Aerlink: conexión Bluetooth entre iOS y Wear OS

Aerlink: Bluetooth connection between iOS and Wear OS

Guillermo Cique Fernández

GRADO EN INGENIERÍA DEL SOFTWARE. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo de fin de grado del Grado en Ingeniería del Software

Curso 2019-2020

Director:

Jorge Jesús Gómez Sanz

Resumen

Un reloj inteligente o *smartwatch* es uno de los dispositivos tecnológicos más personales que podemos poseer, siempre está con nosotros, actuando como una pequeña ventana a nuestro mundo digital y, en muchos casos, monitorizando diversos aspectos de nuestra salud.

Para usuarios de iOS, la opción más obvia puede ser el Apple Watch, pero hay un mundo entero de dispositivos con el sistema operativo Wear OS que pueden convertirse en una opción más económica o menos restrictiva para estos usuarios.

Este proyecto relata la creación de Aerlink, un sistema compuesto por dos aplicaciones que, trabajando juntas, consiguen conectar gracias a la tecnología Bluetooth Low Energy un dispositivo iOS y un dispositivo Wear OS, ofreciendo funcionalidades no disponibles previamente.

Palabras clave

Bluetooth Low Energy
iOS
Android Wear
Wear OS
Reloj inteligente
Wearable
Conexión inalámbrica
Desarrollo móvil

Abstract

A smartwatch is one of the most personal devices we can own, it is always with us, acting as a small window into our digital world and, in many cases, monitoring diverse aspects of our health.

For iOS users, the obvious choice may be the Apple Watch, but there is a whole world of devices with similar functionalities running the operating system Wear OS that can become a more economic or less restrictive option for these users.

This project narrates the creation of Aerlink, a system composed by two applications that, working together, manage to connect through Bluetooth Low Energy an iOS device and a Wear OS device, offering functionalities previously not available.

Keywords

Bluetooth Low Energy
iOS
Android Wear
Wear OS
Smartwatch
Wearable
Wireless connection
Mobile development

Dedicatoria

A mi padre y a mi hermana.

A mi novia.

A los chicos del DAM.

Al director de este TFG.

Índice general

Índice de figuras

Capítulo 1

Introducción

> We love to make products that
> really enrich people's lives.
> We love to make technology
> more personal.
>
> *Tim Cook*

1.1. Motivación

Cada día se ven más y más *smartwatches* por la calle, son dispositivos de funcionalidad reducida pero que siempre están ahí cuando los necesitas. Mucho se ha puesto en duda el verdadero valor de este tipo de dispositivo, no obstante, nos guste más o nos guste menos, el mercado ha decido que están aquí para quedarse.

A pesar de la gran cantidad de dispositivos distintos en venta, estos pueden venir con importantes limitaciones, especialmente para usuarios con dispositivos iOS. Usuarios con un iPhone tienen la opción clara de optar por un Apple Watch, pero este tipo de dispositivos son exclusivamente compatibles con los iPhone, ni siquiera funcionando con un iPad. Esta limitación puede ser pasada por alto por un gran número de usuarios, pero hay usuarios que poseen un dispositivo iOS y un Android, hay usuarios que se plantean cambiar su iPhone por un dispositivo Android en un futuro, hay usuarios que nunca han probado un *smartwatch*... todo esto compaginado con su elevado precio descartan al Apple Watch como opción de muchos usuarios.

La otra opción obvia para estos usuarios sería un dispositivo Wear OS, pero este tipo de dispositivos no vienen libres de limitaciones. Apple se aprovecha de su control sobre la plataforma iOS para poner trabas a dispositivos de la competencia, por otro lado, Google parece más interesada en conseguir usuarios para sus servicios que en intentar ofrecer la mejor experiencia. Aún así, mucha gente acabará optando por este tipo de dispositivos, ya que ofrecen mucha más variedad tanto estéticamente como económicamente.

1.2. Objetivos

Este proyecto relata el desarrollo de **Aerlink**, un sistema creado en 2015 y que ya supera las 180.000 descargas en Android y 50.000 en iOS. Aerlink se posiciona como opción alternativa, independiente de Google y de Apple, para usuarios con un dispositivo Wear OS y un dispositivo iOS.

Este sistema permite a dispositivos Wear OS y dispositivos iOS interactuar entre sí ofreciendo funcionalidades no disponibles de ninguna otra manera, además de seguir ofreciendo las funcionalidades básicas esperadas de un *smartwatch*:

- Gestión de las notificaciones del dispositivo iOS.
- Control de los medios multimedia en el dispositivo iOS.
- Gestión de los datos de la app oficial de Recordatorios del dispositivo iOS.
- Control remoto de la cámara del dispositivo iOS.

Por último, estos objetivos se logran utilizando únicamente la tecnología Bluetooth Low Energy, al ser esta la única opción disponible en los dispositivos iOS debido al fuerte control que ejerce Apple sobre los accesorios Bluetooth Classic que se pueden conectar con los dispositivos iOS.

1.3. Plan de trabajo

Para alcanzar los objetivos descritos en la sección anterior se ha planteado:

- Crear una aplicación iOS que haga al dispositivo descubrible con Bluetooth Low Energy.
- Implementar los servicios de Apple en la aplicación Wear OS.
- Crear los nuevos servicios de Aerlink en la aplicación iOS.
- Implementar estos nuevos servicios en la aplicación Wear OS.
- Trabajar en las interfaces gráficas de ambas aplicaciones.

1.4. Estructura del documento

El documento se compone de 6 capítulos:

- En el capítulo 2 se revisa el contexto comercial de los *smartwatches*, las alternativas disponibles en el mercado y la problemática de la conexión entre dispositivos iOS y Wear OS.
- El capítulo 3 define de forma precisa y comprensible todas las funcionalidades y restricciones del sistema que se desea construir.
- El capítulo 4 describe el diseño del sistema, detallando la funcionalidad de sus componentes, y repasa los servicios implementados por este.
- En el capítulo 5 se describen las pruebas realizadas al sistema ya implementado y en funcionamiento.
- En el capítulo 6 se recogen las conclusiones obtenidas tras la finalización del proyecto y líneas futuras a seguir para continuar con el proyecto.

Capítulo 1

Introduction

> We love to make products that
> really enrich people's lives.
> We love to make technology
> more personal.
>
> *Tim Cook*

1.1. Motivation

Each day we see more and more smartwatches on the street, these are devices of reduced functionality but they are always there when you need them. A lot has been talked about the real value of this type of device, nonetheless, like it or not, the market has decided that they are here to stay.

Despite all the different devices on sale, these can come with important limitations, specially for users with iOS devices. iPhone users have the clear option of choosing an Apple Watch, but these type of devices are exclusively compatible with iPhones, not even working with an iPad. This limitation may be looked over by a high number of users, but there are users with an iOS device and an Android device, there are users who may be thinking in switching over to Android in the future, there are users who have never tried a smartwatch... all of this combined with its elevated price discard the Apple Watch as an option for many users.

The other obvious option for these users would be a Wear OS device, but these devices

do not come free of limitations. Apple takes advantage of its control over the iOS platform to make it harder for competing devices to offer the same functionality, on the other hand, Google seems more interested in increasing the number of users for its platforms than offering the best user experience. Nonetheless, lots of people will end up opting for these type of devices, since they offer much more variety both aesthetically and economically.

1.2. Goals

This project narrates the development of **Aerlink**, a system created back in 2015 and which has already surpassed 180,000 downloads in Android and 50,000 in iOS. Aerlink positions itself as the alternative, independent of Google and Apple, for users with a Wear OS device and an iOS device.

This system allows Wear OS devices and iOS devices to interact with each other offering functionalities not available in any other way, in addition to offering the expected basic functionality of any smartwatch:

- Management of iOS device's notifications.
- Remote control of iOS device's media.
- Remote control for the official iOS app Reminders.
- Remote control for the camera in the iOS device.

Finally, these goals will be achieved using only Bluetooth Low Energy, being this the only available option in iOS thanks to the strict control Apple has over which accessories are allowed to establish a Bluetooth Classic connection with iOS.

1.3. Work plan

To achieve the goals described in the previous section, the following plan will be followed:

- Create an iOS application that allows the device to be discoverable through Bluetooth Low Energy.
- Implement Apple services in the Wear OS application.
- Create the new Aerlink services in the iOS application.
- Implement these new services in the Wear OS application.
- Work on the graphical user interface for both applications.

1.4. Document structure

The document is composed of 6 chapters:

- Chapter 2 reviews the commercial context of smartwatches, the available alternatives in the market and the difficulties of establishing a connection between iOS and Wear OS devices.
- Chapter 3 precisely and comprehensibly defines every functionality and restriction of the system to be built.
- Chapter 4 describes the system's design, detailing the functionality of its components, and reviews the services implemented by it.
- Chapter 5 describes the tests performed on the system already implemented and running.
- Chapter 6 gathers the conclusions arrived at after the completion of the project and future lines to continue work on the project.

Capítulo 2

Estado del Arte

> Un artista copia, un gran artista roba.
>
> *Guillermo Cique*

En este capítulo se revisa el contexto comercial de los *smartwatches*, las alternativas disponibles en el mercado y la problemática de la conexión entre dispositivos iOS y Wear OS.

2.1. Mercado de *smartwatches*

El mercado de *smartwatches* está en su mejor momento con algunas estimaciones (Fig. 2.1) poniendo el **número de ventas en 2020 por encima de los 80 millones**[1]. Situándose en cabeza, el Apple Watch con un market share de más del 50 %[2] muestra el gran interés en este tipo de dispositivos de los usuarios de las plataformas de Apple.

Por debajo del Apple Watch el mercado está bastante fragmentado, hay muchas empresas que intentan usar sistemas operativos propietarios como Samsung, Garmin o Fitbit y a parte tenemos Wear OS con el que Google intenta replicar el éxito que tuvo en el mercado móvil con su sistema operativo Android.

En noviembre de 2019, Google compró uno de sus mayores competidores, Fitbit, por 2.100 millones de dólares con intenciones de seguir invirtiendo en este mercado y con vistas de, en un futuro, acabar presentando relojes inteligentes "hechos por Google"[3].

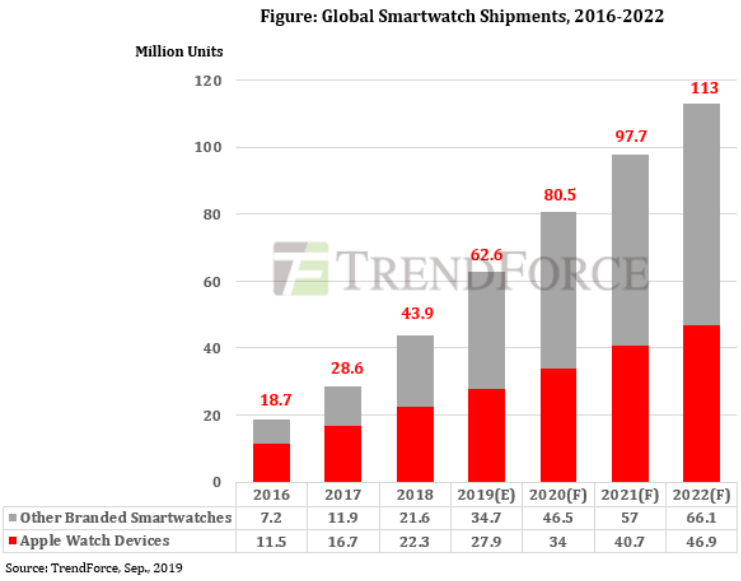

Figura 2.1: *Proyección del mercado de smartwatches*

2.2. Wear OS by Google

El soporte oficial de iOS para relojes Android Wear fue presentado con la app **Wear OS by Google**[4] a principios de septiembre de 2015. Esta aplicación permite vincular un reloj Wear OS con un dispositivo iOS ofreciendo las siguientes funcionalidades (Fig. 2.2):

- Asistente de Google
- Control de la actividad física
- Control de la música
- Notificaciones
- Responder correos de Gmail

Se trata de la única alternativa disponible para poder conectar un reloj Wear OS con un dispositivo iOS, esta conexión requiere enlazar un reloj de cero con un dispositivo iOS haciendo imposible conectar un reloj enlazado con un Android con un dispositivo iOS manteniendo una conexión con ambos.

