:

alpha = t / (t + dt)

siendo t una constante de tiempo para el filtrado por paso bajo.

$$a = \sqrt{x_l^2 + y_l^2 + z_l^2} \qquad \begin{aligned} x_l &= x - g_x \\ y_l &= y - g_y \\ z_l &= z - g_z \end{aligned}$$

Codificación de la solución:

Es necesario codificar estos cálculos en lenguaje JAVA para poder calcularlos de forma dinámica durante el tiempo de ejecución de la aplicación para poder contabilizar las repeticiones correctas del ejercicios que se ha de llevar a cabo.

Así es como se codifican las ecuaciones previamente explicadas usando los datos obtenidos por el acelerómetro del dispositivo:

```
final float alpha = 0.8;

gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];

linear_acceleration[0] = event.values[0] - gravity[0];
linear_acceleration[1] = event.values[1] - gravity[1];
linear_acceleration[2] = event.values[2] - gravity[2];

float absolute=(float) (Math.sqrt((x*x)+(y*y)+(z*z)));
```

Algoritmo de reconocimiento de movimientos:

Una vez convertidos los vectores de aceleración en valores absolutos es posible analizar la composición de movimientos de cada ejercicio usando gráficas para poder generar el conjunto de puntos que más adecuadamente describa el patrón de movimiento necesario para considerarse completada una repetición del mismo. Este conjunto de puntos de control ha de ser probado y afinado hasta obtener resultados de detección consistentes con un nivel de tolerancia aceptable.

Este proceso se lleva a cabo con cada ejercicio que se va a implementar en la aplicación. A continuación se puede ver un ejemplo de este proceso llevado a cabo con uno de los ejercicios.

Ejemplo de implementación de ejercicio: Bicep Curl

En este ejemplo se puede observar el proceso que se realiza para obtener los datos que conforman el fichero de datos del ejercicio "Bicep Curl"

Nombre:

Bicep Curl

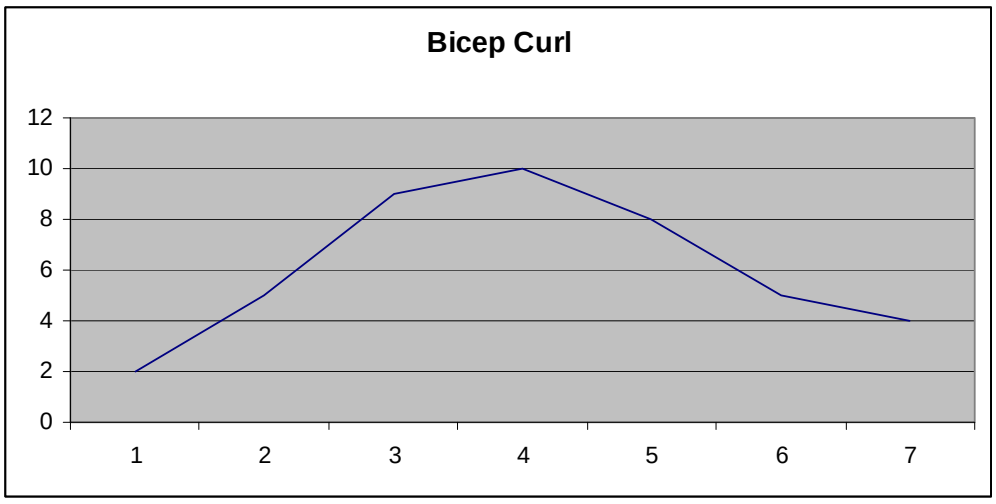
Descripción del ejercicio:

Flex your arm raising the dumbbell, then extend the arm returning to the start position and repeat with the other arm.

