

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

**Aplicación móvil para la monitorización del
ejercicio físico**

A Mobile-App for Physical exercise monitoring

Realizado por
Adrián Cárdenas Jiménez

Tutorizado por
Eduardo Guzmán de los Riscos
Mercedes Amor Pinilla

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2018

Fecha defensa:
El Secretario del Tribunal

Resumen

Este trabajo abarca el desarrollo de una aplicación para el entorno Android que, usando dispositivos de bluetooth de bajo consumo permite a los usuarios monitorizar su actividad física para analizar su progreso. La aplicación cuyo nombre es “BeFit” tiene como objetivo facilitar a las personas el uso de una aplicación común para muchos dispositivos de bluetooth de bajo consumo.

La información que recogen los dispositivos se almacena en el smartphone para ser mostrada y analizada. Si el tipo de datos es más complejo la aplicación decidirá de almacenarlos en una base de datos empotrada. Incluso, alguna información personal puede añadirse sin que sea sincronizada de ningún dispositivo. Esto facilitará a las personas la personalización y la integración en “BeFit”, añadiendo datos como el peso o la altura.

La aplicación Android permitirá escanear los dispositivos cercanos y sincronizarlos para que se muestre la información correspondiente de éstos. Además, contará con el apoyo de gráficas que ayuden visualmente al usuario mostrando un histórico de los datos que se almacenan dentro de la aplicación.

Lista de palabras clave

Android, aplicación móvil, smartphone, Bluetooth Low Energy, Xiaomi MiBand2

Abstract

This project covers the development of an application for the Android environment that, using Bluetooth Low Energy devices, allows users to monitor their physical activity and analyze their progress. The purpose of the application whose name is “BeFit” is to facilitate the use of a common application for many Bluetooth Low energy devices.

The information collected by the devices is stored in the smartphone to be displayed and analyzed. If the data type is complex, the application will decide to store them into an embedded database. In addition, some personal information can be added without being synchronized from any bluetooth device. This will make it easier for people to personalize and integrate in “BeFit” adding data such as weight or height.

The Android application will scan nearby devices and synchronize them so that the corresponding information is displayed. Also, it will be supported by graphics that visually help the user by displaying a history of the data that are stored within the application.

Keywords

Android, mobile application, smartphone, Bluetooth Low Energy, Xiaomi MiBand2

Índice

| | |
|--|----|
| 1. Introducción | 1 |
| 1.1. Introducción y motivaciones | 1 |
| 1.2. Objetivos | 2 |
| 1.3. Aplicaciones en el mercado | 2 |
| 1.4. Metodología | 3 |
| 1.5. Estructura de la memoria | 4 |
| 2. El entorno Android | 7 |
| 2.1. Introducción a Android | 7 |
| 2.2. Un sistema operativo muy particionado | 8 |
| 2.3. La arquitectura del sistema operativo | 12 |
| 2.3.1. Kernel de Linux | 12 |
| 2.3.2. Capa de Abstracción de hardware (HAL) | 12 |
| 2.3.3. Android Runtime (ART) | 13 |
| 2.3.4. Bibliotecas C/C++ nativas | 14 |
| 2.3.5. Java API Framework | 14 |
| 2.3.6. Aplicaciones del sistema | 15 |
| 2.4. Ciclo de vida de una aplicación Android | 17 |
| 3. Bluetooth Low Energy | 19 |
| 3.1. ¿Qué es el Bluetooth? | 19 |
| 3.2. ¿Qué es el Bluetooth Low Energy? | 19 |
| 3.3. GAP (Generic Access Profile) | 20 |
| 3.4. Formas de comunicación | 20 |
| 3.5. ¿Qué es el GATT? | 21 |
| 3.6. La topología de la red con GATT | 23 |
| 3.6.1. Servicios | 25 |
| 3.6.2. Características | 26 |
| 4. Recursos y herramientas utilizadas | 27 |

| | |
|------------------------------------|----|
| 4.1. Xiaomi mi band 2 | 27 |
| 4.2. Beast Sensor | 27 |
| 4.3. HTC U Ultra | 28 |
| 4.4. HTC Desire 820 | 29 |
| 4.5. Android Studio | 29 |
| 4.6. GitHub | 30 |
| 4.7. Magicdraw | 30 |
| 5. Análisis de requisitos | 31 |
| 5.1. Identificación de actores | 31 |
| 5.2. Casos de usos | 31 |
| 5.3. Requisitos funcionales | 33 |
| 5.4. Requisitos no funcionales | 35 |
| 6. Diseño e implementación | 37 |
| 6.1. Prototipado de la aplicación | 37 |
| 6.2. Diagrama de actividad | 39 |
| 6.3. Aplicación | 39 |
| 6.3.1. Diseño de la interfaz | 39 |
| 6.3.2. Desarrollo de la aplicación | 43 |
| 6.4. Diagramas de secuencia | 47 |
| 7. Mantenimiento y pruebas | 51 |
| 7.1. Mantenimiento | 51 |
| 7.2. Pruebas de la aplicación | 52 |
| 8. Conclusiones y líneas futuras | 55 |
| 8.1. Dificultades encontradas | 55 |
| 8.2. Objetivos superados | 56 |
| 8.3. Líneas futuras | 56 |
| 9. Bibliografía | 59 |

Introducción

1.1. Introducción y motivaciones

Los móviles o mejor dicho los “smartphone” ya son hoy por hoy el dispositivo del que todo el mundo dispone. Antiguamente todo el mundo usaba el móvil para llamar por teléfono, en aquella época de los famosos móviles indestructibles. La expansión de los móviles inteligentes tuvo lugar a partir de 2007 con la salida al mercado del iPhone. Después, se unirían al mercado los Android de los que hablaremos más adelante. Los sistemas operativos de ambos móviles ocupan ya casi el 100% del mercado. Aunque han tenido que luchar con otros sistemas operativos como Windows Phone o BlackBerry, éstos no han soportado el empuje de los otros móviles inteligentes y, o bien no han sabido adaptarse al mercado o no estaban preparados para tanta expansión de este mercado.

Las demandas del mercado han hecho que los ordenadores pasen a un segundo plano debido a que la mayoría de las acciones que te podían llevar a usar un ordenador, ahora se pueden hacer en un dispositivo móvil. Y es que un pequeño móvil hoy en día puede tener la misma potencia que un portátil. Y lo además se puede coger con solo una mano y llevarlo a cualquier sitio. Esta es una motivación extra a la hora de desarrollar un proyecto para smartphone y es que tiene un mercado muy grande.

La motivación que lleva a realizar este proyecto parte del hecho de que casi todo el mundo dispone de un dispositivo de monitorización de actividad física. Muchas personas hoy en día debido a las facilidades que disponemos de tener registros de nuestra actividad física nos permite mejorar nuestras marcas personales o proponernos metas. Estos dispositivos nos facilitan aún más la tarea. Podemos conseguir con un pequeño y cómodo dispositivo obtener información sobre nuestro pulso, distancia recorrida etcétera.

Estos dispositivos se están haciendo hoy en día un hueco bastante importante en el mercado, pero, tienen una desventaja y es que cada fabricante suele utilizar una aplicación de monitorización de datos propia. Esto es bueno si sólo tienes

dispositivos de un solo fabricante, pero ¿qué pasaría si quisiéramos un dispositivo de un fabricante y otro dispositivo de otro? Pues ahí tenemos el problema, y es que si deseamos tener una aplicación con la que poder sincronizar todos nuestros dispositivos no podríamos por el hecho de que cada uno posee su correspondiente aplicación. He aquí donde entra el juego este proyecto, cuyo objetivo es unificar en una misma aplicación diferentes dispositivos de distintos fabricantes.

1.2. Objetivos

Este proyecto tiene un objetivo principal. Éste es realizar una aplicación que facilite a las personas el uso de una aplicación que recoja la información de varios dispositivos. Esto hará que simplifique el uso de distintas aplicaciones, cada una de ellas muy diferente entre una y otra. Por tanto, se buscará desarrollar una aplicación sencilla y se intentarán capturar la mayor cantidad de información disponible en los dispositivos.

Además, se realizarán pruebas con distintos usuarios para ver si es una aplicación fácilmente usable por todo tipo de personas. Y sobre todo conseguir una aplicación sencilla y bastante compacta. Es un proyecto bastante complejo debido a que cada fabricante da unas posibilidades a sus dispositivos. Por ejemplo, el sensor de ritmo cardiaco está disponible en algunos dispositivos, pero en otros no.

Por detrás del desarrollo de la aplicación, también tendrá como sub-objetivos, el estudio del funcionamiento del protocolo de Bluetooth de bajo consumo y del estudio de sistema operativo Android.

1.3. Aplicaciones en el mercado

Aquí presentaremos una pequeña recopilación de aplicaciones que son de las más usadas o conocidas por los usuarios de Android similares a la que se va a desarrollar en el marco de este TFG.

- Runtastic: Una de las aplicaciones más importantes y conocidas de Android. Permite la recopilación de diferentes actividades deportivas, como correr o caminar. Permite establecer récords, tiene un reproductor de música integrado. Además, también está sincronizado con Google Fit. También permite compartir con distintos amigos tus avances.
- SportsTracker: Realiza un análisis de tus datos del entrenamiento, con los datos diarios. También tiene información de los distintos amigos y permite compartirlos por redes sociales.
- GoogleFit: La aplicación de Google para la sincronización de datos en la nube. Mucho más simple que las otras porque no tiene reproductores de música y no tiene datos de tus amigos. Sólo muestra datos personales y los objetivos que se proponga.
- Samsung Health: Como las demás registrará la información diaria personal y como las otras también permite compartirlos con los amigos. Añade una funcionalidad mayor con respecto a sus principales competidores. Y es que permite añadir un registro de todos los alimentos tomados o, por ejemplo, agua tomada. Es la más completa y la que permite tener un conocimiento exhaustivo en tu día a día.
- Xiaomi Mi fit: La aplicación de Xiaomi sólo te permite emparejarla con los dispositivos de bajo consumo de Xiaomi, te permite visualizar tu evolución diaria como las otras solo que sólo obtiene información de los dispositivos propios.
- Runkeeper: Como la mayoría de sus compañeras anteriores recopila la información diaria y también dispone de poder compartir información con los amigos. Tiene también un reproductor incorporado y permite personalizar algunos datos a la hora de realizar actividades.

1.4. Metodología

Para llevar a cabo el proyecto se ha utilizado una metodología de desarrollo de software iterativa e incremental. Se han establecido pequeñas tareas para desarrollar el proyecto. Para empezar primero se ha realizado un análisis de las distintas aplicaciones en el mercado para ver cuáles podían ser las necesidades.

En segundo lugar, se ha realizado una prueba de conexión de los distintos dispositivos en el teléfono. Así se comprueba que es lo que realmente necesita cada dispositivo. Se logra el objetivo de conocer cada requerimiento de los dispositivos.

En tercer lugar, se ha llevado a cabo un análisis de requisitos, así como un análisis de la actividad de la aplicación. Se ha desarrollado un prototipado de la aplicación. Aquí se ha logrado obtener los recursos necesarios para el desarrollo de la aplicación

En cuarto lugar, se ha desarrollado la aplicación teniendo en cuenta lo anterior. Por ello fase a fase en el desarrollo han ido surgiendo problemas que han ido solventando separándolos en pequeñas subtareas. Por una parte, ha estado el desarrollo de la interfaz en constante cambio. Aquí se han llevado a cabo varias iteraciones. Primero la aplicación partía de una aplicación sencilla que directamente conectaba con un dispositivo y luego en una posterior iteración se escaneaban los dispositivos y se conectaban. En esta segunda fase se corregían todos los errores en fases anteriores. Como en el apartado del diseño que ha tenido que verse modificado al ir añadiendo información a las pantallas.

Por último, se han desarrollado pruebas con distintas personas para ver la usabilidad de la aplicación, consiguiendo así opiniones de los usuarios.

1.5. Estructura de la memoria

Capítulo 1: Introducción

Contiene la información relativa a los objetivos del proyecto, distintas aplicaciones relacionadas, las fases y las motivaciones que han propiciado el desarrollo de éste. Además, contiene información relativa al contenido del proyecto.

Capítulo 2: El entorno Android

El capítulo 2 abarca todo lo correspondiente al sistema operativo usado en el desarrollo de la aplicación del proyecto, como la historia de su desarrollo o el ciclo de vida de las aplicaciones.

Capítulo 3: Bluetooth Low Energy

Este capítulo abarca todo lo que se refiere al protocolo usado en las comunicaciones entre los dispositivos que recopilan la información y los móviles, así como su funcionamiento.

Capítulo 4: Recursos y herramientas utilizadas

El capítulo 4 contiene un resumen de los recursos y herramientas que se han usado en el desarrollo del proyecto.

Capítulo 5: Análisis de requisitos

Se realiza un análisis de los requisitos del proyecto para tener un conocimiento de qué se va a desarrollar y añadir toda la documentación posible para en un futuro tener la máxima información para una posible continuación del desarrollo.

Capítulo 6: Diseño e implementación

Contiene todo el diseño de la aplicación, así como el desarrollo de la aplicación. Tiene toda la información relativa a los componentes usados y las ventajas e inconvenientes de lo que se ha utilizado en el desarrollo de la aplicación.

Capítulo 7: Mantenimiento y pruebas

Contiene información de las medidas que se han considerado para controlar la evolución del proyecto y de la aplicación, así como las pruebas a las que ha sido sometido.

Capítulo 8: Conclusiones y líneas futuras

En este capítulo se compone de las dificultades que se han encontrado, los objetivos superados y las líneas futuras del proyecto.

Capítulo 9: Bibliografía

Aquí se incluye el apartado de bibliografía del proyecto

El entorno Android

2.1. Introducción a Android

Android es un sistema operativo para dispositivos móviles que ocupa cerca del 80% de cuota de mercado. Su rival directo es iOS con una cuota de menos de un 20% del mercado. Entre ambos se reparten casi todo el mercado de los smartphones. En la siguiente figura podemos ver la gigantesca evolución de Android en los últimos años.

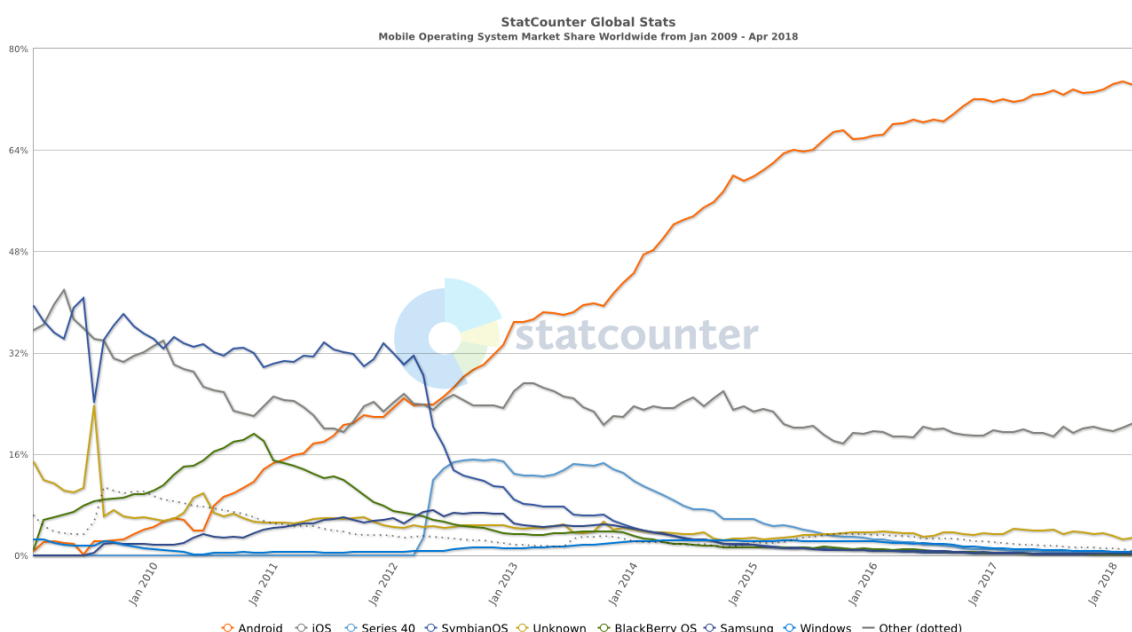


Ilustración 1. Evolución del mercado móvil. Fuente: StatCounter

Lo desarrolló Android Inc., empresa que compró Google. Se desarrolló bajo el kernel de Linux. Una gran ventaja que tiene es que se distribuye un sistema operativo actualizable y muy flexible.

