



ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO DE TELECOMUNICACIONES

Título del proyecto:

CERES: Un Juego Serio para el Fomento de la Actividad Física

JORGE LAMPÉREZ ZUBIA
TUTOR: LUIS SERRANO ARRIEZU
PAMPLONA, 18 DE JULIO DE 2014

Resumen

En esta memoria se explicarán los pasos seguidos en las diferentes etapas del desarrollo de una aplicación para móviles Android dentro del proyecto CERES. Esta aplicación en cuestión se trata de una aplicación de salud y permite mediante los sensores propios del móvil llevar la monitorización de la actividad física así como establecer una conexión con diferentes dispositivos (básculas, termómetros, pulsioxímetro. . .) para la obtención de medidas. Estos datos almacenados por la aplicación pueden ser transferidos a un backend para su posterior análisis o pueden examinarse en el propio móvil. Dado que la aplicación se desarrolla dentro del marco de los juegos serios esto hace que resulte motivante y entretenida.

Agradecimientos

Me gustaría agradecer en primer lugar a mi tutor Luis por su atención estos dos últimos años tanto en mi experiencia erasmus como en la realización de este proyecto, a Alfonso por introducirme en el mundo de las aplicaciones y su inestimable ayuda a la hora de abordar este proyecto y a Miguel por sus consejos así como a todos los integrantes del laboratorio que me han hecho más llevadero el día a día.

En segundo lugar a mi familia la cual ha formado parte desde el principio de mi vida en mi formación, en mi educación en la persona que soy hoy en día y que siempre ha estado ahí para apoyarme.

Por último agradecer a mi cuadrilla y equipo de fútbol del Lagunak por esos días de alegrías y desconexión.

Todos en cierta medida habéis colaborado en este proyecto.

Índice

Resumen	i
Agradecimientos	ii
1. Introducción	4
1.1. Hipotesis	4
1.2. Objetivos	5
1.3. Organización de la memoria	5
2. Aplicaciones de actividad física	6
2.1. Health	7
2.1.1. Motivaciones	7
2.1.2. Características	7
2.1.3. Plataformas	7
2.2. Nike + Running	7
2.2.1. Motivaciones	7
2.2.2. Características	8
2.2.3. Plataformas	8
2.3. Fitbit	8
2.3.1. Motivaciones	8
2.3.2. Características	9
2.3.3. Plataformas	10
2.4. Lift	10
2.4.1. Motivaciones	10
2.4.2. Características	10
2.4.3. Plataformas	11
2.5. Beeminder	11
2.5.1. Motivaciones	11
2.5.2. Características	11
2.5.3. Plataformas	12
2.6. Retrofit Me	12
2.6.1. Motivaciones	12
2.6.2. Características	12
2.6.3. Plataformas	13
2.7. RunKeeper	13
2.7.1. Motivaciones	13
2.7.2. Características	14
2.7.3. Plataformas	14
2.8. Runtastic	14
2.8.1. Motivaciones	14
2.8.2. Características	15
2.8.3. Plataformas	15
2.9. Endomondo	15
2.9.1. Motivaciones	15
2.9.2. Características	16
2.9.3. Plataformas	16
2.10. FitLinxx (Web App)	17

2.10.1. Motivaciones	17
2.10.2. Características	17
2.10.3. Plataformas	17
3. Diario de desarrollo	18
3.1. Requerimientos	18
3.2. Diseño	22
3.2.1. Casos de uso	22
3.2.2. Diseño final	23
3.2.3. Componentes	27
3.2.4. Clases	28
3.2.5. Diagramas de secuencia	33
3.3. Implementación	37
3.3.1. Modelo	39
3.3.2. Almacenamiento de datos	40
3.3.3. Axia™	43
3.3.4. LogIn	43
3.3.5. UI	44
3.3.6. Gráficos	46
3.3.7. Dispositivos utilizados y pruebas	46
3.4. Verificación	48
4. Discusión	49
5. Conclusiones y líneas futuras	51
5.1. Conclusiones	51
5.2. Líneas futuras de trabajo	51
5.3. Valoraciones personales	52
Anexos	53
A. Anexo I: Google+ API	53

Índice de tablas

1. Resumen	6
2. Pruebas realizadas en la aplicación.	48

Índice de figuras

1. Imágenes de la aplicación Health.	7
2. SportWatch	8
3. Aplicación Nike+ para Android	9
4. Productos fitbit	9
5. App. Fitbit para Android	10
6. App. Lift para Android	11
7. App. Beeminder para iphone	12
8. App. Retrofit me para iOS	13

9.	App. Runkeeper para Android	14
10.	App. Runtastic me para Android	15
11.	Empresas con las que trabaja Endomondo	16
12.	App. Endomondo para Android.	16
13.	Aparatos FitLinxx	17
14.	Modelo cascada con feedback	18
15.	Diagrama 1.	19
16.	Diagrama 2.	20
17.	Diagrama 3.	20
18.	Diagrama 4.	21
19.	Diagrama 5.	21
20.	Requerimientos.	21
21.	Caso de uso global.	22
22.	Caso de interacción movil usuario.	23
23.	Primera aproximación.	24
24.	Segunda aproximación.	25
25.	Versión beta.	26
26.	Componentes de la aplicación.	27
27.	Diagrama de consulta de nivel y puntuación.	33
28.	Diagrama de lista de tareas y como empezar su ejecución.	34
29.	Diagrama de las tareas realizadas.	34
30.	Diagrama de la vista de evolución.	34
31.	Diagrama de preferencias.	35
32.	Primer diagrama de logIn.	35
33.	Segundo diagrama de logIn.	36
34.	Diagrama de logOut y borrado de cuenta.	36
35.	Patrón Modelo-Vista-Controlador.	37
36.	Diagrama patrón Composite.	37
37.	Patrón observer.	38
38.	Jerarquía de vistas en Android.	38
39.	Modelo de la aplicación.	39
40.	Línea de comandos de la base de datos.	40
41.	Preferencia de usuario.	41
42.	Estructura de la base de datos.	42
43.	Esquema de logIn.	43
44.	Navigatoin drawer.	44
45.	Lista principal	46
46.	Gráfica para medidas de temperatura.	47
47.	Captura del Android SDK.	53
48.	Consola de desarrollador, activando Google+ API	53
49.	Consola de desarrollador, creación de credenciales	54
50.	Consola de desarrollador, obtención del ID Cliente	54
51.	Añadir librería google play services al proyecto	55

1. Introducción

En los últimos años, la obesidad se ha convertido en un problema de salud importante en todo el mundo que afecta a personas de todas las edades [1, 2]. Esta conduce a muchos problemas de salud graves como la diabetes, enfermedades del corazón, la hipertensión y el cáncer. Una forma simple de medir la obesidad es el índice de masa corporal (IMC), esto es el peso de una persona en kilogramos dividido por el cuadrado de la talla en metros. Según la Organización Mundial de la Salud (OMS) una persona con un IMC igual o superior a 30 es considerada obesa y con un IMC igual o superior a 25 es considerada con sobrepeso [1]. Para combatir la obesidad se prescriben diferentes enfoques, tales como el ejercicio físico regular, auto-control y monitorización, consumo de alimentos sanos y disminución de la cantidad de ingesta diaria de alimentos [1, 2, 3, 4]. Entre éstos, el ejercicio físico es considerado como uno de los enfoques más eficaces que ayuda a la quema de calorías adicionales para prevenir la obesidad [5].

Sin embargo, en la mayoría de los casos, no es fácil monitorizar las condiciones de salud de las personas obesas durante el ejercicio y prescribir los niveles de ejercicio apropiados para ellos. Más específicamente, es difícil de medir y registrar los datos de seguimiento a largo plazo de las calorías consumidas durante el ejercicio y analizarlos para obtener información significativa que pueda ayudar a las personas obesas a ser conscientes de sí mismo y estar motivados para la pérdida de peso.

Con el fin de motivar a las personas obesas a hacer ejercicio físico, uno de los enfoques que se proponen en los últimos años es el de utilizar el concepto de juego serio [6, 7] tanto de interior como de exterior que enseñan a la gente acerca de los temas de la salud. Estos pueden animar a la gente de forma implícita a aumentar el nivel de actividad física durante la ejecución, pero no deben inducir el aburrimiento. Los *exergames* (conjunto de pantalla, videoconsola y un software que obliga a los jugadores a hacer actividad física como baile, deporte u otras variedades) de interior más populares son Dance Dance Revolution[8], Wii Fit [9] y Guitar Hero[10] y de exterior el ARQuake[11].

Si bien todo lo anterior funciona, es muy reciente y tiene éxito en la pérdida de peso y su gestión, no es suficiente ya que la obesidad sigue en aumento[4]. Todo este trabajo, en general, estimula la realización de la actividad física para ayudar a prevenir o tratar la obesidad pero para conseguir todo el potencial de un juego serio motivante que soporte el monitoreo en tiempo real de la actividad física y a su vez se reciba una valoración en relación a la intensidad del ejercicio o cambio de niveles de juego hace falta mucho más trabajo.

En este trabajo se presenta un juego serio para un smartphone con sistema operativo Android teniendo en cuenta los aspectos mencionados anteriormente. Este juego serio consiste en recompensar o penalizar dependiendo de una correcta ejecución de las diferentes tareas programadas. La aplicación permite realizar un seguimiento de los usuarios en tiempo real mediante la integración de diferentes tipos de sensores y el perfil médico permite dar una valoración acerca del ejercicio realizado. Todo este comportamiento se pretende comparar con el de diferentes usuarios para obtener una mayor motivación entre ellos.

1.1. Hipotesis

La hipótesis fundamental trata de si es posible implementar una aplicación en el marco de los juegos serios para un smartphone. Si se diese el caso de que se implementase la aplicación otra de las hipótesis a contrastar sería si los juegos serios ofrecen la motivación necesaria para que las personas realicen actividad física.

1.2. Objetivos

El objetivo de este trabajo es la implementación de una aplicación para realizar una evaluación comparativa entre usuarios que monitorizan sus signos vitales sin usar el juego y los que lo hacen a través de un juego serio. Este juego serio debe ser motivante y no debe inducir al aburrimiento.

1.3. Organización de la memoria

Capítulo 2. Aplicaciones de actividad física. Este capítulo trata sobre las diferentes aplicaciones disponibles en el mercado y que han servido de inspiración a la hora de diseñar la aplicación.

Capítulo 3. Diario de desarrollo. En este capítulo se explica cual ha sido la metodología seguida para desarrollar la aplicación y se aborda las partes del diseño e implementación.

Capítulo 4. Discusión. Este capítulo servirá para presentar el debate realizado para justificar las decisiones tomadas a la hora de implementar la aplicación.

Capítulo 5. Conclusiones y líneas futuras. Este capítulo servirá para expresar cuales han sido los resultados de desarrollar la aplicación y por que líneas se puede seguir trabajando para mejorarla.

2. Aplicaciones de actividad física

El desarrollo de aplicaciones de monitorización de la salud o la actividad física para móviles está en auge en estos momentos. Estas aplicaciones nos permiten tener almacenadas todas las mediciones de salud realizadas a través del dispositivo móvil y su posterior visualización posibilita llevar un control exhaustivo de la actividad.

A continuación se realiza un análisis de aplicaciones disponibles en el mercado centrado en:

- **Motivaciones:** con que fin se desarrolla la aplicación.
- **Características:** sensores utilizados, dispositivos extra, redes sociales ...
- **Plataformas:** Android, iOS, WindowsPhone..

La tabla 1 muestra un pequeño resumen de este análisis en donde las columnas nombre e icono determinan la aplicación, la columna motivación resume aclara el porque de su implementación, las columnas sensores, dispositivos extra, redes sociales y obtención de datos de otras app. equivaldrían al apartado de características y la columna plataforma especifica el sistema operativo en el que están realizadas.











Nombre App	Icono	Motivaciones	Sensores	Dispositivos extra	Redes Sociales	Obtienen datos de otras app	Plataforma
Health		Recopilar todos los datos en una aplicación para tener un mejor control sobre ellos.	No	No	No	Fitbit, Nike+	iOS
Nike +		Aplicación para ofrecer un seguimiento a los amantes del running.	GPS	GPS Nike+ para zapatillas, reloj Nike+ SportWatch GPS	Facebook, Twitter, Path, propia	No	iOS , Android
Fitbit		Seguimiento mediante dispositivos de las actividades realizadas por el usuario.	No	Flex,Zip, One, Aria	Propia	No	iOS , Android, Windows Phone
Lift		Ofrecer al usuario micro tareas para la consecución de una tarea objetivo.	No	No	Comunidad Lift	No	iOS , Android
Beeminder		Hacer que el usuario cumpla con sus tareas propuestas aplicando una penalización monetaria si no son realizadas	No	No	No	RunKeeper, Fitbit	iOS , Android
Retrofit Me		Aplicaciones para llevar un seguimiento de pacientes de manera personalizada.		Pago para acceder a la aplicación, Charlas por Skype		Fitbit	iOS , Android
RunKeeper		Aplicación para corredores que quieren un seguimiento normal o avanzado mediante el uso de dispositivos desarrollados.	GPS	GPS Garmin, Pebble	No	Fitbit	iOS , Android
Runtastic		Aplicación para diferentes tipos de actividades físicas, el seguimiento de estas también puede ser controlado por diferentes dispositivos para obtener mejoras a la hora del entrenamiento.	GPS	Monitor de ritmo cardiaco. Runtastic shop	Facebook, Twitter, propia	No	iOS , Android, Windows Phone, Blackberry App World
Endomondo		Aplicación de diferentes tipos de actividades físicas donde se quiere llevar un control del usuario y motivarlo comparando sus marcas con las de sus amigos.	GPS	Obtiene datos de productos Garmin, Polar, Timex	Facebook, propia	No	iOS , Android, Windows Phone, Blackberry App World
FitLinxx	Web	Modelo de negocio en el control de usuario mediante diferentes dispositivos.	No	Pebble, bpmmonitor y báscula			

Tabla 1: Resumen

2.1. Health

2.1.1. Motivaciones

Health se trata de la app que Apple acaba de estrenar para monitorizar la salud. En Apple están convencidos de la importancia de la salud de sus usuarios, algo que se ha demostrado con la gran variedad de dispositivos que están disponibles a la venta tales como pulseras o relojes inteligentes. Estos dispositivos permiten registrar la actividad física, conocer el pulso del usuario, y otros destinados incluso a la medición del nivel de glucosa en sangre o la presión sanguínea. Cuestiones que, sin embargo, dependen de cada aplicación y dispositivo. Es por ello que Apple ha creado Health para poder reunir todos los datos de salud del usuario en un mismo lugar y crear un completo perfil.

2.1.2. Características

HealthKit, un sistema capaz de trabajar con otras aplicaciones independientes como Nike, Fitbit etc. recogiendo los datos que su propia aplicación se encarga de ordenar. Todo ello pasa finalmente a la aplicación Health para crear el perfil completo del usuario. Un lugar donde consultar todos los detalles medidos y recogidos por otras aplicaciones y dispositivos.

Apple está trabajando para que HealthKit pueda colaborar con la Clínica Mayo para comparar y controlar los parámetros de los usuarios.

En la figura 1 se observa el perfil completo del usuario, donde se recopilari a toda la informaci n, y las gr ficas contruidas en base a esos datos.



Figura 1: (a) Perfil del usuario. (b) Gr ficas a partir de los datos.

2.1.3. Plataformas

Esta aplicaci n s lo se encuentra disponible para iOS.

2.2. Nike + Running

2.2.1. Motivaciones

Es la aplicaci n oficial de esta firma norteamericana. Est  enfocada en el mercado deportivo y se centra b sicamente en la actividad de correr.

2.2.2. Características

Contiene lo esencial para llevar a cabo una buena sesión de entrenamiento: contabilización de calorías, distancia (tanto en kilómetros como en millas), recorrido total post-sesión y intervalos de estadísticas medias por kilómetros. Además te da comentarios en formato de audio cuando corres.

También tiene la opción de compartir la sesión con Facebook, Twitter y Path.

Esta app que registra tus carreras y progresos no necesita de sensores externos puesto que utiliza los del propio dispositivo móvil. Para corredores que quieren llevar esta monitorización a un nivel más avanzado del básico con GPS para rastrear las vueltas o la velocidad Nike a fabricado un SportWatch figura 2.



Figura 2: SportWatch

En la siguiente secuencia de imágenes (figura 3) se observa la estructura de la aplicación para Android :

1. La primera imagen muestra un resumen de la carrera.
2. La segunda muestra el uso de las redes sociales.
3. La tercera muestra la pantalla que aparece al estar realizando la carrera.
4. La cuarta es la pantalla mostrada como recompensa al haber corrido 1000 metros
5. La quinta es el recorrido post sesión realizado.
6. La sexta muestra como compartir el recorrido con tus amigos mediante redes sociales.

2.2.3. Plataformas

Esta app tiene versiones tanto para iOS como para Android.

2.3. Fitbit

2.3.1. Motivaciones

“Reforzarte e inspirarte para que lleves una vida más sana y activa”.

En 2007, los fundadores de la empresa, Eric y James, constataron que las tecnologías inalámbricas y de sensores habían avanzado hasta tal punto que podían aportar experiencias increíbles al terreno del ejercicio físico y la salud. Se embarcaron en una aventura con la idea de crear un producto portátil capaz de transformar nuestra manera de movernos.

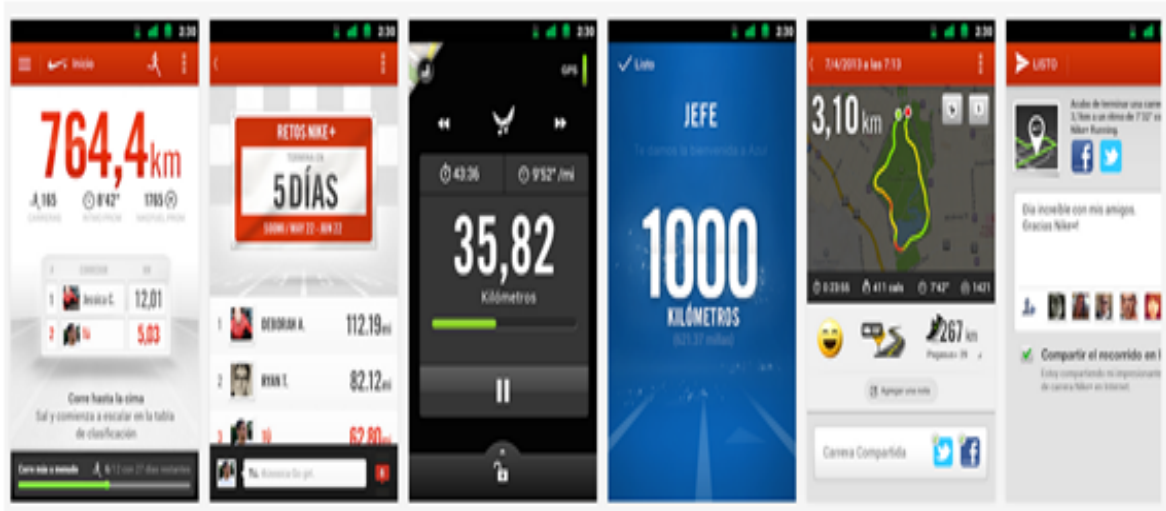


Figura 3: Aplicación Nike+ para Android

2.3.2. Características

Fitbit[12] se sincroniza (fig. 4(a)) de modo inalámbrico con herramientas online y aplicaciones para no perder la motivación. Sin botones, sin cables y sin complicaciones.