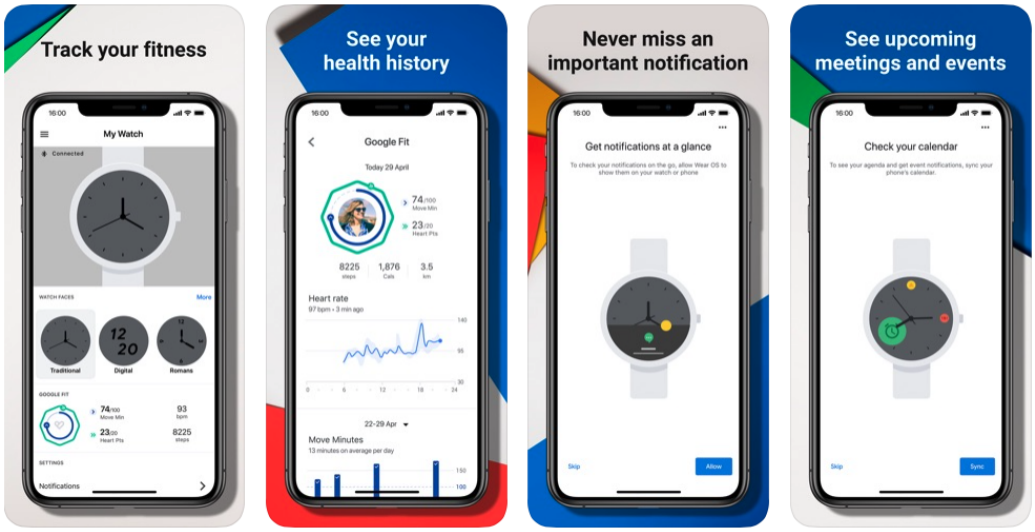

Figura 2.2: *Capturas de pantalla de Wear OS by Google*

La mayor desventaja de está aplicación es su fuerte dependencia de los servicios de Google, todas las funcionalidades extra que ofrece, como por ejemplo el control de la actividad física, requieren que el usuario utilice un servicio de Google para hacer uso de ellas. La mayoría de usuarios de Apple preferiría dar uso a los servicios ofrecidos directamente en su dispositivo y muchos otros simplemente no estarían dispuestos a usar ciertos servicios de Google.

El sistema Aerlink vio la luz por primera vez en 2015, meses antes del lanzamiento de Wear OS by Google para iOS, siendo la aplicación para Wear OS gratuita en el Play Store de Android y teniendo su contra-parte en iOS un precio de 1,99$ en el App Store de iOS. A pesar de ser Wear OS by Google una opción gratuita y oficial, Aerlink ha seguido siendo descargada, llegando a acumular más de **180.000 descargas en Android** y más de **50.000 en iOS** (Fig. 2.3); superando los **100.000 dólares americanos en ventas**. Estos datos **hacen obvia la demanda de los usuarios por una alternativa** a la aplicación oficial de Wear OS by Google.

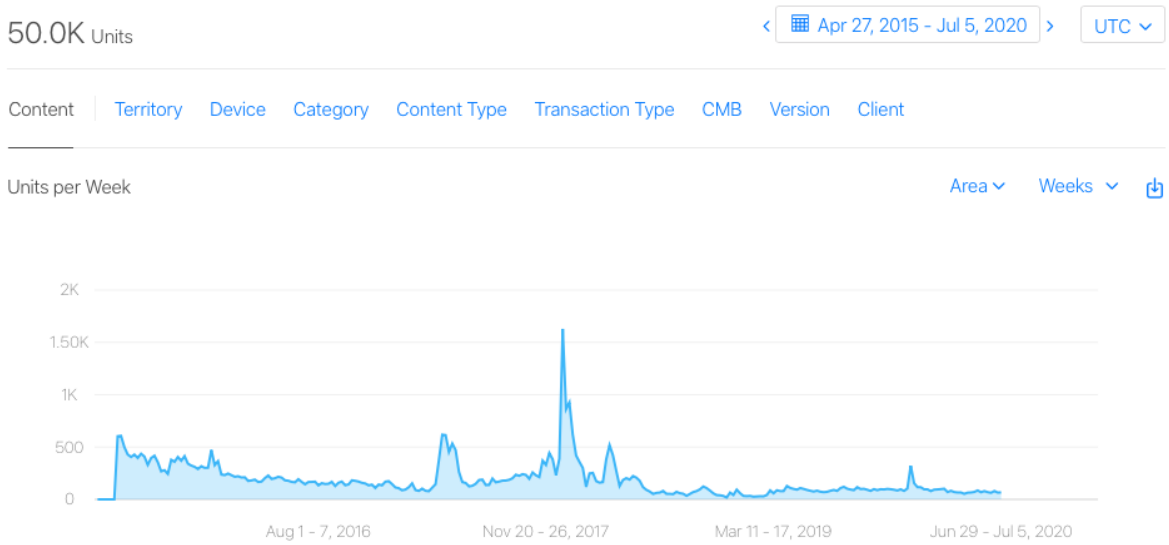

Figura 2.3: *Descargas desde 2015 a 2020 de Aerlink iOS*

2.3. Interoperabilidad mediante Bluetooth

Apple no permite establecer conexiones Bluetooth Classic con dispositivos que no tengan la certificación MFi. La única forma disponible en la actualidad para conectar este tipo de dispositivos entre sí es con Bluetooth Low Energy.

Bluetooth Low Energy (BLE) se trata de la versión de bajo consumo del estándar Bluetooth que nos permite desplegar redes inalámbricas de área personal (PAN) y forma parte de la especificación de Bluetooth 4.0. Su sustancial reducción del consumo lo hace perfecto para gran variedad de aplicaciones como wearables, fitness o seguridad.

El Generic Access Profile (GAP) se encarga de controlar las conexiones y los anuncios en BLE permitiendo hacer la presencia de un dispositivo pública y determinando si dos dispositivos pueden (o no) establecer una conexión entre ellos. En el caso de Aerlink, la parte iOS tomará el papel de periférico e empezará a anunciar su presencia a través de GAP. Será la app de Wear OS la que actúe como central, conectándose al dispositivo iOS para empezar a recibir información sobre este.

La manera en que dos dispositivos BLE se comunican viene definida por el Generic Attribute Profile (GATT). En GATT el periférico se conoce como servidor GATT, contiene

los datos de búsqueda ATT y las definiciones de servicio y características, la central o cliente GATT es quien envía solicitudes a este servidor.

2.3.1. Servicios y Características

Las transacciones GATT en BLE se basan en objetos anidados de alto nivel denominados **Perfiles, Servicios y Características** organizados como se muestra en la Fig. 2.4:

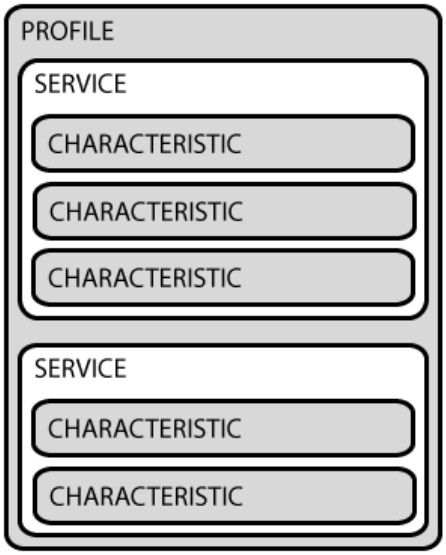

Figura 2.4: *Representación de un Perfil BLE*

Un **perfil** es simplemente una colección de servicios predefinidos por el Bluetooth Special Interest Group, por el fabricante del periférico o, en el caso de Aerlink, por el desarrollador de la aplicación que inicia los servicios.

Los **servicios**, al igual que los perfiles, se utilizan para dividir datos en entidades lógicas, conteniendo a su vez una colección de características. Cada servicio se distingue a través de un Universally Unique Identifier o Identificador Único Universal (UUID) y puede tener una o más características a su vez identificadas por un UUID.

El nivel más bajo en las transacciones GATT son las **características**, que encapsulan un único tipo de dato que variará dependiendo del uso de la característica y cuyo formato deberá ser conocido por la central. Las centrales pueden registrase a ellas para obtener

actualizaciones cuando este dato cambie o pueden escribir en ellas para transmitir un cambio o actualización en el periférico.

2.3.2. Dificultades

A la restricción de Apple de no poder utilizar Bluetooth Classic libremente hay que añadir la estricta gestión de las aplicaciones en segundo plano de iOS, que penaliza especialmente a aplicaciones como Aerlink, aplicaciones que idealmente el usuario no necesitaría estar abriendo continuamente pero se ve forzado a hacerlo periódicamente; en el caso de Aerlink esta puede ser la única forma que tiene de restablecer una conexión perdida con un dispositivo Wear OS.

Por otro lado, el Bluetooth stack de Android es en un pequeño desastre. Un claro ejemplo es la función de reconexión automática de *BluetoothDevice*, una función plagada de errores que parece funcionar de manera aleatoria en algunos dispositivos y en otros directamente no funciona nunca. Esto consigue que el desarrollador simplemente no pueda recurrir a esta funcionalidad y tenga que volver a escanear en busca del dispositivo al que estaba conectado para poder iniciar una conexión de cero manualmente.

Capítulo 3

Especificación de Requisitos

> First, solve the problem. Then, write the code.
>
> *John Johnson*

El objetivo de la **Especificación de Requisitos Software**[5] es definir de forma precisa y comprensible todas las funcionalidades y restricciones del sistema que se desea construir. Se procura que la Especificación de Requisitos Software despeje y clarifique cualquier duda que pueda surgir sobre el funcionamiento del sistema a modo de "guía de usuario".

3.1. Ámbito del sistema

El sistema Aerlink **consiste en dos aplicaciones**, una para dispositivos **Wear OS** y otra para dispositivos **iOS**, que deberán establecer una conexión mediante tecnología Bluetooth LE, permitiendo al usuario recibir información del dispositivo iOS en su dispositivo Wear OS y controlar diversas funcionalidades de su dispositivo iOS remotamente con su dispositivo Wear OS. **No se contempla el caso de conectar un dispositivo Wear OS con un dispositivo Android**, este tipo de conexión queda resuelta por la conexión nativa ofrecida por ambos sistemas operativos que además permite añadir funcionalidades nuevas de forma sencilla.

Se da por supuesto el uso de un dispositivo ejecutando el sistema operativo Wear OS y otro ejecutando el sistema operativo iOS. Además de un correcto funcionamiento de las

capacidades Bluetooth de ambos dispositivos se supone que el usuario acepta los permisos necesarios para la ejecución de ambas aplicaciones; siendo estos el permiso de localización en el dispositivo Wear OS y los permisos de Bluetooth, acceso a cámara, guardar imágenes y acceso a los recordatorios en el dispositivo iOS.

3.2. Restricciones de Diseño

- Apple solo permite establecer conexiones BLE con dispositivos que no formen parte del programa MFi. Al no tener acceso a Bluetooth Classic la velocidad y fiabilidad de la conexión se ven muy afectadas pasando de 2-3 Mbps a tan solo 200 kbps.
- La gestión de aplicaciones en segundo plano de Apple es muy estricta causando que el servicio BLE pueda no encontrarse disponible, forzando al usuario a reabrir la aplicación para relanzar el servicio.
- La única acción permitida en las notificaciones de Apple desde un dispositivo Bluetooth es la eliminación de las notificaciones.
- La gran cantidad de distintos relojes Wear OS hace imposible probar la aplicación en todos pudiendo causar comportamientos extraños en algún dispositivo. Las distintas compañías aplican regímenes distintos de gestión de memoria pudiendo llegar a cerrar conexiones Bluetooth en algunos dispositivos donde otros no se hubieran visto afectados.