Después del éxito que tuvieron los dispositivos con BlackBerry, Google desarrolló un prototipo parecido a estos. Pero después del lanzamiento al mercado de iPhone, Google tuvo que rediseñar su dispositivo. Todo esto ocurrió entre 2006 y 2007. A finales de 2008 se presentó el primer dispositivo Android HTC Dream. A partir de entonces es de sobra conocido lo sucedido con BlackBerry y con los dispositivos con iOS.

2.2. Un sistema operativo muy particionado

Una de las desventajas muy grandes que tiene Android es la cantidad de versiones con las que cuenta. Nació con la promesa de ser un sistema operativo fácilmente actualizable, pero los fabricantes han aprovechado el sistema operativo como una ventaja más a la hora de comercializar los smartphones. Es por esto por lo que contamos en el mercado con muchas versiones de Android.

Pasaremos entonces a hablar de las diferentes versiones de Android centrándonos en las que usaremos en el desarrollo de la aplicación. Como vemos en la siguiente imagen más del 80% de los dispositivos pertenecen a Lollipop (Android 5.0), Marshmallow (Android 6.0), Nougat (Android 7.0) y Oreo (8.0). Debido a ello usaremos estas versiones para el desarrollo.

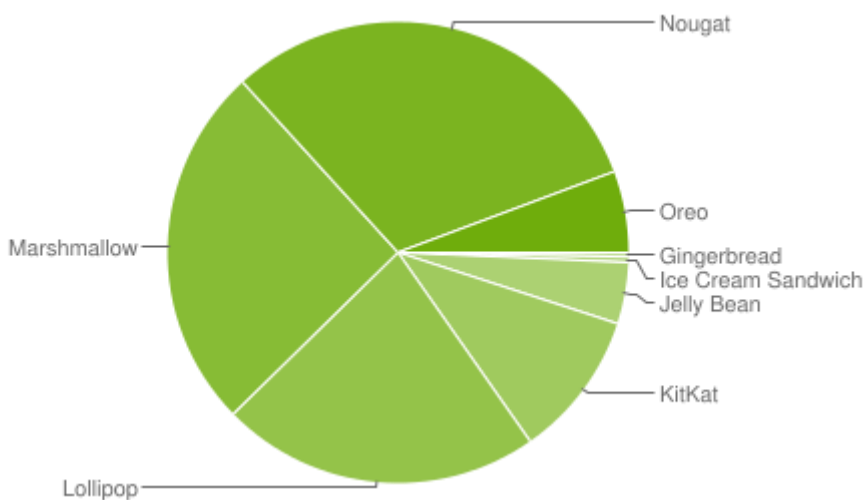


Ilustración 2. Particionamiento de versiones de Android. Fuente: Developer Android

- Android 1.x se introdujo en las primeras aplicaciones. Dentro de ellas ya se empezó a usar el hardware con arquitectura ARM. A partir de la versión de Android 1.5 (Cupcake) nació la tradición de ponerle nombre de dulces a las versiones del sistema operativo. Todas las versiones se centraron en corregir errores y mejorar algunos problemas.
- Android 2.x, a partir de estas versiones supuso el crecimiento gigantesco de Android. Incluían mejoras como el soporte de HTML5, NFC o incorporó

- el famoso Google Maps. Además, Google lanzó su famosa serie de teléfonos Nexus (Nexus One).
- Android 3.x, mejor llamarla la desconocida debido a que fue lanzada para tablets con un rediseño de la interfaz gráfica. Se incluyeron novedades como el soporte de procesadores multinúcleo o la famosa barra con los botones de navegación en la parte inferior, en vez de los botones físicos.
 - Android 4.x, el nacimiento de la era moderna de Android. Después de Android 2.3 (Gingerbread) fue la primera actualización para los dispositivos móviles.
 - Supuso el lanzamiento de los primeros dispositivos con resolución 720p (HD), a partir de ahí fue creciendo la resolución a un ritmo vertiginoso.
 - Fue la última versión en dar soporte a Adobe Flash.
 - Conllevó grandísimas mejoras de rendimiento. Android anteriormente estaba completo de animaciones, los ingenieros de Google se esforzaron porque fuesen a 30 fps (fotogramas por segundo) todas ellas.
 - Además, se introdujo Google Now, pero algo aún básico comparado a lo que hoy en día puede llegar a englobar. Se lanzó como la gran mejora del sistema ya que introducía la búsqueda predictiva.
 - Fue el final del diseño Tron que acompañaba a Android desde sus primeras versiones.
 - Se introdujo ART como sustituto de Dalvik. ART incorporaba compilación anticipada y mejoraba el rendimiento que ofrecía Dalvik.
 - Android 5.x, el lanzamiento más rompedor e importante de la historia de Android.
 - Lo más importante de este lanzamiento fue el desarrollo de “Material Design”. Esta contemplaba una serie de pautas para el desarrollo de interfaces gráficas. Se revisó por completo la IU por completo del sistema operativo.
 - Introdujeron el sistema “developer preview” que permitía disponer de la nueva versión del sistema operativo en forma de beta meses antes de su lanzamiento.

- Se sustituyó oficialmente Dalvik por ART.
- Se introduce además el soporte para procesadores de 64 bits.
- Se revisaron muchas de las aplicaciones predeterminadas que traía el sistema operativo.
- Se introdujo el reconocimiento de voz permanente. Los usuarios podían acceder a la aplicación de Google con simplemente decir “Ok Google”

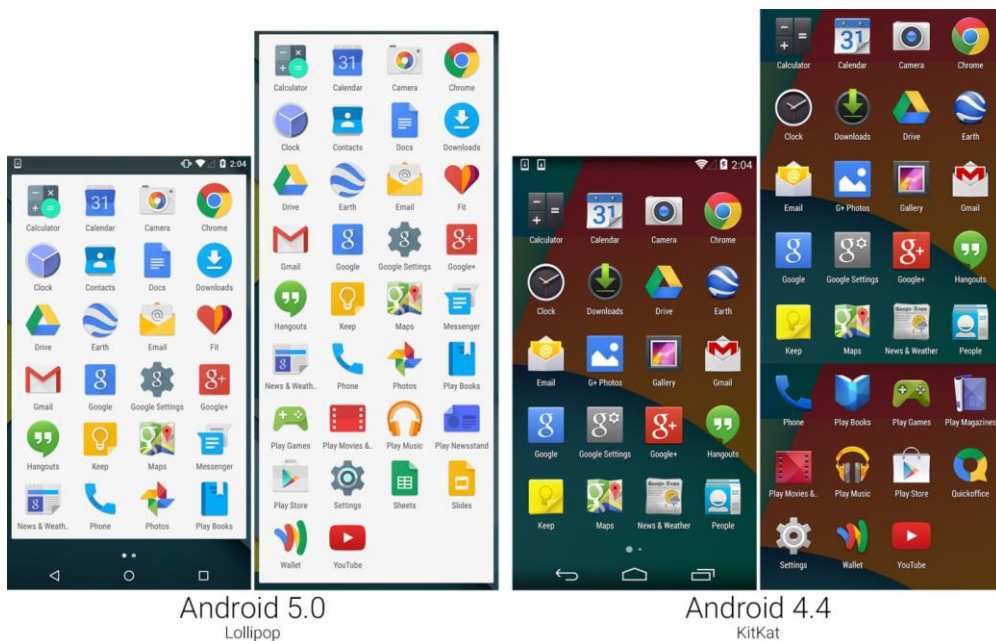


Ilustración 3. Comparación de interfaces gráficas Android 4.4 y Android 5.0. (1) Fuente: ArsTechnica

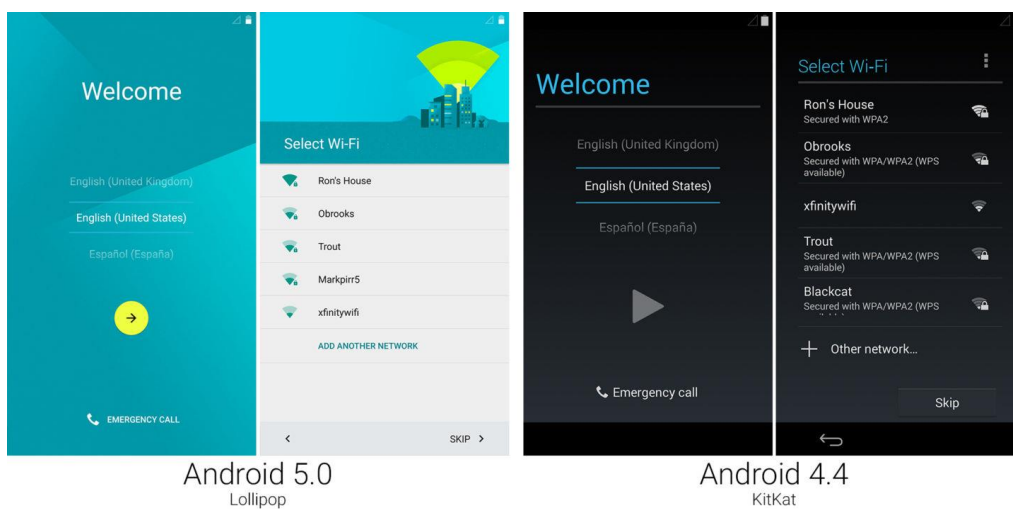


Ilustración 4. Comparación de interfaces gráficas Android 4.4 y Android 5.0. (2) Fuente: ArsTechnica

- Android 6.x, una revisión y mejoría de todas las novedades incluidas en la anterior versión.
 - Mejoras en el rendimiento del sistema operativo.
 - Se incluyeron grandes mejoras en la seguridad. Con esta versión las aplicaciones no deben pedir todos los permisos al instalarse, sólo serán requeridos antes de usarse. Se podían rechazar o aceptar dichos permisos.
 - Se introdujeron mejoras de Google Now, como el acceso rápido a ésta dejando pulsado el botón de inicio del smartphone.
 - Introducción de la "Fingerprint API". Con ella las aplicaciones podían incluir el uso de la huella dactilar para ciertas acciones.
 - Mejoras de consumo con respecto a Lollipop. Introducción de "Doze" cómo sistema de ahorro de energía.
 - Soporte para el USB de tipo C.
- Android 7.x, pequeñas mejoras (como el amplio de funciones de "Doze") y algunos rediseños de las versiones anteriores (como el de las notificaciones).
 - Se introduce el modo de visualización de pantalla dividida. Esto hace que puedas tener dos aplicaciones a la vez con la pantalla dividida.
 - Introducción de vulkan, la nueva API de OpenGL para la ejecución de gráficos 3d con un mayor rendimiento.
 - Cambió del Java Runtime Environment (De Apache Harmony se cambia a Open JDK)
 - Mejoras de seguridad (introducción de mecanismos para evitar ejecutar código malicioso en el kernel de linux)
 - Introducción de Google Daydream, que conlleva al desarrollo del mundo de la realidad virtual en dispositivos móviles.
- Android 8.x, mejoras en las notificaciones, optimización de aplicaciones en segundo plano y algunos cambios de diseño. Se introduce una versión del sistema operativo para móviles de gama baja llamada Android Go. La mejora más importante es la llamada Project Treble. Consiste en crear una capa de abstracción para que los fabricantes de móviles reciban actualizaciones del sistema sin tener que adaptar su interfaz gráfica.

- Android 9.x, esta será la futura versión de Android que incluirá algunas mejoras en el sistema operativo. Pero aún queda tiempo para su lanzamiento y para que conozcamos todas sus mejoras. Aun así, debemos mencionarla porque es parte del futuro de la compañía.

Además de todas las versiones aquí citadas, cabe destacar que cada fabricante suele traer su capa de personalización. Aunque cada vez menos común, los fabricantes en las primeras versiones de Android solían utilizar su propia capa de personalización. Estas capas de personalización pueden ser a veces parecidas a Android nativo, pero otras veces, son muy distintas, como el caso de Huawei o Xiaomi. Hay que tener también en cuenta esto a la hora de desarrollar una aplicación porque se mezclan muchas combinaciones de dispositivos con distintas versiones de Android y distintas capas de personalización

2.3. La arquitectura del sistema operativo

Android realmente es una pila de protocolos que usa un kernel de Linux modificado. Es una plataforma de código abierto y cubre una gran cantidad de dispositivos (con distinto hardware, entre ese hardware se encuentra multitud de procesadores o pantallas de distintas resoluciones).

2.3.1 Kernel de Linux

La base sobre la que se ejecuta Android es Linux. ART, una capa superior usa funcionalidades del kernel, como la creación de subprocesos o el administro del uso de memoria de bajo nivel. Otra de las ventajas es que puede aprovechar el uso de funciones de seguridad que proporciona.

2.3.2 Capa de Abstracción de hardware (HAL)

Otorga al sistema una serie de interfaces que agrupan las funcionalidades del hardware del dispositivo. Las capas de niveles más altos puedes acceder a estas bibliotecas de funcionalidades. Entonces, por ejemplo, una biblioteca implementa la interfaz de un componente hardware como podrían ser la cámara, o el audio. Su funcionamiento es fácil, cuando una capa superior, desea acceder a las funcionalidades de la cámara carga el módulo correspondiente.

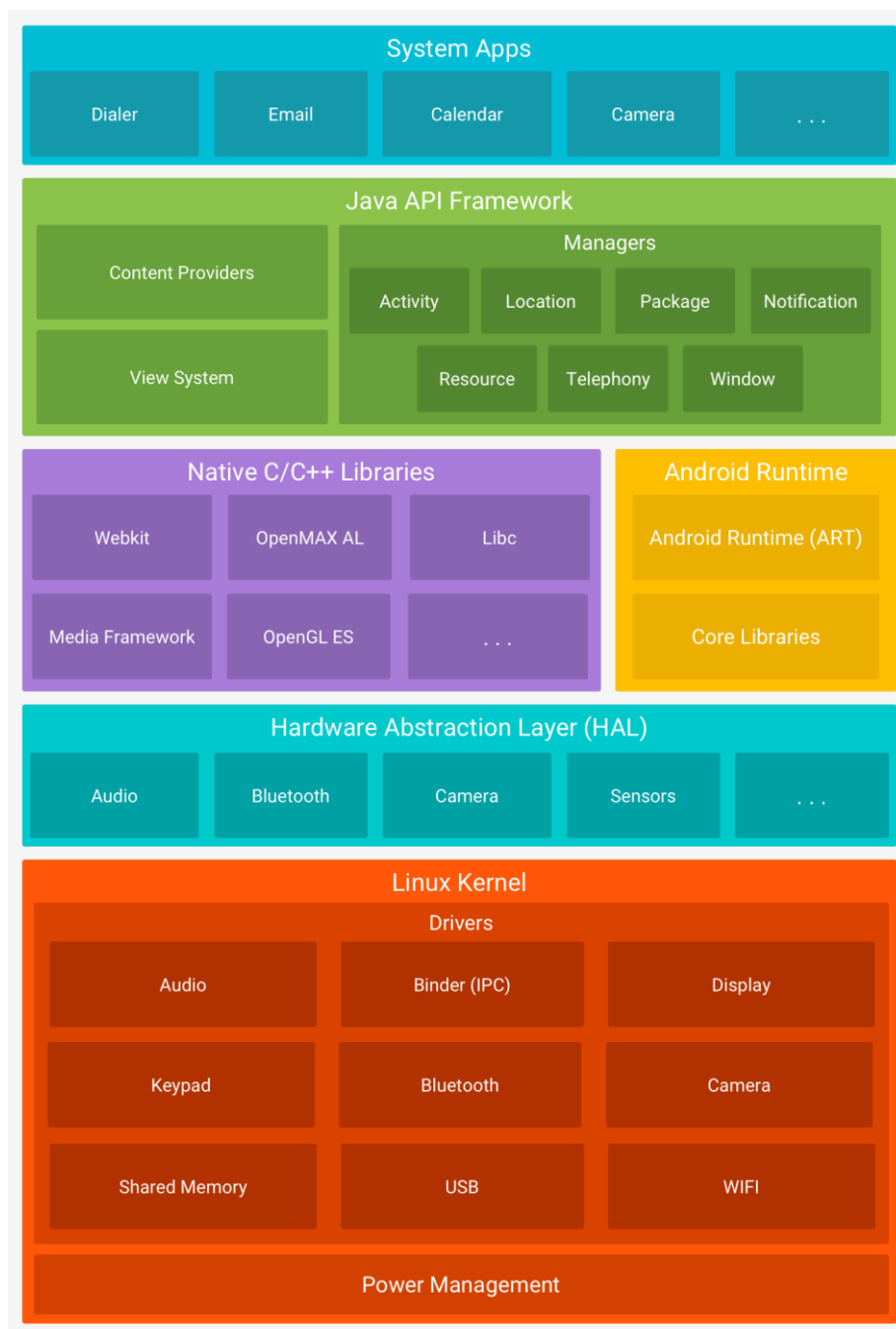


Ilustración 5. Capas del sistema operativo Android. Fuente: Android Developer

2.3.3 Android Runtime (ART)

Como ya vimos en la historia de Android, ART fue lanzado oficialmente en Android 5.0 (Lollipop). Las aplicaciones se ejecutan en su propio proceso y con su propia instancia del ART. Android Runtime está escrito para lanzar múltiples máquinas virtuales en dispositivos de poca memoria ejecutando archivos DEX.