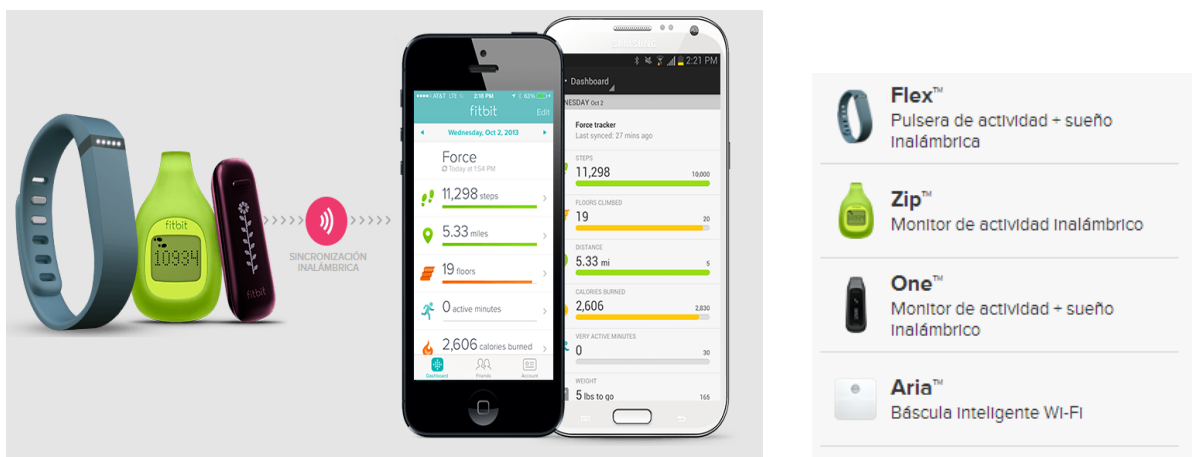


Figura 4: (a) Sincronización de dispositivos (b)Dispositivos disponibles

Esta sincronización se lleva a cabo con el monitor de FitBit en tiempo real. La aplicación de Fitbit utiliza Bluetooth Low Energy para sincronizar los datos; de este modo se pueden observar siempre las últimas estadísticas, tablas y gráficos en el smartphone.

Si el dispositivo Android tiene NFC (Near Field Communication), simplemente tocando el teléfono con la Fitbit Flex iniciará la aplicación automáticamente. No es necesario abrir la aplicación.

Los dispositivos externos que ofrece esta compañía se pueden ver en la figura 4(b), mientras que los dispositivos móviles con los que se puede sincronizar pueden consultarse en [12].

Fitbit dispone de una API que permite a los desarrolladores interactuar con sus datos en sus propias aplicaciones, productos y servicios. En la figura 5 se observa la estructura de la aplicación:

1. La primera imagen muestra la información obtenida por uno de los dispositivos exter-

nos por pantalla.

2. La segunda imagen muestra la interacción en redes sociales.
3. Las imágenes tres y seis muestran la evolución en el tiempo de pasos y peso.
4. Las imágenes cuatro y cinco muestran la configuración de dispositivos externos.

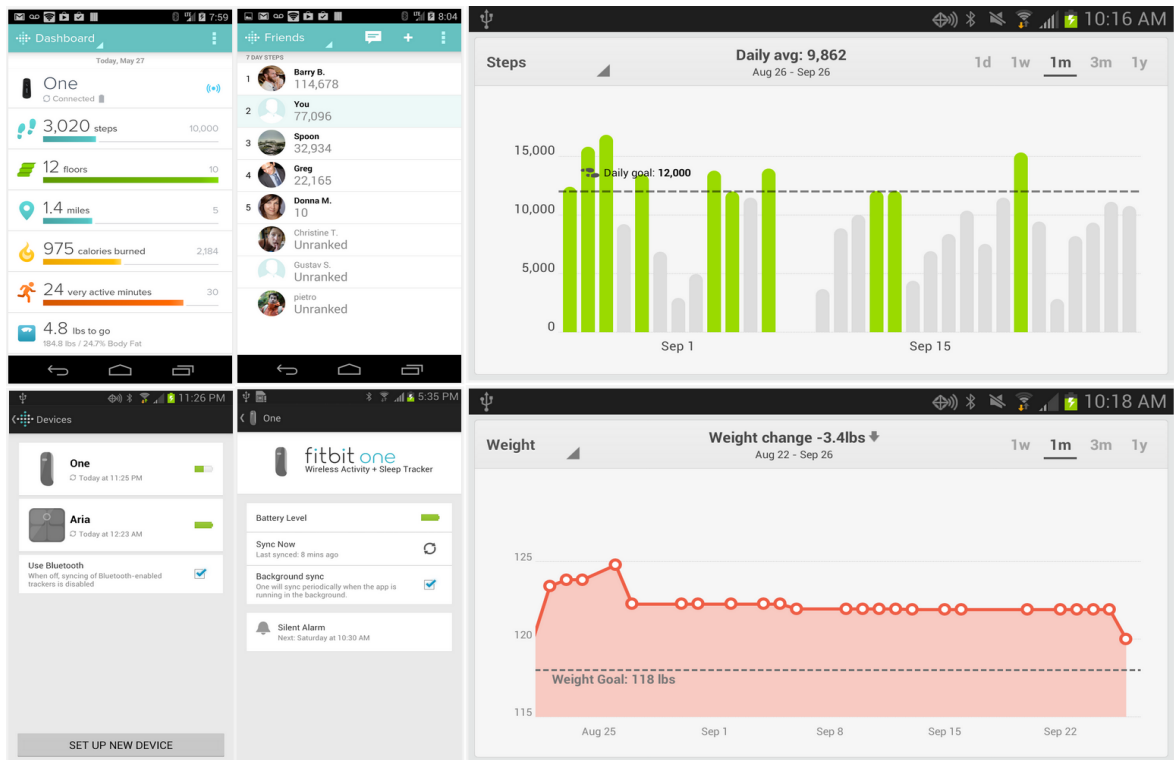


Figura 5: App. Fitbit para Android

2.3.3. Plataformas

Esta app se encuentra tanto para iOS, Android y Windows Phone.

2.4. Lift

2.4.1. Motivaciones

La filosofía detrás de Lift[13] es simple: ayudar a sus usuarios a cumplir sus metas. Generalmente se habla de aplicaciones que ayudan a hacer esto de forma laboral, pero Lift ofrece una vuelta de tuerca interesante. Se puede tener metas de variadas índoles, como por ejemplo, llevar una vida más sana. En este caso, entra en juego la aplicación que desglosa nuestras metas para hacerlas más realizables.

2.4.2. Características

Esta aplicación forma pequeños hábitos que ayudan a lograr la meta propuesta. La comunidad de Lift puede crear hábitos, que los usuarios pueden usar para su beneficio, o crear

nuevos de forma individual. Toda la actividad de Lift se hace de forma pública, para que el resto de los usuarios que forman parte de la comunidad puedan alentar al resto a cumplir sus metas.

Se tiene acceso a la información personal de “logros” a través de un análisis estadístico detallado, mostrándo que tan consistente se ha sido con las actividades, cuánto falta para llegar a la meta, y, en fin, estar más motivado para seguir adelante.

En la figura 6 se muestra una captura de pantallas para la app Android donde,

1. La primera imagen muestra los pequeños hábitos completados y por hacer..
2. La segunda muestra que planes se pueden realizar.
3. La tercera muestra la una búsqueda de hábitos para completar la meta.

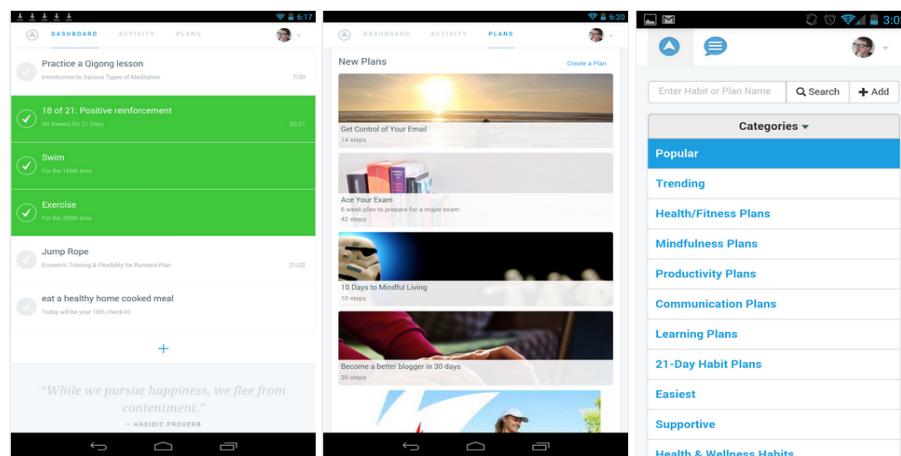


Figura 6: App. Lift para Android

2.4.3. Plataformas

Esta aplicación esta disponible tanto para iOS como para Android.

2.5. Beeminder

2.5.1. Motivaciones

Beeminder[14] tiene una propuesta interesante a la hora de monitorizar la actividad física. Como Lift, busca brindar un motor de motivación, aunque a través de una mecánica diferente y un poco invasiva. A través de Beeminder, se puede introducir una serie de metas que tendrán que ser seguidas al pie de la letra, a menos que realmente se quiera que la aplicación cobre un precio predeterminado cada vez que se falle.

2.5.2. Características

Beeminder cada vez que se vuelve a fallar en cumplir alguna de las metas, cobra un precio más alto, hasta que finalmente se logra hacerla. La motivación que se anda buscando llega en forma de una penalización monetaria que impulsa al usuario a que finalmente vaya a

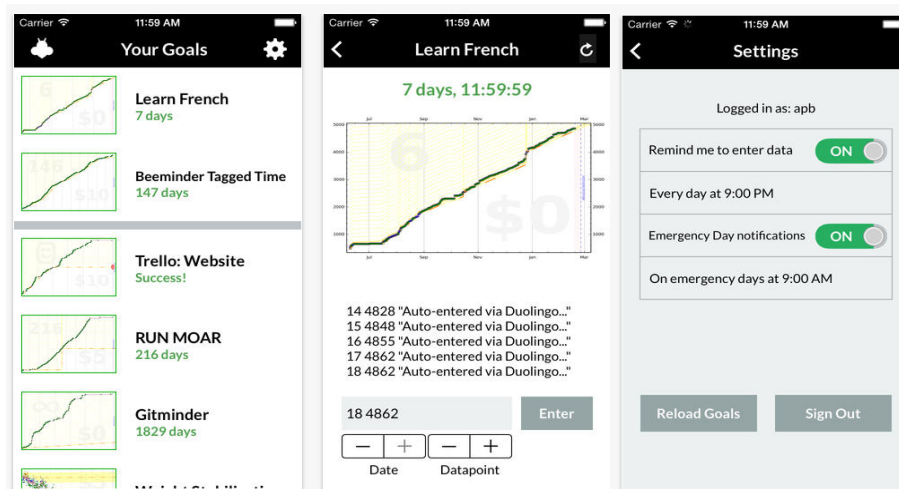


Figura 7: App. Beeminder para iphone

trabajar en bicicleta, o coma un poco más sano en el almuerzo. Es una idea creativa, aunque un poco nociva para el bolsillo.

Beeminder en lugar de ofrecer rastrear los propios movimientos, usa la información que puede obtener de otros servicios que se tengan instalados en el móvil, como puede ser Run-Keeper y también cuenta con sincronización con Fitbit para obtener todas las estadísticas. Cualquier meta cuantificable que se tenga puede ser medida fácilmente a través de Beeminder.

En la figura 7 se puede ver:

1. La pantalla donde aparecen los objetivos que se quieren realizar.
2. El seguimiento que se lleva del objetivo Learn French y como se insertan datos.
3. Pantalla de configuración

2.5.3. Plataformas

Esta aplicación esta disponible para iOS y para Android de forma gratuita.

2.6. Retrofit Me

2.6.1. Motivaciones

Esta aplicación está más orientada a los fanáticos del fitness, brindando una amplia gama de funcionalidades relacionadas con el mundo del ejercicio. Con Retrofit Me[15] se puede rastrear los movimientos pero se orienta más hacia la pérdida de peso.

2.6.2. Características

A través de Retrofit los usuarios pueden ponerse pequeñas metas de pérdida de peso para motivarse.

Retrofit combina una variedad de funciones que van más allá del simple monitoreo de actividad física. Por ejemplo, está incluido un plan de mentoring de especialistas en salud y

ejercicio para obtener mejores resultados. Estas sesiones de mentoring se realizan mediante Skype y son una propuesta interesante, porque permite tener un soporte en la vida “real”, más allá de algunos números en la pantalla, que para algunas personas podrían no ser suficientes.

Por otro lado, de la misma forma que Beeminder, se pueden importar los datos que son obtenidos a través de dispositivos como Fitbit y balanzas inalámbricas. Sin embargo, lo que realmente ofrece el valor diferencial es este consejo especializado por parte de los expertos.

Retrofit Me suele estar incluida en el programa (paquete) no gratuito Retrofit. En este paquete también se incluyen el precio del mentoring y las charlas por Skype.

A continuación la figura 8 muestra el estilo de la aplicación.

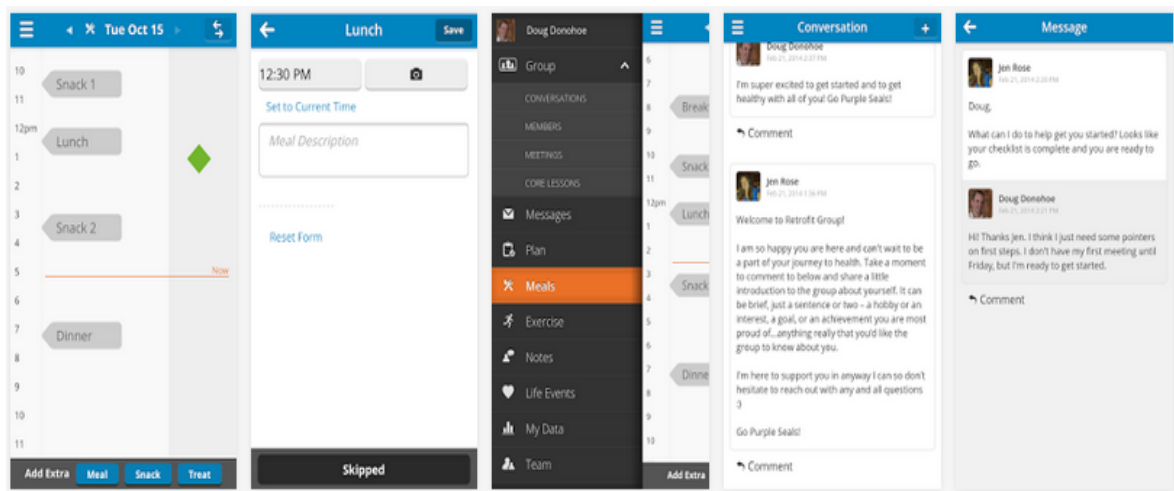


Figura 8: App. Retrofit me para iOS

Donde,

- La primera muestra el seguimiento de las comidas realizado.
- La segunda muestra como añadir descripciones de comidas.
- La tercera muestra el navigation bar de la app y sus opciones.
- Las dos últimas muestran las conversaciones que se tienen con los especialistas.

2.6.3. Plataformas

Tiene versiones tanto para iOS como para Android.

2.7. RunKeeper

2.7.1. Motivaciones

RunKeeper[16] esta orientada a aquellos que toman la actividad física como una actividad también relajante. Por un lado, es una herramienta eficiente de medición de la actividad física, mientras que por el otro lado, también ofrece una visión estadística de las cosas que se han realizado para tener una vida más sana.

2.7.2. Características

RunKeeper ofrece la posibilidad de monitorear diferentes actividades físicas como: caminar, andar en bicicleta, correr etc. Esto lo hace a través del GPS que viene instalado en el equipo, manteniendo estadísticas detalladas de todo lo que realizado.

Por otro lado, sin tener la misma dimensión de comunidad que Lift, ofrece un programa de motivación para que no abandonar las metas.

Con RunKeeper se pueden componer planes de entrenamiento personalizados, además de poder sincronizar la información con otros dispositivos como Fitbit.

En la figura 9 se puede observar la estructura de la aplicación:

1. Mapa donde se irá dibujando el recorrido realizado.
2. Estadísticas detalladas de una tarea realizada.
3. Comunidad de usuarios de la aplicación.
4. Lista de tareas realizadas con un pequeño resumen.

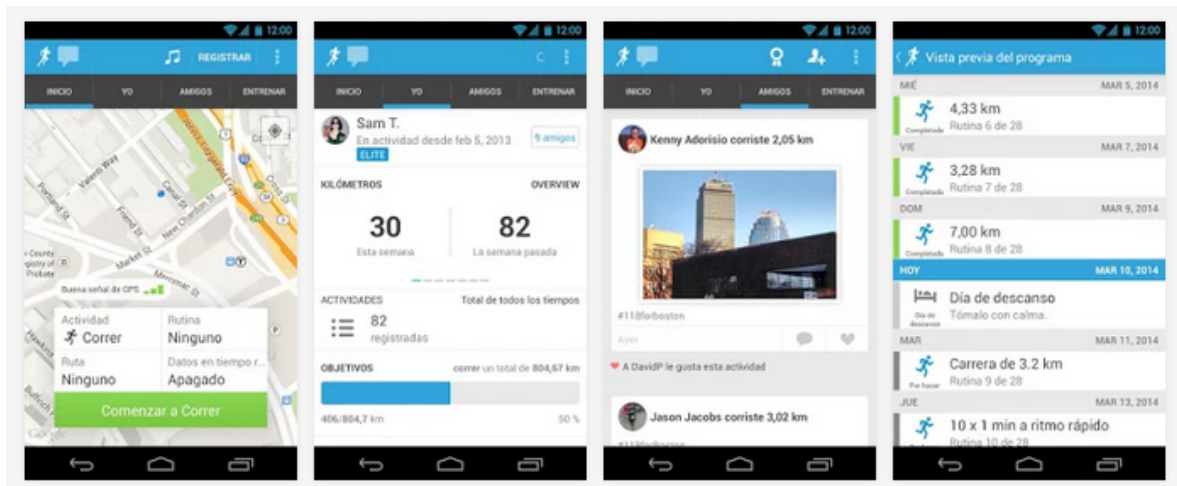


Figura 9: App. Runkeeper para Android

2.7.3. Plataformas

RunKeeper ofrece una experiencia interesante, multiplataforma, con la posibilidad de descargar tanto para iOS como para Android, de forma completamente gratuita.

2.8. Runtastic

2.8.1. Motivaciones

Runtastic[17] mide todos los datos necesarios para optimizar los resultados del entrenamiento y comparar el progreso en relación a actividades pasadas o con amigos.

2.8.2. Características

Runtastic permite tener acceso a diferentes planes de entrenamiento o de pérdida de calorías, a un programa particular de voice coaching, y a un análisis detallado de toda la información recolectada del ejercicio realizado.

Los datos almacenados para el análisis detallado son, por ejemplo, la velocidad de movimiento, la elevación a la que se realiza ejercicio, la distancia que se recorre etc.

En cuanto a esta información un aspecto interesante es que puede ser almacenada en la nube, para poder tener acceso a ella de cualquier forma, desde cualquier dispositivo.

La aplicación además de permitirnos monitorear nuestra actividad fuera de casa, también permite fijarse en qué es lo que se está haciendo, por ejemplo, en el gimnasio, con la medición de diferentes tipos de actividades físicas. También cuenta con un monitor del ritmo cardíaco.