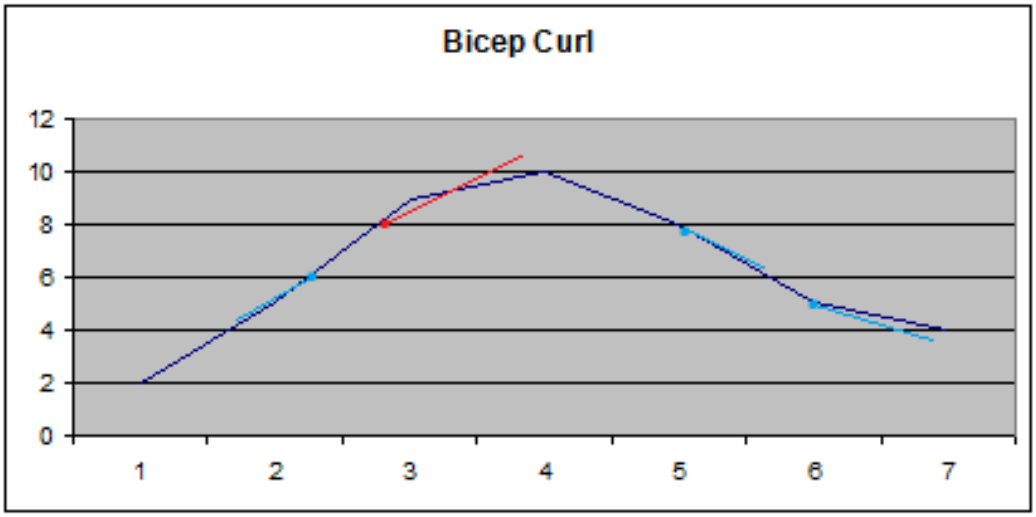
Imagen:

Gráfica del patrón de aceleración:

La siguiente gráfica se obtiene a partir de los valores de aceleración obtenidos durante la realización de una repetición del ejercicio:



Una vez obtenido la gráfica, se seleccionan los puntos de control y si es necesario sobrepasar o no el umbral. Sobrepasar el umbral implica que se requiere una aceleración ascendente, mientras que lo contrario implica una disminución de la aceleración del dispositivo.



En el caso del "Bicep Curl" se seleccionan los siguientes puntos de control para representar la curva generada al realizar el ejercicio:

1. $a < 6ms^2$
2. $a > 8ms^2$
3. $a < 7{,}5ms^2$
4. $a < 5ms^2$

La tolerancia del reconocimiento de movimientos se puede manipular aumentando o disminuyendo el número de puntos de control a usar. No obstante, aunque a mayor número de puntos de control, mayor la precisión del reconocimiento, se limita cada vez más el margen de error de cada usuario al realizar el ejercicio.

Desarrollo de vistas y controladores:

Siguiendo el patrón de diseña para dispositivos Android, cada vista se desarrolló en XML en la carpeta de "layout" del proyecto y para cada una de ellas existe una clase que la controla y extiende la clase "Activity". Las diferentes vistas son cargadas por los controladores mediante la instanciación de la clase Intent en la cual se inyectan los objetos que se han de pasar de vista en vista.

Cada vista ha sido generada por el IDE Eclipse usando una herramienta de creación y edición de documentos XML integrada en el entorno de desarrollo y el Android SDK, y editadas en su formato de texto cuando fuese necesario.

El diseño de cada vista es una adaptación del diseño obtenido en la fase de Diseño dentro de los límites que permite el lenguaje XML y la necesidad de incluir los componentes necesario para que la clase de actividad (controlador) de cada vista pueda llevar a cabo su función.

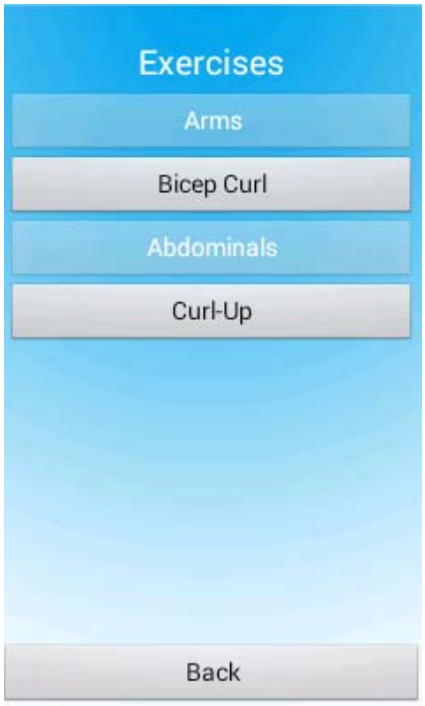
A continuación se puede ver la implementación de cada una de las vistas de la aplicación y el funcionamiento asociado a cada una de ellas:

1. MainMenuActivity - Menu Principal

Es la pantalla inicial de la aplicación y permite el acceso al menú de Ejercicios y de Rutinas instanciando ExerciseMenuActivity y RoutineMenuActivity respectivamente.