Esto es un formato diseñado especialmente para Android, y que ocupa un espacio de memoria muy reducido.

Las principales características de ART son:

- Compilación anticipada (AOT – Ahead of time) y justo a tiempo (JIT – Just in time). Gracias a ello, permite ahorrar energías y ahorrar tiempo a la hora de ejecutar cualquier programa.
- Al igual que java usa un recolector de recursos no utilizados, solo que está optimizado para Android.
- Establece muchas facilidades a la hora de depurar el código. Algo que facilita a los desarrolladores ayudándoles a obtener más información sobre posibles errores.

2.3.4 Bibliotecas C/C++ nativas

Varios componentes de Android como el ART y la capa del HAL usan bibliotecas nativas escritas en C y C++. Android proporciona el Java API Framework que tiene algunas de estas funcionalidades. Pero también podemos acceder a algunas librerías nativas en C/C++ para desarrollar una aplicación gracias al NDK. Esto es el Kit de desarrollo nativo que permite reutilizar código C/C++, este código se ejecuta directamente en el procesador, en cierto modo hace que funcione más rápido la aplicación, pero añade mucha complejidad en el desarrollo.

2.3.5 Java API Framework

Todas las funcionalidades de Android están disponibles mediante la API escrita en Java. Sirven como capa de abstracción para simplificar el uso de los componentes del sistema.

- Contiene un sistema de vistas que facilita el desarrollo de interfaces agregando componentes como listas, cuadros de texto, etcétera
- Un administrador de recursos que proporciona un acceso rápido a imágenes, cadenas de texto o archivos de diseño.
- Un administrador de notificaciones que nos permite personalizar alertas en la barra de estado.

- Un administrador de las actividades, que maneja el ciclo de vida de una pantalla y una pila en la que se almacenan las actividades.
- Un proveedor de contenido que permite que unas aplicaciones accedan a otras para obtener datos de ellas.

2.3.6 Aplicaciones del sistema

Hay aplicaciones nativas como el calendario o los contactos que están siempre disponibles para poder usarlas, son completamente iguales que las aplicaciones desarrolladas externamente. Además, las aplicaciones que se desarrollen podrá ser usadas para reutilizar funcionalidades ya implementadas en estas. Por ejemplo, en la aplicación de las llamadas, podemos reutilizarla desde otra para marcar un teléfono desde ésta.

También vamos a ver las partes de una aplicación Android. Se divide en diferentes componentes. El más importante es el componente llamado Actividad.

- **Actividad:** Representa una pantalla con interfaz de usuario. Pongamos un ejemplo. La aplicación de contactos tiene una pantalla con la lista de todos los contactos que tenemos. Eso sería una actividad. Aplicaciones pueden tener una o muchas actividades. Todo esto depende de la complejidad de nuestra aplicación.
- **Servicios:** Este componente se ejecuta en segundo plano realizando operaciones de larga duración o tareas de procesos externos. Entonces un servicio no proporciona una interfaz como si la proporciona la actividad. Por ejemplo, un servicio puede ser el que se encargue de ejecutar música en segundo plano mientras que estamos en otra aplicación.
- **Proveedores de contenido:** Podemos almacenar datos compartidos dentro del sistema de archivos, en una base de datos, en la web, etcétera. Esto es algo lógico debido a que es normal compartir los datos y hoy en día cada vez más se usa la nube.
- **Receptores de mensajes:** Es un componente que se encarga de recoger los anuncios que envía el sistema. Estos pueden ser enviados por él mismo, pero también por aplicaciones. No tienen una interfaz gráfica, pero por ejemplo pueden crear notificaciones al recibir eventos, por ejemplo,

que se ha descargado un archivo. Generalmente su trabajo es mínimo y sirve para comunicar unos componentes con otros.

- Otras partes de la aplicación
 - Fragments: representa una parte de la interfaz de la actividad. A veces se puede dar el caso de que una actividad tenga una interfaz que solo contenga un fragment que lo contenga todo.
 - Views: Todos los elementos que se incluyen en una vista son views. Así que dentro de este grupo podemos tener listas, botones, campos de texto, etcétera.
 - Layouts: Son la estructura de las vistas de la aplicación. Todos los objetos que tengamos se engloban en la estructura de un layout. Cuando se pinta uno de ellos, se pinta con todos sus componentes.
 - Intent: Es una parte de la aplicación que se encarga de describir la acción a realizar. Se usan los intent para inicializar otros componentes como actividades o servicios.
 - Recursos: una aplicación también dispone de los recursos independientes para usar en la aplicación como imágenes, videos, sonidos (Archivos .png, .mp4, .mp3, por ejemplo). También se pueden definir animaciones, estilos, colores, cadenas de texto para todo ello usando XML. La gran ventaja de disponer de estos archivos independientes a la aplicación es que podemos tener diferentes archivos de idiomas para las cadenas de texto y dependiendo del idioma del dispositivo podremos usar un archivo u otro. También podemos hacer lo mismo para definir los UI dependiendo de las configuraciones del dispositivo.
 - Archivo de manifiesto: Para que el sistema pueda iniciar los componentes de una aplicación debe leer el “Manifest”. Aquí se declaran todos los componentes que se incluyen dentro de la aplicación. Además, dentro del manifiesto se declaran todos los permisos requeridos por la aplicación, las bibliotecas que se usan, el nivel de API de Android, todos los datos con respecto a la aplicación en general.

Algo importante que podemos destacar en Android es que, si por ejemplo una funcionalidad está implementada en otra aplicación, podemos usarla desde nuestra aplicación. Pongamos el ejemplo de compartir en una red social: si queremos compartir en una red social algo, podemos abrir el navegador desde la aplicación o mejor podríamos abrir esa aplicación, compartir lo que queramos y podríamos volver a nuestra aplicación original.

2.4. Ciclo de vida de una aplicación Android

En la navegación en la aplicación se usan diferentes actividades. Se puede modificar el uso de la aplicación según las diferentes funciones que podemos encontrar dependiendo del estado en que se encuentre la aplicación.

Cuando lanzamos una actividad, se ejecuta la creación de ésta y ahí es donde se carga la vista de la aplicación. No siempre debemos usar todas las funciones que nos facilita Android (`onPause()` o `onResume()`, por ejemplo). Siempre se usa el método de creación, porque no tendría sentido crear una actividad vacía.

Cuando vamos navegando entre aplicaciones se van ejecutando los diferentes métodos de los que dispone Android para ahorrar los recursos del sistema. Por ejemplo, tenemos un reproductor de audio, en ese caso cuando vayamos al segundo plano no debemos parar el reproductor. Sin embargo, en la reproducción de un video si podríamos pararlo. Para eso podríamos usar el método `onPause()` que hace referencia a que la actividad no está en primer plano.

Todos estos métodos sirven al sistema operativo para administrar los estados de las aplicaciones. Y dependiendo de las situaciones, destruirían algunos procesos. Si una aplicación lleva mucho tiempo sin usarse, lo normal es que se destruya el proceso y más adelante se vuelva a inicializar cuando se inicie la aplicación. Aunque hay algunas aplicaciones que siempre se están ejecutando en segundo plano, ya sean del propio sistema o externas. Por eso también existen las diferencias entre parar o pausar una aplicación.

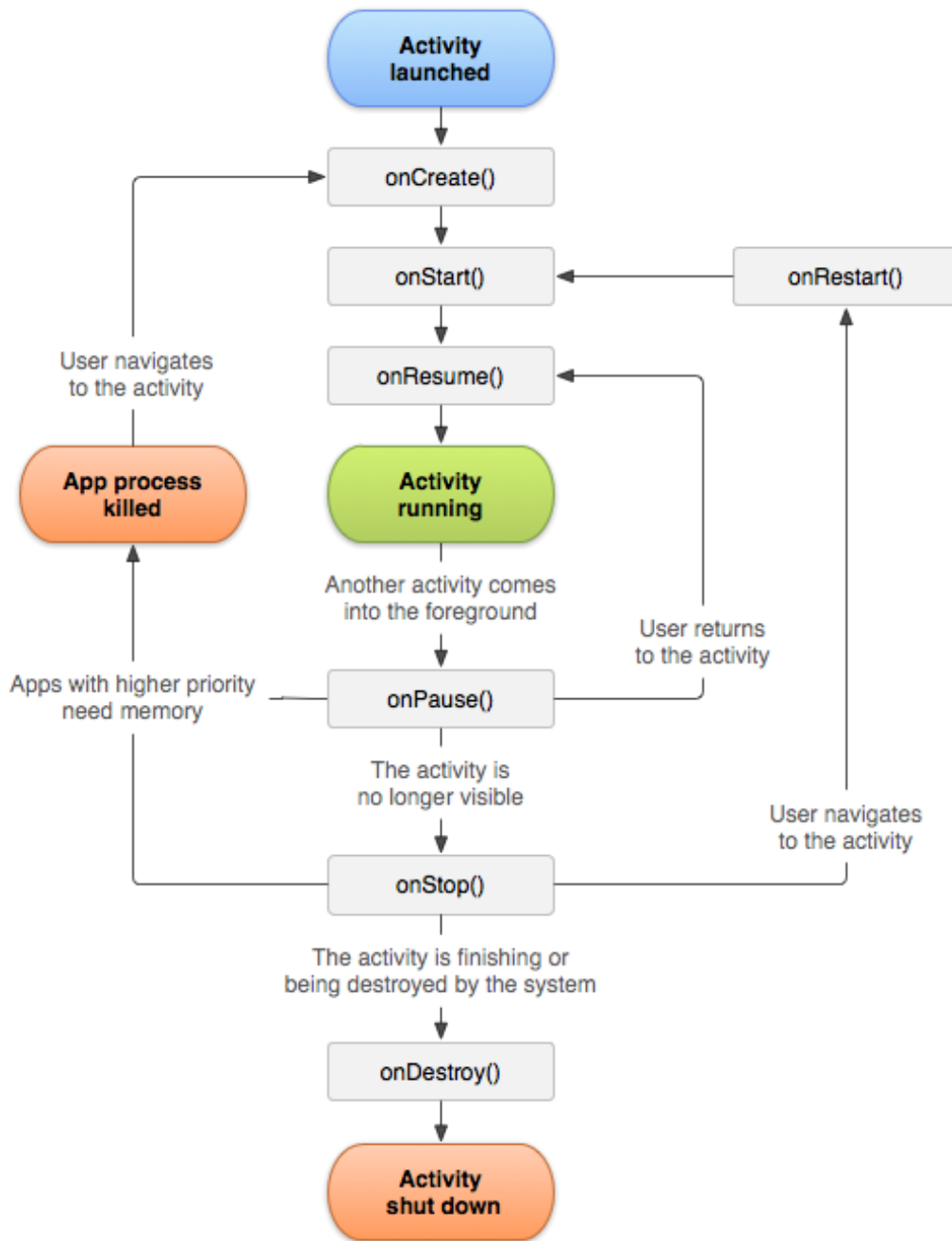


Ilustración 6. Ciclo de vida Android. Fuente: <https://developer.android.com/>

Bluetooth Low Energy

3.1. ¿Qué es el Bluetooth?

Es un protocolo de redes inalámbricas estandarizado para el envío de datos en la banda de 2.4GHz. Estandarizado por el IEEE (802.15.1), sirve para crear redes PAN (Redes de área personal). Es un protocolo seguro para el envío de datos a corta distancia. Tiene un consumo bastante reducido. Usa un modelo maestro esclavo, como veremos más adelante. Un maestro puede tener conectados hasta siete esclavos.

Este protocolo utiliza una dirección de 48-bit, de los cuales los 24 más significativos los usa para identificar al fabricante y los 24 menos para el dispositivo. También se le puede asignar nombres amigables al usuario que es el que se suele usar a la hora de presentar el dispositivo al usuario.

3.2. ¿Qué es el Bluetooth Low Energy?

Bluetooth Low Energy (BLE) es también llamado “Bluetooth Smart”. Esto es debido porque es un subconjunto del clásico Bluetooth. Se introdujo como parte del núcleo de Bluetooth 4.0. Es el gran conocido desconocido, puesto que vive con nosotros día a día. Hoy en día la mayoría de las personas tienen dispositivos para medir la frecuencia cardíaca, distancia recorrida etcétera.

Lo más interesante que tiene BLE es que al ser más ligero permite diseñar algo fácilmente que pueda comunicarse con cualquier plataforma móvil moderna. Para Apple es el único método que permite la interacción entre periféricos, sin ningún requisito legal.

Dispositivos que soportan BLE:

- IOS 5 o superior (preferiblemente IOS 7 o superior)
- Android 4.3 o superior (Grandes correcciones de errores en 4.4 o superior)
- Apple OS X 10.6 o superior
- Windows 8 (Los anteriores solo son compatibles con Bluetooth 2.1 como mucho.)
- GNU/Linux Vanilla BlueZ 4.93+

Cabe destacar que la gran característica que ofrece BLE con respecto al original es que suponiendo que el consumo de energía es de un 100% para el Bluetooth, el hermano pequeño tiene unos consumos entre el 1% y el 50%. Esto es debido a que, hasta que no se requiere el dispositivo, este queda en un estado de suspensión, lo que hace que el consumo sea el mínimo. Tanto Bluetooth como Bluetooth Low Energy operan entre dos dispositivos con el modelo maestro-esclavo para ellos los dispositivos primero deben emparejarse y luego ocurre la transmisión de información.

3.3. GAP (Generic Access Profile)

Es un estándar de la ETSI (European Telecommunications Standards Institute). Se encarga del control de las conexiones y anuncia a través del bluetooth un dispositivo. Así se encarga de que haga visible al mundo exterior y determinen qué dispositivos pueden interactuar con uno determinado o cuáles no.

En GAP se definen dos roles:

- Los dispositivos centrales, que son normalmente dispositivos móviles que tienen gran capacidad de procesamiento.
- Los dispositivos periféricos, que son dispositivos pequeños, de baja potencia y con pocos recursos. Éstos pueden conectarse a los dispositivos centrales mucho más potentes. Ejemplos de ellos son sensores de ritmo cardiaco, un beacon, sensores de pasos etcétera.