La figura 10 muestra la estructura de la versión más reciente de la aplicación:

1. Navigation Bar con la foto del usuario y opciones.
2. Lista con los diferentes tipos de actividades realizadas
3. Información extra de la tarea dada por el usuario.
4. Análisis detallado de las tareas.

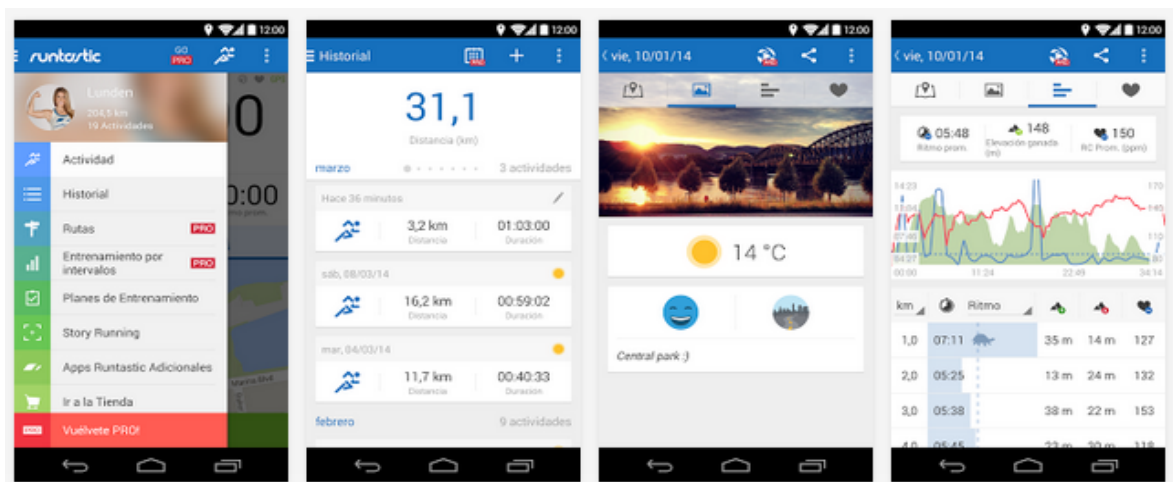


Figura 10: App. Runtastic me para Android

2.8.3. Plataformas

Runtastic, es una aplicación muy completa, que está disponible para Android, iOS, Windows Phone, y hasta Blackberry OS y Bada.

2.9. Endomondo

2.9.1. Motivaciones

Aplicación que lleva a cabo el seguimiento del entrenamiento por GPS, válido tanto para running como para otras disciplinas.

2.9.2. Características

Esta aplicación cuenta con las mismas características detalladas en la aplicación Nike+ con la salvedad de que se pueden escoger varias modalidades deportivas en lugar de estar enfocada solo a una.

Endomondo[18] trabaja con (fig. 11):



Figura 11: Empresas con las que trabaja Endomondo

La estructura de la aplicación es similar a la de las demás como se puede apreciar de la figura 12.

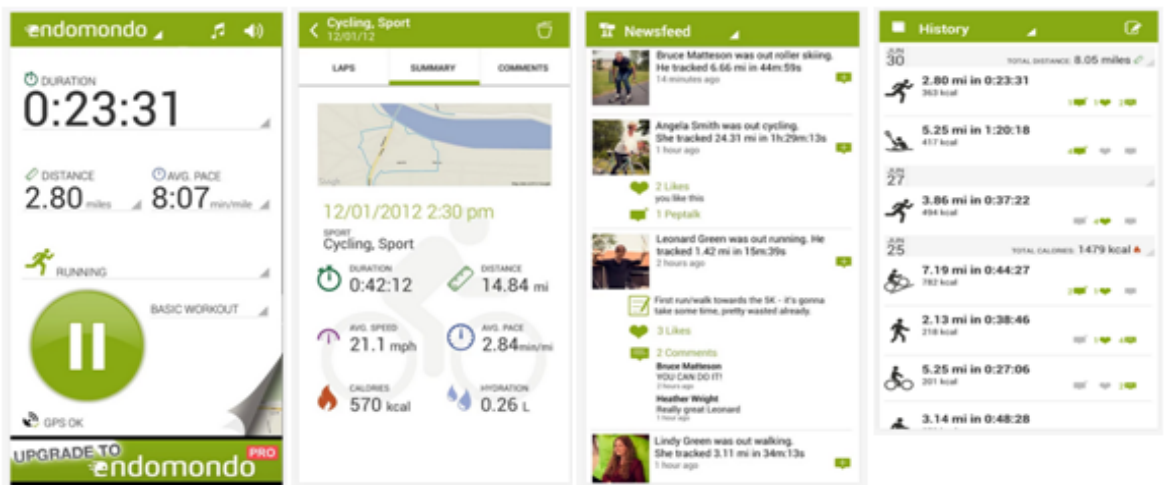


Figura 12: App. Endomondo para Android.

Donde,

1. Se muestra la realización de la prueba.
2. Análisis detallados de las tareas.
3. Comunidad de usuarios.
4. Lista con todas las actividades realizadas.

2.9.3. Plataformas

Endomondo se encuentra tanto para iPhone como para Android.

2.10. FitLinxx (Web App)

2.10.1. Motivaciones

FitLinxx[19] es una página web que ofrece productos para motivar a las personas a tener una vida activa y saludable. Sus aplicaciones de bienestar y productos fitness son usados por unos 4 millones de personas cada día para realizar un seguimiento de la actividad diaria, entrenamientos de fitness y indicadores críticos de la salud como el peso o la presión sanguínea. Sus productos ayudan a mejorar la forma física y en la pérdida de peso y gestión de enfermedades crónicas como diabetes e hipertensión.

2.10.2. Características

FitLinxx posee un monitor de actividad wireless, el Pebble y aparatos para la salud wireless que hacen que sea fácil la captura de actividad diaria, presión sanguínea y datos de peso. También poseen aplicaciones web divertidas y animadas para que el usuario pueda observar todas estas estadísticas, establezca un seguimiento saludable y metas en su actividad, unirse a comunidades de usuario, ganar recompensas etc.

En la figura 13 se observan los diferentes aparatos que poseen de monitorización.



Figura 13: (a) Pebble. (b) bpmonitor. (c) báscula

FitLinxx provee productos para facilitar el fitness que apoyan el éxito del usuario con entrenamiento interactivo en equipamiento de fuerza o entrenamiento de cardio, y el sistema da facilidades para ponerse en contacto con miembros del staff. Estos productos de seguimiento fitness están en red y todos los datos del ejercicio son enviados a aplicaciones web. Este programa lo denominan Health and Fitness Tracking.

2.10.3. Plataformas

No tiene aplicación para teléfonos móviles.

3. Diario de desarrollo

En la realización de este proyecto se ha seguido una metodología orientada a objetos y el modelo de cascada con feedback (fig. 14) puesto que al ir avanzando en la realización de la aplicación el conocimiento de las herramientas de trabajo ha ido en aumento y este ha podido ser reflejado en las diferentes etapas. Las etapas en las que mayor hincapié se ha realizado han sido las de diseño, implementación y verificación, partiendo de algo muy sencillo hasta conseguir una primera versión de la aplicación.

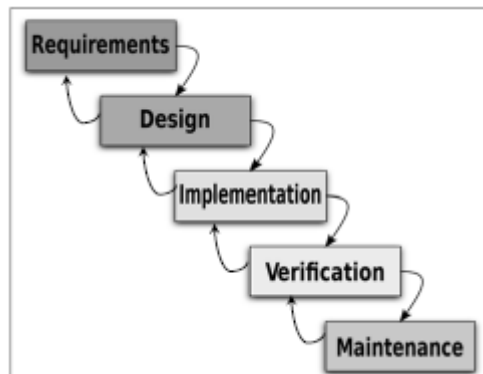


Figura 14: Modelo cascada con feedback

A continuación se presenta la descripción de las etapas empleadas en la realización del proyecto, las cuales se desarrollarán más detenidamente en apartados posteriores:

1. **Requerimientos:** descripción de las funcionalidades deseadas del sistema.
2. **Diseño:** teniendo como base los requerimientos se define como funcionará el sistema
3. **Implementación:** aplicando los métodos y medidas necesarios se implementa el diseño definido.
4. **Verificación:** se comprueba que la aplicación cumple las especificaciones de partida.
5. **Mantenimiento:** se provee una plataforma para la corrección de errores.

3.1. Requerimientos

El análisis de usuario, tarea y dominio alimentan al proceso más general llamado análisis de Requerimientos, por lo que primero se analizará estos.

- Análisis de usuario

El análisis de usuario es tan obvio que normalmente es saltado, incrementando el riesgo de un mal diseño de interfaz: los diseñadores pueden pensar en ellos como el usuario y esto no es así.

Por lo que respecta al usuario de la aplicación, se considera que su rango de edad puede ser variado: desde personas jóvenes hasta personas con una cierta edad, tanto hombres como mujeres, que tenga cierta experiencia en el manejo de teléfonos móviles y que quieran mejorar su condición física aún teniendo algún problema de salud. Estos usuarios utilizarán la aplicación frecuentemente ya que serán informados de las diferentes tareas a realizar y verán sus resultados en el propio teléfono.

■ Análisis de tarea

El análisis de tareas tiene como objetivo identificar las tareas del programa. Estas tareas serán expresadas como una meta: *que es lo que hay que hacer (no como)*.

Las tareas a las que tendrá acceso el usuario serán las siguientes:

- Recibir notificaciones de las misiones encomendadas.
- Monitorización de la posible correcta consecución de las misiones, haciendo uso de sensores de acelerometría y posición (GPS).
 - Acelerómetro para controlar misiones de ciertos ejercicios físicos (flexiones, abdominales)
 - GPS para el control de los kilómetros en las misiones que fomenten andar o ir en bici.
- Interactividad.
 - Actualizando la información de las misiones tras haberlas realizado.
 - Recompensa al usuario con puntos virtuales o pérdida de ellos en base a su cumplimiento y mejoras.
 - Visualización de tablas y gráficas de resultados y evolución.
 - Comparación consigo mismo o con otros usuarios del programa.
- Integración con redes sociales.

■ Análisis de dominio

El análisis de dominio descubre los elementos y como son relacionados entre sí dentro del proyecto CERES. En los siguientes diagramas se puede observar como se relaciona al usuario con las diferentes partes de la aplicación:

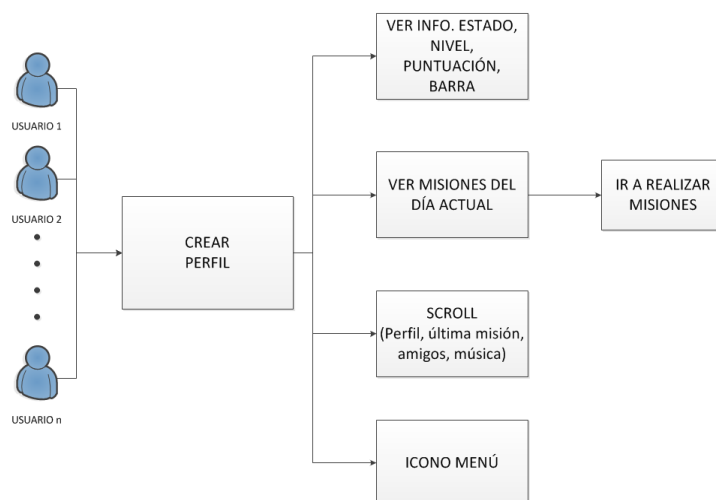


Figura 15: Diagrama 1.

En este primer diagrama (fig. 15) cada usuario entra con su propio perfil y tendrá la posibilidad de realizar acciones en la pantalla principal como las que se pueden ver.

Este segundo diagrama (fig. 16) hace referencia al patrón de diseño del navigation drawer. Pulsando el icono de la aplicación se desplegará una lista con las diferentes

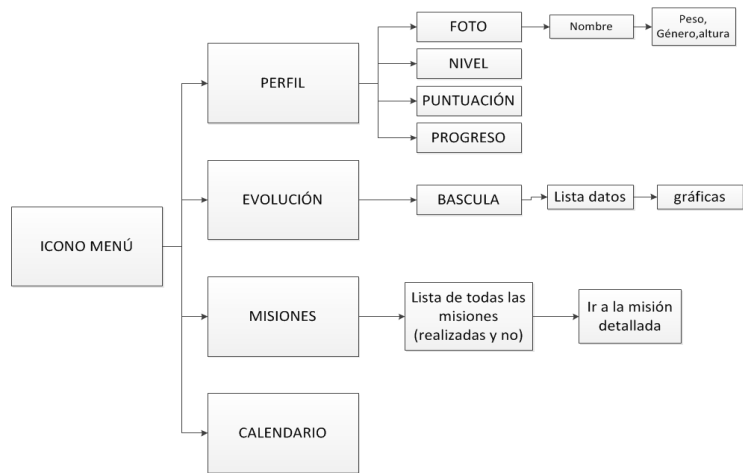


Figura 16: Diagrama 2.

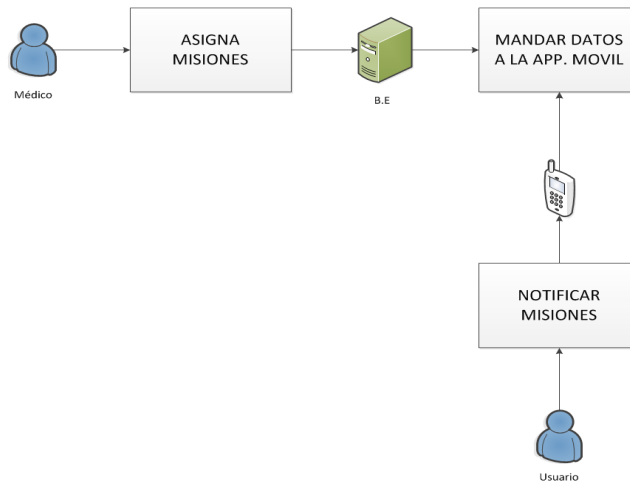


Figura 17: Diagrama 3.

opciones implementadas para interactuar con el usuario como se puede observar del diagrama.

En este tercer diagrama (fig. 17) el usuario es notificado en su móvil de las tareas asignadas por el médico.

En el cuarto diagrama (fig. 18) se muestra el modelo de la aplicación, la actualización del valor de las recompensas y la interfaz de usuario.

Por último el diagrama número cinco (fig. 19) muestra que pasa en la aplicación cuando un usuario quiera iniciar una tarea.

Una vez terminada la etapa de análisis anterior uniendo estas tres partes se obtiene el documento de requerimientos (fig. 20) necesario para comenzar a diseñar la aplicación.

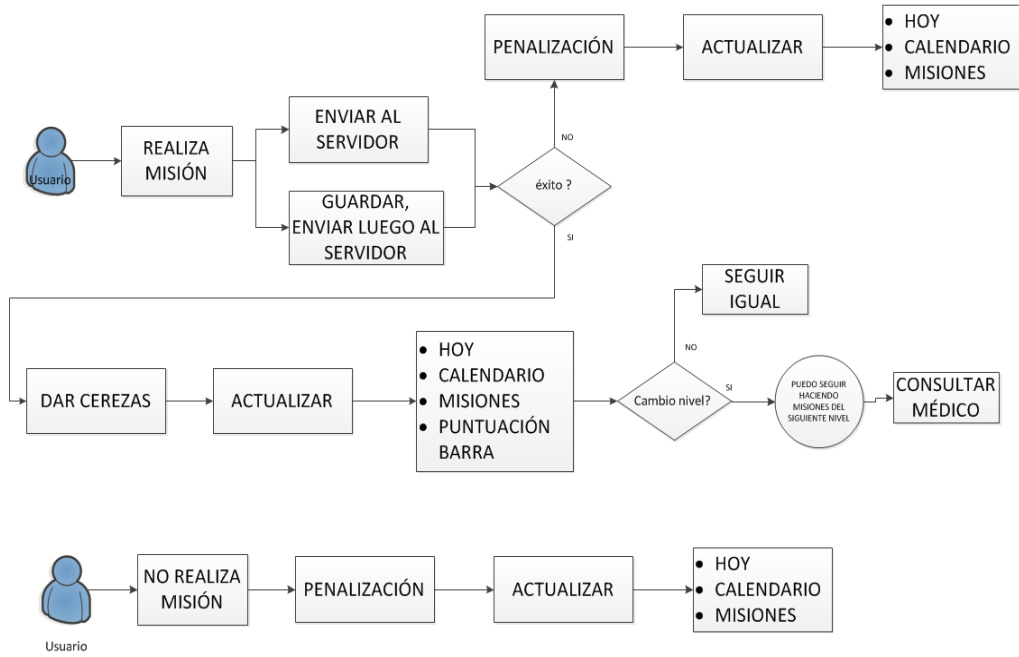


Figura 18: Diagrama 4.

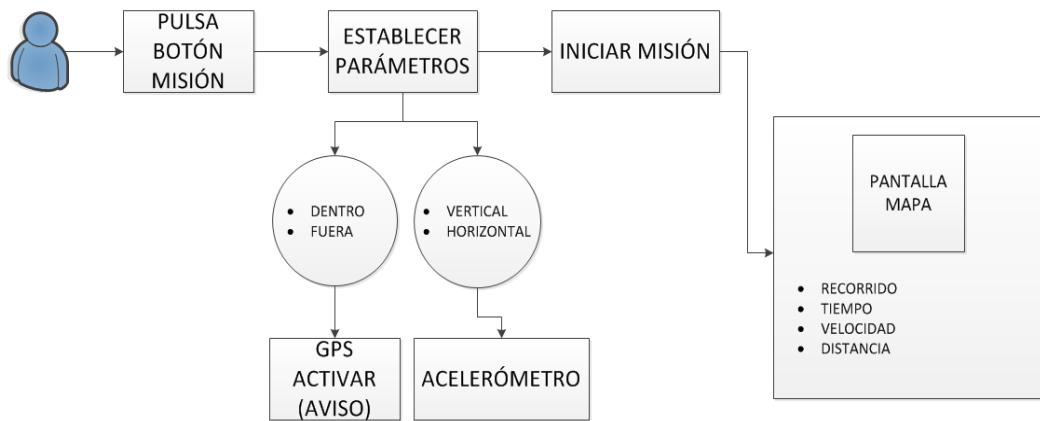


Figura 19: Diagrama 5.

• Requirements: what should the system do?

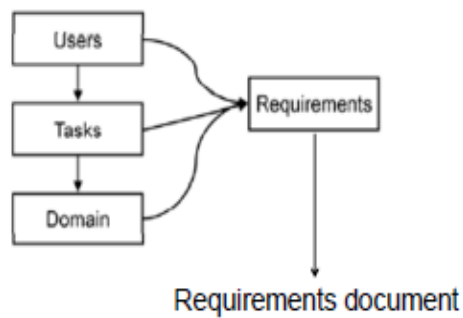


Figura 20: Requerimientos.