Esta es la actividad encargada de controlar la carga inicial de los ficheros a la memoria interna de la aplicación sin sobrescribir aquellos ficheros de rutinas ya existentes, para mantener los cambios cada vez que se lanza la aplicación.

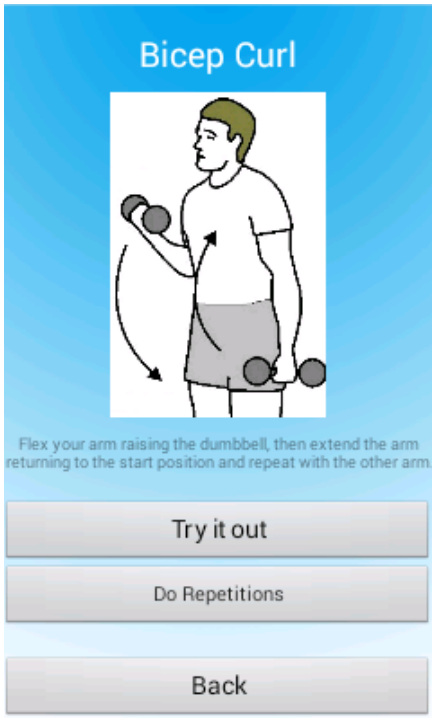
2. ExerciseMenuActivity - Menu de Ejercicios



Esta es la pantalla del Menú de Ejercicios que contiene una lista de ejercicios, ordenado por grupos.

Cada ejercicio está vinculado a un botón que al ser pulsado lleva a la vista de información detallada del ejercicio instanciando ExerciseInfoActivity, el controlador encargado de carga la información del ejercicio.

3. ExerciseInfoActivity - Información de cada Ejercicio



Esta es la pantalla de información detallada del ejercicio, donde se puede observar la imagen y descripción del ejercicio además de dos botones.

Ambos botones llevan a la vista de contador de repeticiones en ejercicio, pero cada uno en un modo diferente: "Try it out" instancia el modo de repeticiones libres del ejercicio, mientras que "Do repetitions" instancia el modo de repeticiones objetivo.

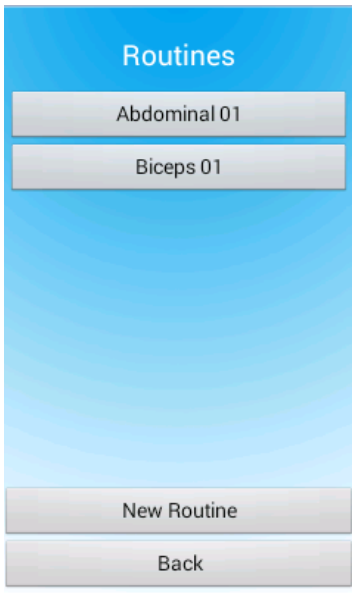
4. ExerciseCounterActivity - Contador de repeticiones de Ejercicio

Estas son las pantallas de Contador de repeticiones de ejercicios en sus dos posibles modalidades. Esta actividad es la responsable de contabilizar las repeticiones del ejercicio que se realiza, comprobando que el patrón de movimiento se corresponde con los puntos de control del ejercicio generando una señal sonora y aumentando el contador de repeticiones cada vez que se completa una repetición

El modo de repeticiones objetivo permite al usuario fijar un número objetivo de repeticiones a realizar antes de comenzar con el ejercicio. Al alcanzar ese número de repeticiones se generará una señal sonora diferente y se finalizará el ejercicio.

El modo de repeticiones libres no tiene un número limite de repeticiones, simplemente permite al usuario realizar cuantas repeticiones desee del ejercicio. Es, principalmente un modo de prueba para el usuario.