3.4. Formas de comunicación

Hay dos maneras de enviar información. El “Advertising Data Payload” y el “Scan Response Payload”.

Ambos payload (Carga útil) son idénticos y pueden contener 31 bytes de datos, pero solo “Advertising Data Payload” es obligatorio ya que se manda continuamente desde el periférico. Esto lo hacen para que el nodo central pueda saber su existencia.

El “Scan Response Payload” es opcional que puede ser solicitado por el nodo central. Permite que los periféricos manden información adicional al nodo central. Por ejemplo, el nombre del fabricante, nombre del dispositivo o alguna

información especial del dispositivo. Básicamente se trata de información extra que no es relevante.

Primero un periférico emite el “Advertising Data Payload”. El periférico especifica un intervalo para mandar esta información. Si el intervalo de tiempo se ha superado, vuelve a retransmitir dichos datos. Dependiendo del retraso del intervalo, se ahorra más energía. Es decir, si tenemos un intervalo de 1 segundo ahorra más energía que si es un intervalo de 10 milisegundos.

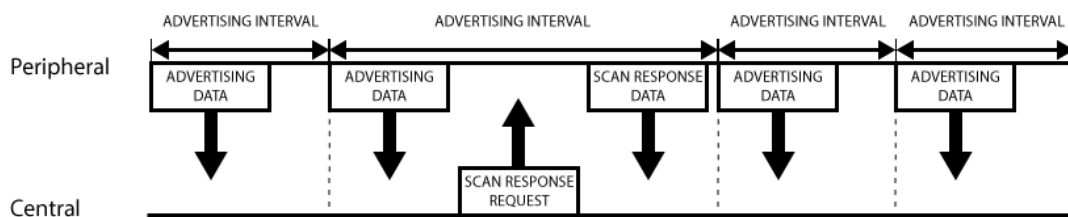


Ilustración 7. Envío de información en dispositivos GATT. Fuente: <http://learn.adafruit.com>

Si un nodo central está interesado en la carga útil que recibe de los periféricos, puede solicitar a éste que le transmita los datos adicionales.

3.5. ¿Qué es el GATT?

La mayoría de los dispositivos se anuncian para poder establecer una conexión y se utilizan los servicios y las características del GATT (Generic Attribute Profile). Esto permite obtener mucha más información entre los dispositivos, pero hay casos en los que solo se desea anunciar datos.

Un ejemplo de esto es cuando un periférico desea transmitir a varios dispositivos a la vez. Esto solo es posible con el “Advertising Data Payload” debido a que los datos enviados y recibidos solo pueden verse entre los dos dispositivos conectados.

Podemos incluir datos personalizados en el envío de información (“Advertising Data Payload” o “Scan Response Payload”), permitiendo que un periférico envíe de forma unidireccional a todos los dispositivos centrales dentro del rango. Esto se conoce como broadcasting en BLE.

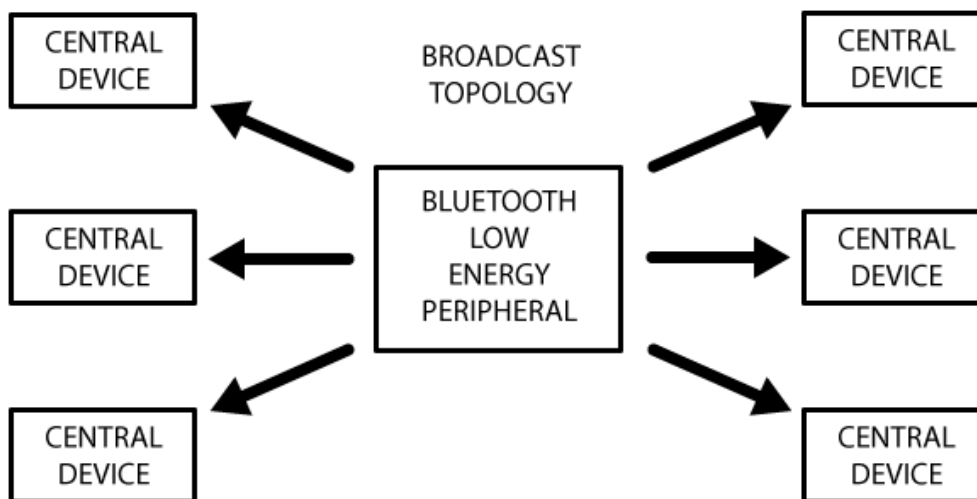


Ilustración 8. Topología de red para el envío de información en modo broadcast. Fuente: <http://learn.adafruit.com>

Una vez se establezca la conexión entre un nodo central y un periférico, se detendrá el proceso de advertising y utilizará los servicios y características que ofrece el GATT para comunicarse entre ambos dispositivos.

Como hemos visto el GAP es el encargado de manejar la conexión y el advertising entre dispositivos para poder reconocerse entre sí, pues el GATT es el encargado de la transferencia de datos entre los dispositivos BLE. Los datos se transmiten de manera que un dispositivo pueda almacenarlos en su memoria.

El GATT es un acrónimo de Generic Attribute Profile. Hace uso del protocolo de transmisión de datos ATT (Attribute Protocol), que se usa para almacenar servicios, características y datos relacionados en una tabla de búsqueda usando un ID de 16 bits para cada entrada de la tabla.

Un aspecto importantísimo del GATT es que es exclusivo. Esto es que un dispositivo BLE solo se puede conectar a un dispositivo central a la vez (móvil). Cuando el dispositivo establece la conexión, éste deja de anunciarse entonces otros dispositivos ya no podrán verlo ni conectarse hasta que haya desaparecido la conexión entre los primeros dispositivos.

Otro aspecto importante es que, en contrapartida con el GAP, éste establece una conexión que permite enviar datos de forma bidireccional. El GAP solo enviaba datos de manera unidireccional.

Existen igual que en el GAP dos roles:

- El cliente GATT, se suele subscribir a un servidor GATT. Se encarga de consultar los datos y de enviar datos al servidor para que este los almacene.
- El servidor GATT, que es el encargado de almacenar los datos para que un cliente pueda consultarlos y que éste los establezca como disponibles para su uso.

3.6. La topología de la red con GATT

Como ya hemos visto antes esta topología tiene lugar cuando los dispositivos están conectados que es además cuando se usa el GATT. Un periférico solo puede ser conectado con un dispositivo central. Eso sí, un dispositivo central, como un smartphone, puede ser conectado con múltiples periféricos BLE.

Se plantea un problema: ¿Y si queremos intercambiar información entre dos periféricos? Pues como está definido el GATT a simple vista no podríamos. La solución sería implementar un buffer o un sistema de almacenamiento en el dispositivo central y a través de éste ambos dispositivos podrían comunicarse entre sí.

Como vemos en la siguiente imagen el GATT transmite datos de manera bidireccional a diferencia de que el GAP lo hacía de manera unidireccional.

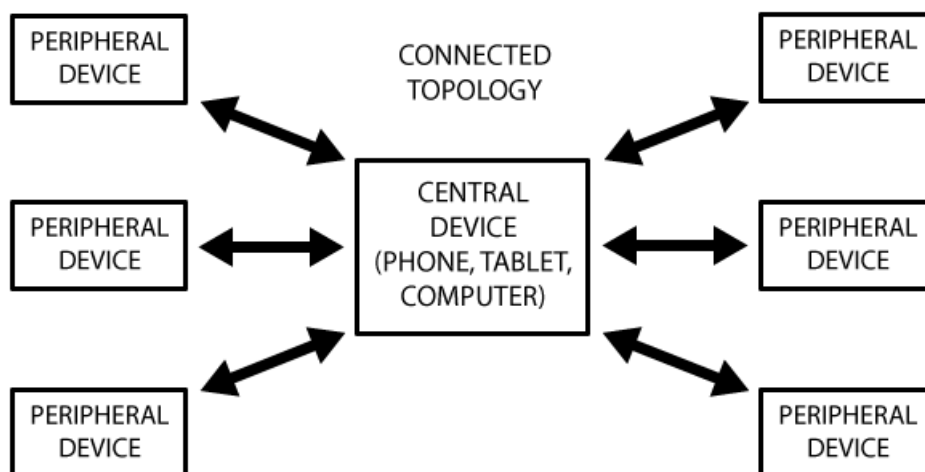


Ilustración 9. Topología de red para el envío de información en con los dispositivos conectados emparejados. Fuente: <http://learn.adafruit.com>

La relación entre el servidor y el cliente es muy importante. El periférico se conoce como el “GATT Server” que contiene los datos de búsqueda del ATT y las definiciones de servicio y características. El cliente GATT, un teléfono o una tableta, es el que envía las peticiones al servidor. Todas las transacciones están iniciadas por un dispositivo maestro, el cliente. Éste recibe la respuesta del dispositivo esclavo, el servidor.

Al establecer la conexión, el periférico envía al dispositivo central una sugerencia de “Intervalo de conexión”. Este tiempo es el que usará el maestro para consultar si hay nuevos datos disponibles. Hay que tener en cuenta que este intervalo podría no cumplirse si, por ejemplo, el tiempo de conexión acaba y tiene que volver a consultar datos, podría darse el caso de que el dispositivo central estuviese consultando datos de otro periférico o usando los recursos del sistema requeridos, entonces no podría cumplir el intervalo de conexión. Es por esto por lo que este tiempo es una sugerencia, porque puede que no se pueda cumplir.

Aquí podríamos observar una traza entre un dispositivo maestro y un dispositivo esclavo:

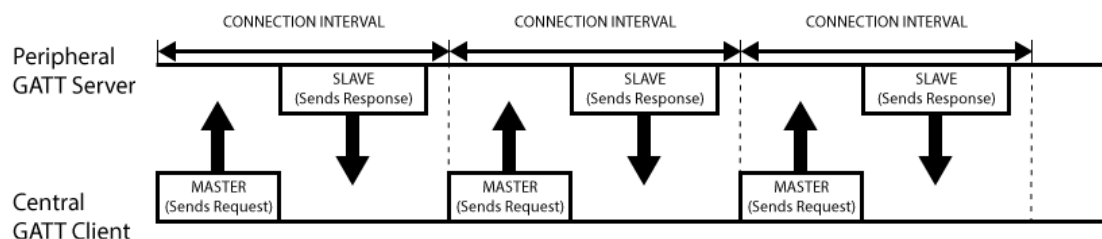


Ilustración 10. Envío de información con dispositivo conectados. Fuente: <http://learn.adafruit.com>

Estas transacciones entre servidor y cliente se basan en objetos anidados de alto nivel, llamados perfiles, servicios y características. El nivel superior es el perfil, que es el que engloba uno o más servicios necesarios para cumplir el caso de uso que cubre el objetivo de perfil. Por ejemplo, el perfil de frecuencia cardíaca combina el servicio de frecuencia cardíaca con el de información del dispositivo.

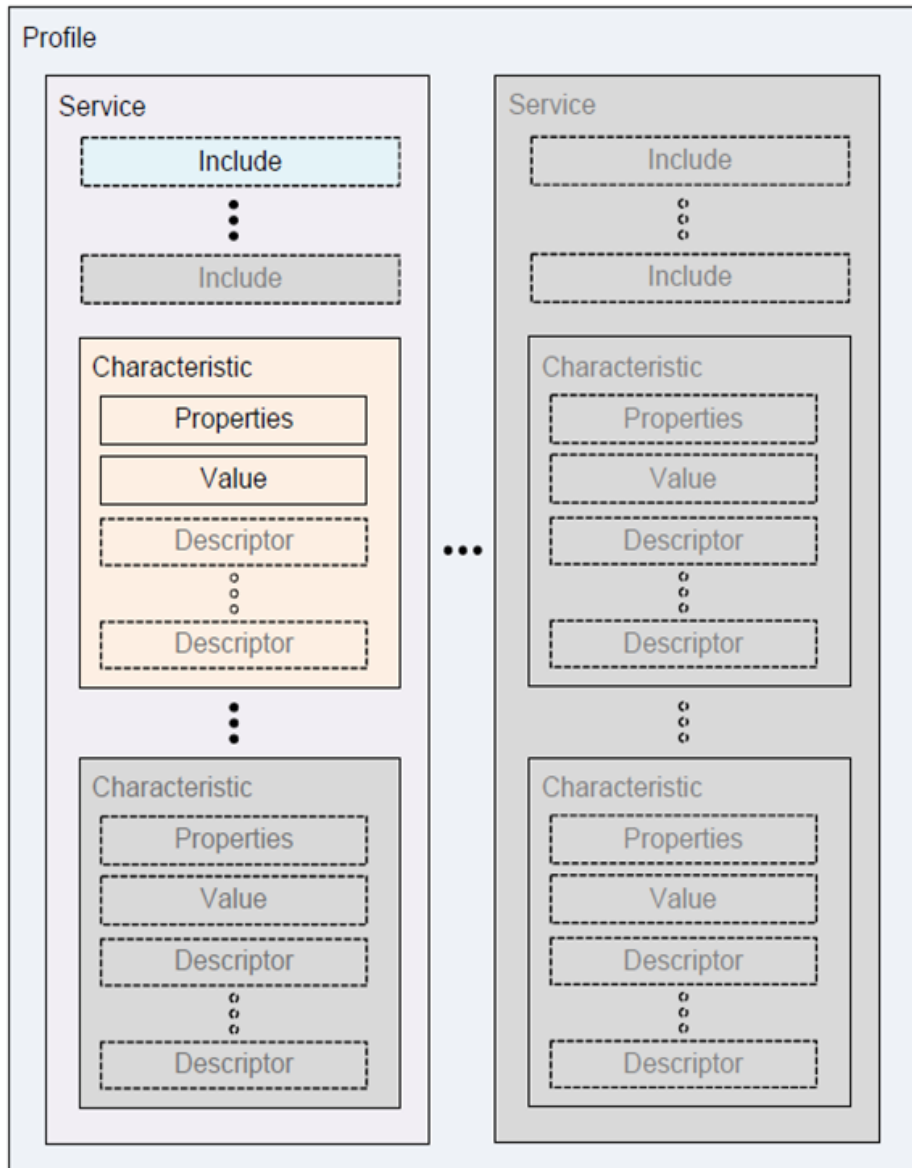


Ilustración 11. Jerarquía del GATT. Fuente: Bluetooth.com

3.6.1. Servicios

Los servicios se usan para dividir los datos en entidades lógicas. Contienen fragmentos específicos de datos. Por ejemplo, la información básica del dispositivo. Se identifican por UUID UUID, que puede ser de 16-bits (para servicios BLE adoptados oficialmente) o de 128-bits (para servicios personalizados). Un UUID es un identificador único universal, que sirve en común a todos los dispositivos sin tener usar un generador de códigos.

3.6.2. Características

Es el nivel más bajo en las transacciones del GATT. Las características es el nivel base sobre el que se interactúa en BLE. Contiene sólo un nivel lógico de datos. Por ejemplo, el servicio de frecuencia cardíaca tiene hasta tres características entre ellas la medición de la frecuencia cardíaca.

Se distingue de manera similar a los servicios. Se usan UUID de 16-bits o 128-bits, usando las oficiales o las personalizadas al igual que en los servicios. La diferencia está en que el UUID comienza con un valor de 8bits que describe el formato de los datos (UINT8 o UINT16). Los demás datos dependen del tipo de medición que se realice.

Recursos y herramientas utilizadas

4.1. Xiaomi miband2



Ilustración 12. Xiaomi mi band 2. Fuente: <http://mi.com>

Pulsera inteligente para monitorización de algunos de los parámetros que se usan en la aplicación. Datos como número de pasos diarios o ritmo cardiaco.

- Pantalla OLED 0,42"
- Bluetooth 4.2 BLE
- Batería de 70 mAh

4.2. Beast Sensor



Ilustración 13. Sensor Beast. Fuente: <https://www.thisisbeast.com>

Es un dispositivo que contiene varios tipos de sensores dentro, como un acelerómetro. Se encarga de medir la fuerza y el número de repeticiones que realizamos.