3.3. Características de los Usuarios

El sistema solo tiene un tipo de usuario, el usuario final de las aplicaciones. Estos usuarios tendrán un conocimiento técnico por encima de la media al haberse instalado en sus dispositivos una forma alternativa a la oficial de Google de conectar sus relojes con sus dispositivos iOS.

3.4. Requisitos específicos

Diferenciaremos entre requisitos **funcionales**, es decir aquellos relacionados con los servicios o prestaciones que debe cumplir el sistema y **no funcionales** o atributos de calidad que hacen referencia a las características de funcionamiento.

Para cada requisito identificado se especifica:

ID	Identificador numérico único
Nombre	Nombre descriptivo del requisito
Descripción	Descripción específica del requisito y su función
Entrada	Acciones y parámetros solicitados al usuario para llevar a cabo la funcionalidad
Salida	Respuesta del sistema
Contexto	App y pantalla en la que se desarrolla la acción

3.4.1. Especificación de Atributos de Calidad

1. **Seguridad y privacidad:** El sistema debe asegurar la protección de los datos e información personal del usuario.

2. **Usabilidad:** El sistema debe ser eficaz, intuitivo y ofrecer al usuario una experiencia agradable.

3. **Compatibilidad:** El sistema deberá ser compatible con el mayor número de dispositivos posibles tanto en Wear OS como iOS.

4. **Actualizable:** El sistema deberá permitir actualizaciones que añadan funcionalidad con facilidad sin afectar a la ya existente.

3.4.2. Especificación de Requisitos Funcionales

ID	1
Nombre	Inicio del servicio Aerlink
Descripción	Un usuario puede iniciar el servicio Aerlink para poder establecer la conexión entre dispositivos.
Entrada	El usuario marca como activa la opción "Aerlink"
Salida	Si el usuario concedió el permiso de localización el servicio se inicia, si no, se le pide que lo conceda.
Contexto	Wear OS app. Pantalla principal

ID	2
Nombre	Establecer conexión
Descripción	Una vez iniciado el servicio Aerlink este debe intentar establecer la conexión con un dispositivo iOS cercano.
Entrada	-
Salida	Los dispositivos iOS y Wear OS quedan conectados.
Contexto	Wear OS app. Servicio Aerlink ejecutándose tanto en iOS como en Wear OS

ID	3
Nombre	Ver nivel de batería
Descripción	Un usuario puede ver el nivel de batería de su dispositivo iOS
Entrada	-
Salida	El nivel de batería es visible
Contexto	Wear OS app. Estando conectado. Pantalla principal

ID	4
Nombre	Recibir notificación
Descripción	Un usuario puede recibir notificaciones de su dispositivo iOS
Entrada	Se recibe una notificación en el dispositivo iOS
Salida	Se recibe la misma notificación en el reloj
Contexto	Wear OS app. Estando conectado

ID	5
Nombre	Descartar una notificación
Descripción	Un usuario puede descartar una notificación recibida
Entrada	El usuario desliza una notificación hacia la derecha
Salida	La misma notificación es descartada en el dispositivo iOS
Contexto	Wear OS app. Estando conectado. Viendo una notificación

ID	6
Nombre	Ver información multimedia
Descripción	Un usuario puede ver la información del contenido multimedia en reproducción actualmente en su dipositivo iOS
Entrada	-
Salida	Se muestra la información del contenido multimedia
Contexto	Wear OS app. Estando conectado. Reproduciendo contenido multimedia

ID	7
Nombre	Controlar reproducción multimedia
Descripción	Un usuario puede controlar la reproducción multimedia de su dipositivo iOS
Entrada	El usuario selecciona reproducir, pausar, adelantar, retrasar, subir o bajar volumen
Salida	La acción es replicada en su dispositivo iOS
Contexto	Wear OS app. Estando conectado. Reproduciendo contenido

ID	8
Nombre	Tomar foto
Descripción	Un usuario puede tomar una foto remotamente con la cámara de su dispositivo iOS
Entrada	El usuario pulsa el botón de captura
Salida	Se inicia una cuenta atrás en el dispositivo iOS que resulta en una foto siendo capturada
Contexto	Wear OS app. Estando conectado. Pantalla de captura de foto y cámara abierta en el dispositivo iOS

ID	9
Nombre	Ver foto tomada
Descripción	Un usuario puede ver la foto tomada remotamente con la cámara de su dispositivo iOS
Entrada	-
Salida	Se muestra la foto tomada en la pantalla del dispositivo
Contexto	Wear OS app. Estando conectado. Tras haber tomado una foto remotamente

ID	10
Nombre	Ver listado de listas de recordatorios
Descripción	Un usuario puede ver el listado de listas de recordatorios
Entrada	-
Salida	Se muestra el listado de listas de recordatorios
Contexto	Wear OS app. Estando conectado. Pantalla de listado de listas de recordatorios

ID	11
Nombre	Ver listado de recordatorios
Descripción	Un usuario puede ver un listado específico de recordatorios
Entrada	El usuario selecciona un listado de recordatorios
Salida	Se muestra el listado de recordatorios
Contexto	Wear OS app. Estando conectado. Pantalla de listado de listas de recordatorios

ID	12
Nombre	Cambiar estado de un recordatorio
Descripción	Un usuario puede cambiar remotamente el estado de un recordatorio
Entrada	El usuario selecciona un recordatorio
Salida	El estado del recordatorio cambia en su dispositivo iOS o se muestra un error
Contexto	Wear OS app. Estando conectado. Pantalla de listado de recordatorios

Capítulo 4

Diseño e implementación del sistema

> Without requirements or design, programming is the art of adding bugs to an empty text file.
>
> *Louis Srygley*

El **sistema Aerlink se compone de dos aplicaciones**, como se puede ver en Fig. 4.1 una se ejecutará en un **dispositivo Wear OS** y la otra se ejecutará en un **dispositivo iOS**, estos dispositivos entablarán una conexión **Bluetooth LE**. En el caso de un dispositivo Wear OS conectado a un dispositivo Android Aerlink no será necesario y no afectará de ninguna forma esta conexión. Opcionalmente, un usuario podrá mantener su su dispositivo Wear OS conectado de forma nativa a un dispositivo Android y a través de Aerlink con un dispositivo iOS, manteniendo conexiones independientes con ambos dispositivos a la vez.

En los diagramas de clase de este capítulo se pueden distinguir **las clases implementadas por el sistema**, que estarán **representadas en amarillo**, de **las propias de cada plataforma, representadas en blanco**.

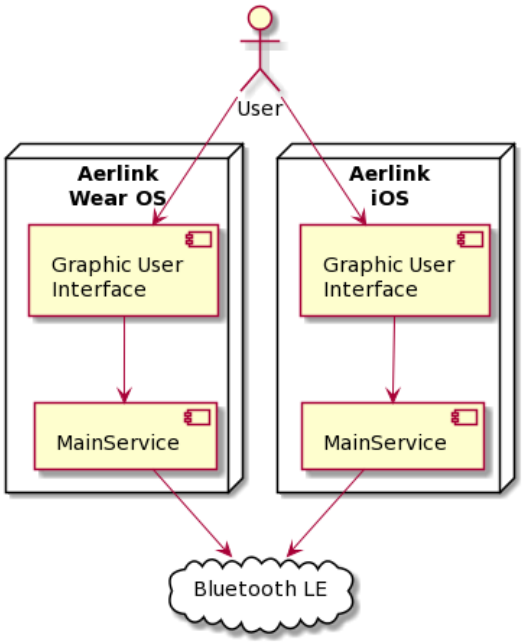

Figura 4.1: *Sistema Aerlink*

4.1. Aerlink Wear OS

4.1.1. MainService

El servicio Android[6] **MainService** es iniciado por el usuario a través de la aplicación Aerlink en Wear OS y se mantendrá activo siempre que se quiera establecer o mantener una conexión. Este servicio se encarga de coordinar las clases responsables de descubrir dispositivos Aerlink, iniciar un enlace seguro y establecer una conexión. Además el servicio MainService permite acceder al estado actual de la conexión y a los servicios disponibles en el dispositivo conectado en caso de que haya uno.

4.1.2. DiscoveryManager

Esta clase es responsable de descubrir dispositivos compatibles que se encuentren lo suficientemente cerca para establecer una conexión. La clase **DiscoveryManager** permite iniciar y detener el descubrimiento de dispositivos, y tiene una interfaz **Callback** para recibir los dispositivos que se vayan encontrando.

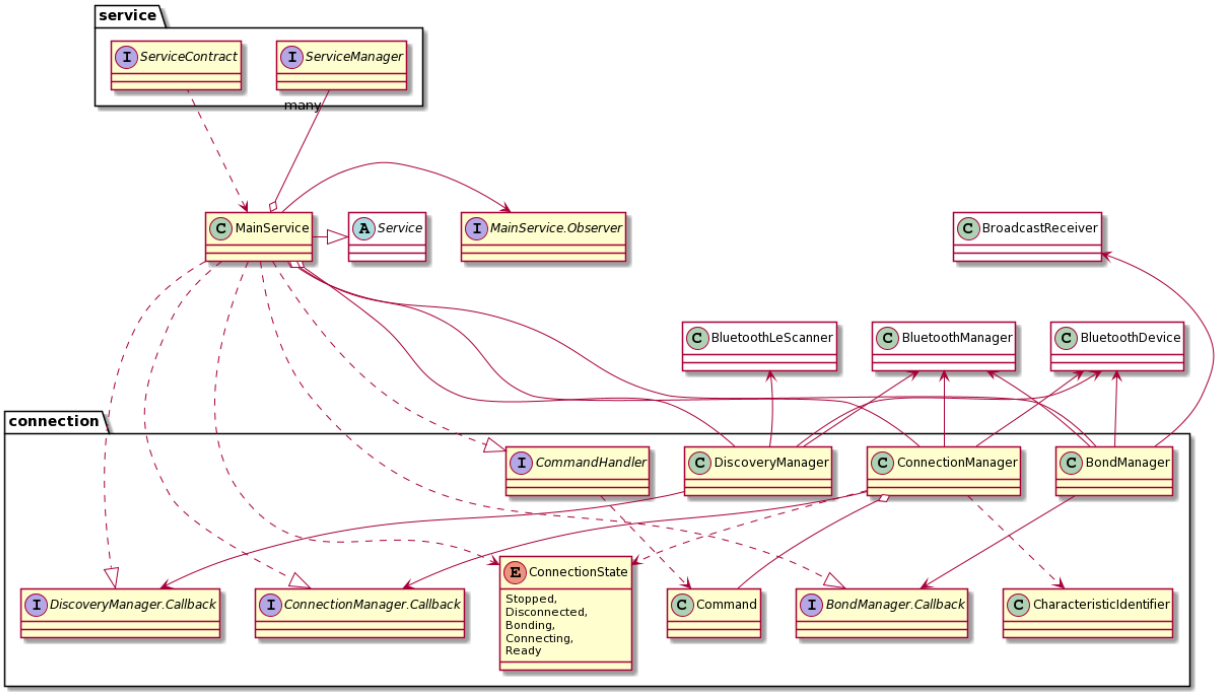

Figura 4.2: *Wear OS: Diagrama de clases responsables de la conexión*

Al iniciar el descubrimiento comprueba si el Bluetooth está activado, activándolo y reintentando iniciar el descubrimiento en caso de que no este activo; a continuación, comprueba si el descubrimiento ya se ha iniciado para no volver a iniciarlo; por último comprueba la fecha del último escanamiento para no realizar escaneamientos demasiado seguidos, hay dispositivos que pueden llegar a matar procesos que intenten abusar del Bluetooth ya que puede llegar a consumir mucha batería mantenerlo activo indefinidamente. Si se pasan todas las comprobaciones anteriores intenta iniciar el descubrimiento y comprueba si se ha iniciado exitosamente para poder reintentarlo en caso contrario.