3.2. Diseño

Una vez obtenidos los requerimientos se comienza a realizar el diseño de la aplicación. Esta fase se dividirá de la siguiente manera:

1. **Casos de uso:** las diferentes acciones que los actores realizan en nuestro sistema.
2. **Diseño final:** explicación de la evolución del diseño desde la primera aproximación hasta la versión final.
3. **Componentes:** los bloques funcionales que constituyen la aplicación además de detallar los diferentes tipos de comunicaciones que se establecen entre ellos así como con los sistemas externos.
4. **Clases:** descripción de las responsabilidades de cada una de las clases Java que componen la aplicación Android.
5. **Diagramas de secuencia:** especificación de los flujos básicos de los mensajes entre las diferentes clases, módulos o actores que intervienen en el sistema

3.2.1. Casos de uso

La figura 21 muestra el diseño de la aplicación de una manera global. En cuanto a los actores presentes en el sistema se hará una pequeña descripción de estos:

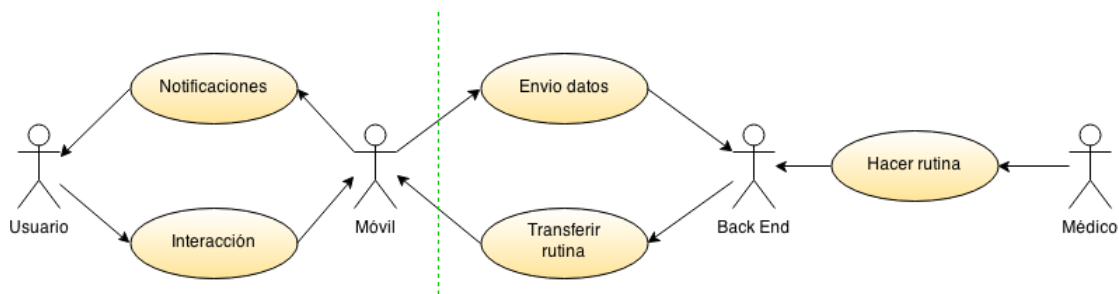


Figura 21: Caso de uso global.

- Usuario: usuario que tendrá la aplicación instalada en el telefono y podrá interactuar con ella.
- Móvil: aparato donde se instalará la aplicación. Será el encargado de por ejemplo, mostrar por pantalla la rutina correspondiente al usuario o enviar datos al Back End.
- Back End: el encargado de guardar en un lugar remoto los datos del usuario y comunicar el cambio de tareas.
- Médico: encargado de realizar una rutina y enviarla al back end para un posterior envío al usuario.

En la fig. 21 hay dos partes diferenciadas por una línea discontinua. Esto se debe a que en el momento de escribir esta memoria la parte del lado derecho del diagrama no ha sido todavía implementada en la aplicación. La parte izquierda del diagrama sin embargo si que ha sido desarrollada y es la que se explicará a continuación haciendo hincapié sobre todo

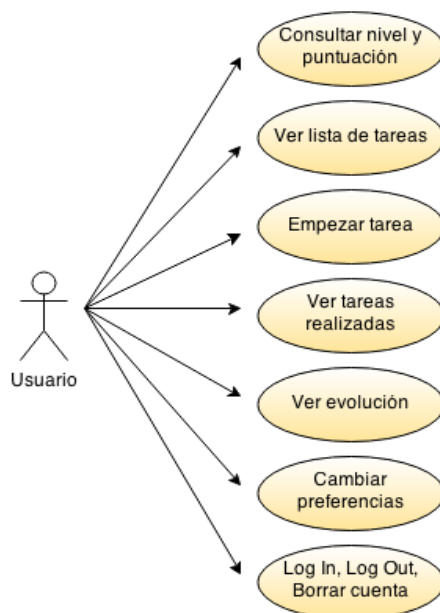


Figura 22: Caso de interacción móvil usuario.

en el caso de la interacción móvil usuario. Para ver este comportamiento se ha creado el siguiente diagrama.

En este caso de uso (fig. 22) se especifican cuales son las operaciones que se le permiten realizar al usuario en el teléfono.

3.2.2. Diseño final

En este apartado se hará un breve resumen de los cambios que se han ido dando en el diseño de la aplicación, comenzando por la aproximación más simple y acabando en la versión final.

1ª Aproximación

Una primera aproximación de la aplicación fue la siguiente(fig. 23):

Se trataba de una aproximación muy simple, pero que daba una primera visión de la aplicación. En la pantalla principal situada a la izquierda se inserta una lista de tareas. Pulsando en una tarea de esta lista de la pantalla principal se lanzaría esta para poder ser realizada. Desde la pantalla principal también se puede acceder tanto al perfil como a una serie de opciones como son evolución, misiones o calendario. Si por ejemplo la opción misiones era pulsada se saltaba a la siguiente lista con todas las tareas realizadas y pulsando en una de ellas se saltaría a esta tarea específica como se observa con la pantalla del mapa.

2ª Aproximación

A medida que se implementaban nuevas funcionalidades esta primera aproximación se fue modificando llegando a lo que se puede llamar segunda aproximación (fig.24).

En esta segunda aproximación se contaba con un launcher y seguido aparece la opción de logearse en la aplicación mediante Google+. Una vez dentro de la aplicación la pantalla principal sigue el mismo esquema que en la primera aproximación donde se puede ver el estado (puntuación y nivel del usuario) y la lista con las tareas a realizar.

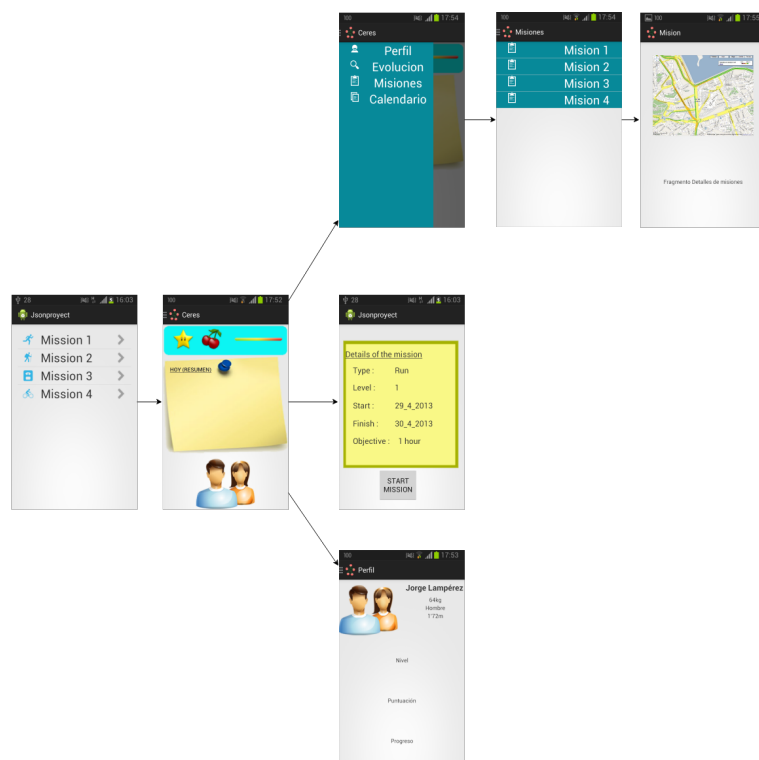


Figura 23: Primera aproximación.

A partir de esta aproximación ya se empieza a dividir las tareas del usuario en dos tipos, toma de medidas y de seguimiento de localización. Las pantallas de tareas de toma de medidas se encuentran en la parte de abajo de la figura. Al comenzar una tarea de este tipo se lanza un dialogo que permite elegir si esta medida va a ser recogida por un dispositivo vinculado, si se tiene que realizar una búsqueda de dispositivos o tomar la medida manualmente. En las pantallas de tareas de seguimiento de localización (dos pantallas en línea horizontal) antes de dicha realización se informa al usuario de cual es el objetivo y una vez dentro había un cronómetro en el cual se podía ver el transcurso de los segundos mientras se obtenía la localización.

En esta aproximación la navigation drawer tambien esta presente desde la cual se puede entrar al perfil (con imagen personalizada de la cuenta de google+) o a la lista de tareas realizadas de los dos tipos, si la tarea se trata de seguimiento de localización se muestra un mapa y si es de toma de medida, esta es reflejada en la pantalla.

3ª Aproximación versión beta

Esta versión beta de la aplicación sigue el modelo de diseño de las dos anteriores. El diseño de las pantallas es realizado en Photoshop y en base a ellas se reestructura todo el código de la aplicación elaborado hasta ese momento. Comparando esta aproximación con la anterior se puede ver como algunas de las pantallas han sido eliminadas o modificadas con el fin de conseguir una mejor usabilidad por parte del usuario. Un ejemplo de esto es la pantalla de perfil de usuario que ha sido eliminada y insertando esta información dentro de la navigation drawer.

Por lo que respecta al diseño de la aplicación esta sigue teniendo una pantalla de login. Al realizar login en la aplicación si se dispone de más de una cuenta gmail en el telefono aparecerá un *account picker* para permitir escoger la cuenta deseada. Si se trata de la primera

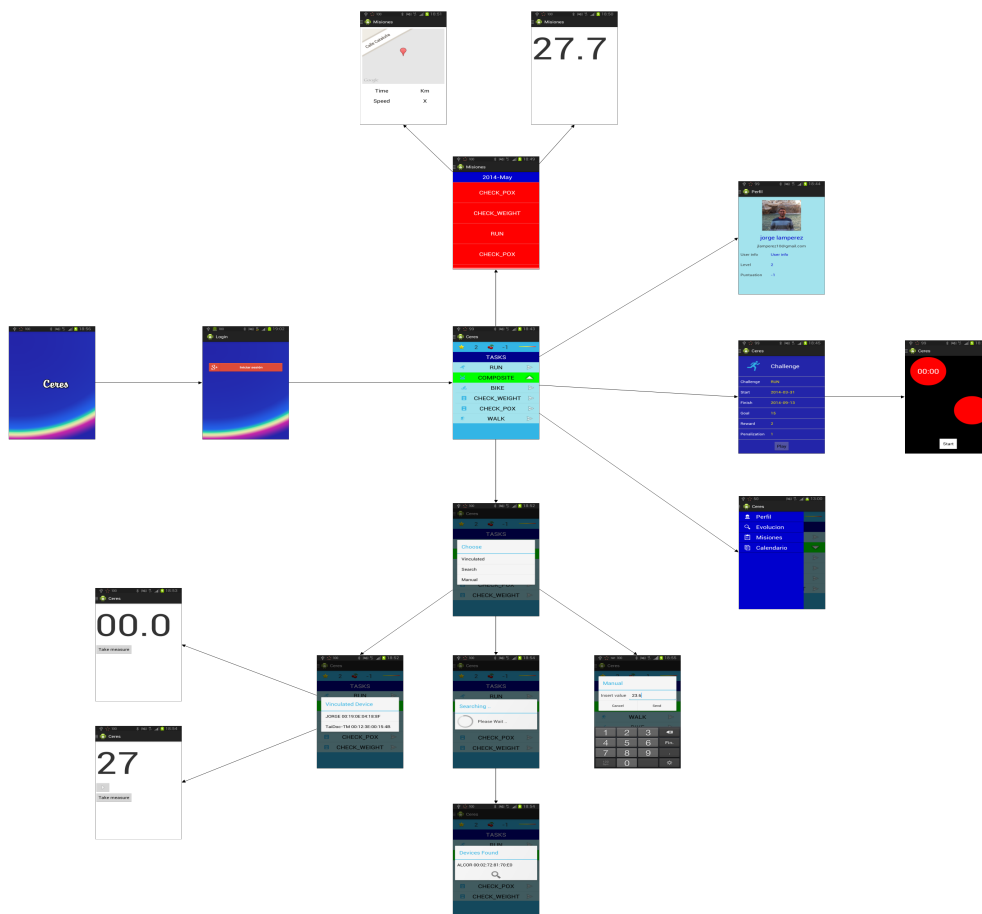


Figura 24: Segunda aproximación.

vez que el usuario utiliza la aplicación este deberá conceder los permisos necesarios.

Una vez dentro de la aplicación el diseño es el mismo, información de nivel y puntuación del usuario y la lista con las tareas a realizar.

El diseño de las diferentes tareas ha sido unificada como se puede observar teniendo la parte de arriba de la pantalla un espacio en el que se especifica cual es el objetivo de la tarea y un campo abajo que se va actualizando una vez se esta realizando la tarea. Las tareas de seguimiento de localización mostrarán un dialogo cada vez que el sensor de posición este desactivado como se muestra en la línea horizontal de la figura, mientras que las tareas de toma de medidas siguen el mismo diseño que en la segunda aproximación con el único matiz de que al acabar estas tareas la pantalla actual es refrescada para mostrar el resultado.

El diseño de la navigation drawer, como se ha comentado anteriormente, ha sido modificado para la inclusión de la información de perfil dentro de ella. Desde esta navigation drawer se puede acceder tanto a todas las tareas realizadas, como a observar las gráficas de las tomas de medida como peso y temperatura o a la pantalla de cambio o borrado de cuenta. En la pantalla de tareas realizadas en las que sea necesario mostrar el recorrido de estas se habilita un botón mediante el cual este recorrido puede ser divisado en un mapa.



Figura 25: Versión beta.

3.2.3. Componentes

Los diferentes componentes de la aplicación se representan mediante la figura 26.

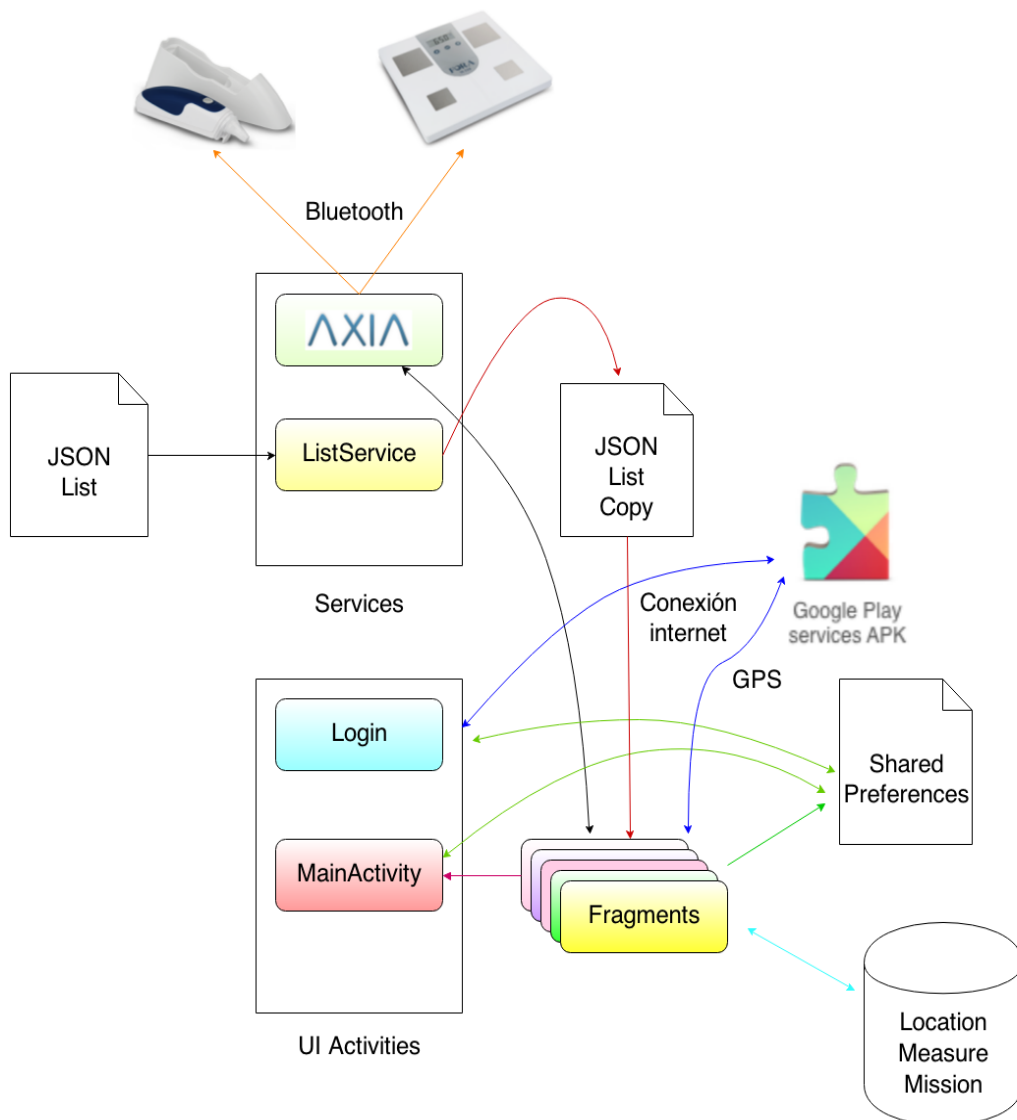


Figura 26: Componentes de la aplicación.

Como se puede observar se tiene una lista en formato JSON de las tareas a realizar por el usuario. Esta lista se renova cada x tiempo mediante el servicio ListService que copiará las tareas a realizar y enviará el fichero al fragmento correspondiente para su posible modificación. La aplicación consta de dos actividades que serán Login y MainActivity. La primera de ellas servirá para logearse en la aplicación y hará uso de las shared preferences así como de la API de google play services. Google play services proporciona a la aplicación las últimas características proporcionadas por Google como Maps o Google+. Los fragments son fragmentos de código reutilizable que se irán insertando en el contenedor correspondiente de la MainActivity. Estos diferentes fragmentos son los que trabajarán con las shared preferences, bases de datos, localización de google play services o requerirán establecer una conexión bluetooth con los diferentes dispositivos para la toma de medidas mediante la plataforma AXIA.

3.2.4. Clases

Las clases que constituyen la aplicación están agrupadas en 8 paquetes: axia, controller, graphics, login, model, record, util, view. Además de estos 8 paquetes se encuentra la clase principal MainActivity.

1. PAQUETE AXIA

- PAQUETE *BTDISCOVERER*
 - **DeviceDiscoverer**: Clase para encontrar dispositivos.
 - **Plugin**: Plugin para cargar el SearchEngine.
 - **SearchEngine**: Clase que sirve para buscar dispositivos y servicios.
 - **ServiceDiscoverer**: Clase para buscar servicios.
- PAQUETE *BTSP*
 - **ClientSocket**: Creación de un socket de tipo cliente.
 - **Plugin**: Plugin para cargar el Transport.
 - **ServerSocket**: Creación de socket de tipo servidor.
 - **Transport**: Clase que define el transporte bluetooth y servicio SPP.
- PAQUETE *LOG*
 - **Log**: Clase que muestra por pantalla diferentes mensajes.
 - **Plugin**: Plugin para cargar el Log.
- PAQUETE *SERVICE*
 - **PlatformService**: Servicio que servirá a la aplicación para cargar plugins y realizar las conexiones mediante la plataforma AXIA.

2. PAQUETE CONTROLLER

- PAQUETE *BRS*
 - **BootReceiver**: Clase que manda una notificación de la aplicación al encender el telefono móvil.
- PAQUETE *GAME*
 - **GameController**: Clase que se encarga de llevar el control del nivel y la puntuación en la aplicación.
- PAQUETE *Preferences*
 - **Preferences**: Clase que se encarga de crear, almacenar o modificar las preferencias tanto de la aplicación como de los usuarios de esta.

3. PAQUETE GRAPHICS

- **FEvolution**: Clase que se encarga de mostrar por pantalla las diferentes gráficas en un fragmento.
- **GraphicLayout**: Se trata de un LinearLayout en el que se insertan distintas vistas como gráfica, tipo, valor máx. y min.
- **MyCustomXYPlot**: Clase en la que se diseña la forma del gráfico.