5. RoutineMenuActivity - Menu de Rutinas

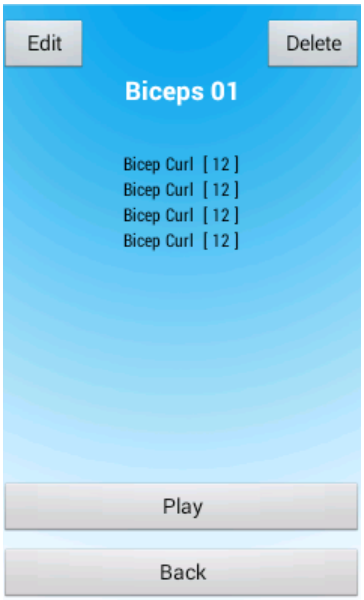


Esta es la pantalla del menú de rutinas que contiene la lista de rutinas disponibles en la aplicación y permite la creación de rutinas nuevas.

Cada rutina está vinculada a un botón que al ser pulsado lleva a la vista de información detallada de la rutina instanciando RoutinaInfoActivity, el controlador encargado de carga la información de la rutina.

El botono "New Routine" instancia la actividad RoutineEditActivity en la modalidad de creación.

6. RoutineInfoActivity - Información de cada Rutina

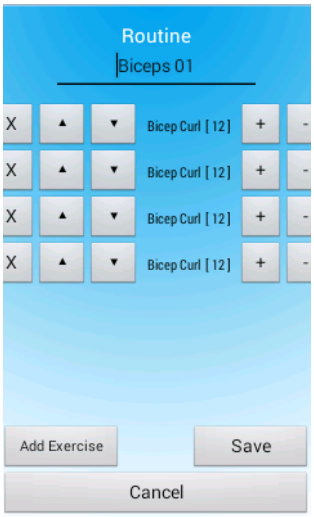


Esta es la pantalla de información detallada de la rutina de ejercicios. Muestra la lista de ejercicios de los que se compone la rutina y el número de repeticiones de cada uno.

En esta pantalla el usuario puede eliminar la rutina, lo que requiere que el controlador elimine el fichero de datos de la rutina y la referencia a la rutina en el índice.

También se puede editar la rutina pulsado el botón "Edit" que instancia la actividad RoutineEditActivity en la modalidad de edición; y comenzar la rutina pulsando el botón "Play" que instancia la actividad RoutineCounterActivity

7. RoutineEditActivity - Editor/Creador de Rutinas

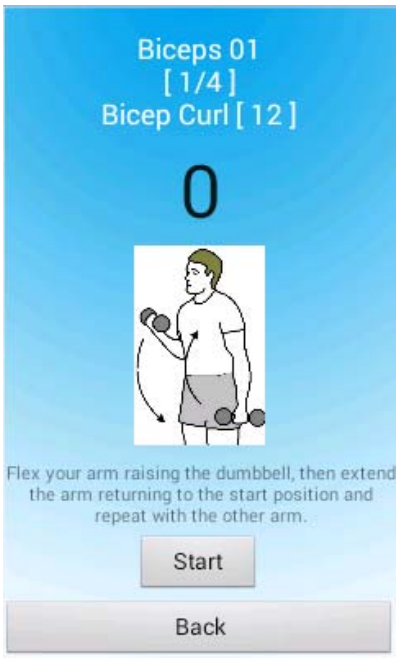


Esta es la pantalla de edición y creación de rutinas, en ambas modalidades funcionan igual, la única diferencia radica en que en la modalidad de edición, la información de la rutina es cargada para que esta se pueda editar; mientras que en la modalidad de creación aparece la misma pantalla y opciones pero esta vez son datos, vacía.

Al guardar los cambios en una rutina se requiere que el controlador sobrescriba estos cambios en el fichero de datos de la rutina y en el índice si fuera necesario.

Lo mismo ocurre si se crea una nueva rutina, sin embargo en este caso no se sobrescribe ningún fichero sino que se crea uno nuevo.