Los resultados del *BluetoothLeScanner* se reciben en un *ScanCallback*, al encontrar un dispositivo Bluetooth se comprueba si su nombre es uno de los permitidos, **los dispositivos con la aplicación Aerlink de iOS instalada aparecerán como "Aerlink"**, además se permite intentar conexión con dispositivos que aparezcan como "BLE Utility" o "Blank", algunas aplicaciones gratuitas de iOS permiten anunciar la presencia del dispositivo con estos nombres, esto permite el uso de la aplicación Aerlink en Wear OS con estas aplicaciones

gratuitas a pesar de que no vayan a tener acceso a todas las funcionalidades de la aplicación Aerlink de iOS. En caso de que el descubrimiento genere un error se reinicia el descubrimiento para reintentarlo de nuevo.

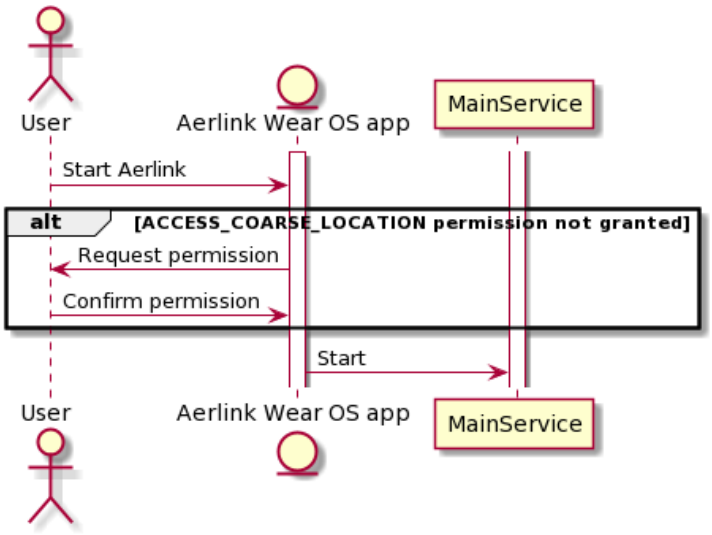

Figura 4.3: *Requisito funcional 1: Inicio del servicio Aerlink*

4.1.3. BondManager

La clase **BondManager** se encarga del proceso de enlazar el dispositivo WearOS con un dispositivo iOS previamente descubierto, permite únicamente iniciar el proceso de enlace y observar si ha sido un exitoso o no a través de una interfaz **Callback**.

La clase BondManager tiene un *BroadcastReceiver*[7] que recibe *intents* con la acción *BluetoothDevice.ACTION_BOND_STATE_CHANGED*, esto significa que se ejecutará cuando haya cambios en el estado de enlazamiento de los dispositivos Bluetooth. Este *BroadcastReceiver* comprueba que el cambio en estado ha ocurrido con el dispositivo con el que estamos intentando enlazar y el estado previo y actual para decidir si ha sido un enlace exitoso o si ha fallado.

4.1.4. ConnectionManager

El **ConnectionManager** se encarga de establecer una conexión con un dispositivo Bluetooth previamente descubierto. Al ser la clase encargada de la conexión permite también comprobar si un servicio está disponible en el dispositivo conectado y ejecutar comandos en este. A través de su **Callback** se reciben mensajes cuando el dispositivo este listo para recibir suscripciones a sus servicios, cuando ocurra algún cambio en el estado de la conexión, cuando el dispositivo con el que estamos intentando conectarnos necesita un un enlace seguro o cuando ha habido cambios en una de las Characteristics de los servicios a los que nos hemos suscrito.

Al intentar conectarse con un dispositivo el ConnectionManager comprueba primero si existe un enlace con este, pidiendo uno a través de su Callback si fuese necesario. En caso de tratarse de un dispositivo ya enlazado se inicia el proceso de conexión. Una vez establecida la conexión lo primero que se intenta es aumentar la Maximum Transmission Unit o Unidad Máxima de Transferencia (MTU) para permitir el envío y recepción de paquetes más grandes. Se consiga o no este cambio, el proceso de conexión continua con el descubrimiento de los servicios disponibles en el dispositivo conectado.

Una vez descubiertos los servicios disponibles se pide al Callback las características a las que quiere suscribirse, se añaden a una cola y se procede a intentar suscribirse a todas; en caso de error o de que no haya ningún servicio disponible se considera que la conexión ha fallado.

Por cada característica se intenta la suscripción un máximo número veces antes de considerar que la conexión ha fallado, al suscribirse a una característica se comprueba de nuevo la cola para continuar con la siguiente hasta que no quede ninguna, en este caso se considerará que la conexión está lista.

Para acabar con el ConnectionManager, la gestión de comandos se hace también a través de una cola que se comprueba siempre que se añade un nuevo comando o se completa la ejecución del último.

Todas las acciones realizadas en el ConnectionManager tienen un tiempo límite de ejecución antes de que se considere que la conexión ha fallado, esto se hace para poder informar al usuario de que la conexión se ha perdido e intentar restablecerla.

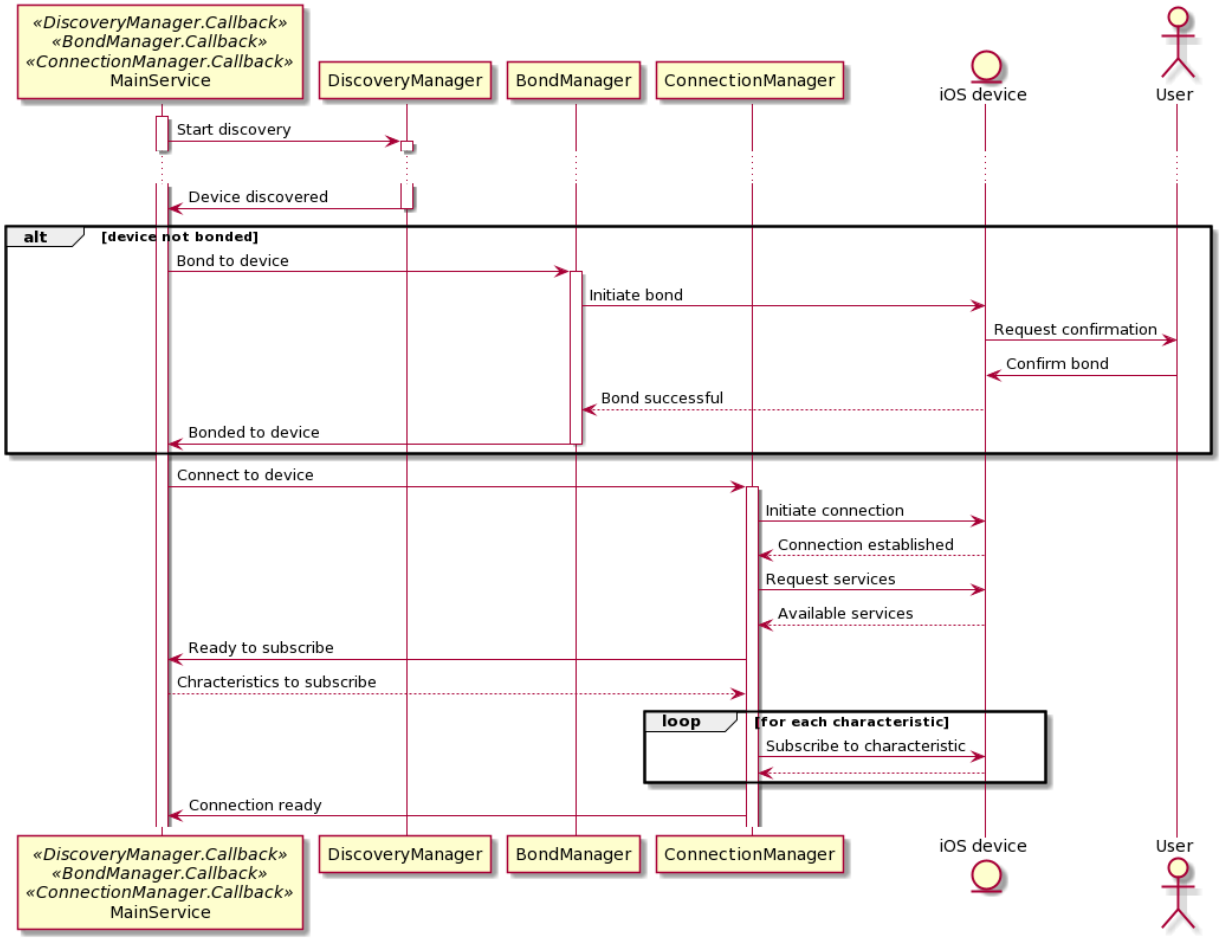

Figura 4.4: *Requisito funcional 2: Establecer conexión*

4.1.5. ServiceContract y ServiceManager

ServiceContract es una interfaz que define el UUID del servicio, las características a suscribirse y un método para crear un ServiceManager. La interfaz **ServiceManager** incluye métodos para inicializar el ServiceManager, comprobar si puede manejar una característica, gestionar una característica que haya recibido un cambio y para cerrar el ServiceManager. Cada servicio GATT necesita su ServiceContract y su ServiceManager para que se pueda

trabajar con él.

4.1.6. CharacteristicIdentifier

La clase **CharacteristicIdentifier** simplemente se usa para identificar una característica GATT, esta compuesta del UUID del servicio y del UUID de la característica.

4.1.7. Command

Un **Command** tiene la información necesaria para realizar una acción de lectura o escritura en el dispositivo enlazado. Contiene la información para identificar la característica donde se va a ejecutar, el paquete de datos en caso de tratarse de una acción de escritura y lleva la cuenta de intentos fallidos.

4.2. Aerlink iOS

La gran parte del código que gestiona la conexión entre los dispositivos se encuentra en la aplicación Wear OS pero **el usuario necesitará también la aplicación iOS** para que esta pueda **anunciar su presencia** y **ofrecer las funcionalidades exclusivas** de Aerlink.

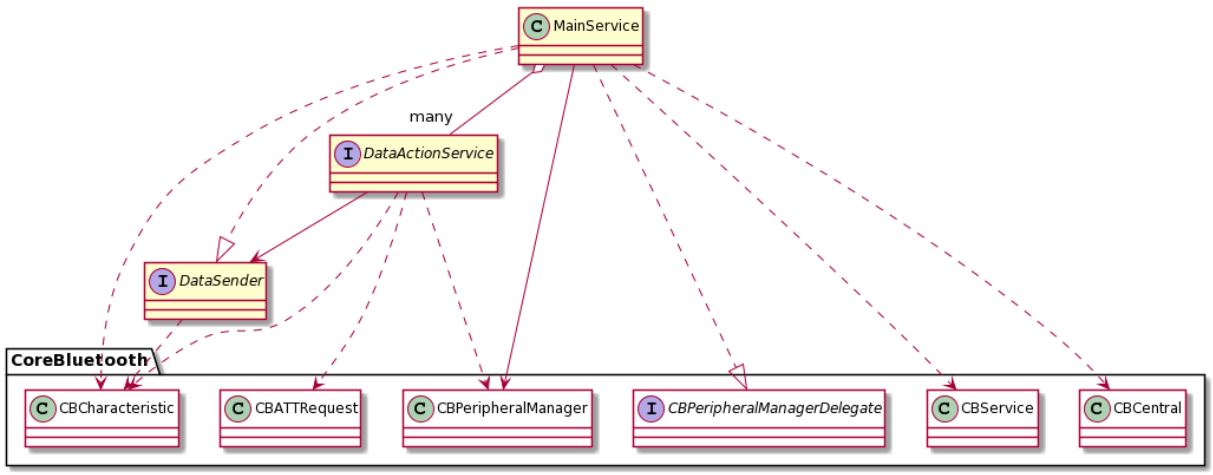

Figura 4.5: *iOS: Diagrama de clases responsables de la conexión*

4.2.1. MainService

La clase **MainService** actúa de delegado de un *CBPeripheralManager*[8], esto le permite recibir actualizaciones cuando, entre otras cosas, ha habido un cambio en la disponibilidad del Bluetooth en el dispositivo, cuando se ha realizado una conexión o desconexión, o cuando un dispositivo se ha registrado a una característica. Al recibir un estado válido se inicia un servicio y se le añaden las características de los servicios exclusivos de Aerlink; además, al recibir un estado inválido, deja de anunciarse y reinicia ambos servicios a su estado inicial.