Características:

- Como ya hemos dicho contiene varios tipos de sensores. Para ser más exactos contiene tres acelerómetros, tres giroscopios y tres brújulas.
- Tiene una batería de alta duración con hasta 8 horas
- Bluetooth 4.0

4.3. HTC U Ultra



Ilustración 14. Smartphone HTC U Ultra. Fuente: <https://htc.com>

Un dispositivo de última generación. Permitirá la prueba y la mayoría del desarrollo de la aplicación móvil.

Características:

- Pantalla Quad HD (5.7 pulgadas)
- Procesador Snapdragon 820 (4x2.15Ghz)
- Bluetooth 4.2
- Memoria RAM de 4 GB
- Android 8.0
- Almacenamiento de 64 GB

4.4. HTC Desire 820



Ilustración 15. Smartphone HTC Desire 820. Fuente: <https://htc.com>

Este dispositivo permite la prueba de la aplicación con un dispositivo con características inferiores al anterior. Es un antiguo dispositivo de gama media que se enmarcaría ahora mismo en los teléfonos inteligentes de gama inferior.

Sus características principales son:

- Pantalla HD (5.5 pulgadas)
- Procesador Snapdragon 615 (4x1.5Ghz y 4x1.0Ghz)
- Bluetooth 4.0
- Memoria RAM de 2 GB
- Android 6.0
- Almacenamiento de 16 GB

4.5. Android Studio

Es el entorno de desarrollo desarrollado por JetBrains a partir de IntelliJ IDEA. Está diseñado expresamente para el desarrollo de Android y disponible para Linux, macOS y Windows. Sustituyó a Eclipse como entorno principal de desarrollo Android.

Se anunció en la conferencia Google I/O de 2013 y se lanzó la versión definitiva (1.0) a finales de 2014. Ahora mismo la versión estable actual es la 3.1.3.

Permite una depuración sencilla de aplicaciones en tiempo real, en un dispositivo virtual o en un dispositivo real. Además, tiene la integración de Gradle, que es un sistema de automatización de la construcción del proyecto (similar a Maven).

4.6. GitHub

Para el desarrollo se ha utilizado un gestor de versiones para poder tener un histórico de versiones y poder observar cómo va evolucionando el proyecto. Es una plataforma de desarrollo software colaborativa, que permite tener el código en un repositorio en la web, abierto para todo el mundo.

4.7. Magicdraw

Es una herramienta de modelado visual UML facilita el diseño de sistemas orientados a objetos y con numerosos lenguajes compatibles, así como compatibilidad con el diseño de esquemas de bases de datos.

Análisis de requisitos

5.1. Identificación de actores

Simplemente el único actor es la persona que use la aplicación para la sincronización de los dispositivos con el smartphone. Es el que recogerá todos los casos de uso que se puedan dar en la aplicación. Para simplificar la información dentro de los casos de uso no haremos referencia a este debido a que siempre es el mismo y no tiene sentido incluirlo.

5.2. Casos de usos

Analizaremos los casos de uso generales del sistema y posteriormente se llevará a cabo una especificación más a fondo del sistema.

| | |
|----------------------|--|
| Caso de uso | Buscar dispositivos |
| Descripción | El sistema se encargará de escanear todos los dispositivos Bluetooth LE y los mostrará por pantalla |
| Precondición | <ul style="list-style-type: none"> • El Bluetooth del dispositivo debe estar encendido • El usuario debe pulsar el botón de escaneo y aceptar los permisos del dispositivo |
| Postcondición | El sistema mostrará una lista con todos los dispositivos encontrados y preparará el dispositivo para la conexión |

| | |
|----------------------|--|
| Caso de uso | Personalizar Datos |
| Descripción | El sistema permitirá al usuario modificar toda la información personal que se recoge dentro de la aplicación para facilitar la mayor precisión de la información y satisfacer una mayor personalización de ésta. |
| Precondición | - |
| Postcondición | El sistema actualizará los datos del usuario dentro de la aplicación. Además, permitirá modificarlos una vez dentro, mostrando los datos actuales. |

| | |
|-----------------------|---|
| Caso de uso | Visualización de datos |
| Descripción | El sistema mostrará los datos recopilados por el sensor seleccionado, así como un resumen de su información almacenada en la aplicación. Algunos de los datos los analiza la aplicación y los almacena en ella para su uso. |
| Precondiciones | <ul style="list-style-type: none"> • Navegar por las pestañas de la aplicación • Sincronizar los datos • El Bluetooth del dispositivo debe estar encendido |
| Postcondición | El sistema muestra la información actualizada y modifica los antiguos datos actualizándolos con los nuevos. |

| | |
|-----------------------|--|
| Caso de uso | Sincronización de los datos |
| Descripción | El sistema sincroniza los datos del dispositivo para que pueda actualizarlos. |
| Precondiciones | <ul style="list-style-type: none"> • El Bluetooth del dispositivo debe estar encendido • Haber pulsado en uno de los dispositivos que se encuentren en la lista de dispositivos escaneados |
| Postcondición | El sistema actualizará los datos del dispositivo |

| | |
|-----------------------|---|
| Caso de uso | Compartir los datos |
| Descripción | El sistema permite compartir los datos diarios con otras aplicaciones |
| Precondiciones | <ul style="list-style-type: none"> • Haberse conectado a un dispositivo • Pulsar en el botón de compartir |
| Postcondición | Se comparten los datos en la aplicación correspondiente y volvemos a nuestra a aplicación |

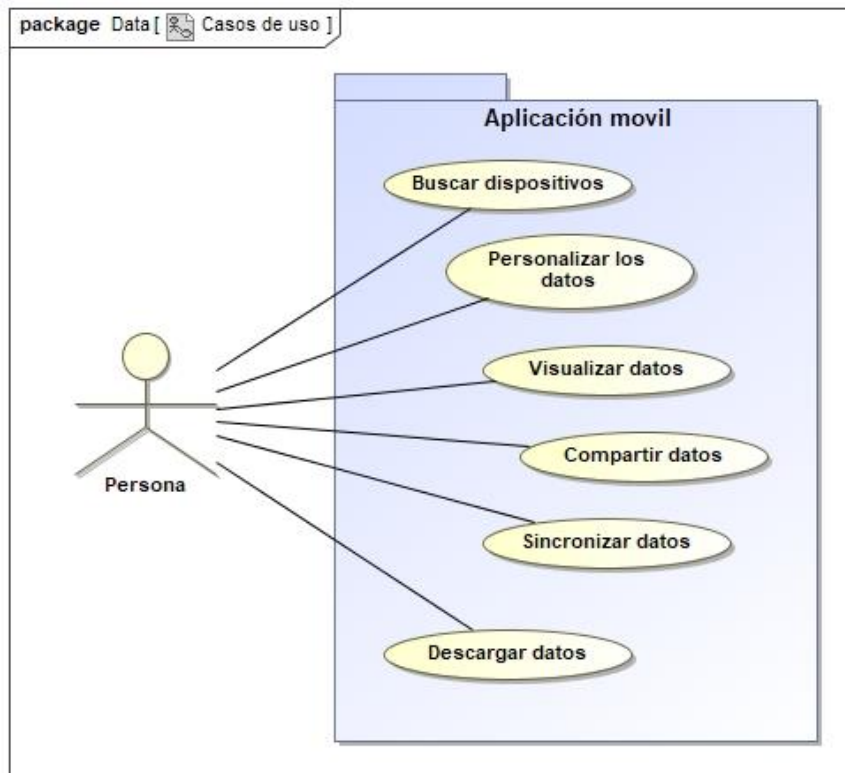


Ilustración 16. Casos de uso de la Aplicación

| | |
|-----------------------|---|
| Caso de uso | Descargar los datos |
| Descripción | El sistema permite descargar el histórico de los datos |
| Precondiciones | <ul style="list-style-type: none"> • Haberse conectado a un dispositivo para poder descargar los datos |
| Postcondición | El sistema actualizará los datos del dispositivo |

5.3. Requisitos funcionales

| | |
|--------------------|--|
| Nombre | Escanear dispositivos |
| Descripción | El sistema permitirá el escaneado de todos los dispositivos BLE cercanos |

| | |
|--------------------|--|
| Nombre | Detener escaneo de dispositivos |
| Descripción | El sistema permitirá detener el escaneo de un dispositivo. |

| | |
|--------------------|---|
| Nombre | Configurar datos |
| Descripción | El sistema permitirá la configuración de los datos personales como: <ul style="list-style-type: none"> ▪ Peso ▪ Altura ▪ Edad ▪ Meta de pasos diarios ▪ Nombre ▪ Foto |

| | |
|--------------------|---|
| Nombre | Visualización de datos para miband 2 |
| Descripción | El sistema mostrará la información relativa a la pulsera inteligente con datos como los pasos o el ritmo cardiaco |

| | |
|--------------------|---|
| Nombre | Visualización de gráficas de datos |
| Descripción | El sistema gracias al escaneo o la introducción de información como el peso elaborará gráficas para que estén disponibles para el usuario |

| | |
|--------------------|--|
| Nombre | Visualización de Tablas |
| Descripción | Como complemento a la visualización de datos de gráficas se darán tablas para facilitar su visualización |

| | |
|--------------------|--|
| Nombre | Descarga de datos |
| Descripción | El usuario podrá descargar su histórico de datos para su posterior análisis. |

| | |
|--------------------|---|
| Nombre | Visualización de errores |
| Descripción | El sistema informará visualmente siempre que ocurra un error. |

| | |
|--------------------|--|
| Nombre | Compartir datos |
| Descripción | El sistema permitirá el compartir los datos diarios personales en distintas aplicaciones |

5.4. Requisitos no funcionales

| | |
|--------------------|--|
| Nombre | Tiempo de escaneo |
| Descripción | Debido a que el escaneo de los dispositivos suele tardar porque los dispositivos BLE emiten una señal cada un cierto tiempo, este requisito realmente depende de nosotros. |

| | |
|--------------------|---|
| Nombre | Requisitos de fiabilidad |
| Descripción | Estableceremos la fiabilidad de la aplicación al ser joven, no ser un sistema crítico y depender de otros dispositivos en un 95%. |

| | |
|--------------------|--|
| Nombre | Requisitos de compatibilidad |
| Descripción | El sistema será compatible con todos los dispositivos superiores a Android 5.0 |

| | |
|--------------------|--|
| Nombre | Requisitos de eficiencia |
| Descripción | El sistema no afectará gravemente a la batería de los dispositivos en los que se establezca la conexión y tendrá un tiempo de respuesta inferior a los 5 segundos evitando los bloqueos. |

| Nombre | Requisitos de usabilidad |
|--------------------|--|
| Descripción | <ul style="list-style-type: none"><li data-bbox="576 250 1361 398">• El sistema permitirá el 80% de las veces con un máximo de 3 pulsaciones obtener la información deseada.<li data-bbox="576 421 1361 515">• Estará disponible en los idiomas de inglés, francés y español. |

Diseño e implementación

6.1. Prototipado de la aplicación

Como podemos observar en las capturas siguientes, atendiendo a los requisitos, se ha desarrollado este prototipo de aplicación. Podemos encontrar por una parte una ventana de configuración la cual nos permitira insertar toda la información correspondiente al usuario.

Por otro lado, encontraremos un botón de escaneo, que irá actualizando la lista de dispositivos cercanos a los que conectarse. Una vez en ella, cuando pulsemos, nos llevará a la pantalla que contiene la información general de los dispositivos.



Ilustración 18. Pantalla de configuración

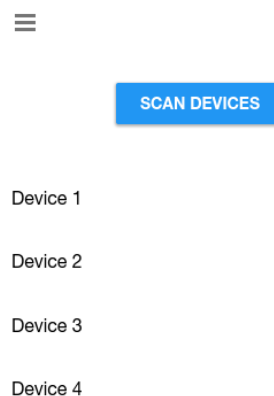


Ilustración 17. Pantalla de escaneo

Mostraremos la información actual del dispositivo. Además podremos consultar en las distintas pestañas gráficas con información acerca de la aplicación.

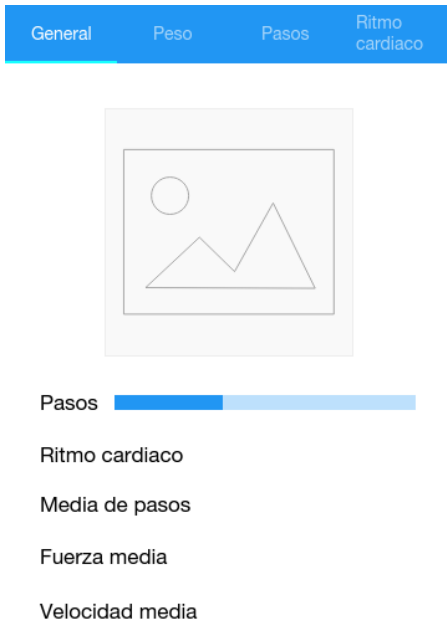


Ilustración 19. Evolución de peso



Ilustración 20. Información general



Ilustración 21. Evolución de ritmo cardíaco Evolución de pasos



Ilustración 22. Evolución de ritmo cardíaco

6.2. Diagramas de actividad

Aquí veremos una traza completa de la aplicación, mostrando todas las posibles interacciones dentro de ella. Ésta posteriormente nos servirá para su desarrollo.

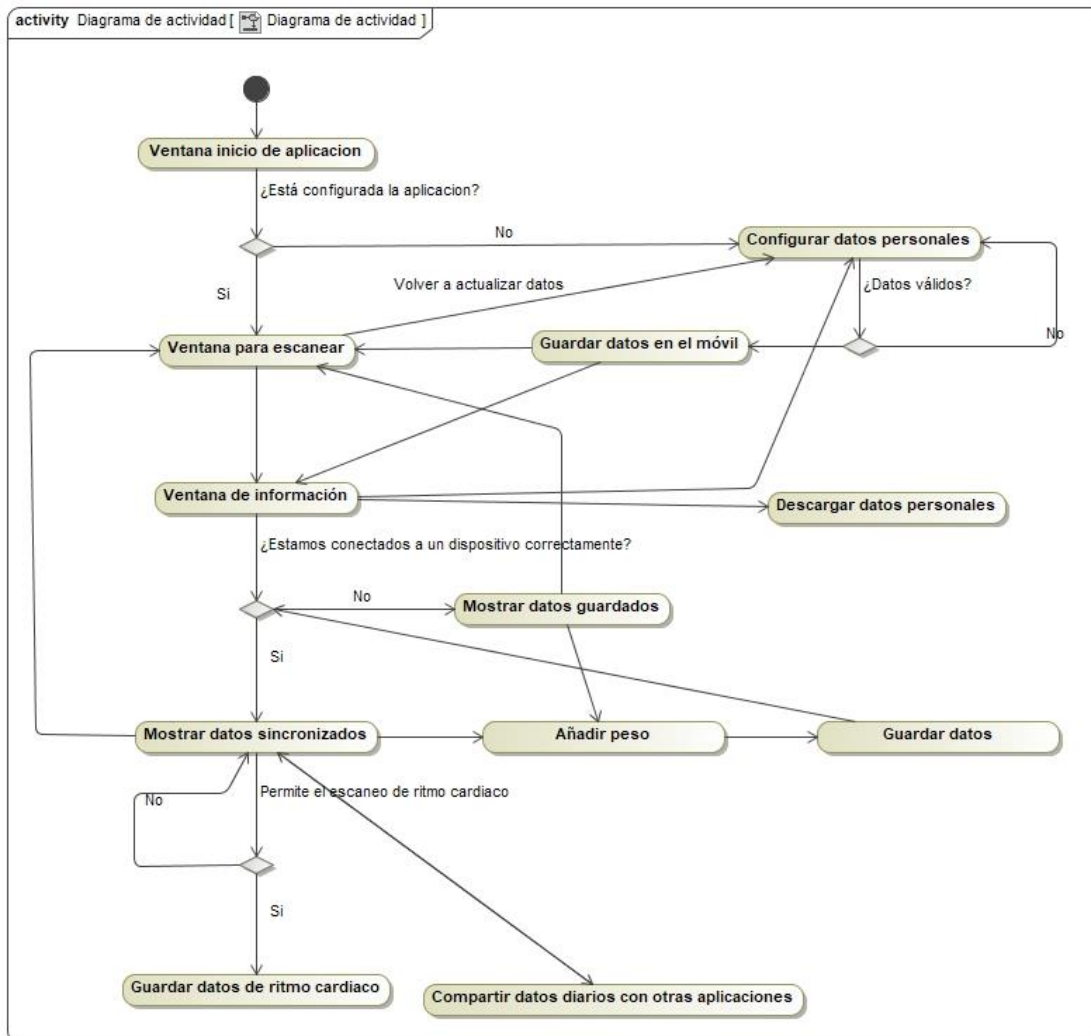


Ilustración 23. Diagrama de Actividad de la aplicación

6.3. Aplicación

6.3.1. Diseño de la interfaz

Para empezar, se ha realizado una pantalla de bienvenida que dota a la aplicación de un aspecto más agradable. Para ello, una vez entrada la primera vez a la aplicación y configurados sus datos personales, se mostrará un mensaje personalizado con su nombre.