4. PAQUETE *LOGIN*

- **LoginActivityPlus**: Clase que lanzará la primera actividad de la aplicación con el botón de logIn.
- **LoginHelper**: Clase que servirá de apoyo para realizar una conexión con el GoogleApiClient y implementa los métodos callback. Utiliza patrón Observer.

5. PAQUETE *MODEL*

- PAQUETE *Ceres*
 - **AChallenge**: Clase abstracta que implementa IChallenge.
 - **Challenge**: Clase que se refiere a un tarea simple.
 - **ChildrenStuff**: Clase que dice como es el comportamiento de las tareas hijas dentro de una tarea compuesta.
 - **GameStuff**: Clase que detalla las recompensas o penalizaciones en la aplicación.
 - **IChallenge**: Interfaz de la clase challenge. Esta tarea puede ser tanto Challenge como MultiChallenge y tiene definidos diferentes tipos mediante el enum ChallengeType.
 - **IChildrenStuff**: Interfaz de la clase ChildrenStuff.
 - **IGameStuff**: Interfaz de la clase GameStuff.
 - **IPlanning**: Interfaz de la clase Planning.
 - **IReward**: Interfaz de la clase Reward.
 - **MultiChallenge**: Clase correspondiente a un Composite. Diferentes clases dentro de ella. La workList es una clase de este tipo.
 - **Planning**: Clase que determina con que periodicidad se realizará la tarea correspondiente.
 - **Reward**: Clase que determina los puntos a recibir o restar.
- PAQUETE *worklist*
 - **IWorklist** : Interfaz de una worklist
 - **JsonWorklist**: Clase que convierte la worklist de tipo JSON a objeto Multichallenge.
 - **ManualWorklist** : Clase que crea una worklist manualmente mediante código.
 - **MultiChallengeToJSON**: Clase que sirve para actualizar la lista de tareas. Esta guarda el objeto MultiChallenge en formato JSON.

6. PAQUETE *RECORD*

- PAQUETE *DATABASE*
 - PAQUETE *DAO*: Data Acces Object, clases que permiten a los diferentes objetos de la aplicación no tener que estar pendiente del código con la base de datos.
 - **LocationDao** : Clase para trabajar con objetos del tipo Location insertados en la base de datos.

- **MeasureDao** : Clase para trabajar con objetos del tipo Measure insertados en la base de datos.
- **MissionDao** : Clase para trabajar con objetos del tipo Mission insertados en la base de datos.
- **PAQUETE HELPER**
 - **RecordDbHelper** : Clase que crea la base de datos y sus tablas. Extiende a SQLiteOpenHelper.
- **PAQUETE MODEL**
 - **Measure** : Clase que permite trabajar con objetos del tipo medida.
 - **Mission** : Clase que permite trabajar con todas las tareas realizadas.
 - **MyLocation** : Clase que permite trabajar con objetos del tipo location.
- **PAQUETE SCHEMA**
 - **LocationSchema** : Clase que define la implementación de la tabla location.
 - **MeasureSchema** : Clase que define la implementación de la tabla measure.
 - **MissionSchema** : Clase que define la implementación de la tabla mission
- **PAQUETE DRAW**
 - **PAQUETE EXPLV**
 - **FMissionExplvAdapter** : Adaptador para la explv de todas las misiones.
 - **FMissionsExplv** : Fragmento que contiene una explv y muestra todas las tareas realizadas.
 - **PAQUETE MISSION**
 - **FMissionLocation** : Fragmento que muestra el mapa de recorrido de una misión.
- **PAQUETE SAVE**
 - **LocationClientHelper** : Clase que sirve para realizar una conexión con GooglePlayServices y obtener la localización del usuario.
 - **LocationRequestHelper** : Clase que sirve para crear una petición de localización y establecer el intervalo de actualización de esta.
- **FSpecRecordLocation** : Fragmento en el que se le propone al usuario comenzar una actividad de recorrido. Este guardará la localización en la base de datos.
- **FSpecRecordMeasur** : Fragmento en el que se le propone al usuario comenzar una actividad de medida. Este guardará la medida en la base de datos.

7. PAQUETE UTIL

- **EnumSelector** : Clase que pasa parámetros de string a enum.
- **ImageSelectorUtil** : Clase para mostrar el icono de correspondiente resolución.
- **MeasureTypeEditor** : Clase para editar los tipos measure.
- **TimeUtils** : Clase que establece la hora de la alarma y obtiene el día.

- **Util_ISO8601** : Clase que trabaja con el formato de fecha y hora definido por el ISO8601.

8. PAQUETE *VIEW*

- PAQUETE *CUSTOM*

- **Circle** : View en el que se pinta un círculo.
- **CustomChronometer** : View en el que se pinta un cronómetro dentro de un círculo.

- PAQUETE *FRAGMENTS*

- PAQUETE *DIALOG*

- PAQUETE *CHECK*

- ◊ **CheckDialogFragment** : Diálogo que muestra por pantalla estas tres opciones: Manual, Search y Vinculated.
- ◊ **ManualDialogFragment** : Diálogo en el que puede ser insertada la medida manualmente.
- ◊ **SearchDialogFragment** : Diálogo que permite la búsqueda de dispositivos Bluetooth no vinculados.
- ◊ **VinculatedDialogFragment** : Diálogo que permite seleccionar el dispositivo vinculado.
- ◊ **GPSDialogFragment** : Diálogo que aparece al estar el GPS apagado en las tareas de recorrido.
- ◊ **WifiDialogFragment** : Diálogo para encender el Wifi del móvil.

- PAQUETE *UI*

- PAQUETE *MAIN*

- ◊ **FTasks** : Fragmento de la lista de tareas.
- ◊ **TaskExplvAdapter** : Adaptador de la lista de tareas.
- ◊ **ListService** : Servicio que actualizará la lista de tareas y mandará notificaciones.
- ◊ **TaskAlarm** : Clase que crea una alarma para actualizar la lista de tareas.
- ◊ **FMain** : Fragmento principal donde irá la lista de tareas , el nivel y la puntuación.
- ◊ **FInfo** : Fragmento de nivel y puntuación.
- ◊ **FPreferences** : Fragmento de preferencias de usuario.

- PAQUETE *NAVDRAWER*

- **AdapterNavDraw** : Adaptador para la lista de opciones del navigation drawer.
- **CustomProfileImageDraw** : Clase para dibujar la imagen de perfil de forma circular y con un marco blanco.
- **NavDrawOptions** : Modelo de clase para las diferentes opciones de la nav. draw.

- **MainActivity** : Clase que se lanza una vez logeado dentro de la aplicación. Es la actividad principal donde se implementa el diseño del navigation drawer, es

decir, un contenedor en el que se irán insertando los diferentes fragmentos y la barra de opciones escondida a la izq. de la pantalla.

3.2.5. Diagramas de secuencia

Típicamente un diagrama de secuencia captura el comportamiento de un único escenario. Estos diagramas muestran un número de objetos de ejemplo y los mensajes que se pasan entre objetos dentro de un caso de uso. Utilizando los casos de uso descritos en el apartado 3.2.1 se obtienen los siguientes diagramas:

1. Consultar nivel y puntuación.

En la fig. 27 puede observarse como la actividad principal hace uso del fragmento FMain y que el fragmento FInfo se encuentra metido en el. El objeto Preferences mantiene este fragmento actualizado.

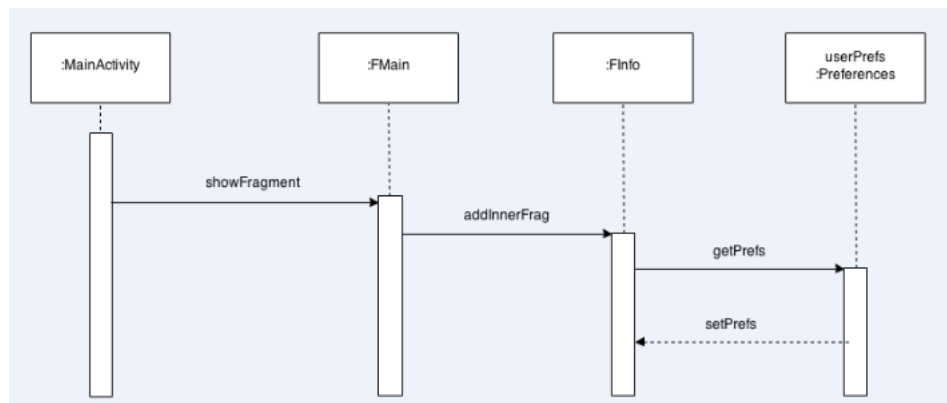


Figura 27: Diagrama de consulta de nivel y puntuación.

2. Ver lista de tareas y empezar tarea.

En la fig. 28 puede observarse también que el fragmento FTasks se encuentra dentro de FMain. Este fragmento puede ser actualizado cada día utilizando los objetos TaskAlarm y ListServices o puede ser creado a partir de un fichero. Por último puede verse como al hacer click sobre cada tarea el fragmento FTasks es reemplazado por el correspondiente de Measure o Location.

3. Ver tareas realizadas.

La fig. 29 muestra como se rellena la expandable list con los datos de las tareas realizadas y como puede haber un campo opcional en las misiones de localización las cuales al hacer click sobre el botón dan más detalles acerca del recorrido.

4. Ver evolución.

La fig. 30 refleja como se hace una búsqueda de las misiones que contienen *MDC* y como se establecen los parámetros al objeto *GraphicLayout* para actualizar la vista de *FEvolution*.

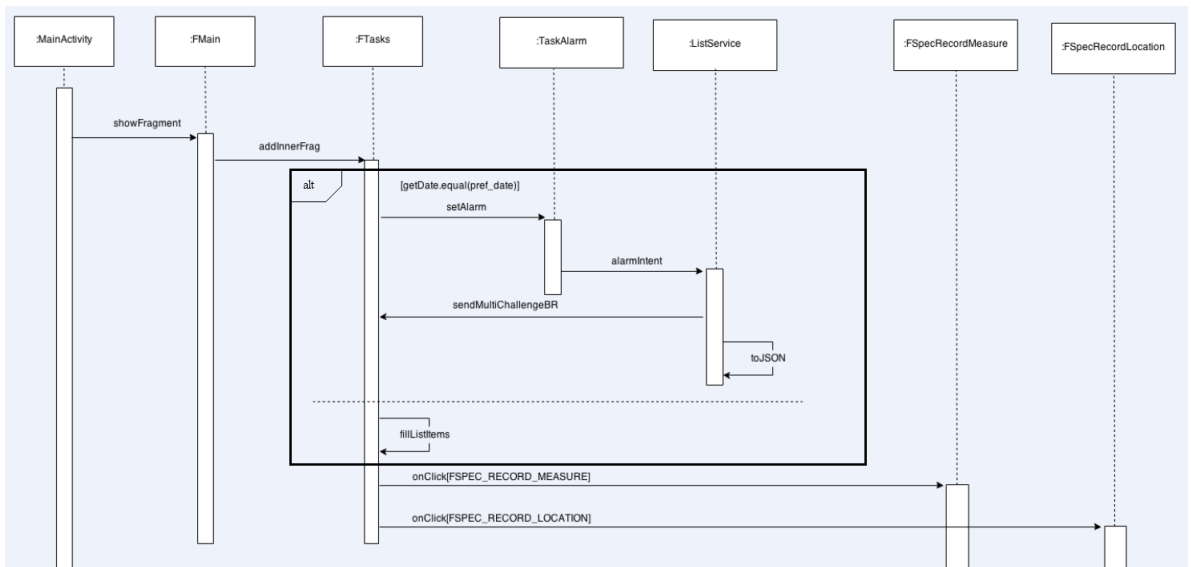


Figura 28: Diagrama de lista de tareas y como empezar su ejecución.

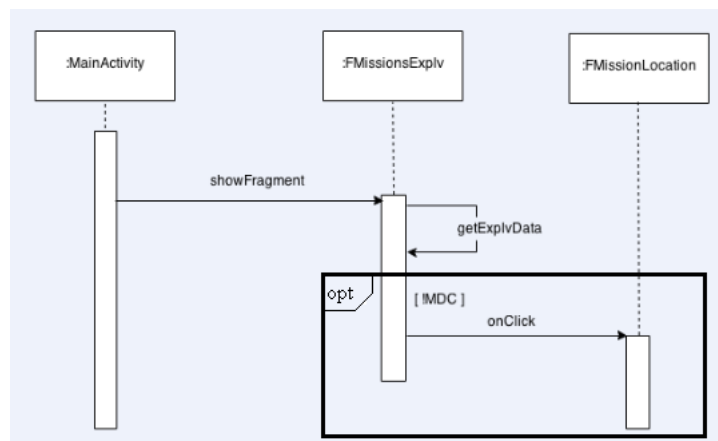


Figura 29: Diagrama de las tareas realizadas.

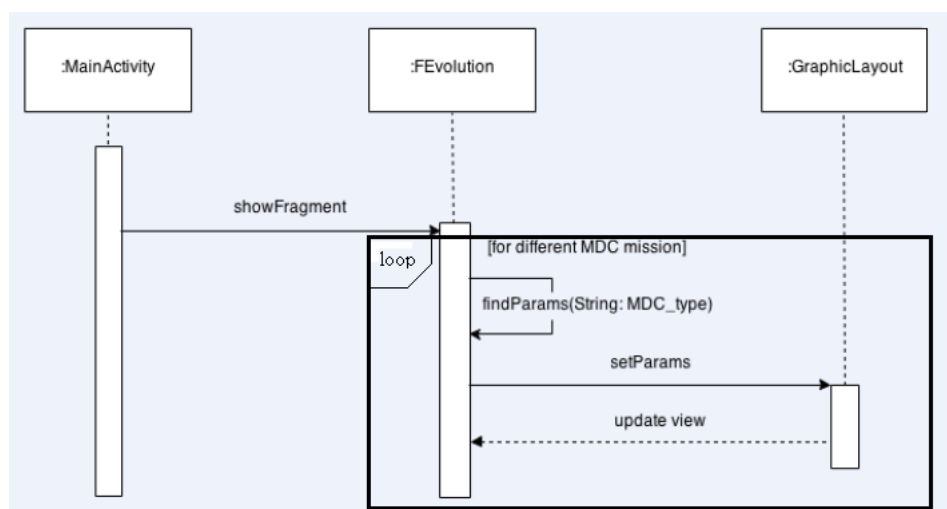


Figura 30: Diagrama de la vista de evolución.

5. Cambiar preferencias.

Esta fig. 31 muestra los objetos Preferencia utilizados para y como se modifican.

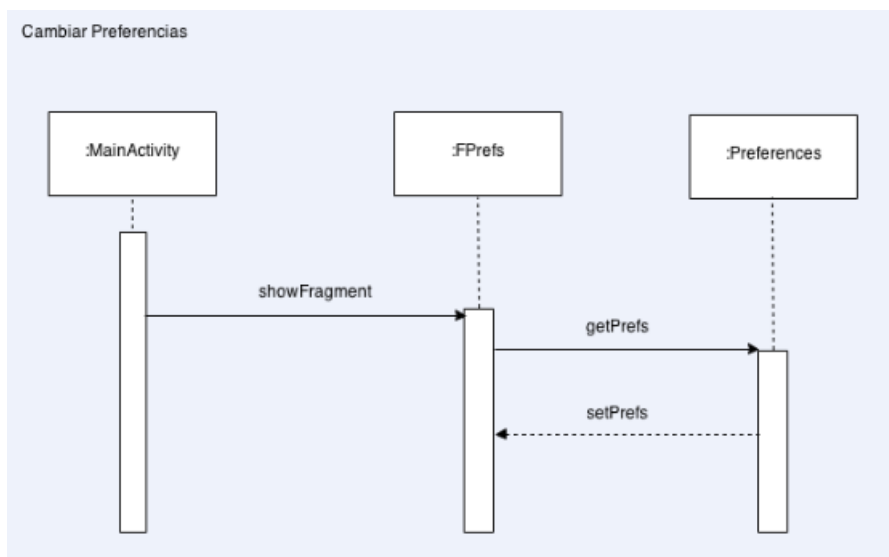


Figura 31: Diagrama de preferencias.

6. logIn, logOut, borrar cuenta.

Por último se muestran los diagramas de logIn, logOut y borrado de la aplicación. El login consta de dos diagramas el primero de ellos(fig. 32) se debe a que un usuario ya registrado no tiene porque volver a registrarse ya que su email queda registrado en preferencias de la aplicación y el segundo(fig. 33) a que pasa al tener que hacer click en el botón para registrarse. Los diagramas de login y borrado de cuenta son muy parecidos y aparecen representados en la misma figura 34.

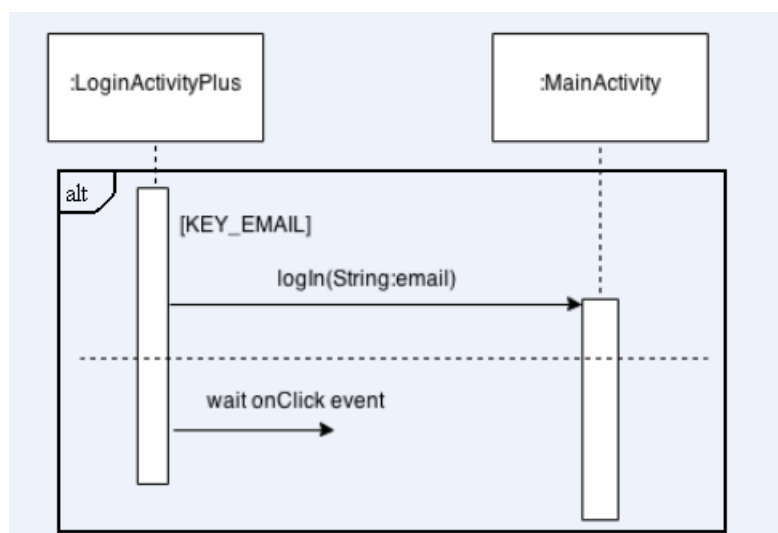


Figura 32: Primer diagrama de logIn.