La clase MainService también se encarga de enviar los datos a los dispositivos Wear OS conectados, divide los datos a enviar en paquetes con un MTU válido y en caso de error avisará a los servicios cuando puedan volver a enviar datos.

4.2.2. DataActionService

Para simplificar el desarrollo, los servicios implementados en la aplicación iOS tendrán un par de características con un funcionamiento análogo. Cada **DataActionService** tendrá una característica de datos a la que la aplicación Wear OS se podrá suscribir para recibir datos, y una característica de acciones a través de la cual la aplicación Wear OS podrá enviar comandos.

4.3. Diseño de servicios Apple

En esta sección se describen los servicios preexistentes a los que tendrá acceso la aplicación Wear OS. Los servicios de notificaciones, control multimedia y batería son servicios implementados por Apple y que se vuelven disponibles una vez se ha establecido la conexión entre los dispositivos del sistema. Estos servicios **solo requieren ser gestionados en la aplicación Wear OS**.

4.3.1. Notificaciones

El **Apple Notification Center Service (ANCS)**[9] es un servicio disponible en dispositivos iOS que permite a los dispositivos registrados obtener el feed de notificaciones y realizar determinadas acciones sobre ellas.

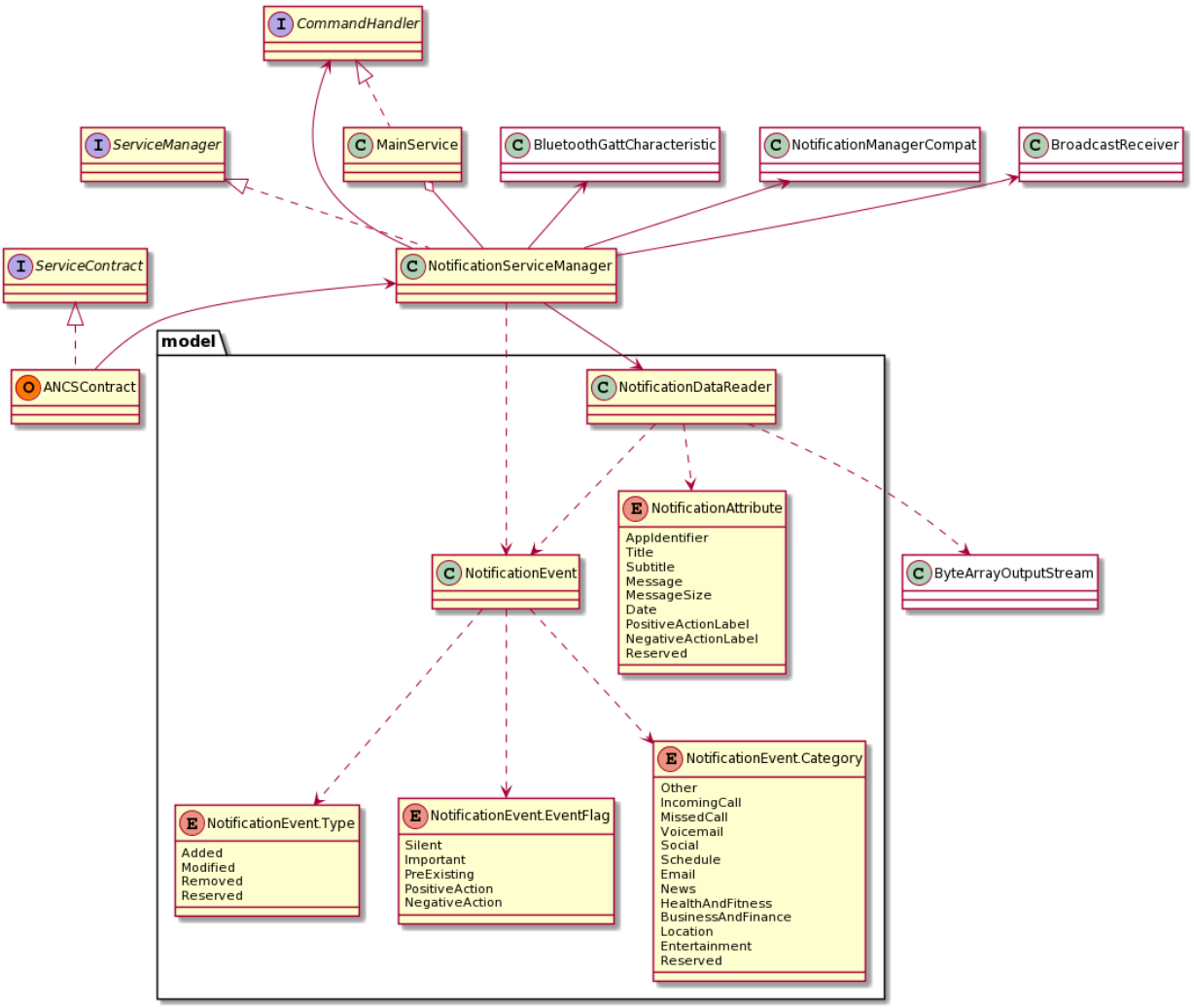

Figura 4.6: *Diagrama de clases ANCS*

La aplicación Wear OS utiliza este servicio para mostrar todas las notificaciones recibidas en el dispositivo con el que está emparejado. El usuario podrá descartarlas deslizando el dedo sobre la notificación eliminándose a su vez de su dispositivo iOS.

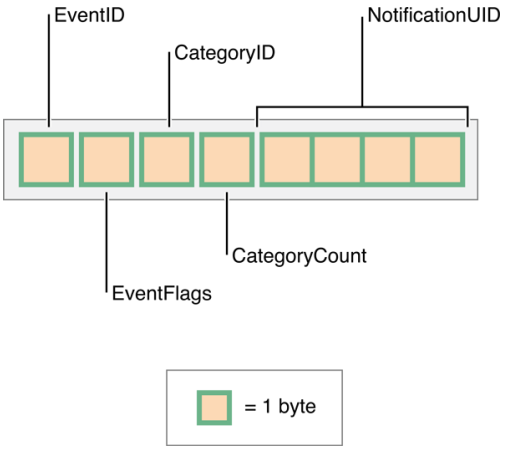

Figura 4.7: *Ejemplo de evento*

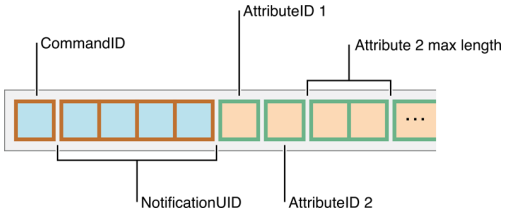

Figura 4.8: *Ejemplo de comando*

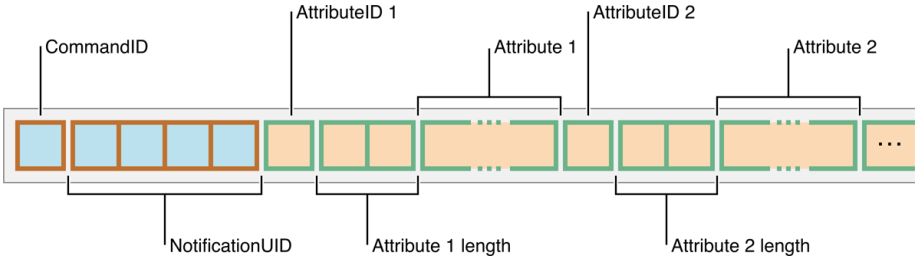

Figura 4.9: *Ejemplo de respuesta*

Eventos (Fig. 4.7) son recibidos en la característica ***Fuente de notificaciones*** como un paquete de bytes con la información necesaria para pedir la información sobre la notificación que sea necesaria.

El **NotificationUID** de estos se eventos se puede usar para pedir a la característica ***Punto de control*** atributos sobre la notificación como pueden ser el título, el mensaje o la aplicación origen de la notificación enviando un comando como el de la Fig. 4.8.

Si la petición se realiza correctamente, la característica ***Fuente de datos*** responderá con la información pedida utilizando paquetes con el formato en Fig. 4.9.

El **NotificationServiceManager** se encarga de gestionar los cambios en estas características y la creación de los eventos necesarios para obtener la información necesaria. Los eventos recibidos se añaden a una cola para poder gestionarlos de uno en uno, por cada evento se crea un comando para pedir la información completa y se crea un **Notifica-**

tionDataReader para juntar todos los paquetes que la conforman y poder manejar la información una vez la respuesta este completa. Una vez se han recibido todos los atributos necesarios se puede crear la notificación y publicar usando el *NotificationManager* para que el usuario pueda verla.