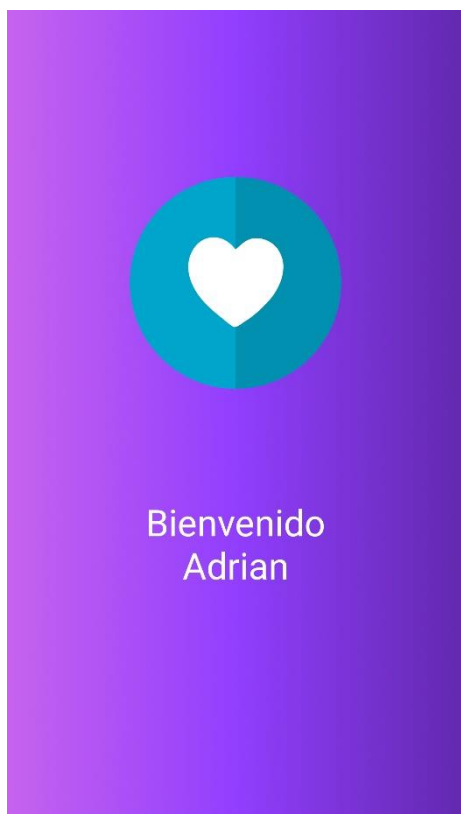


Ilustración 24. Pantalla de bienvenida

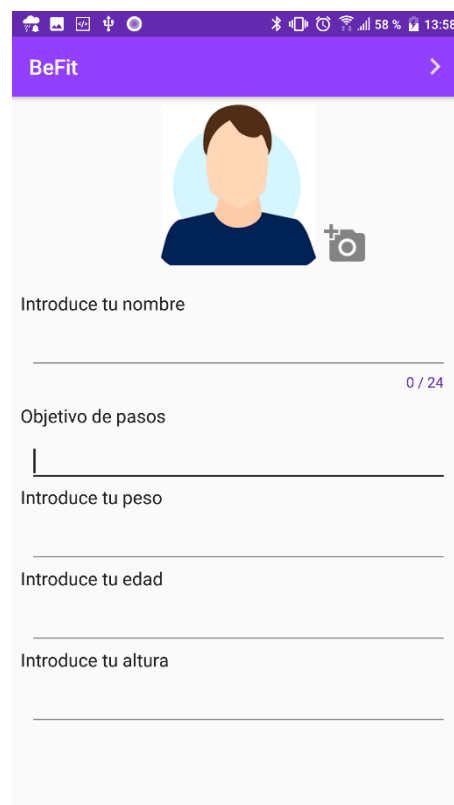


Ilustración 25. Pantalla de configuración

Como ya vimos en el prototipo, tendremos una pantalla de configuración para personalizar los datos del usuario. En él se pedirán los datos que posteriormente se usarán para, por ejemplo, calcular el índice de masa muscular. Otros datos simplemente servirán para fijar una meta de pasos diarios o como el caso del nombre para el mensaje de bienvenida. También se puede añadir una foto con el hecho de poder personalizar más la aplicación.

Una vez configurados los datos con el botón que encontramos en la parte superior, se guardarán los datos en el teléfono y la aplicación procederá a redirigirnos a la pantalla encargada de buscar todos los dispositivos cercanos. Una vez en esta ventana podremos escanear los dispositivos o parar la búsqueda de dispositivos si queremos detenerla. Además, también podremos volver a configurar los datos desde esta pantalla, ya que es la pantalla principal de nuestra aplicación y se encargará de encontrar los dispositivos y una vez pulsado en el dispositivo que nos interese se encargará de recopilar toda la información.

Como vemos en las ilustraciones, se recoge la información correspondiente al dispositivo. Además, una vez sincronizados los datos obtendremos unas gráficas como la mostrada a continuación de la distancia recorrida en metros. Todas las pestañas harán referencia a una gráfica mostrando un histórico ordenado por fecha de la información correspondiente. Además de la gráfica para obtener la información de manera más precisa será apoyada por una tabla de datos.

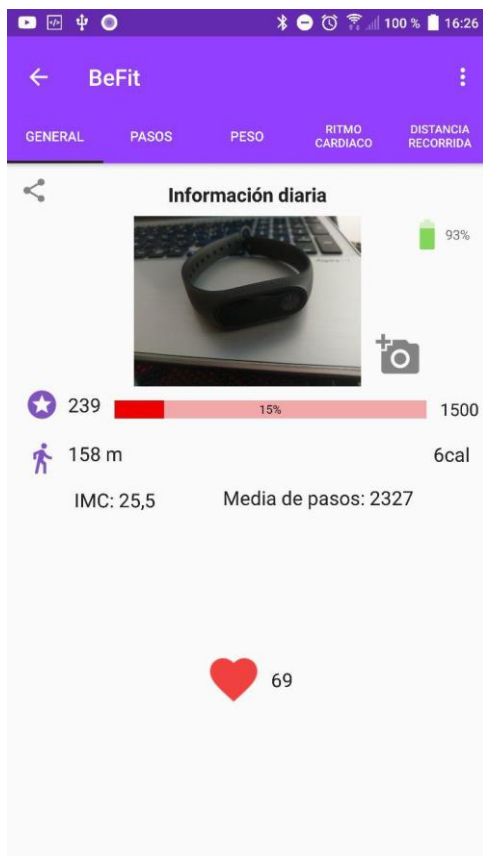


Ilustración 27. Pantalla de información general

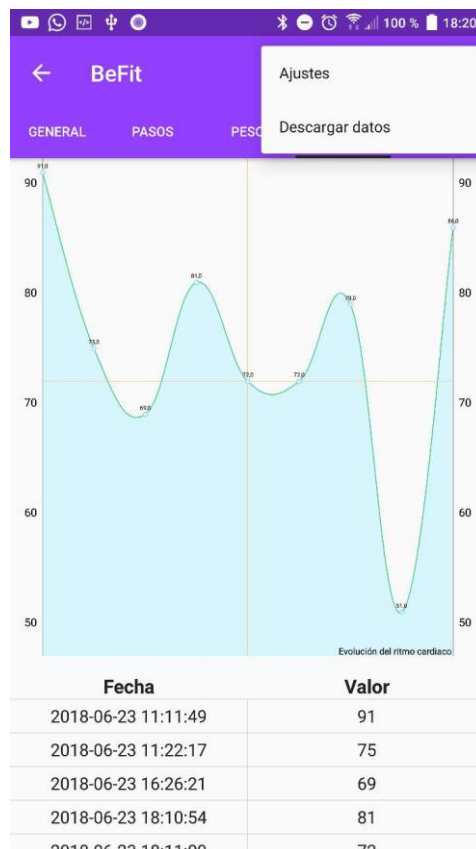


Ilustración 26. Pantalla de gráficas de datos

Además, una vez conectado al dispositivo se podrá volver a configurar la información personal (foto, meta de pasos diaria, etcétera). También podremos descargar la información de nuestras gráficas en archivos csv independientes que encontraremos en la carpeta de descargas del dispositivo. Y por último también podremos compartir nuestros datos diarios con nuestras otras aplicaciones, como vemos en las siguientes imágenes.

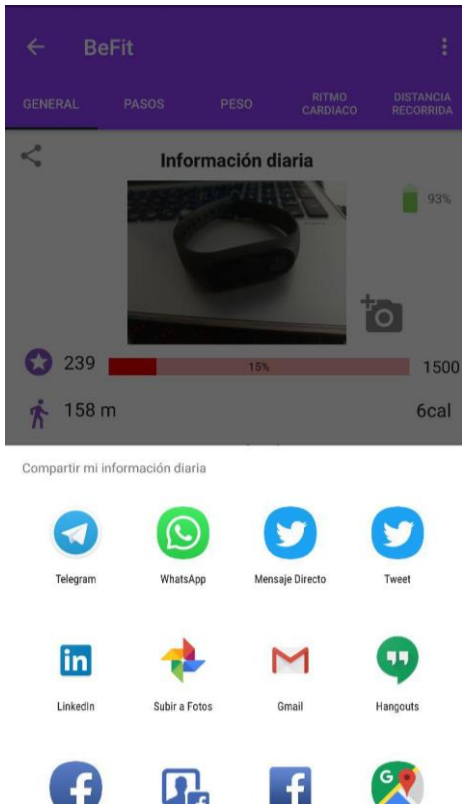


Ilustración 30.

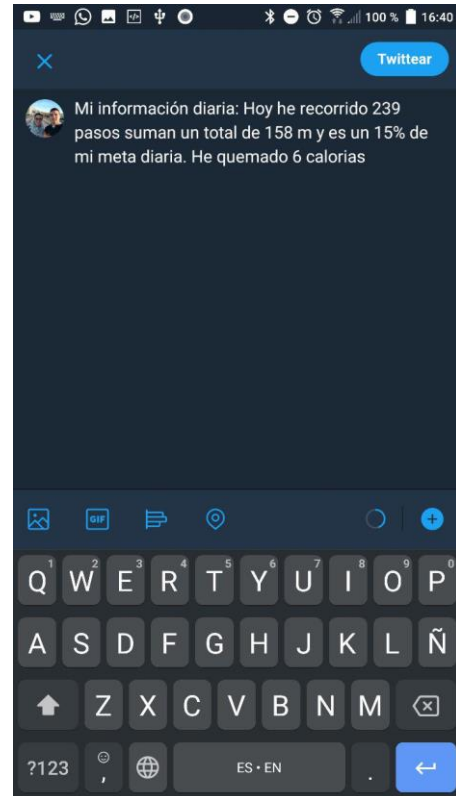


Ilustración 31. Ejemplo de compartir información con alguna aplicación

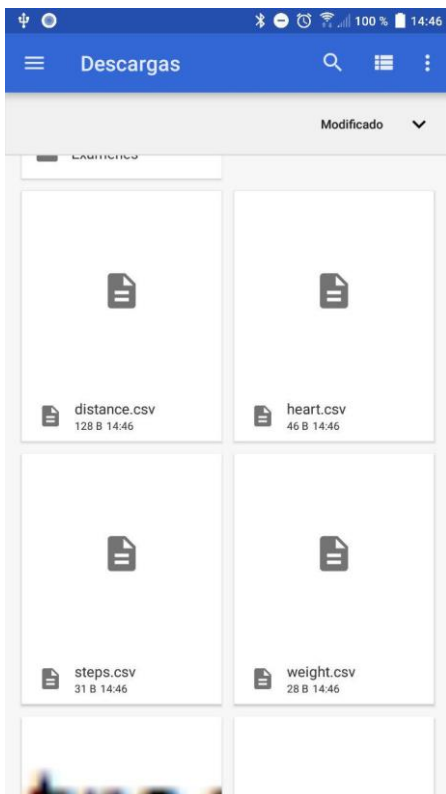


Ilustración 29. CSV Importados en la carpeta descargas

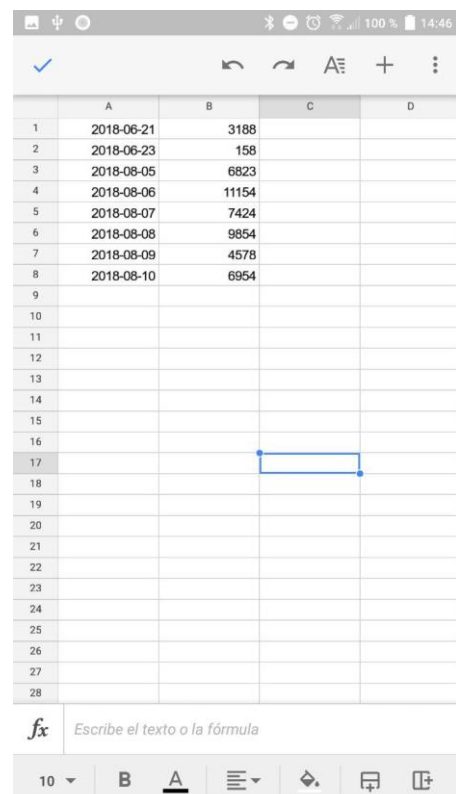


Ilustración 28. CSV en exportado en formato estándar

6.3.2. Desarrollo de la aplicación

6.3.2.1. Librerías

Empezaremos hablando de librerías externas a Android que hemos usado en el proyecto. Todas ellas nos facilitan alguna de las funcionalidades que se incorporan en la aplicación.

- Glide. Es una librería que permite la carga de imágenes tanto mediante un enlace web como una dirección de almacenamiento en el teléfono. Es de código abierto y administra la cache de los recursos del sistema para las fotos. Es simple y fácil de usar y evitamos tener que usar un bitmap para representar una imagen evitando así complejidad y ganando legibilidad en el código. Proporciona, además, también la posibilidad de mostrar gifs.
- Butterknife. Es una librería que permite rápidamente el asociamiento entre archivos gráficos y archivos de programación (binding) como un ImageView. Un ejemplo sencillo:

Para relacionar un archivo en Android debemos usar lo siguiente:

```
Button myButton = findViewById(R.id.my_button)
```

Sin embargo, gracias a esta librería podemos referenciarlo de esta manera.

```
@BindView(R.id.id.my_button) Button myButton;
```

Esto simplifica el código además de como pasaba en el caso anterior favorece a la legibilidad, haciendo un código más fácil de entender. Se pueden también referenciar, recursos como texto, o imágenes.

- MPAndroidChart. Nos permite de una forma y sencilla añadir a nuestra interfaz gráficas que son fáciles de usar y de personalizar. Hay gran cantidad de gráficas, como gráficos de barras o circulares, pero en nuestro caso hemos usado solo gráficas lineales ya que van acorde con la aplicación.

- FastCSV es una biblioteca CSV compatible con RFC 4180 ultra rápida y simple para java, en este caso Android. RFC 4180 es el formato de separación por comas que se ajusta a los archivos CSV.
- Librerías nativas. Android proporciona una serie de librerías nativas como las de diseño que facilitan el desarrollo de las aplicaciones con los últimos componentes disponibles o la de testing (Espresso) que permite la realización de pruebas con test basándose en Junit.