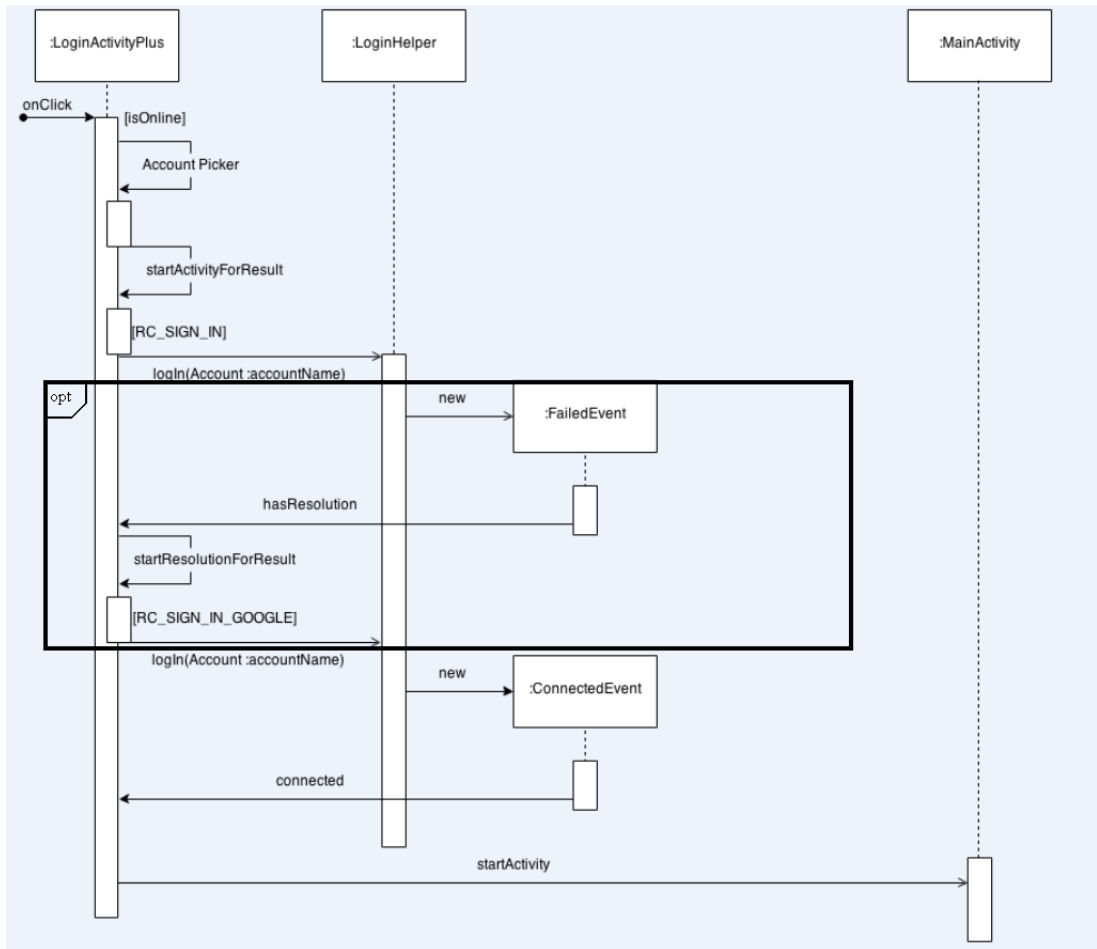


Figura 33: Segundo diagrama de login.

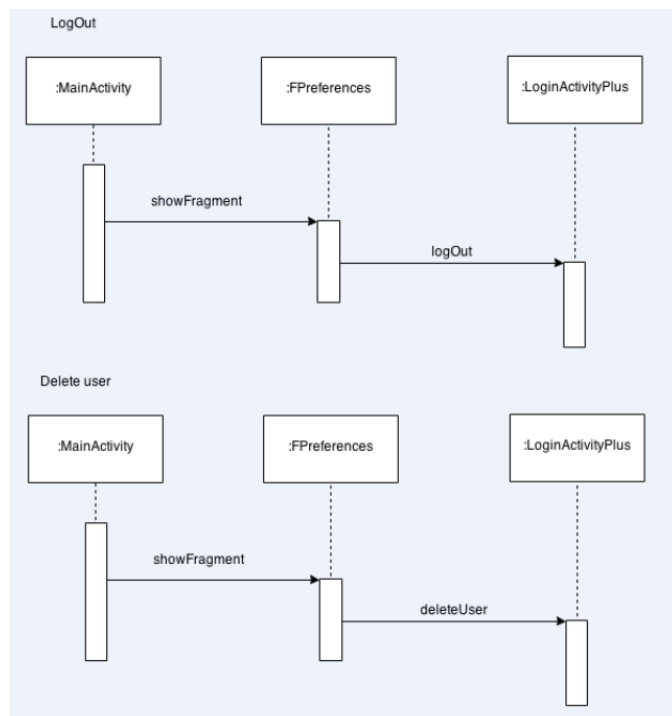


Figura 34: Diagrama de logOut y borrado de cuenta.

3.3. Implementación

Una vez definido el diseño se comienza a implementar la aplicación. Aquí es donde entran en juego los diferentes *patrones de diseño*. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Una visión más extendida de estos patrones se puede encontrar en la bibliografía.

Por lo que respecta a la aplicación, el patrón sobre el cual gira la estructura de código ha sido el *MVC(Model-View-Controller)*. Este patrón propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

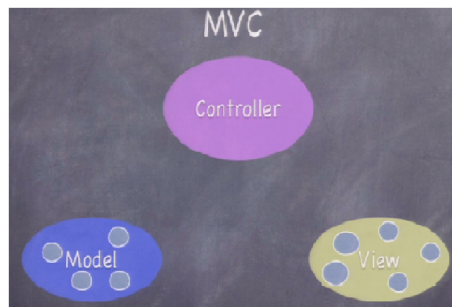


Figura 35: Patrón Modelo-Vista-Controlador.

Los componentes modelo, vista y controlador de la figura 35 se pueden definir como:

- Modelo: Qué es la aplicación(pero no como es mostrado).
- Controlador: Cómo el modelo se presenta al usuario (lógica UI).
- Vista: esclavos de los controladores, diferentes componentes de la interfaz.

Otro de los patrones utilizado para implementar el modelo funcional de la aplicación ha sido el patrón Composite. Este patrón se usa para componer objetos en estructuras tipo árbol para representar su jerarquización, y permitir tratar los objetos tanto individual como grupalmente de manera uniforme basado en una composición recursiva. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera. Un diagrama general del patrón sería el siguiente(fig. 36).

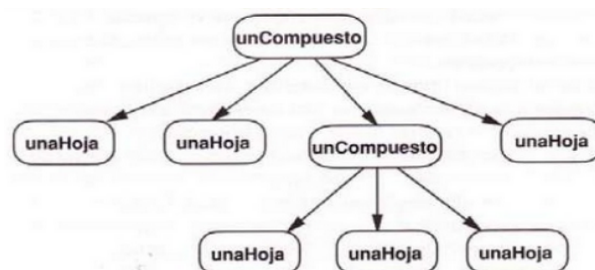


Figura 36: Diagrama patrón Composite.

En este diagrama se puede observar esta estructura de árbol. En el primer nivel se tienen tres objetos simples (hojas) y un compuesto que a su vez un nivel por debajo contiene tres objetos simples.

El patrón de diseño Observador (en inglés: Observer) define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Un símil se puede ver en la figura 37.

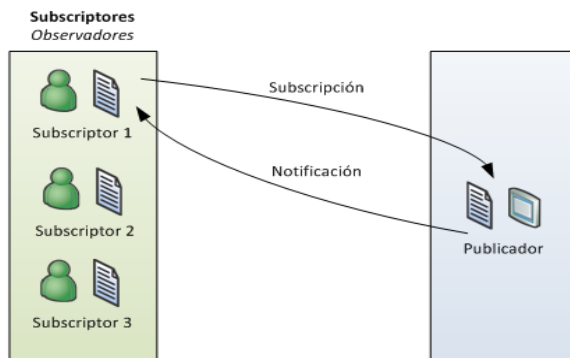


Figura 37: Patrón observer.

En esta figura se puede ver como existe un objeto que será observado(publicador) por ciertos observadores. Estos observadores serán notificados de cualquier cambio en el publicador.

Dentro de la aplicación el uso de patrón se puede ver cuando se trabaja con la plataforma AXIA o en la implementación del login.

Se ha trabajado con más patrones en la implementación de la aplicación como el patrón Singleton (tener una instancia única de un objeto) o Template (estructura de las Activities en Android) pero no se entrará en más detalle.

Por último y antes de hacer un resumen de la implementación de las diferentes funcionalidades se hablará de la jerarquía de vistas (view hierarchy) en Android y que es una característica importante a la hora de diseñar la interfaz de la aplicación. Una vista dentro de otra vista crea una jerarquía, la vista exterior viene a ser el padre de la vista interior y la vista interior se convierte en su hija. He aquí un ejemplo(fig. 38):

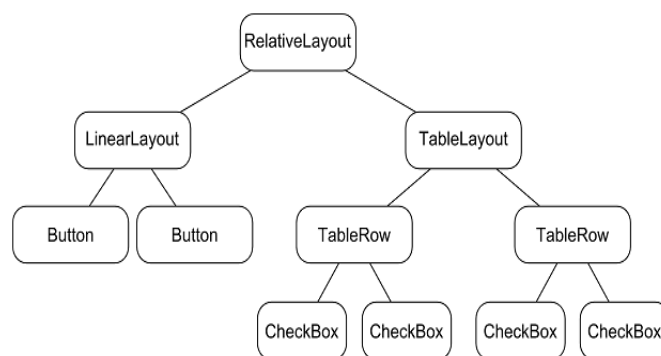


Figura 38: Jerarquía de vistas en Android.

En esta figura se puede ver este comportamiento, donde el RelativeLayout es el padre de todas las vistas y las demás vistas son las hijas pudiendo ser a la vez padres de otras vistas en su interior.

3.3.1. Modelo

En este apartado de la implementación se especifica el modelo de la aplicación. Como se ha explicado en la introducción sobre los patrones anteriores, se ha seguido el patrón Composite. En esta estructura el objeto más simple se denomina Challenge y MultiChallenge se llamará al objeto que contiene tanto nuevas MultiChallenges como Challenges. Estas dos clases implementan la interfaz IChallenge en la que se establece las pautas de comportamiento a seguir por ellas mediante tres interfaces IGameStuff, IPlanning y IChildrenStuff así como el tipo de la tarea a realizar. También existe la clase abstracta AChallenge la cual sirve para cuando un método de la interfaz no puede ser implementado por una de las dos clases. Todo esto se puede ver en el siguiente diagrama (fig. 39).

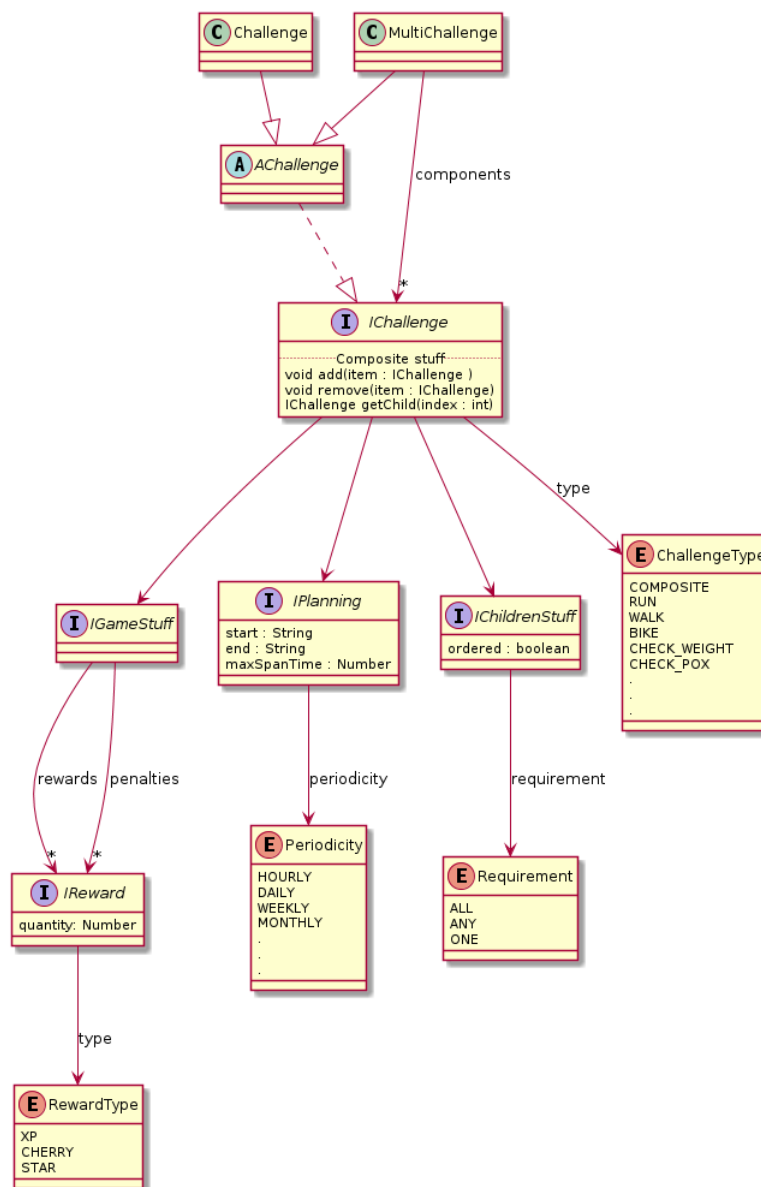


Figura 39: Modelo de la aplicación.

3.3.2. Almacenamiento de datos

En muchas ocasiones se va a necesitar almacenar información de la aplicación de manera permanente. Las alternativas más habituales para conservar esta información son los ficheros, las bases de datos o servicios a través de la red.

A lo largo de esta sección describiré cuales han sido las técnicas utilizadas y como han sido implementadas dentro de la aplicación:

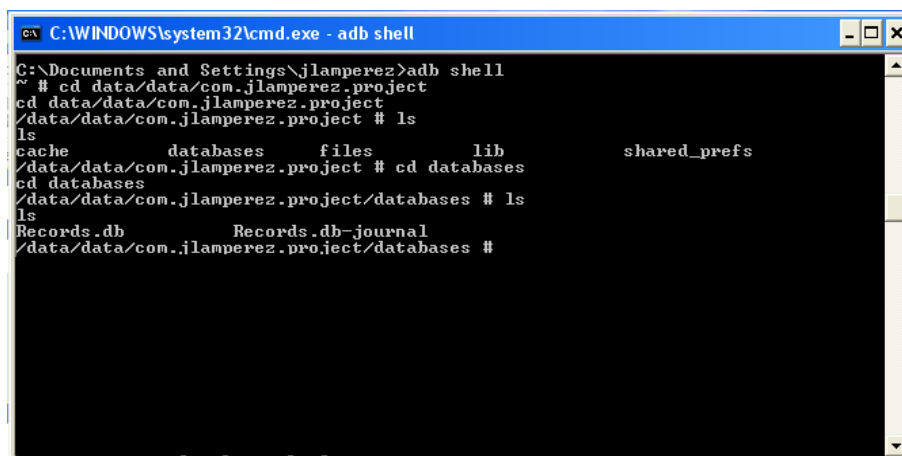
- **Base de datos**

SQLite es una base de datos OpenSource. SQLite soporta características estándar de base de datos relacionales como sintaxis SQL, transacciones y declaraciones preparadas. La base de datos requiere memoria limitada en tiempo de ejecución (approx. 250 KByte) lo que le hace candidata a ser insertada dentro de otros tiempos de ejecución.

SQLite soporta los tipos de datos *TEXT* (similar a String en Java), *INTEGER* (similar a long en Java) y *REAL* (similar a double en Java). Todos los demás tipos deben ser convertidos a uno de estos campos antes de ser guardados en la base de datos. SQLite por sí sola no valida si el tipo escrito en la columna es realmente del tipo definido, es decir, puedes escribir un entero en una columna de String y viceversa. Para más información sobre SQLite bibliografía[20].

Pero ¿porqué SQLite? SQLite esta insertado en cada dispositivo Android. Utilizar una base de datos SQLite en Android no requiere un procedimiento de inicialización o administración de la base de datos. Lo único necesario es la definicion de declaraciones SQL para creación y actualización de la base de datos. Despues de esto la base de datos es automaticamente gestionada para ti por la plataforma Android.

La base de datos de la aplicación es guardada por defecto en el directorio: *data/data/com.jlamperez.project/databases* como puede verse en la siguiente imagen(fig. 40) de línea de comandos.



```
cs C:\WINDOWS\system32\cmd.exe - adb shell
C:\Documents and Settings\jlamperez>adb shell
~ # cd data/data/com.jlamperez.project
cd data/data/com.jlamperez.project
/data/data/com.jlamperez.project # ls
ls
cache          databases      files          lib            shared_prefs
/data/data/com.jlamperez.project # cd databases
cd databases
/data/data/com.jlamperez.project/databases # ls
ls
Records.db      Records.db-journal
/data/data/com.jlamperez.project/databases #
```

Figura 40: Línea de comandos de la base de datos.

En la aplicación la clase **RecordDbHelper** que se trata de una subclase de SQLiteOpenHelper es responsable de la creación de la base de datos. Esta base de datos a su vez está constituida por tres tablas que serán creadas mediante las clases **MissionSchema**, **LocationSchema** y **MeasureSchema**. Las clases **Mission**, **Location** y **Measure** son el modelo de datos utilizado para almacenar u obtener la información de la base de datos y las clases DAO son las encargadas del mapeo objeto-relacional (ORM).

Mediante el software SQLiteadmin se puede llevar un seguimiento de las tablas de la base de datos que será como el que se puede ver en la figura 42.

■ Shared Preferences

Las preferencias no son más que datos que una aplicación debe guardar para personalizar la experiencia del usuario, por ejemplo información personal, opciones de presentación, etc.

En el punto anterior ya se ha comentado uno de los métodos disponibles en la plataforma Android para almacenar datos, como son las bases de datos SQLite. Las preferencias de una aplicación se podrían almacenar por su puesto utilizando este método, y no tendría nada de malo, pero Android proporciona otro método alternativo diseñado específicamente para administrar este tipo de datos: las *Preferencias compartidas* o *Shared preferences*.

Cada preferencia se almacena en forma de clave-valor, es decir, cada una de ellas está compuesta por un identificador único (p.e. “email”) y un valor asociado a dicho identificador (p.e. “prueba@email.com”). Además, y a diferencia de SQLite, los datos no se guardan en un fichero binario de base de datos, sino en ficheros XML.

La aplicación tiene una clase para tratar las preferencias llamada **Preferences**. Esta clase tiene dos constructores, uno de ellos permite crear la preferencia por defecto de la aplicación llamada CeresPreferences y el otro permite crear las preferencias de los diferentes usuarios y que pueden tener cualquier nombre. En esta clase también se definen ciertos métodos para el almacenamiento o recuperación de los datos.