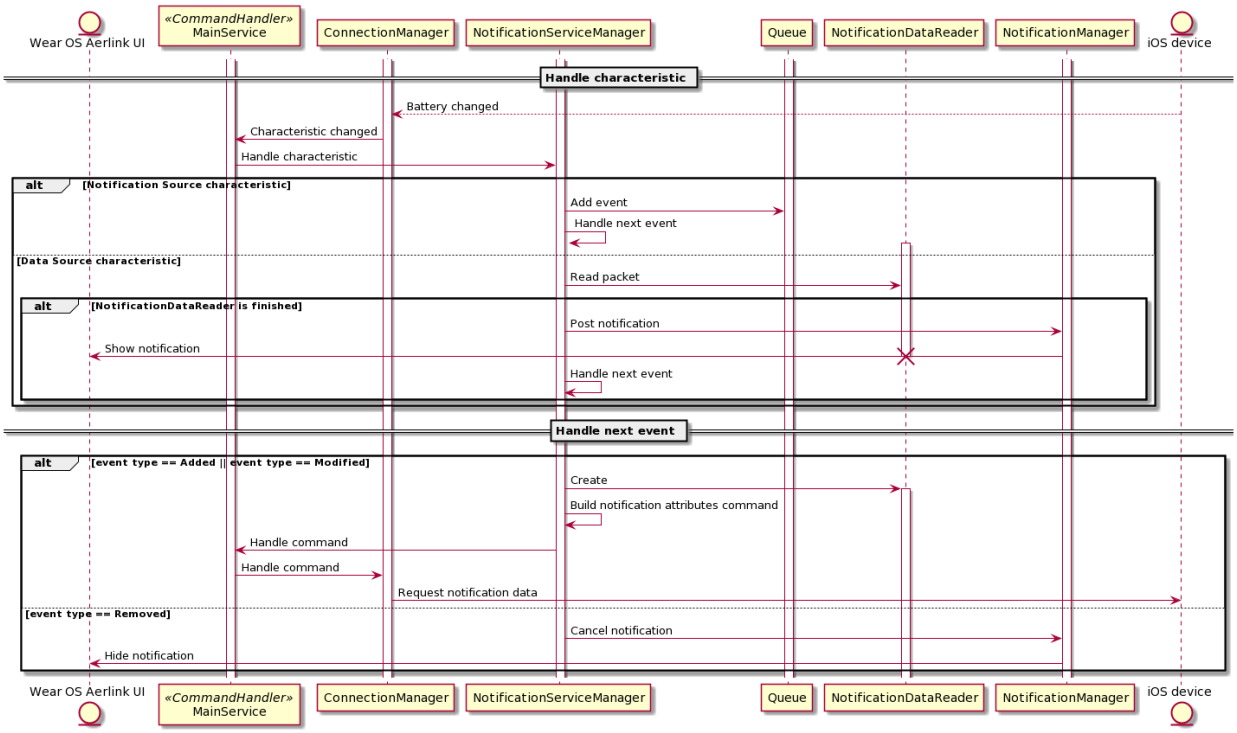

Figura 4.10: *Requisito funcional 4: Recibir notificación*

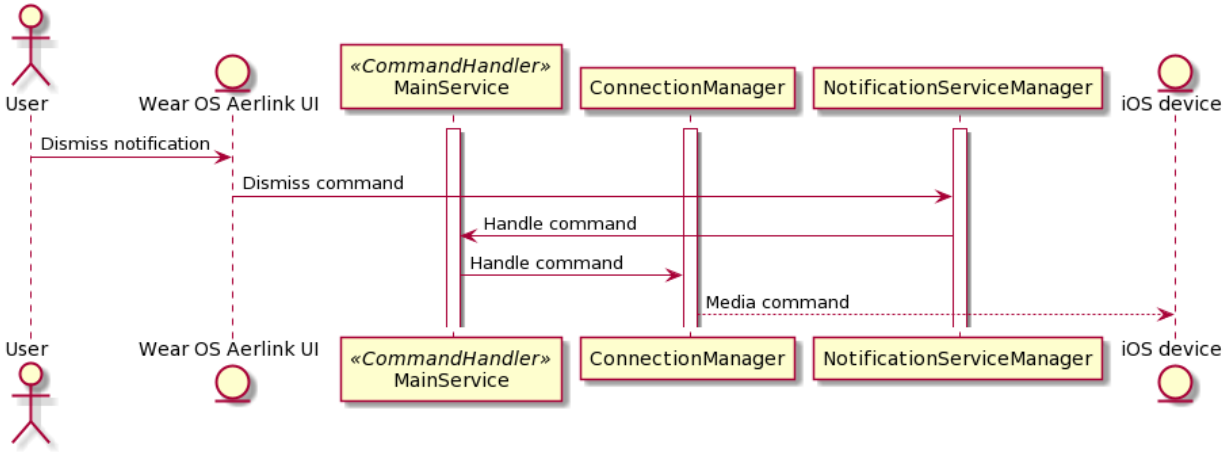

Figura 4.11: *Requisito funcional 5: Descartar una notificación*

4.3.2. Control multimedia

El **Apple Media Service (AMS)**[10] es un servicio disponible en dispositivos iOS que permite a los dispositivos registrados controlar remotamente aplicaciones multimedia y obtener información de los medios multimedia siendo consumidos.

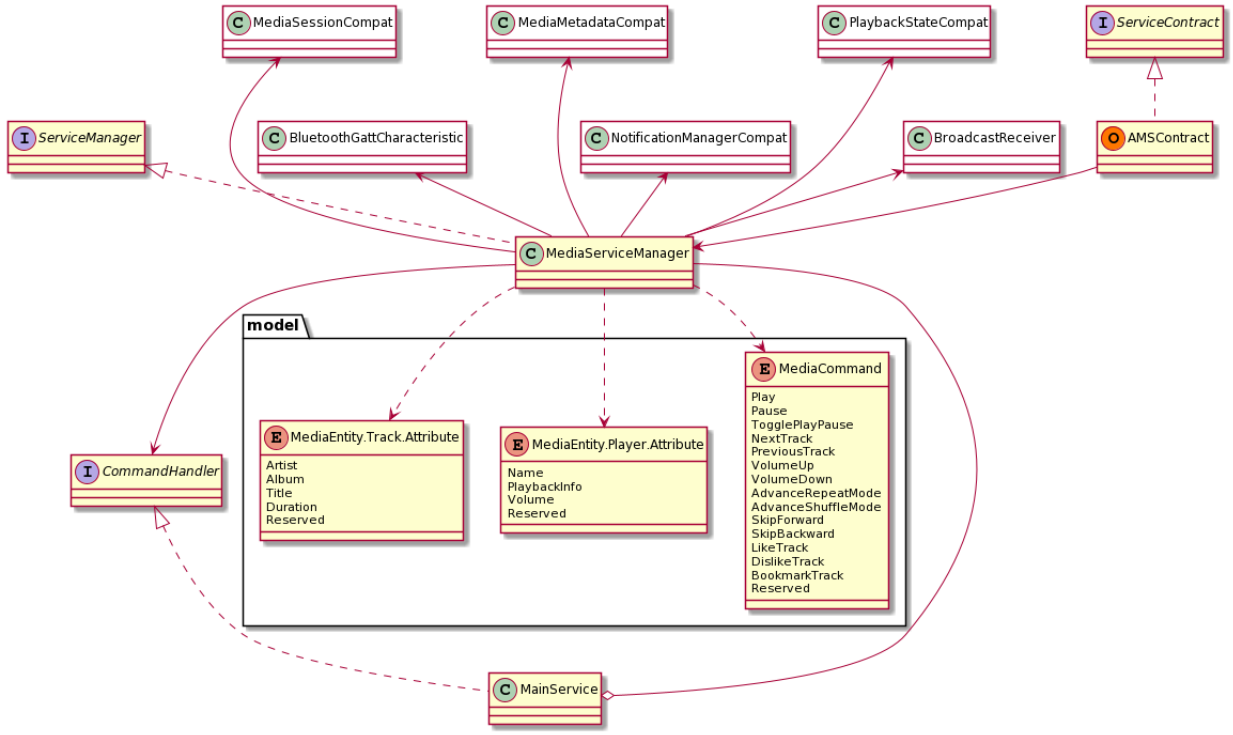

Figura 4.12: *Diagrama de clases AMS*

Los dispositivos registrados a este servicio pueden registrarse a la característica ***Actualización de entidades*** para recibir actualizaciones cuando haya cambios en determinados atributos como pueden ser el álbum, el nombre o el cantante de la canción actual. También se puede controlar el estado de reproducción escribiendo comandos en la característica ***Comandos remotos***.

El **MediaServiceManager** se encarga de registrarse a las características necesarias, recibir las actualizaciones, enviar los comandos para controlar el estado de reproducción y hacer todo esto disponible al usuario.

La aplicación Wear OS utiliza este servicio para mostrar la información de los medios multimedia en reproducción en el dispositivo con el que está emparejado. Además, el usuario podrá controlar remotamente estos medios, pudiendo retroceder, avanzar, reproducir, pausar o controlar el volumen.

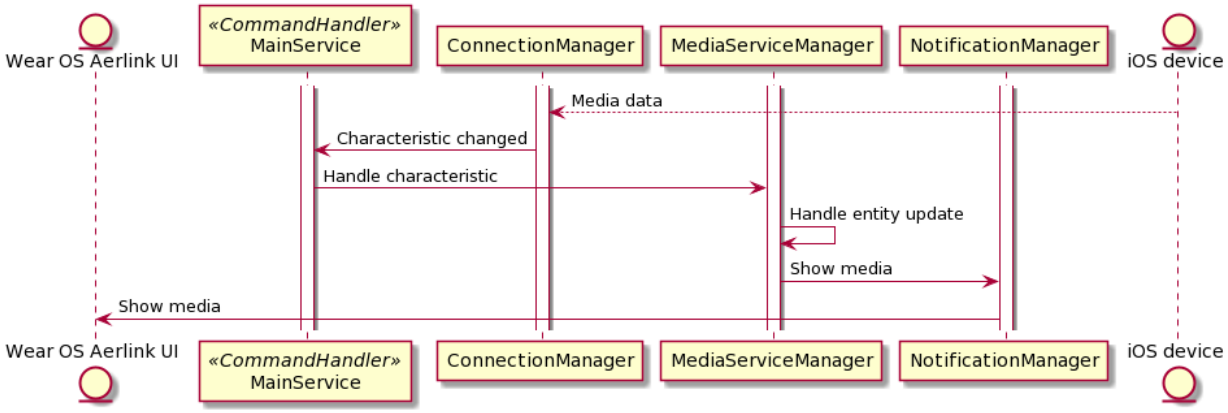

Figura 4.13: *Requisito funcional 6: Ver información multimedia*

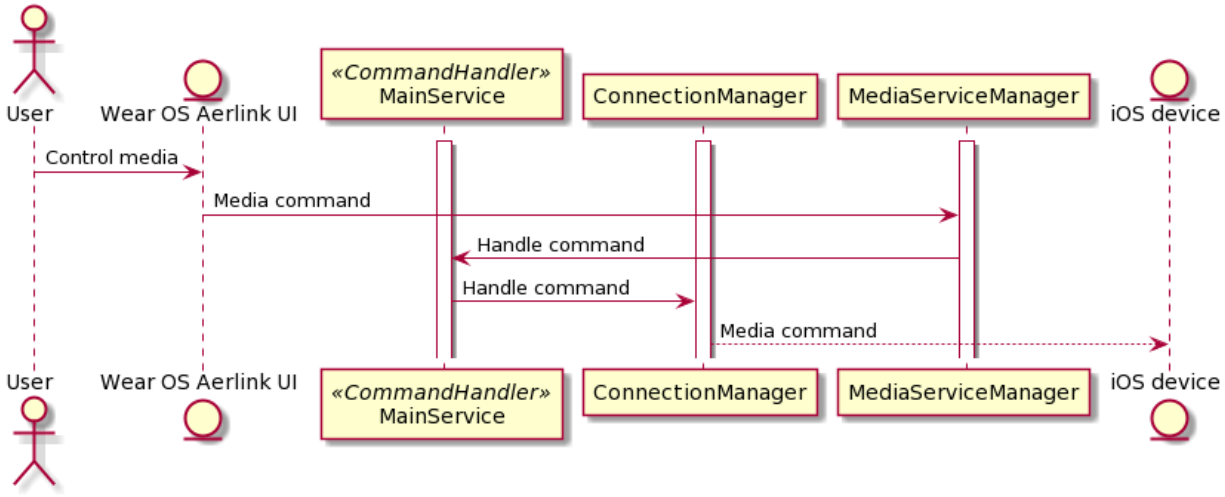

Figura 4.14: *Requisito funcional 7: Controlar reproducción multimedia*

4.3.3. Batería

El **Battery Service**[11] es un servicio incluido en la especificación oficial de Bluetooth LE y disponible en dispositivos iOS.

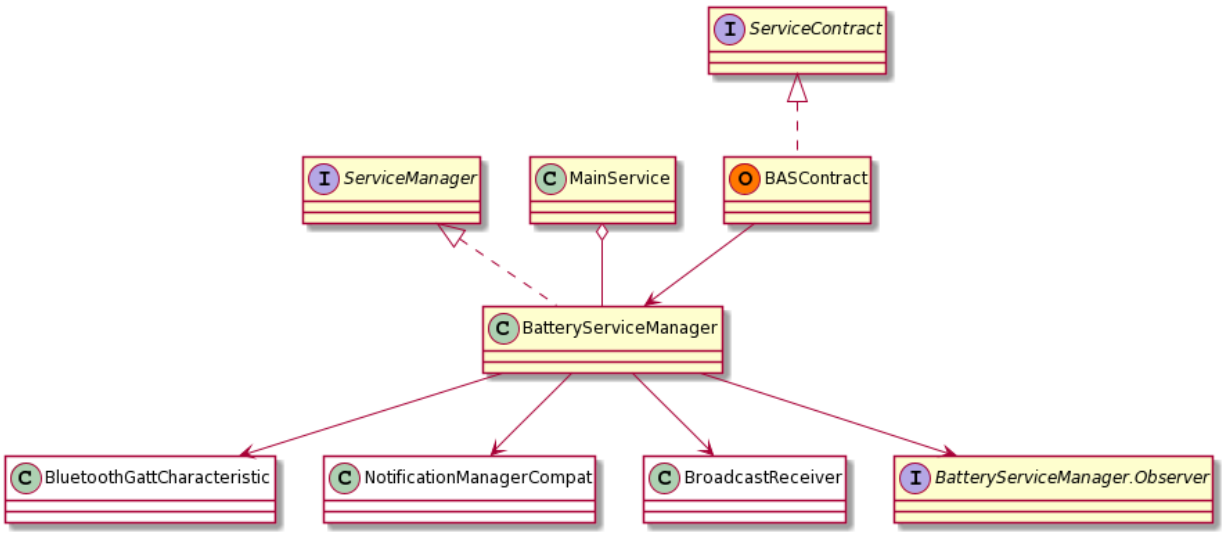

Figura 4.15: *Diagrama de clases Battery Service*

Los dispositivos registrados a este servicio recibirán actualizaciones cuando el nivel de batería del dispositivo con el que está emparejado cambie.