6.3.2.2. Vistas

Antes de entrar a más detalle vamos a ver cómo funcionan las vistas en Android. Para empezar las vistas se declaran en una parte de los recursos de la aplicación llamada layouts (así como puede haber otras como drawable para imágenes). Estos archivos como la gran mayoría usan el formato XML para su declaración, y se genera así un árbol entre padres (vistas o layouts) e hijos que son objetos como botones, campos de texto u otras vistas.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:id="@+id/root_cell"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginEnd="8dp"
  android:layout_marginStart="8dp"
  app:cardUseCompatPadding="true">

  <android.support.constraint.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
      android:id="@+id/name_device"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginEnd="8dp"
      android:layout_marginStart="8dp"
      android:layout_marginTop="8dp"
      android:text=""
      android:textSize="18sp"
      app:layout_constraintEnd_toStartOf="@+id/guideline"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
```

Ilustración 32. Ejemplo de XML de una vista

Antiguamente Android disponía de RelativeLayout, LinearLayout o GridLayouts. El problema de estas vistas era que cuanto más complejidad tenía una vista más grande se hacía el árbol. Android a la hora de pintar

una interfaz lo realiza por partes pequeñas del árbol, pintando de vista en vista, entonces esto supone una sobrecarga de rendimiento. Si anidamos en una vista veinte dentro de ella el rendimiento a la hora de pintarse la vista será menor.

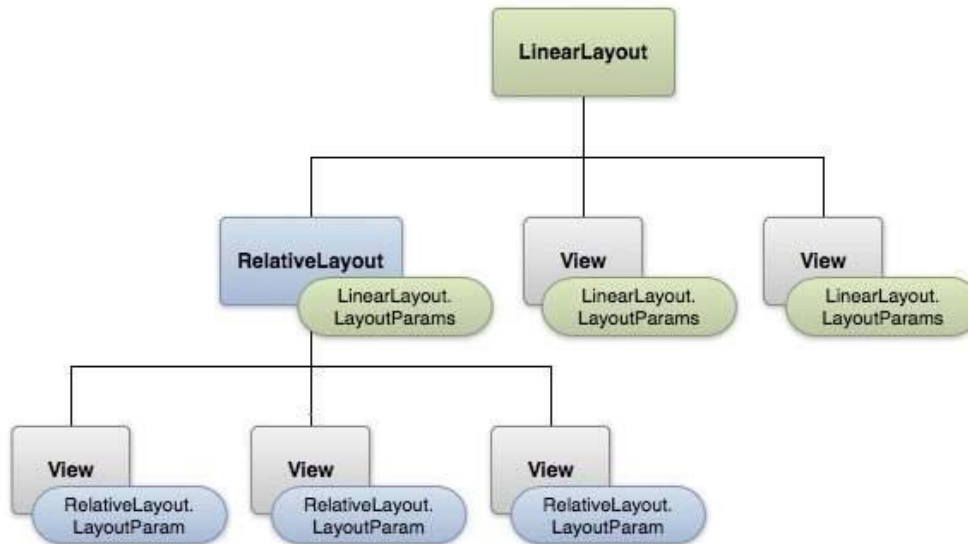


Ilustración 33. Árbol de vistas en Android. Fuente: Tutorialspoint

La solución a esto se llama ConstraintLayout. Busca la utilización de la menor cantidad de layouts anidados dentro de una vista. Usa las restricciones para establecer en qué lugar estará cada elemento. La ventaja de esto es, como ya hemos hablado, la eliminación de sobrecarga de una vista.

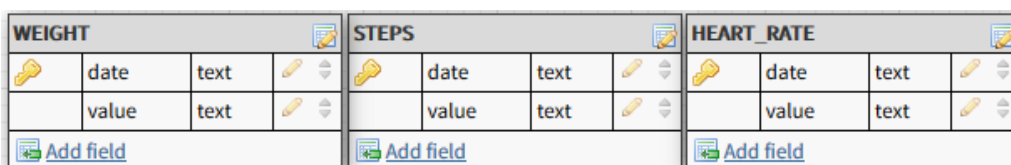
Otra parte que se ha usado gracias a simplicidad y mejora de rendimiento se llama RecyclerView. Antes de esto las listas de elementos se hacían con un ListView, pero esto tenía un gran problema: Cargaba todos los elementos de la lista, aunque no estuviesen en pantalla. Esto conlleva a una sobrecarga innecesaria porque a veces no nos hace falta ver toda la lista. En nuestro caso las listas de elementos serán pequeñas probablemente no tengamos que hacer desplazar la vista para buscar los dispositivos. Pero, gracias a este elemento, se encarga de solo cargar aquellos elementos que estén en pantalla. Una vez se desplace la vista hacia abajo reciclará el elemento y pintará el nuevo.

6.3.2.3. Almacenamiento

El almacenamiento de la información se lleva a cabo en dos lugares dentro de la aplicación. La información pequeña y sencilla se lleva a cabo en lo que en Android se denomina “Shared Preferences”. Esto permite almacenar información clave-valor. Por ejemplo, en nuestro caso almacenamos el nombre con una clave que se llama “name” y el valor que introduzca el usuario en la configuración.

Por otro lado, ya un poco más complejo se encuentra una base de datos SQLite que se encarga de almacenar toda la información con respecto a las lecturas de datos, de los pasos, ritmo cardiaco o peso. SQLite es sistema de gestión de bases de datos relaciones que permite el almacenamiento de datos en dispositivos empotrados. Es de código abiertos y utiliza el lenguaje de SQL lo que facilita su uso.

Para el caso del peso, y de los pasos son datos diarios y en la base de datos se almacenarán de manera que solo se pueda insertar un dato por día. Una persona solo tiene un número de pasos diario y el peso tampoco varía demasiado diariamente. Sin embargo, para el caso del ritmo cardiaco, se puede insertar un dato por fecha completa (con horas, minutos y segundos), no como en el caso anterior. Esto es debido porque el pulso está en constante cambio y nos permitirá monitorizarlo de una manera más precisa.



| WEIGHT | | | |
|---------------------------|-------|------|------|
| key | date | text | edit |
| | value | text | edit |
| Add field | | | |

| STEPS | | | |
|---------------------------|-------|------|------|
| key | date | text | edit |
| | value | text | edit |
| Add field | | | |

| HEART_RATE | | | |
|---------------------------|-------|------|------|
| key | date | text | edit |
| | value | text | edit |
| Add field | | | |

Ilustración 34. Ejemplos de tablas que integra la base de datos de la aplicación

6.3.2.4. Conexión con dispositivos

Para llevar a cabo una conexión primero se lleva a cabo la búsqueda de los dispositivos que haya cercanos. Para realizar la búsqueda de los dispositivos de bajo consumo se usa un escáner que proporciona el adaptador de bluetooth del dispositivo.

Con este podremos empezar y parar el escaneo de dispositivos. Nos permitirá establecer una función para cada vez que un dispositivo sea detectado. En ese momento nosotros añadiremos dicho dispositivo a la lista de dispositivos que tenemos a nuestro alrededor.

Una vez se tenga el dispositivo cercano se usará el GATT para conectarse con el dispositivo. Una vez conectado pediremos a los servicios que dispone el dispositivo que nos facilite la información. Una vez recibida la información se encargará de mostrarla por pantalla y actualizar los datos en la base de datos.

6.3.2.5. Permisos

Algo esencial hoy en día es la seguridad. Para ello dentro de Android nos encontramos los permisos del sistema. Éstos son los encargados de facilitar el acceso a los recursos dentro del sistema como la cámara o el bluetooth en nuestro caso.

En Android Lollipop los permisos se pedían todos al principio al instalar las aplicaciones. A partir de Android Marshmallow los permisos se deben pedir en tiempo real y sólo y exclusivamente cuando sean necesarios. Por ejemplo, dentro de nuestra aplicación se requieren también los permisos de almacenamiento para guardar la imagen de perfil del usuario. Para ello pediremos permiso cuando tengamos la foto y solo tengamos que almacenarla en el dispositivo. Mientras tanto no será necesario pedir los permisos.

6.4. Diagramas de secuencia

Para la documentación correcta del funcionamiento de la aplicación también se aportan diagramas de secuencia con las principales funcionalidades de la aplicación. Con el podremos ver la interacción entre las distintas partes de la aplicación del sistema y como se comunican entre ellas a alto nivel. Los primeros diagramas son los que no es necesario establecer la conexión entre los dispositivos. El más importante es el de sincronización de los datos que incluye el escaneo de dispositivos.

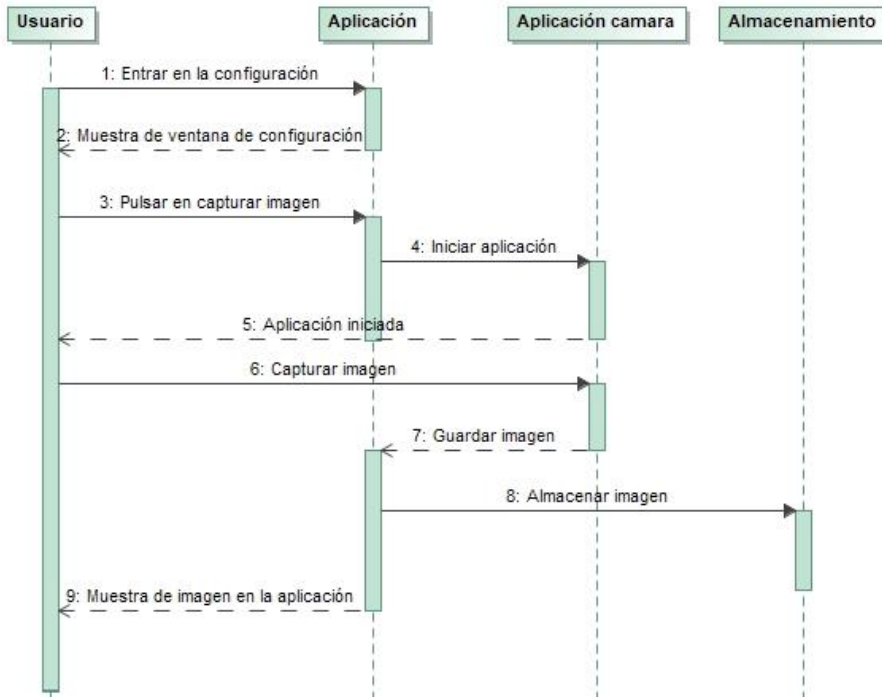


Ilustración 35. Diagrama de secuencia para la captura de imágenes de perfil en la aplicación

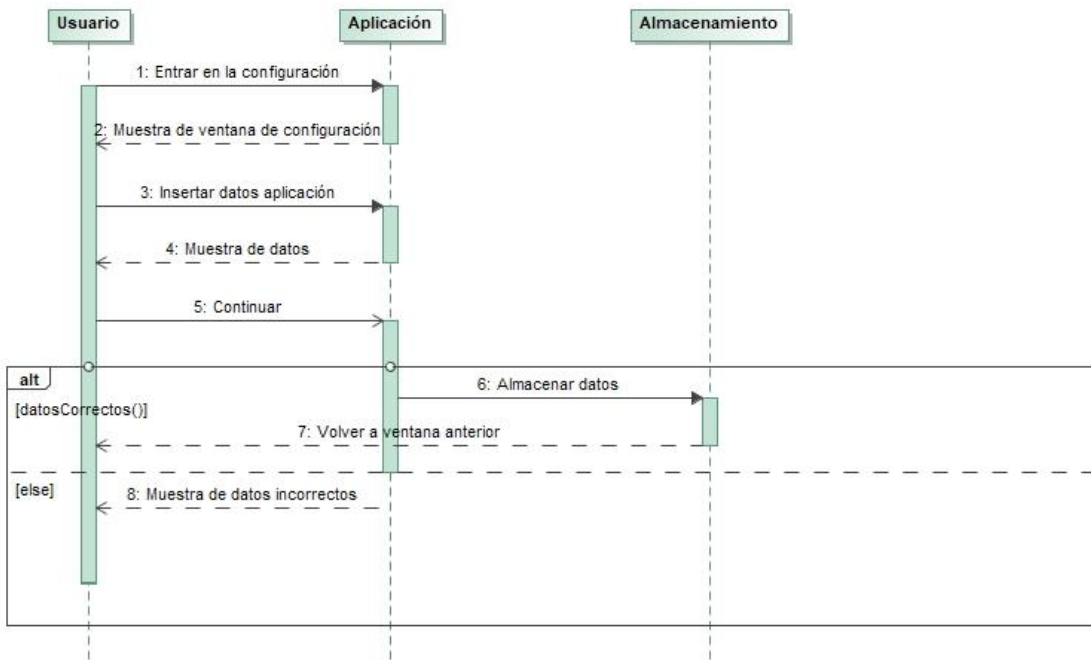


Ilustración 36. Diagrama de secuencia del almacenamiento de los datos personales dentro de la aplicación

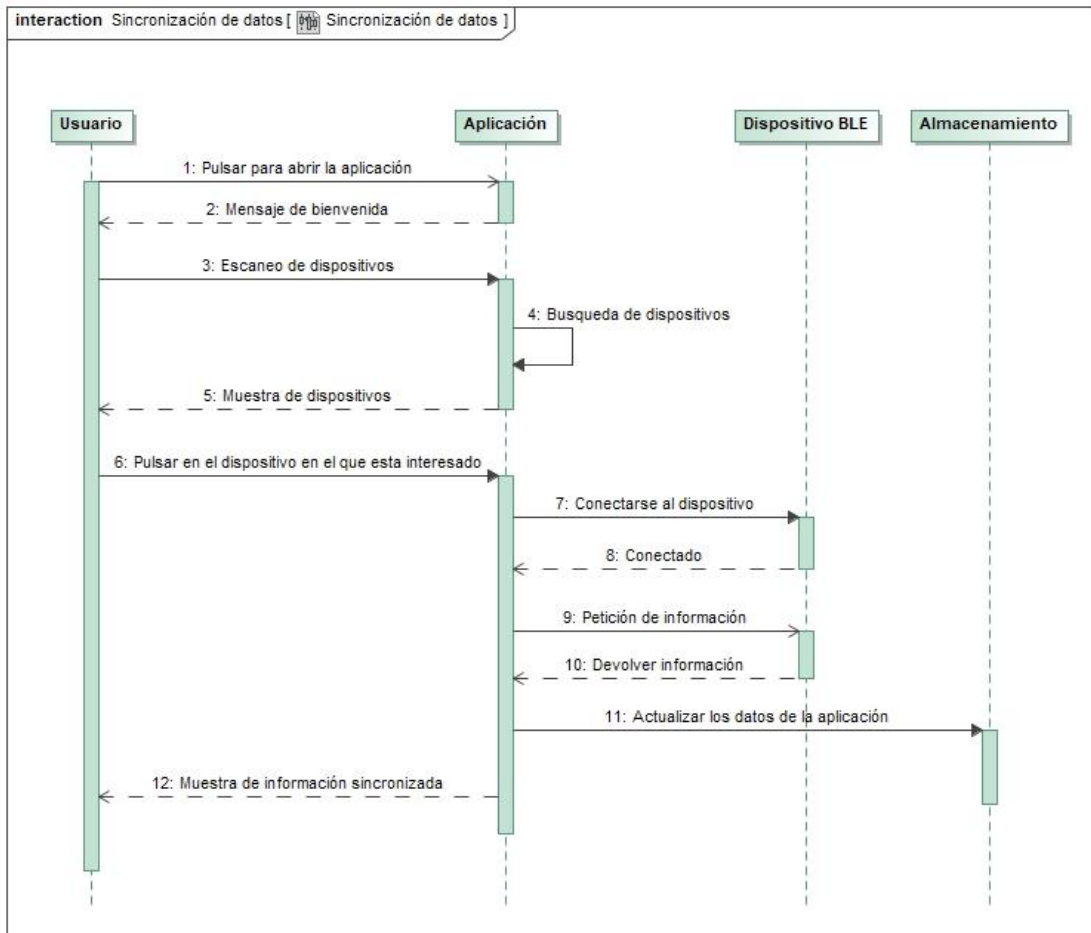


Ilustración 37. Diagrama de secuencia sobre la sincronización de los datos de la aplicación

A partir de estos diagramas, partimos de que los datos han sido sincronizados.

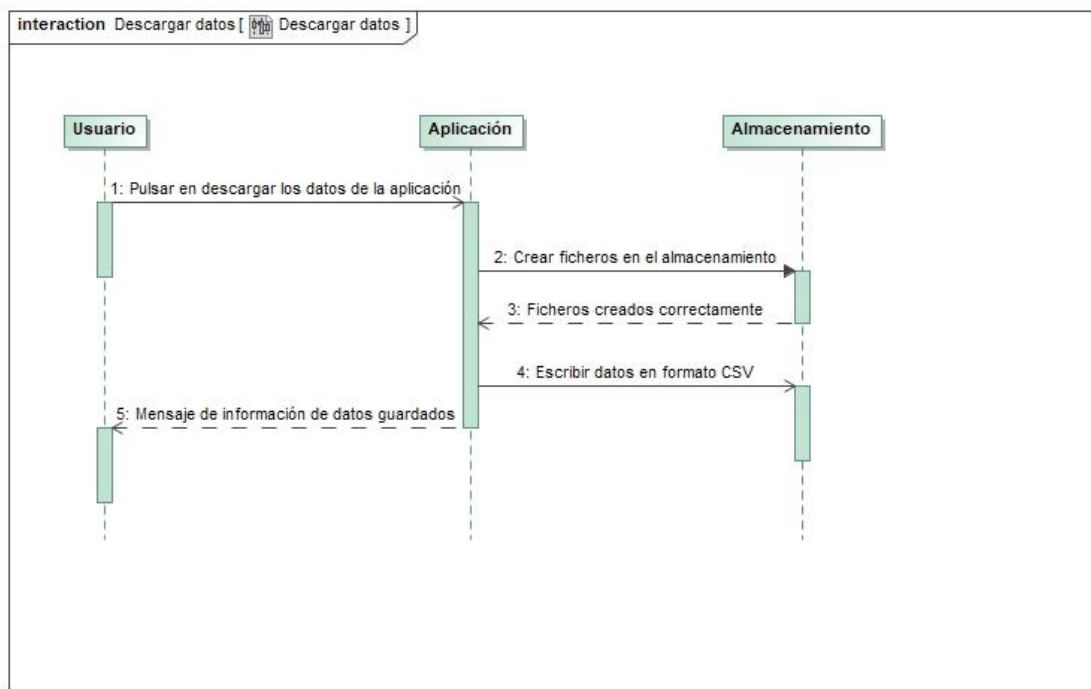


Ilustración 38. Diagrama de secuencia de la funcionalidad de descargar los datos

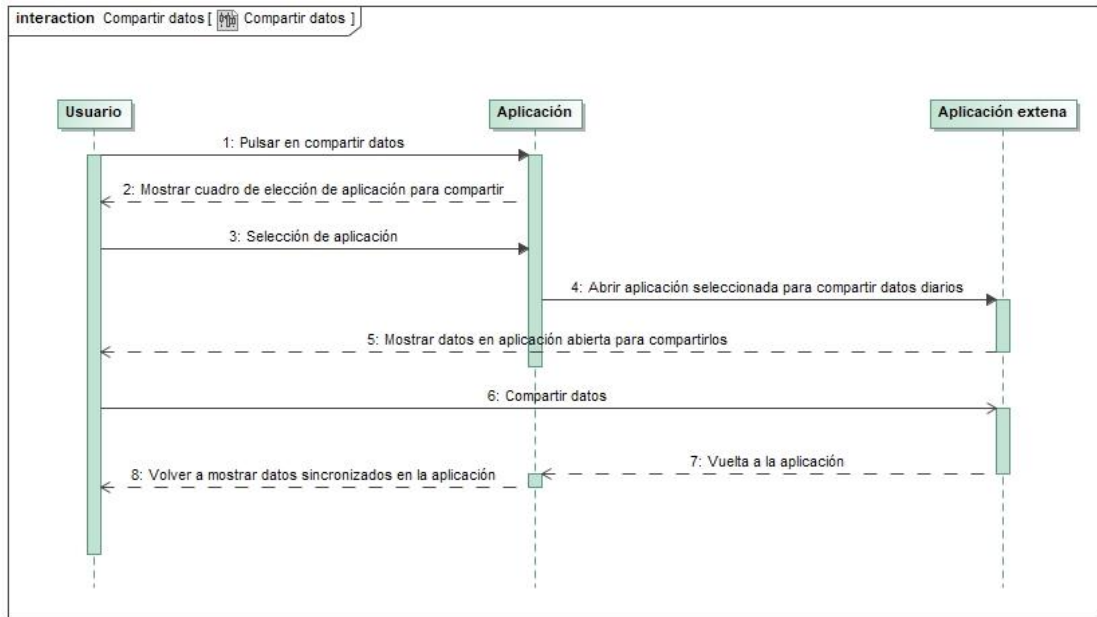


Ilustración 39. Diagrama de secuencia para la funcionalidad de compartir los datos sincronizados diarios con una aplicación externa

Mantenimiento y pruebas

7.1. Mantenimiento

Para el desarrollo de la aplicación se ha tenido siempre en cuenta el mantenimiento del software. Para ello se ha documentado de varias maneras, otorgando una serie de requisitos, donde podemos encontrar las funcionalidades que abarca la aplicación. Se han generado también una serie de diagramas que son vitales a la hora de entender el funcionamiento de la aplicación.

Por otro lado, a la hora del desarrollo software se ha intentado utilizar el código más limpio posible, intentando evitar la duplicación innecesaria de código. Aquí podemos tener, por ejemplo, clases auxiliares con constantes u operaciones que se usan siempre en diferentes pantallas facilitando así su reutilización.

Otra parte importante a la hora del mantenimiento es tener el proyecto en un repositorio como es GitHub. Si alguien decide usar el proyecto y ve que hay algún problema nos podrían notificar evitando así posteriores errores.

El mantenimiento es necesario para que otros desarrolladores puedan reutilizar el código si hay que modificarlo o corregir errores. Debería ser obligatorio siempre complementar el código con diagramas para facilitar su comprensión.

Como plan de mantenimiento se tiene las siguientes variantes:

- Comprobar que los dispositivos con sus posteriores versiones son compatibles. Hay que tener cuidado con algunas versiones más recientes que pierden la retrocompatibilidad de lo anterior; por ello, habrá que estar atento a las actualizaciones de los dispositivos.
- Mejora de rendimiento en general del sistema. Un sistema nunca es perfecto, siempre es mejorable, entonces aquí se tendría en cuenta la mejora de la velocidad y eficiencia de la aplicación.
- Corrección de problemas futuros, en el software siempre hay problemas porque como ya se ha dicho antes, nada es perfecto. Entonces, si hubiese

algún problema de diseño o implementación se debería avisar al desarrollador para corregirlo lo antes posible.

7.2. Pruebas de la aplicación

Para empezar, se han realizado pruebas a la hora del desarrollo. Se ha generado además una comprobación de errores si ocurriese algún problema dentro de la aplicación para estar siempre informado de cualquier problema. Además, se debería generar para mayor seguridad una batería de pruebas para asegurar las funcionalidades del sistema. Pero muchas de éstas dependen de la interfaz gráfica.

Por esto hemos recurrido al uso de la librería de Espresso la que nos permite hacer una simulación de uso de la interfaz. Contiene una pequeña API con las funciones básicas de las interfaces de usuario, como pulsar, insertar un texto. Además, permite la automatización de las pruebas dentro de la aplicación