Si se descarga por ejemplo la preferencia jlamperez10@gmail.com.xml su contenido sería el siguiente (fig. 41):

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="EMAIL">jlamperez10@gmail.com</string>
  <string name="IMAGE">https://lh5.googleusercontent.com/-2u7iB8raawc/
  AAAAAAAAAAI/AAAAAAAAABI/rRvwKuZvMIE/photo.jpg?sz=400</string>
  <string name="NAME">jorge lamperez</string>
  <int name="KEY_REWARD" value="4" />
  <int name="KEY_LEVEL" value="2" />
</map>
```

Figura 41: Preferencia de usuario.

En este XML se puede observar cómo se han almacenado cinco preferencias, con sus claves y valores correspondientes que pertenecen al usuario.

■ FicheroJSON

El fichero JSON permite tener almacenado el estado de la lista en un fichero, así cuando el usuario realiza una tarea esta se elimina y el fichero JSON se sobrescribe para poder tener la lista que se le presenta al usuario actualizada. Este fichero es creado en memoria interna por la aplicación. La clase que se encarga de crear este fichero en la aplicación es **MultiChallengeToJSON**. Para más información acerca del formato JSON bibliografía[21] [22].

SQLite Administrator - Records.db

Base De Datos Tabla Indice Ver Trigger Consulta Datos Ayuda

RECORDS

- Tablas
 - Location
 - Measure
 - Mission
 - android_metadata
 - Vistas
 - Consultas

Consulta SQL Resultado Editar Datos

Tabla: Location

_id	mission_id	latitude	longitude	time	speed	altitude	bearing	accuracy	meters
1	1118151960	42.8043032	-1.6365059	2014-06-16T18:31:49+02:00	0.0	0.0	0.0	1303.0	0.0
2	1118151960	42.8043032	-1.6365059	2014-06-16T18:32:04+02:00	0.0	0.0	0.0	1303.0	0.0
3	1117169552	42.8043032	-1.6365059	2014-06-16T18:35:38+02:00	0.0	0.0	0.0	1303.0	0.0
4	1117169552	42.8043032	-1.6365059	2014-06-16T18:35:53+02:00	0.0	0.0	0.0	1303.0	0.0

Tabla: Measure

_id	mission_id	device_name	device_address	type	absoluteTime	unit	data
4	1113672112	Manual	Manual	MDC_MASS_BODY_ACTUAL	2014-06-12T18:39:21+02:00	CHECK_WEIGHT	299
5	1113757912	Manual	Manual	MDC_MASS_BODY_ACTUAL	2014-06-16T18:44:19+02:00	CHECK_WEIGHT	33
6	1119407112	Manual	Manual	MDC_PULS_OXIM_PULS_RATE	2014-06-16T18:46:09+02:00	CHECK_POX	22

Tabla: Mission

_id	email	mission_id	type	start	end	reward	penalty
4	pepecanalla66@gmail.com	1113672112	MDC_MASS_BODY_ACTUAL	2014-06-12T18:39:21+02:00	2014-06-12T18:39:21+02:00	2	0
5	pepecanalla66@gmail.com	1118151960	RUN	2014-06-16T18:31:48+02:00	2014-06-16T18:32:08+02:00	2	0
6	jamperez10@gmail.com	1117169552	RUN	2014-06-16T18:35:37+02:00	2014-06-16T18:35:54+02:00	2	0
7	jamperez10@gmail.com	1113757912	MDC_MASS_BODY_ACTUAL	2014-06-16T18:44:18+02:00	2014-06-16T18:44:18+02:00	2	0
8	pepecanalla66@gmail.com	1119407112	MDC_PULS_OXIM_PULS_RATE	2014-06-16T18:46:09+02:00	2014-06-16T18:46:09+02:00	10	0

Figura 42: Estructura de la base de datos.

3.3.3. Axia™

Axia™ es una plataforma interoperable desarrollada sobre Java(1.6) que provee las herramientas necesarias para simplificar la gestión de las comunicaciones inalámbricas (corto/largo alcance) y por cable.

En lo que se refiere a la aplicación AXIA se ha utilizado para establecer comunicaciones Bluetooth entre diferentes dispositivos. Estos dispositivos concretos para esta primera versión de la aplicación han sido un termómetro y una báscula de la marca ForaCare aunque esta plataforma permite realizar comunicaciones con más dispositivos. Para más información acerca de esta plataforma y como emplearla consultar con la empresa LQTAI.

3.3.4. LogIn

Google+ sign-in[23] permite al usuario registrarse a la app Android con su cuenta Google y obtener información de su perfil como el nombre, email, imagen y otros detalles. La mayor ventaja de integrar G+ login es que puedes introducir a más usuarios a la aplicación y proveer un proceso de registro más fácil y simple.

En el Anexo 1 se explica como instalar el servicio google plus en la aplicación. Una vez instalado la aplicación para hacer uso de los servicios de google va a requerir ciertos permisos que deben de ser añadidos en el AndroidManifest.xml así como meta-datos para trabajar con el. Una vez hecho esto ya se podrá establecer una conexión con los servicios.

La clase de la aplicación que se encarga de esta conexión es el LoginHelper. Esta clase extiende la clase Observable de java. Es decir este objeto será observado y cuando haya en él un cambio todos los objetos que estén observandolo recibirán una actualización de su estado. Este comportamiento es lo que se ha explicado al comienzo denominado patrón Observer. Esta clase establece una conexión llamando al método connect del objeto GoogleApiClient y mediante las interfaces *ConnectionCallbacks* y *emphOnConnectionFailedListener* actualizará el estado des sus observadores dependiendo de si esta conexión ha sido exitosa o fallida.

El siguiente esquema (fig. 43) explica el funcionamiento que se ha seguido para hacer login a la aplicación:

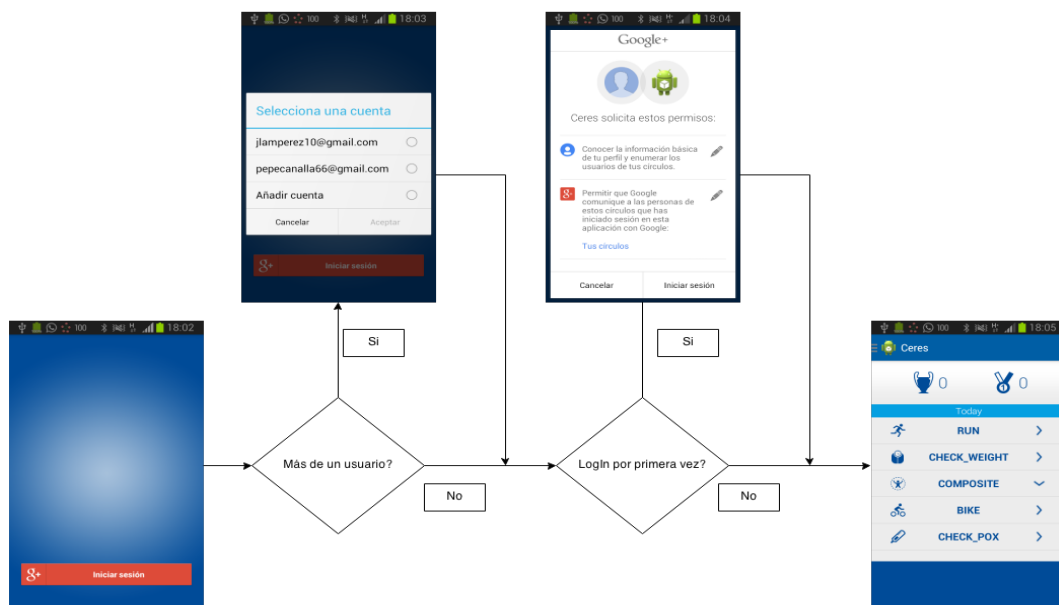


Figura 43: Esquema de logIn.

3.3.5. UI

La interfaz de usuario de la aplicación se basa en la navigation drawer, en los fragmentos y en la lista principal. A continuación, sin entrar en mucho detalle de código, se explicará cada uno de ellos.

■ Navigation drawer

La navigation drawer[24] se trata de un panel de transición en el borde izquierdo de la pantalla que muestra las principales opciones de navegación en la aplicación.

El usuario puede hacer que aparezca la navigation drawer en la pantalla deslizando el dedo desde el borde izquierdo de la pantalla o pulsando el icono de la aplicación en el action bar.

Una vez expandida la navigation drawer, esta solapa el contenido de la aplicación exceptuando el action bar. Para salir de esta existen cuatro opciones:

- Tocar el contenido fuera de esta.
- Deslizando hacia la izquierda en cualquier parte de la pantalla.
- Pulsando el icono de la app en el action bar.
- Pulsando el botón back.

A continuación se ilustra como es el diseño de la navigation drawer dentro de la aplicación.

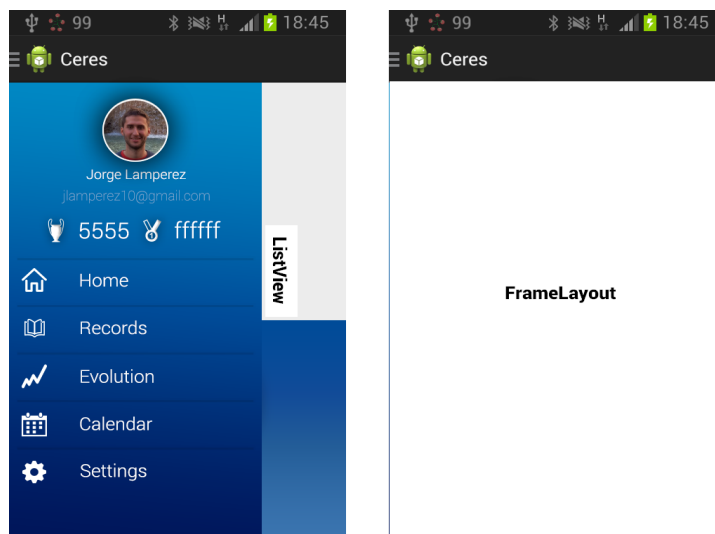


Figura 44: Navigatoin drawer.

Como se puede ver en la figura 44, existen dos vistas hijo dentro del contenedor principal que es el Drawer Layout. Estas dos vistas hijo son una ListView, permanece escondida en la parte izquierda de la pantalla y aparecerá como se ha explicado anteriormente, y un Frame Layout en el que se insertan los diferentes fragmentos de la aplicación dependiendo de la opción elegida. De los fragmentos se habla más detalladamente en el siguiente punto.

■ Fragmentos

Un fragmento representa un comportamiento o una porción de la interfaz de usuario en una Activity. Múltiples fragmentos pueden ser combinados en una única actividad para construir una UI multi panel o pueden ser reutilizado en múltiples actividades. Un fragmento debe de ir siempre embebido en una activity y su ciclo de vida está afectado por el ciclo de vida de la activity en la que este incrustado.

Los fragmentos aparecen en Android a partir de la versión 3.0(API 11) y puesto que la aplicación se ha desarrollado para versiones mayores de la versión 2.3.3(API 10) para su uso ha sido necesario la utilización de la librería de soporte. Estos fragmentos dentro de la aplicación pueden diferenciarse dado que el nombre de la clase comienza con una F mayúscula o finaliza con la palabra fragment y son insertados en la activity principal mediante el Frame Layout del apartado anterior.

La siguiente lista detalla los diferentes fragmentos dentro de la aplicación:

- **FEvolution:** Fragmento que sirve para mostrar gráficas por pantalla.
- **FMissionExplv:** Fragmento en el que se inserta una expandable list view para las tareas realizadas.
- **FMissionLocation:** Fragmento en el que se muestra en mapa.
- **FSpecRecordLocation:** Fragmento de especificación y grabación de tareas de localización.
- **FSpecRecordMeasure:** Fragmento de especificación y grabación de tareas de medida.
- **CheckDialogFragment:** Fragmento de elección del método de medida.
- **ManualDialogFragment:** Fragmento de medida manual.
- **SearchDialogFragment:** Fragmento de búsqueda de dispositivos.
- **VinculatedDialogFragment:** Fragmento de conexión con dispositivos vinculados.
- **GPSDialogFragment:** Fragmento de advertencia de activación GPS.
- **WifiDialogFragment:** Fragmento de advertencia de activación Wifi.
- **FTasks:** Fragmento con la lista de tareas a realizar.
- **FInfo:** Fragmento donde se inserta el nivel y puntuación del usuario.
- **FMain:** Fragmento principal de la aplicación donde se insertan FTasks y FMain
- **FPreferences:** Fragmento de preferencias de usuario.

■ Lista Principal

La lista principal de tareas a realizar está contenida en el fragmento FTasks. El número de niveles puede variar modificando el adaptador de esta lista que permite mostrar por pantalla el comportamiento tipo árbol del patrón composite. Esta lista de tareas se ha acotado a dos niveles y implementado como una expandable list view para la última versión de la aplicación. El comportamiento tipo árbol puede verse en la siguiente imagen (fig.45).

La fig. 45(a) muestra el primer nivel de la lista de tareas que contiene una tarea composite donde van agrupadas una serie de tareas. Pulsando el icono de expansión de

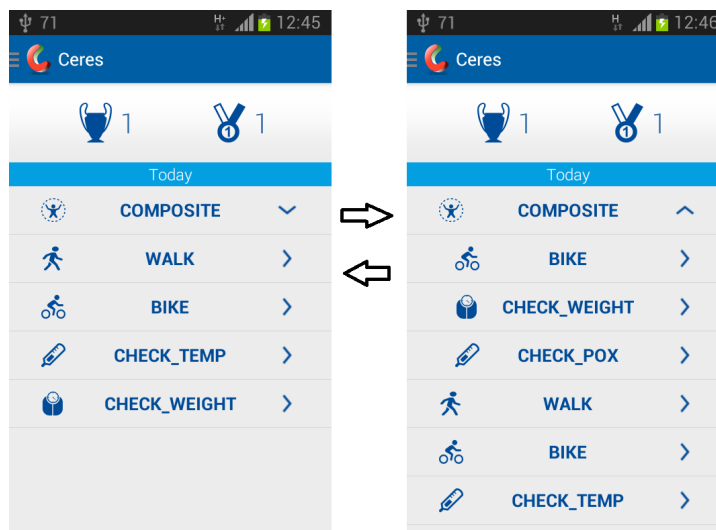


Figura 45: (a)Lista principal sin expandir. (b)Lista principal expandida.

esta tarea composite se expande el segundo nivel y pueden verse estas tareas como se muestra en fig. 45(b).

Por otro lado esta lista de tareas se va actualizando dependiendo de la hora y de las tareas realizadas en el día. Para la actualización por horas, en esta primera versión 24h, se utiliza un servicio que se encarga de recoger la lista obtenida, hace una copia y la muestra por pantalla enviando un broadcast intent. El refresco de la lista cuando se realiza una tarea se hace modificando esta copia que se trata de un fichero JSON.

3.3.6. Gráficos

Para la realización de los gráficos de la aplicación se ha utilizado la API AndroidPlot[25]. Esta API esta diseñada desde cero para la plataforma Android, y es compatible con todas las versiones Android desde 1.6 en adelante y es usada por 500+ apps en el Play Store.

Para poder realizar un gráfico mediante esta librería es necesario tener una instancia del objeto **XYPlot**. Este objeto puede ser modificado para obtener el gráfico deseado, es decir, se puede modificar los márgenes, el padding, los datos, las leyendas etc. que se encuentran en él.

Para insertar los datos dentro del objeto **XYPlot** se utiliza el objeto **SimpleXYSeries** y la orden correspondiente *addSeries()*. Estos datos se dibujan utilizando un formato establecido por el objeto **LineAndPoitFormatter**.

En la aplicación el objeto **MyCustomXYPlot** es el encargado de dibujar los gráficos y estos se insertan en **GraphicLayout**. Por último se dibujan estos GraphicLayout uno por cada tipo de medidas que se encuentren en la base de datos dentro del fragmento FEvolution.

El diseño de estos gráficos se muestra en la figura 46 donde el tipo de medida en este caso es de temperatura.

3.3.7. Dispositivos utilizados y pruebas

La implementación se ha llevado a cabo con el siguiente equipo y software:

- **Hardware:**

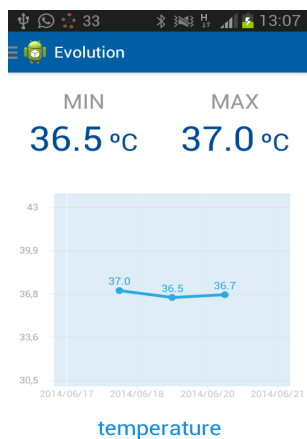


Figura 46: Gráfica para medidas de temperatura.

- Smartphone Samsung Galaxy S II i9100
 - PC Intel® Pentium® 4 a 3.40GHz con 3GB de RAM y tarjeta gráfica con salida dual
 - Dos monitores
 - Báscula y termómetro.
- **Software:**
- Eclipse
 - Android Development Tools (ADT) plugin para Eclipse.
 - Java SDK 1.7
 - TortoiseGIT
 - Adobe Photoshop CS6
 - Adobe Illustrator CS6

3.4. Verificación

La tabla 2 muestra las pruebas a las que se ha sometido a la aplicación. La columna type detalla en que parte de la aplicación se realiza la verificación, la columna # fija el número de cada prueba, la columna test especifica cual a sido la prueba, la columna result presenta su corrección y por último la columna OS presenta la versión de sistema operativo para la que se ha verificado.

TYPE	#	TEST	RESULT	OS
LogIn	1	El botón de registro muestra AccountPicker.	✓	4.1.2
LogIn	2	Si el usuario no se ha registrado con anterioridad solicitud de permisos.	✓	"
LogIn	3	Usuario que ha dado permisos no necesita volver a darlos no aparece ese dialogo.	✓	"
LogIn	4	Usuario registrado entra directamente a la aplicación.	✓	"
LogIn	5	Sin conexión a internet pulsando el botón de registro no se puede entrar en la aplicación.	✓	"
LogOut	6	Pulsando el botón Log Out se borra la pref email de la app	✓	"
Delete	7	Pulsando botón Delete User se borran las preferencias del usuario y las tareas guardadas en la base de datos	✓	"
UI	8	La lista de FTasks muestra las tareas a realizar obtenida del JSON.	✓	"
UI	9	Al realizar una tarea se actualiza la lista	✓	"
UI	10	Al realizar una satisfactoriamente se recompensa al usuario.	✓	"
UI	11	Al fallar una tarea se penaliza al usuario.	✓	"
UI	12	Las tareas realizadas(fallidas y completadas) se muestran en Records.	✓	"
UI	13	Pulsando botón en tareas de localización se muestra el recorrido.	✓	"
UI	14	Se muestra imagen en navigation drawer cuando hay conexión a internet.	✓	"
UI	15	No se muestra imagen en navigation drawer cuando no ha hay conexión a internet.	✓	"
UI	16	En la navigation drawer se muestra el nombre y email del usuario.	✓	"
UI	17	Pulsando botón Home se vuelve a la lista de tareas.	✓	"
UI	18	Pulsando en tarea de medida sale la pantalla correspondiente.	✓	"
UI	19	Pulsando en tarea de localización sale la pantalla correspondiente.	✓	"
UI	20	Si el GPS no esta activado al lanzar la pantalla de localización aparece el dialogo correspondiente para activarlo.	✓	"
UI	21	Al pulsar Start en tarea de localización se obtienen la localización.	✓	"
UI	22	La localización se actualiza cada 15 segundos.	✓	"
UI	23	Al pulsar Stop en tarea de localización se para la obtención de la localización.	✓	"
UI	24	Al pulsar Start en tarea de medida aparece el dialogo de opciones.	✓	"
UI	25	Opción Vinclated muestra los dispositivos vinculados.	✓	"
UI	26	Opción Search realiza busqueda de dispositivos.	✓	"
UI	27	Opción Manual permite insertar valor manualmente.	✓	"
Gráficos	28	Pulsando en Evolution si no hay tareas completadas MDC pantalla en blanco	✓	"
Gráficos	29	Pulsando en Evolution con una tarea MDC completa aparece un gráfico	✓	"
Gráficos	30	Pulsando en Evolution con más de una tarea MDC completa se puede hacer scroll para ver los gráficos.	✓	"
Axia	31	Conexión bluetooth con termómetro.	✓	"
Axia	32	Conexión bluetooth con báscula.	✓	"
Preferencias	33	Al entrar en la app. se guarda el email en preferencias de la aplicación.	✓	"
Preferencias	34	Cada usuario al entrar en la aplicación crea su preferencia.	✓	"
Preferencias	35	Posible modificar preferencia de notificación.	✓	"
BD	36	Almacenamiento correcto del resumen de las misiones.	✓	"
BD	37	Almacenamiento correcto de las medidas.	✓	"
BD	38	Almacenamiento correcto de la localización.	✓	"

Tabla 2: Pruebas realizadas en la aplicación.

4. Discusión

Dentro de este apartado se intenta definir el porqué de las decisiones tomadas dentro de la implementación a modo de debate.

- *Navigation Drawer*: Dentro de las aplicaciones para móviles se toma este diseño puesto que es el más utilizado en este ámbito. Este diseño permite definir diferentes niveles dentro de la aplicación.
- *Todo en uno*: En un comienzo esta aplicación iba a estar enfocada solamente en el seguimiento de la actividad física dejando la obtención de medidas para otra aplicación. Estas dos aplicaciones podían estar conectadas haciendo uso del content provider que permite conectar datos entre dos aplicaciones diferentes. Esta opción se desechó debido a que era más cómoda la opción de incluir AXIA dentro de la misma aplicación la cual provee una manera fácil de realizar conexiones con dispositivos y obtener sus datos. Esto a su vez permite tener todo dentro de una aplicación y aumentar la motivación del juego serio teniendo diferentes tareas de distintos tipos.