El **BatteryServiceManager** se registra a los cambios en esta característica y mantiene actualizado el nivel de batería mostrado al usuario. Además, cuando el nivel de batería es inferior al 20 %, muestra una notificación al usuario con el nivel de batería.

La aplicación Wear OS utiliza este servicio para mostrar el nivel de batería del dispositivo con el que está emparejado.

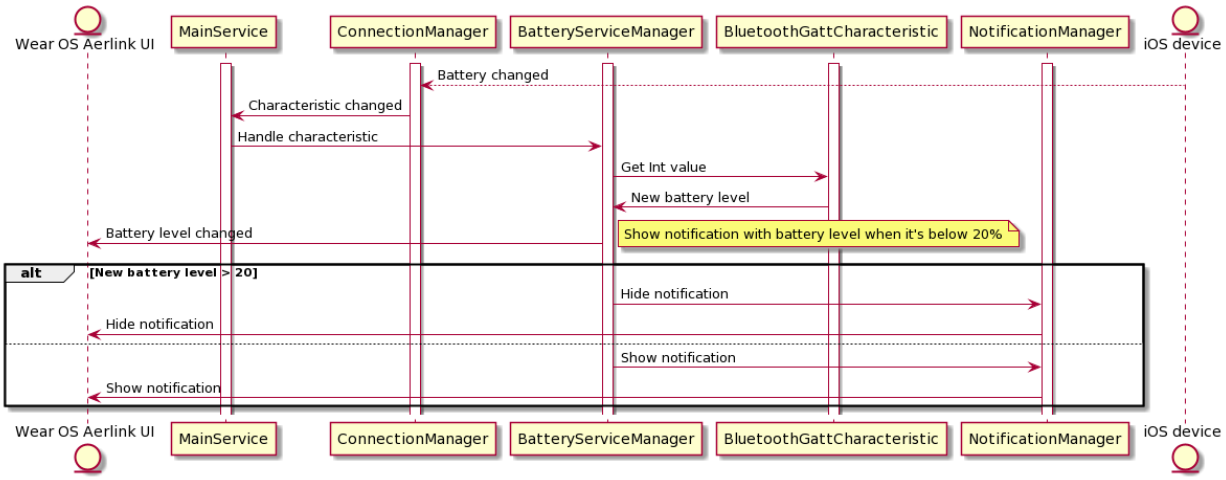

Figura 4.16: *Requisito funcional 3: Ver nivel de batería*

4.4. Diseño del servicio Aerlink

El **servicio Aerlink** es el servicio **implementado por la aplicación Aerlink de iOS**. Ofrece características y funcionalidades a las cuales, sin esta aplicación, no sería posible acceder. Cada subservicio de Aerlink tendrá una característica ***Acciones*** a través de la cual podrá recibir comandos, y otra característica ***Datos*** a través de la cual se enviarán los datos pertinentes del servicio. Este servicio **ha sido desarrollado específicamente para la aplicación Aerlink de Wear OS** por lo que su uso es pleno. Este servicio está formado por los servicios de control remoto de cámara y recordatorios.

4.4.1. Control remoto de cámara

Los dispositivos registrados a este servicio podrán tomar fotografías remotamente a través de la característica ***Acciones***; recibiendo la captura a través de la característica ***Datos***.

iOS

En la aplicación iOS, el servicio **CameraRemoteService** se encarga de recibir actualizaciones de la interfaz gráfica de la aplicación Aerlink para determinar si la cámara esta abierta o cerrada, y para recibir las imágenes tomadas y enviarlas al dispositivo Wear OS co-

nectado; también permitirá tomar fotos a través de un comando recibido en su característica ***Acciones***.

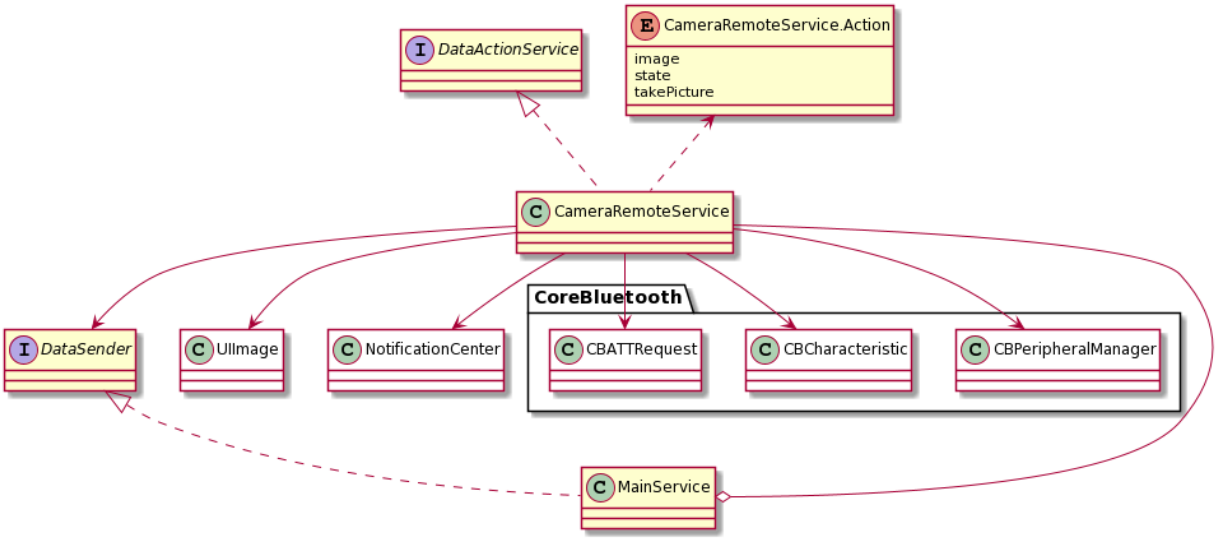

Figura 4.17: *iOS: Diagrama de clases del servicio de control remoto de cámara*

Wear OS

El **CameraRemoteServiceManager** se encarga de gestionar la suscripción a este servicio. Procesa los datos recibidos, formando imágenes a partir de los paquetes de datos recibidos, envía actualizaciones a la interfaz gráfica y construye los comandos para pedir actualizaciones del estado de la cámara o tomar fotos remotamente.

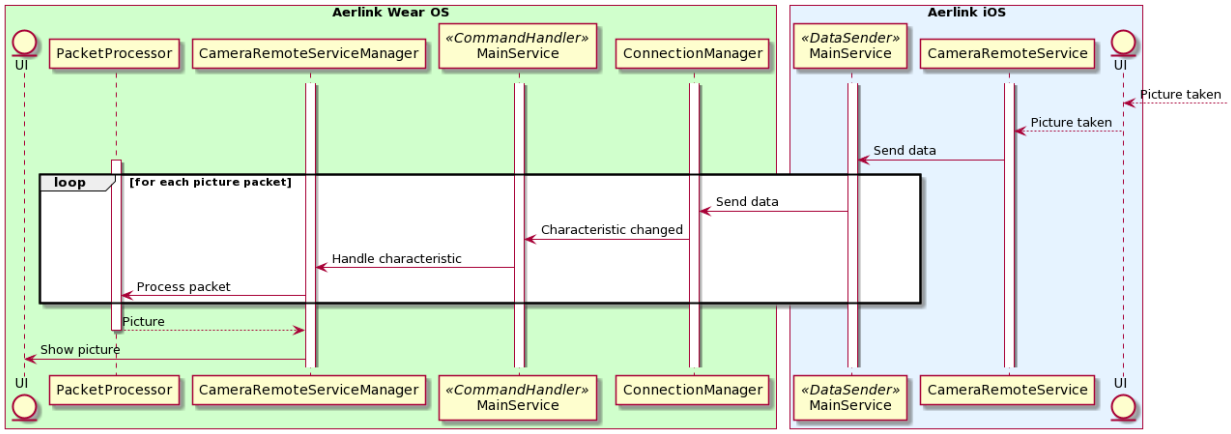

Figura 4.18: *Requisito funcional 8: Tomar foto*

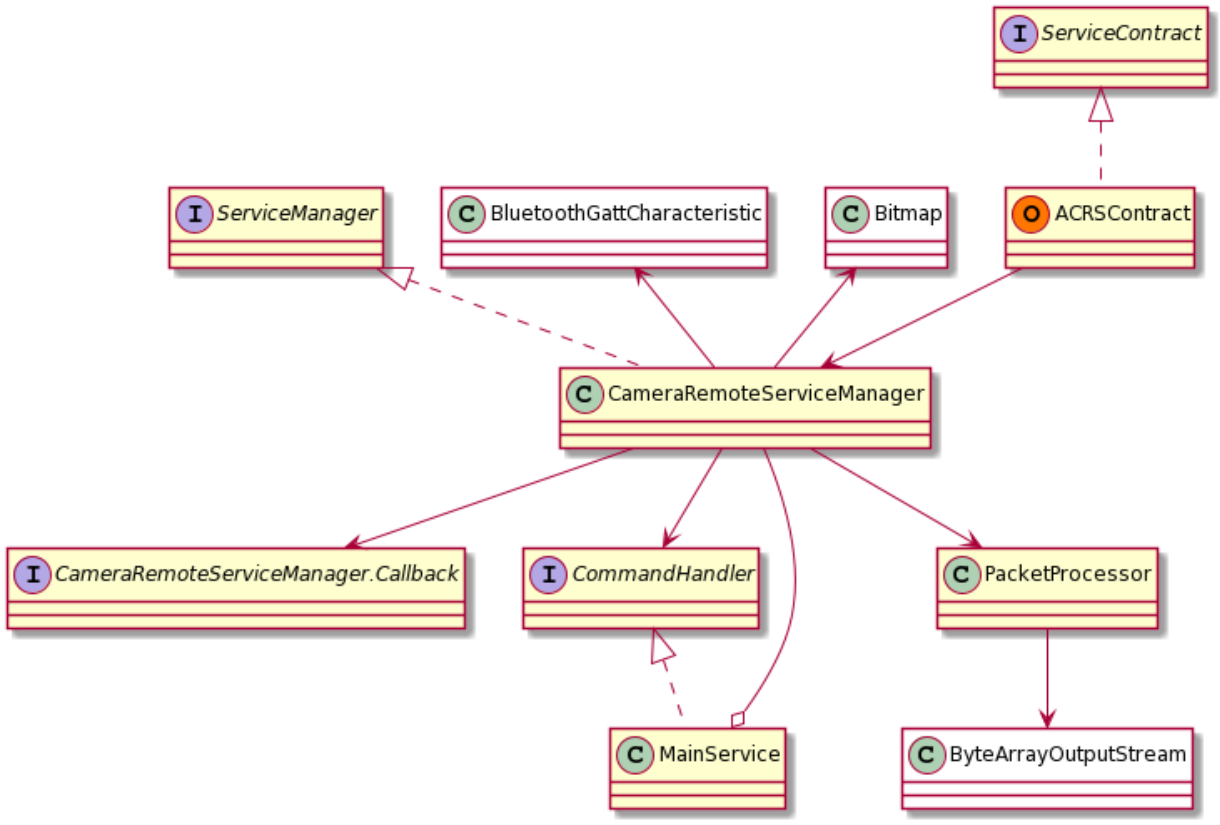

Figura 4.19: *Wear OS: Diagrama de clases del servicio de control remoto de cámara*

4.4.2. Recordatorios

Los dispositivos registrados a este servicio podrán pedir los listados de recordatorios y su contenido a través de la característica *Acciones*; recibiendo la respuesta a través de la característica ***Datos***. También podrán cambiar el estado de los recordatorios a través de la característica ***Acciones***.

iOS

En la aplicación iOS, el servicio **RemindersService** se encarga de lidiar con el framework *EventKit* para obtener datos de la app Recordatorios del dispositivo iOS, procesa estos datos para enviar solo los datos esenciales formateados en JSON y gestiona un caché de los datos ya enviados al dispositivo Wear OS para evitar reenviar datos ya disponibles en este.