8. RoutineCounterActivity - Contador de repeticiones de Ejercicios de la Rutina



Esta es la pantalla del contador de repeticiones de ejercicios de la rutina. Funciona igual que la pantalla controlada por la actividad ExerciseCounterActivity con la diferencia que en esta pantalla se ofrece información adicional de la rutina (ejercicios restantes, número de repeticiones objetivo, etc.) y en lugar de finalizar la actividad una vez se completa un ejercicio, se carga el siguiente ejercicio y se repite el proceso hasta completar la rutina; en ese momento es cuando se finaliza la actividad.

CONCLUSIONES Y TRABAJOS FUTUROS

La aplicación permite crear y organizar diversos tipos de rutinas de forma fácil y rápida y se ha comprobado que no solo sirve que aquellos usuarios que tienen la necesidad de organizar su tiempo para llevar a cabo un mínimo de ejercicio físico diario, sino gracias a su versatilidad se puede aplicar a cualquier tipo de usuario si se configura de manera apropiada. Además se ha visto el potencial para ser usado como una aplicación auxiliar para llevar la cuenta de repeticiones y series de ejercicios en el ámbito de los gimnasios, aunque esto requiero del desarrollo de nuevas funcionalidades para la aplicación y la incorporación de dispositivos periféricos.

Durante la implementación de los ejercicios se observó una clara debilidad de la aplicación, ya que al usar un dispositivo móvil como sensor los puntos de referencias desde los que se pueden realizar las mediciones están limitados a aquellas zonas del cuerpo en las que se puede sostener el dispositivo, por lo que aquellos ejercicios que depende de movimientos de piernas, cintura o cabeza no pueden ser implementados.

Una vez comprobado el impacto de la aplicación existen ideas para su futura extensión para cubrir más requisitos y servicios aumentando el valor del producto para sus usuario, y, por tanto, aumentar su longevidad.

Estos son ideas a desarrollar en un futuro si se considera una opción viable:

- Incluir un nuevo actor: El Entrenador, que sería un nuevo tipo de usuario responsable de monitorizar el proceso de un grupo de usuarios en la realización de sus rutinas, y crear nuevas o modificarlas según las necesidades del usuario. Esto requiere de la colaboración con entrenadores personales y personal de gimnasios para desempeñar la labor.
- Incluir una forma de ajustar la tolerancia del reconocimiento de movimientos de forma dinámica para facilitar el uso de la aplicación a todos sus usuarios.
- Desarrollar un sistema de dispositivos periféricos conectados al dispositivo principal, idealmente de forma inalámbrica, para poder desarrollar perfiles para ejercicios que requieren más de un punto de referencia para reconocerlo, mejorar el actual algoritmo de reconocimiento e implementar nuevos ejercicios previamente no desarrollados al depender del movimiento de extremidades o partes de cuerpo en los que no es posible colocar el dispositivo original.

FUENTES DE INFORMACIÓN

Información del entorno del problema:

1. Artículos de ejercicios y rutinas:

- NYTimes - The Advanced 7-Minute Workout: http://well.blogs.nytimes.com/2014/10/24/the-advanced-7-minute-workout/
- 7 Minute Workout. http://www.7-min.com/

Información técnica para el desarrollo de la solución:

1. Entorno y leguaje de desarrollo:

- Android SDK: http://developer.android.com/sdk/index.html
- API del Android SDK: http://developer.android.com/guide/index.html
- Entorno de desarrollo Eclipse: http://www.eclipse.org/

2. Calibración de sensores y Mediciones:

- API de sensores de movimiento (Acelerómetro y giroscopio): http://developer.android.com/guide/topics/sensors/sensors_motion.html
- Cálculos de movimientos: http://www.slideshare.net/paller/better-motion-control-using-accelerometergyroscope-sensor-fusion

MANUAL DE USUARIO Y SOFTWARE

El manual de usuario explica al usuario, de forma sencilla, como funciona la aplicación y le permite familiarizarse con la interfaz de la misma antes de empezar a usarla.

Ver: ANEXO I: Manual de Usuario - ExerciseApp.

ANEXO I: Manual de Usuario - ExerciseApp.