- *Metodología*: En este caso la metodología de cascada con feedback utilizada ha sido la más correcta por el hecho de que se carecía de una experiencia previa en este tipo de implementaciones. Esta metodología permite variar el planteamiento seguido y volver sobre tus pasos para mejorar ciertos aspectos desarrollados. La metodología de cascada sin feedback, por ejemplo, no hubiese sido correcta por los cambios sufridos durante el desarrollo.
- *Actualización de la lista principal*: En lugar de ir actualizando la lista principal a medida que el usuario realiza las diferentes tareas estas podrían seguir apareciendo en la lista principal marcadas o diferenciadas pero para ofrecerle al usuario una mejor motivación dentro de los juegos serios se opta por un borrado de la tarea realizada. Así el usuario a menos tareas por realizar puede estar más motivado para acabarlas.
- *AndroidPlot*: para la realización de gráficos en android en vez de utilizar esta librería se podía haber utilizado otra como afreechart pero dada la buena documentación existente y su continua actualización así como su uso en más de 500 aplicaciones de la play store con resultados increíbles se opta por su utilización para crear los gráficos correspondientes.
- *Axia*: Se podía haber optado por la realización de la comunicación con dispositivos médicos implementado el protocolo de comunicaciones dentro de la aplicación paso por paso pero dado que Axia ya tiene incorporado el protocolo para la marca de dispositivos Fora y que permite crear comunicaciones con estos fácilmente sin tener que adentrarse a nivel de Bluetooth se decide por su implementación dentro de la aplicación agregando los modulos correspondientes.
- *G+*: El hecho de registrar un usuario con la cuenta gmail se debe a que la mayoría de usuarios que disponen de teléfono con sistema operativo Android para optar a descargarse una aplicación del market necesita una cuenta de este tipo y de este modo el registro es fácil y no hace falta crear un usuario para la aplicación. Además el uso de esta red social permite que la aplicación tenga un toque personal teniendo cada uno su fotografía.

- *Fragmentos*: Esta aplicación en vez de utilizar fragmentos podía haber estado compuesta solo de Activities de cara a la interfaz de usuario pero se opta por los fragmentos dado que estos ofrecen una interfaz de usuario reutilizable y es una de las apuestas fuertes de Google en cuanto al diseño de interfaz de usuario.
- *Base de datos*: ¿Por qué utilizar tres tablas dentro de la base de datos y no más o menos?. Esta organización es establecida para tener una tabla principal donde se encuentra un resumen de cada tarea y de las que cuelgan las otras dos tablas. Estas dos últimas sirven para obtener más detalles acerca de las tareas de la tabla principal y pueden ser accedidas mediante un índice. Por lo tanto esta organización permite una estructura adecuada para los dos tipos de tareas que se pueden realizar en la aplicación.
- *SharedPreferences*: Se podría crear una tabla en la base de datos en lugar de guardar las preferencias pero esto resultaría en una pérdida de tiempo ya que las SharedPreferences tienen estructura key/value que hacen que la lectura de datos sea muy fácil y no necesitan ser estructurados y gestionados como con la base de datos.
- *Google Play Services*: Estos servicios que pueden actualizarse mediante el Google Play y proveen a la aplicación de las últimas ventajas en implementación de Google, es por ello que se usan a la hora de hacer login con G+ y ver los Mapas donde se dibuja el recorrido realizado.
- *Patrones*: Los patrones dentro de la aplicación son utilizados para la búsqueda de soluciones a problemas comunes en el desarrollo de software, pueden ser implementados o no pero son una ayuda dado que se ha comprobado su efectividad resolviendo problemas similares en ocasiones anteriores.
- *Medallas y trofeos*: Esto se debe a que se le quiere dar al juego una dinámica más seria en lugar de cerezas y estrellas.

5. Conclusiones y líneas futuras

En este capítulo se presentan las conclusiones obtenidas tanto a nivel de trabajo como a nivel personal. También se muestran futuras líneas de trabajo para una mejora de la aplicación.

5.1. Conclusiones

La versión final de la aplicación Ceres no ha variado mucho desde el diseño de la primera aproximación que se realizó de la aplicación. Si que se han introducido muchas nuevas funcionalidades pero la estructura siempre ha estado basada en el patrón de diseño de la *Navigation Drawer*.

Este diseño fue elegido así una vez realizada una búsqueda de aplicaciones del mismo tipo que la implementada para este proyecto. Las aplicaciones que más influencia han tenido han sido *runtastic* y *Nike+*. La primera de ellas dado que se centraba en diferentes deportes y como llevar a cabo el seguimiento del entrenamiento (mapas, tiempo, gráficas) y la segunda debido a su atractivo diseño y uso de las redes sociales.

El diseño de una lista de tareas en la pantalla principal de la aplicación es debida a que se le quiere dar a la aplicación un apoyo médico para que el usuario pueda estar controlado en todo momento. Esta lista de tareas ha sido motivo de debate y a partir de ella se ha podido llegar a implementar el modelo de la aplicación. Un modelo que en un principio era un muy extenso y que se simplificó para poder llegar a implementar esta primera versión. A partir de aquí se pudieron diferenciar dos conjuntos de tareas, simples y compuestas.

Dentro de las tareas simples se han diferenciado dos tipos: tareas de recorrido(de las que se hace un seguimiento mediante el GPS del móvil) y tareas de toma de medidas(conexión con dispositivos externos).

El modelo funcional fue la piedra angular de la aplicación. Una vez definido, implementar las diferentes funcionalidades de la aplicación fue más sencillo. A partir de aquí ayudado por diferentes APIs o librerías se ha implementado la base de datos(con su modelo de datos), las preferencias de usuario, gráficos, conexión con dispositivos externos etc.

El logIn en la aplicación ha sido una de las partes que más me ha costado implementar por como se ha querido que funcionase en la aplicación puesto que si el usuario concreto sale de la aplicación pero sin hacer logOut o borrar la cuenta esta usuario es recordado y no hay necesidad de volver a mostrar la pantalla de registro.

La obtención de la lista de tareas a partir del Back End ha sido el único punto no implementado dentro de la aplicación pero que puede ser implementado en el futuro.

5.2. Líneas futuras de trabajo

Esta versión final ofrece al usuario el modo de llevar el control de su actividad física mediante el teléfono móvil y puede utilizarse a modo de experimento para validar si el usuario objetivo tiene interés en monitorizar su actividad. Un proceso de análisis de uso de la aplicación sería fundamental para ofrecer mejoras de diseño o usabilidad al usuario.

Desde el punto de vista de la implementación se podría mejorar los siguientes aspectos:

- **Obtención de lista de tareas:** Autenticación de la aplicación con el Back End para la obtención de la lista tareas.
- **Soporte médico:** Hablar con un especialista médico para creación de las listas de tareas.

- **Redes sociales:** Mejorar el uso de las redes sociales dentro de la aplicación o crear una red social para la aplicación.
- **Diseño:** Modificar el diseño de la aplicación para una mejor visión o usabilidad.
- **Notificaciones:** Implementar notificaciones de voz..
- **Dispositivos:** Conexión a más dispositivos mediante la plataforma AXIA.

5.3. Valoraciones personales

Durante la realización de este proyecto he encontrado numerosos problemas y retos que he conseguido superar satisfactoriamente. Mi experiencia previa en desarrollo de aplicaciones para móviles era nula y con una pequeña formación inicial he podido llevar a cabo este proyecto. Esta formación inicial ha sido la que más me ha costado dado mi poco conocimiento del lenguaje Java y la programación orientada a objetos.

Al principio del proyecto no tenía muy claro que era lo que tenía que hacer, no sabía para que necesitaba Java para programar aplicaciones para móviles Android ni como realizar una aplicación para mejorar la actividad física pero poco a poco una vez metido en faena todas las cosas que había utilizado en la formación inicial empiezan a tener sentido.

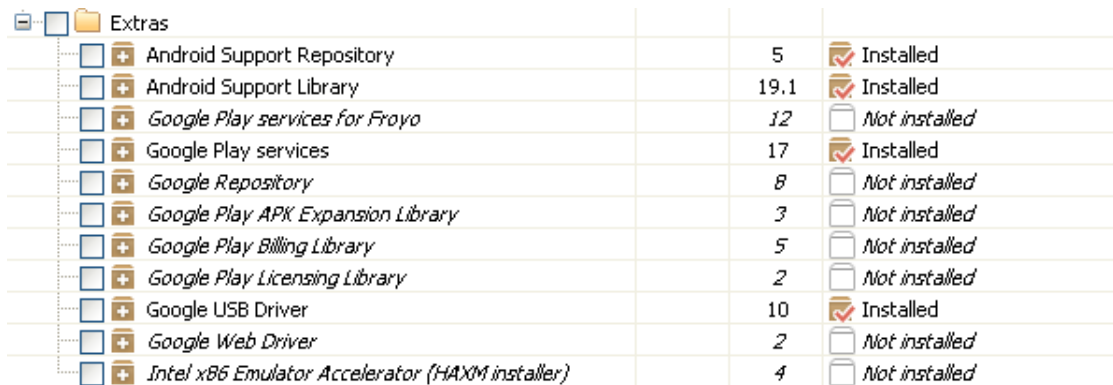
Gracias a la realización de este proyecto he conseguido cierta destreza en Java[26] y el entorno de desarrollo eclipse[27], he logrado conocimientos sobre el lenguaje UML[28] y de control de versiones (SVN/GIT) [29] y me he iniciado en el mundo de las aplicaciones para Android[30]. He podido desarrollar una metodología para crear estas aplicaciones y he contemplado las dificultades a la hora de implementarlas. He leído acerca de proyectos desarrollados por diferentes personas y he conseguido insertarlos dentro de mi proyecto para que la aplicación implementara diferentes funcionalidades como crear una conexión Bluetooth con diferentes dispositivos y lograr guardar estos datos de la conexión en la aplicación o crear un gráfico a partir de una librería de código abierto.

Este proyecto también me ha servido para trabajar en equipo y ver cómo piensan otras personas sobre la aplicación aparte de mi mismo y que me ha servido de inspiración y ayuda para su implementación.

Anexos

A. Anexo I: Google+ API

Google+ es parte de la API Google Play Service para poder utilizarlo se necesita descargar google play services de Android SDK manager como se aprecia en la figura 47 . Es muy importante tener google play services actualizado a la última versión.



Component	Version	Status
Extras		
Android Support Repository	5	Installed
Android Support Library	19.1	Installed
Google Play services for Froyo	12	Not installed
Google Play services	17	Installed
Google Repository	8	Not installed
Google Play APK Expansion Library	3	Not installed
Google Play Billing Library	5	Not installed
Google Play Licensing Library	2	Not installed
Google USB Driver	10	Installed
Google Web Driver	2	Not installed
Intel x86 Emulator Accelerator (HAXM installer)	4	Not installed

Figura 47: Captura del Android SDK.

Para realizar la autenticación y poder comunicarte con la API de Google+, primero se debe registrar el certificado público del archivo .apk firmado digitalmente en la consola de las API de Google.

PASOS

1. En la consola de las API de Google se crea el proyecto **Lab Project** para la aplicación.
2. En la izquierda, bajo la sección APIs & auth hacer click en APIs y activar el servicio Google+ API.(fig 48)

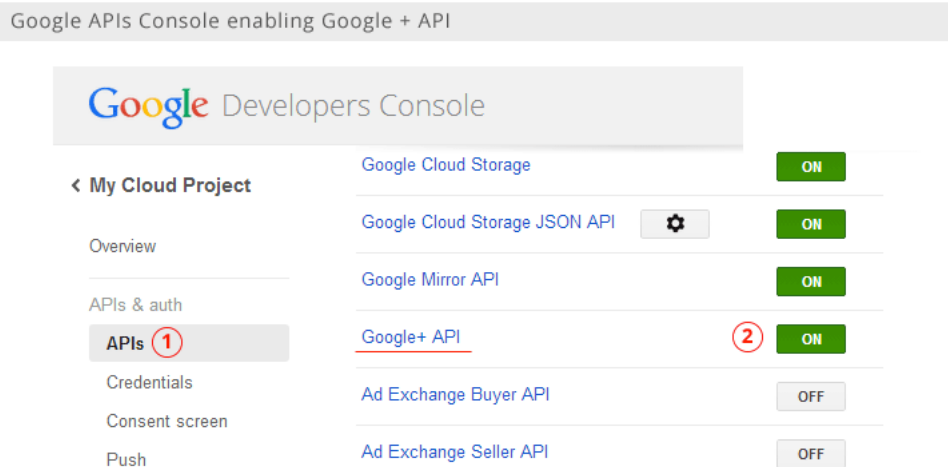


Figura 48: Consola de desarrollador, activando Google+ API

Google APIs Console creating new client ID

< My Cloud Project

Overview

APIs & auth

APIs

Credentials **1**

Consent screen

Push

OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

[Learn more](#)

CREATE NEW CLIENT ID

2

Figura 49: Consola de desarrollador, creación de credenciales

Client ID for Android application

Client ID	285463835543-e1m9hh1uikcg7d4nvrhtrgh4jg7rglta.apps.googleusercontent.com
Redirect URIs	urn:ietf:wg:oauth:2.0:oob http://localhost
Package name	com.jlamperez.project
Certificate fingerprint (SHA1)	1e:9e:2f:3b:9a:47:f6:34:38:db:79:39:76:4f:96:84:74:93:f5:4f
Deep linking	Enabled

Edit settings

Download JSON

Delete

Figura 50: Consola de desarrollador, obtención del ID Cliente

3. Otra vez en la izquierda hacer click en Credentials y darle a CREATE NEW CLIENT ID(fig. 49).
4. Una vez creado este Cliente ID con la huella SHA-1 y el nombre del paquete se obtiene(fig. 50):

Una vez hecho esto lo único necesario es importar la librería Play Services a Eclipse siguiendo estos pasos.

1. En Eclipse ir a **File** → **Import** → **Android** → **Existing Android Code Into Workspace**.
2. Hacer click en Browse y seleccionar el proyecto Google Play Services de la carpeta android sdk. Este esta localizado en **android-sdk-windows\extras\google\google_play_services\libproject\google-play-services_lib**.

3. Seleccionar **Copy projects into workspace** que permitirá tener una copia de play services en el espacio de trabajo de eclipse.
4. Después de tener la librería en el entorno de trabajo lo único que falta es añadirla al proyecto **com.jlamperez.project**(fig. 51).

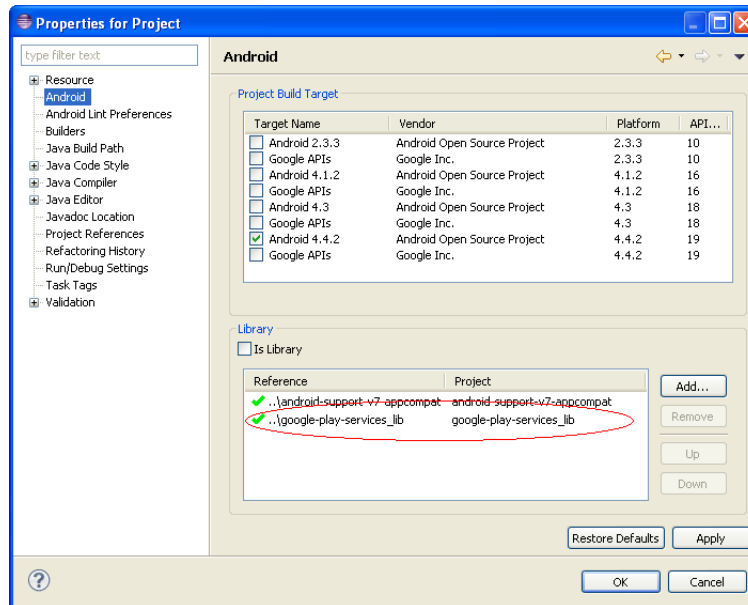


Figura 51: Añadir librería google play services al proyecto

Bibliografía

- [1] P. Puska, C. Nishida, and D. Porter, “Obesity and overweight, world health organization (who) global strategy on diet, physical activity and health fact sheets,” *Fact Sheet*, vol. 2, 2003. cited By (since 1996)4.
- [2] “Nhlbi,” *National Heart Lung and Blood Institute (Nhlbi) Obesity Education Initiative, the Practical Guide to the Identification, Evaluation and Treatment of Overweight and Obesity in Adults*, 2000. cited By (since 1996)1.
- [3] G. Villalobos, R. Almaghrabi, P. Pouladzadeh, and S. Shirmohammadi, “An image processing approach for calorie intake measurement,” *Medical Measurements and Applications Proceedings (MeMeA)*, pp. 1–5, 2012. cited By (since 1996)3.
- [4] L. Görgü, A. Campbell, K. McCusker, M. Dragone, M. O’Grady, N. O’Connor, and G. O’Hare, “Freegaming: Mobile, collaborative, adaptive and augmented exergaming,” pp. 173–179, 2010. cited By (since 1996)7.
- [5] Y. Jang, H. Noh, I. Lee, Y. Song, W. Jang, and S. Lee, “Development of an integrated obesity management waist belt system composed of calorie tracking and waist circumference measuring module for long term monitoring.,” *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2011, pp. 2172–2175, 2011. cited By (since 1996)3.
- [6] “Evaluating the impact of a cloud-based serious game on obese people,” *Computers in Human Behavior*, vol. 30, no. 0, pp. 468 – 475, 2014.
- [7] S. M. Arteaga, V. M. González, S. Kurniawan, and R. A. Benavides, “Mobile games and design requirements to increase teenagers’ physical activity,” *Pervasive and Mobile Computing*, vol. 8, no. 6, pp. 900 – 908, 2012. Special Issue on Pervasive Healthcare.
- [8] Konami *Dance Dance Revolution from Konami Digital Entertainment*, 2013. cited By (since 1996)1.
- [9] *Wii Fitness Game for Nintendo Wii, Wii Fit Plus*, 2013. cited By (since 1996)1.
- [10] “Activision,” *Guitar Hero from Activision Publishing Inc*, 2012. cited By (since 1996)1.
- [11] W. Piekarski and B. Thomas, “Arquake: the outdoor augmented reality gaming system,” *Communications of the ACM*, vol. 45, no. 1, pp. 36–38, 2012. cited By (since 1996)1.
- [12] <http://www.fitbit.com/es>.
- [13] <https://lift.do/>.
- [14] <https://www.beeminder.com/>.
- [15] <http://www.retrofitme.com/>.
- [16] <http://runkeeper.com/>.
- [17] <https://www.runtastic.com/es>.

- [18] <https://www.endomondo.com/>.
- [19] <http://www.fitlinxx.net/>.
- [20] <http://www.sqlite.org>.
- [21] <http://www.json.org/>.
- [22] <http://wiki.fasterxml.com/JacksonInFiveMinutes>.
- [23] <https://developers.google.com/>.
- [24] <http://developer.android.com/index.html>.
- [25] <http://androidplot.com/>.
- [26] B. Eckel, *Thinking in Java*. Prentice Hall, 2006.
- [27] <http://www.eclipse.org/>.
- [28] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. 3a ed. USA: Pearson Education, 2004.
- [29] <https://github.com/>.
- [30] C. K. W. Frank Ableson, Robi Sen, *Android In Action*. Manning, 2011.