```
@Test
public void testIcheckStepsFieldIsWritten() {
    onView(withId(R.id.steps_daily_user)).perform(typeText(mSteps));
    closeSoftKeyboard();
    onView(withId(R.id.steps_daily_user)).check(matches(withText(mSteps)));
}
```

Ilustración 40. Test con Espresso para la prueba de interfaces

Además, para complementar su uso hemos usado JUnit para integrar las pruebas de interfaces de usuario con pruebas de unidad. Así podemos comprobar que los campos introducidos son correctos.

```
@Test
public void checkStepsIsInsert() {
    Date cDate = new Date();
    String fDate = new SimpleDateFormat( pattern: "yyyy-MM-dd").format(cDate);
    databaseOperations.insertWeight(fDate, mSteps);
    assertTrue(databaseOperations.getWeights().containsKey(fDate));
    assertEquals( message: "Weight inserted", Float.parseFloat(mSteps), databaseOperations.getWeights().get(fDate));
}
```

Ilustración 41. Test JUnit para la comprobación de que se inserta un dato correctamente

Las pruebas que se han llevado a cabo han sido en el apartado de la base de datos para comprobar la inserción de datos y su consulta correctamente. Por otro lado, las pruebas en la parte de la interfaz gráfica han tenido que ver completamente con el apartado de configuración de los datos de usuario para

comprobar que se están almacenando y que los campos se rellenan correctamente.

```
@Test
public void testFieldSavesInSharedPreferences() {
    onView(withId(R.id.name_user)).perform(typeText(mName));
    closeSoftKeyboard();
    onView(withId(R.id.steps_daily_user)).perform(typeText(mSteps));
    closeSoftKeyboard();
    onView(withId(R.id.weight_user)).perform(typeText(mWeight));
    closeSoftKeyboard();
    onView(withId(R.id.age_user)).perform(typeText(mAge));
    closeSoftKeyboard();
    onView(withId(R.id.height_user)).perform(typeText(mHeight));
    closeSoftKeyboard();
    onView(withId(R.id.button_next)).perform(click());
    assertEquals( message: "Error name not equals", mName, sharedPreferences.getString(Constants.NAME, s1: ""));
    assertEquals( message: "Error steps not equals", mSteps, sharedPreferences.getString(Constants.STEPS_GOAL, s1: ""));
    assertEquals( message: "Error age not equals", mAge, sharedPreferences.getString(Constants.AGE, s1: ""));
    assertEquals( message: "Error height not equals", mHeight, sharedPreferences.getString(Constants.HEIGHT, s1: ""));
}
```

Ilustración 42. Test mezclado entre pruebas con la interfaz y JUnit

Las ventajas de utilizar las pruebas es que nos permiten detectar errores a tiempo. Por otro lado, también nos genera un código limpio y de calidad al utilizar las pruebas durante todo el proceso de desarrollo software. Además, facilitan el flujo de información que se va a seguir en la aplicación pudiendo detectar a tiempo problemas de diseño o rendimiento.

Conclusiones y líneas futuras

Aquí encontraremos un resumen con los problemas que se han encontrado en el desarrollo de la aplicación, así como objetivos que se han cumplido. Además, encontraremos también apartados de posibles mejoras futuras.

8.1. Dificultades encontradas

Se han encontrado numerosas dificultades a la hora de conocer la información que se comparte en los dispositivos, debido a que no hay muchos fabricantes que quieran facilitar dicha información. Entonces no es posible conseguir la máxima información posible de los dispositivos.

Nos encontrábamos por una parte el problema de entender el funcionamiento del protocolo de Bluetooth Low Energy y, posteriormente, ver su funcionamiento dentro de un dispositivo normal inteligente como son los móviles actuales.

Por otro lado, la dificultad de sincronización entre los dispositivos no es ideal y, por tanto, no es inmediata. Entonces se ha tenido que diseñar un sistema capaz de conseguir toda la información de manera que sea capaz de reaccionar a fallos y casos en los que se pierda la conexión o no sea posible extraerla de los dispositivos.

Se han quedado además atrás algunos dispositivos que debían integrarse, pero cada dispositivo requiere un conocimiento exhaustivo de la transmisión de información en cada caso. Hasta el punto de conocer byte a byte como es la transmisión de esta información y como se ha dicho anteriormente los fabricantes no suelen dar estas facilidades. Para superar esta desventaja se tendrá que trabajar en un futuro con las distintas empresas fabricantes de los dispositivos y lograr obtener la mayor cantidad de información sobre éstos. Aunque podría ser una tarea dificultosa lograr obtener la información sobre el funcionamiento de los servicios de cada dispositivo. Esto en general es debido a que las empresas fabrican su propio software que facilitan con el mismo hardware.

8.2. Objetivos superados

Se ha logrado integrar una aplicación sencilla, y bastante liviana que se pueda usar en cualquier dispositivo que hoy en día haya en el mercado (cabe destacar que el mercado en Android ya sólo distribuye al mercado versiones de Android 7.0 hacia adelante).

Además, se ha logrado el uso de una base de datos embebida para el tratamiento de los datos. Para su posterior análisis se han usado gráficas para entender y ver visualmente la evolución de éstos.

También se ha llevado a cabo una investigación del funcionamiento y la historia de sistema operativo Android y del protocolo de Bluetooth Low Energy.

Se han realizado pruebas de interfaces de usuario para corroborar el funcionamiento correcto de la aplicación.

8.3. Líneas futuras

Por último, se han elaborado una serie de líneas futuras que pueden ser factibles de aplicar y que son potencialmente aplicables dentro del proyecto.

- Como ya se ha mencionado anteriormente la inclusión de más dispositivos. El objetivo principal es recopilar la información de muchos dispositivos móviles para facilitar el uso de las personas y que no tengan que usar varias aplicaciones, aprovechando así el ahorro de almacenamiento de memoria dentro del dispositivo.
- Mejora en las interfaces de usuario, habría que prestar atención a lo que demande el mercado y actualizarse con las nuevas opciones que nos permita el sistema. Así como actualizar la información de las pantallas o su restructuración para aumentar la legibilidad de la información.
- Ampliación de la funcionalidad. Ahora mismo la funcionalidad se limita a los dispositivos que disponemos, pero si lográsemos tener más dispositivos podríamos obtener información combinada entre ellos consiguiendo todavía más precisión en los datos. Obtener datos como por

ejemplo el número de pasos en una sesión cuando vayamos a correr o el número de kilómetros recorridos.

- Uso de una API como Google Fit para sincronizar los datos, en vez de usar una base de datos local para ello. Esto permitiría tenerlo en múltiples dispositivos.
- Integración con redes sociales. Hoy en día todos estamos conectados con nuestros dispositivos. Una opción sería el poder compartir los datos con distintas personas para ir viendo su evolución.

En definitiva, con estos aspectos mostramos que el proyecto tiene potencial de crecimiento en un futuro, evitando las actuales restricciones temporales.

Bibliografía:

Todos los enlaces han sido recuperados de páginas webs en el año 2018, que ha sido el año de realización de este proyecto:

<https://punchthrough.com/bean/docs/guides/everything-else/how-gap-and-gatt-work/>

http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_40_00_45/docs/blestack/ble_user_guide/html/ble-stack-3.x/gatt.html

<https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>

[https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))

<http://gs.statcounter.com/os-market-share/mobile/worldwide>

<https://developer.android.com/guide/platform/?hl=es-419>

<https://arstechnica.com/gadgets/2016/10/building-android-a-40000-word-history-of-googles-mobile-os/11/>

<https://source.android.com/devices/architecture/treble>

<https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido?lang=es>

<https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction>

<https://www.bluetooth.com/specifications/gatt/generic-attributes-overview>

https://en.wikipedia.org/wiki/Generic_access_profile

<https://developer.android.com/guide/components/activities/activity-lifecycle>

<https://developer.android.com/guide/components/fundamentals?hl=es-419>

<https://learn.sparkfun.com/tutorials/bluetooth-basics>

http://www.noalaobesidad.df.gob.mx/index.php?option=com_content&view=article&id=52&Itemid

<https://developer.android.com/guide/components/intents-filters?hl=es-419>

<https://developer.android.com/guide/components/fragments?hl=es-419>

<https://developer.android.com/training/animation/screen-slide>

<https://developer.android.com/guide/topics/ui/menus>

<https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419>

<https://developer.android.com/training/constraint-layout/?hl=es-419>

<https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=es-419>

<https://developer.android.com/training/transitions/custom-transitions?hl=es-419>

<https://developer.android.com/training/data-storage/sqlite>

<https://developer.android.com/training/data-storage/shared-preferences>

Recursos utilizados:

<https://www.htc.com/es/smartphones/htc-u-ultra/>

<https://www.htc.com/in/smartphones/htc-desire-820/>

<https://www.mi.com/es/miband2/>

https://en.wikipedia.org/wiki/Android_Studio

<https://developer.android.com/studio/>

<https://www.nomagic.com/products/magicdraw>

<https://www.thisisbeast.com/en>