Por otro lado, convierte los comandos recibidos en su característica ***Acciones*** en un efecto visible en los datos de recordatorios del dispositivo iOS.

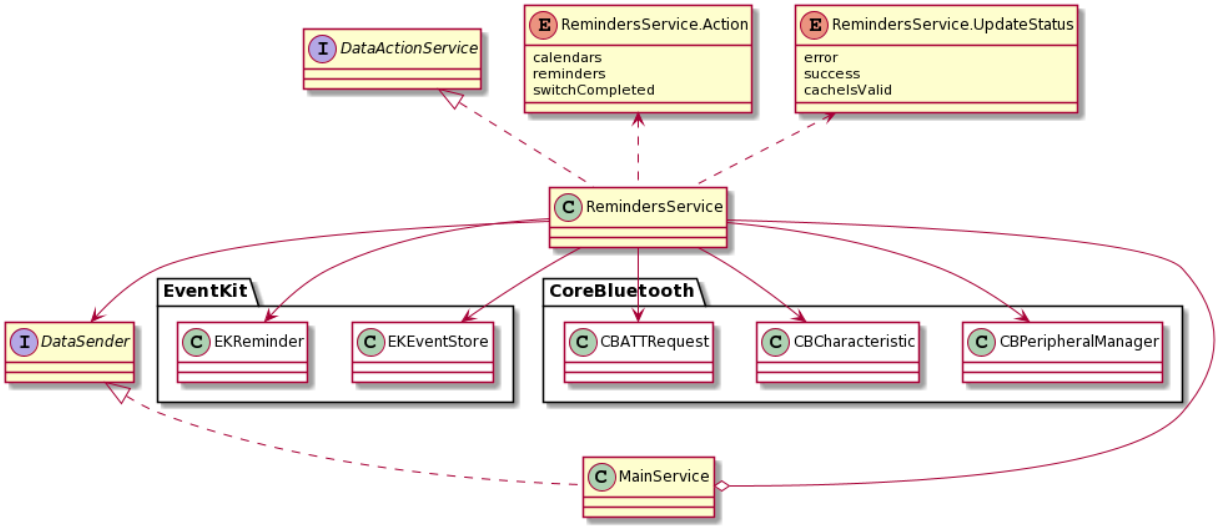

Figura 4.20: *iOS: Diagrama de clases del servicio de recordatorios*

Wear OS

El **RemindersServiceManager** se encarga de gestionar la suscripción a este servicio. Procesa los datos recibidos, parseando los datos recibidos y convirtiéndolos en datos procesables por la interfaz gráfica, permite también recibir acciones de está para convertir en comandos y modificar el estado de un recordatorio remotamente. Al igual que en iOS, este ServiceManager, gestiona un caché para poder evitar el reenvío de datos innecesario.

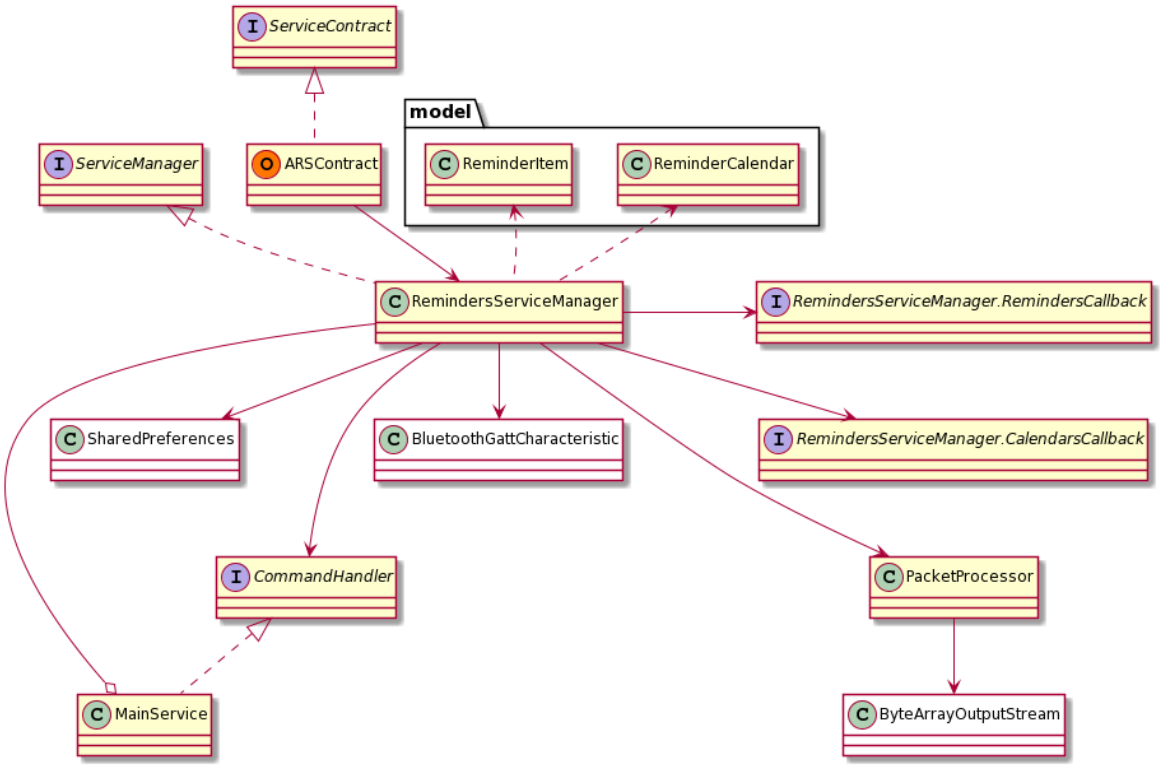

Figura 4.21: *Wear OS: Diagrama de clases del servicio de recordatorios*

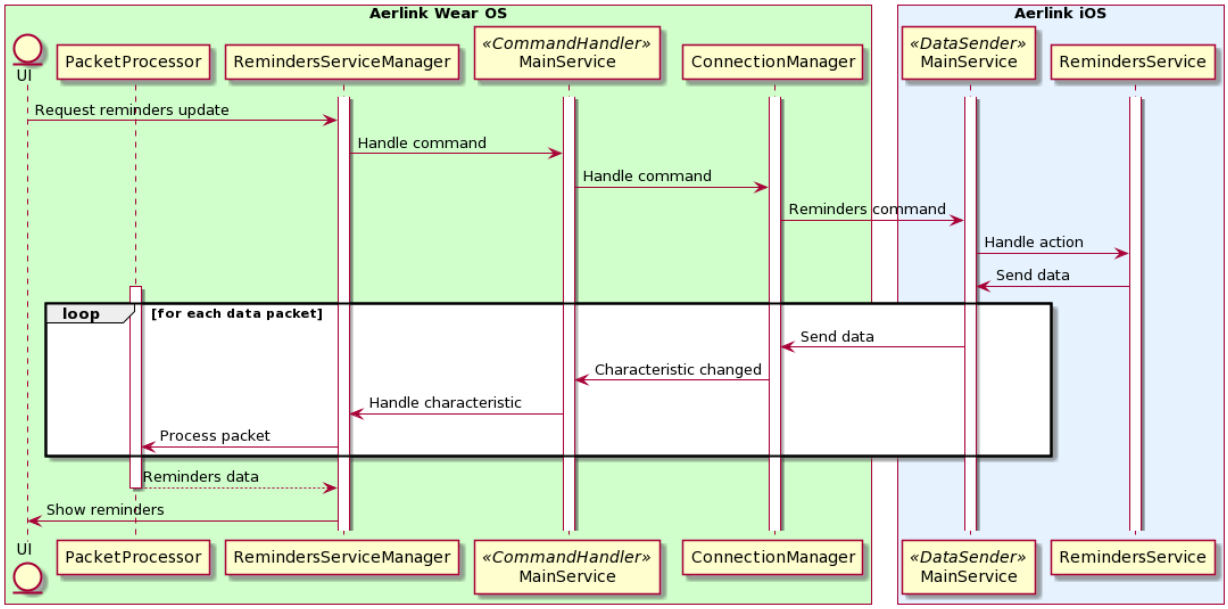

Figura 4.22: *Requisito funcional 11: Ver listado de recordatorios*

Capítulo 5

Experimentación

> Don't worry if it doesn't work right. If everything did, you'd be out of a job.
>
> *Mosher's Law of Software Engineering*

En este capítulo se presentan las pruebas realizadas en el sistema ya implementado y los resultados obtenidos.

Los *smartwatches* en los que se realizaran las pruebas de la aplicación Wear OS serán:

- **LG G Watch** con Android Wear 1.5.0 (Fig. 5.1).
- **Ticwatch E Express** con Wear OS 2.17 (Fig. 5.2).
- **Sony SmartWatch 3** con Android Wear 1.5.0 (Fig. 5.3).
- **Motorola Moto 360 V2 Sport** con Wear OS 2.35 (Fig. 5.4).

Figura 5.1: *LG G Watch*

Figura 5.2: *Ticwatch E Express*

Figura 5.3: *Sony SmartWatch 3*

Figura 5.4: *Motorola Moto 360 V2 Sport*

La aplicación iOS se instalará en un iPhone 11 Pro con iOS 13 (Fig. 5.5) que actuará de dispositivo iOS para todos los dispositivos Wear OS.

Por último, se utilizará un Nexus 5 con Android 6.0.1 para facilitar el despliegue y depuración de la aplicación Wear OS en varios dispositivos a la vez, este dispositivo es necesario ya que, a pesar de no realizar función alguna en el sistema Aerlink, hay algunos dispositivos Wear OS que solo permiten depuración por Bluetooth a través de un dispositivo Android enlazado.

Figura 5.5: *iPhone 11 Pro*

Figura 5.6: *Nexus 5*

5.1. Pruebas

5.1.1. Notificaciones

En la Fig. 5.7 se puede ver como una notificación recibida en el dispositivo iOS es también recibida y mostrada al usuario correctamente en los dispositivos Wear OS.

Figura 5.7: *Ejemplo de notificación. De izquierda a derecha: Moto 360, Ticwatch, iPhone 11 Pro, G Watch, SmartWatch 3.*

Las notificaciones pueden ser descartadas desde un dispositivo Wear OS, esto hace que desaparezcan también del centro de notificaciones del dispositivo iOS. Aunque según la documentación del servicio de notificaciones estas pueden venir acompañadas de una acción positiva y otra negativa, por lo general estas vienen solo con una acción negativa equivalente a descartar la notificación; en las pruebas solo se ha encontrado que al recibir una llamada, esta se recibe como una notificación cuya acción positiva es descolgar y cuya acción negativa es colgar, al tratarse de un caso especial este tipo de notificaciones se gestiona de forma especial mostrando una interfaz especifica.

5.1.2. Control multimedia

En la Fig. 5.8 se puede ver como la información multimedia del dispositivo iOS se muestra en los dispositivos Wear OS correctamente. El control funciona correctamente desde todos los dispositivos a pesar de tratarse de versiones diferentes de Wear OS.

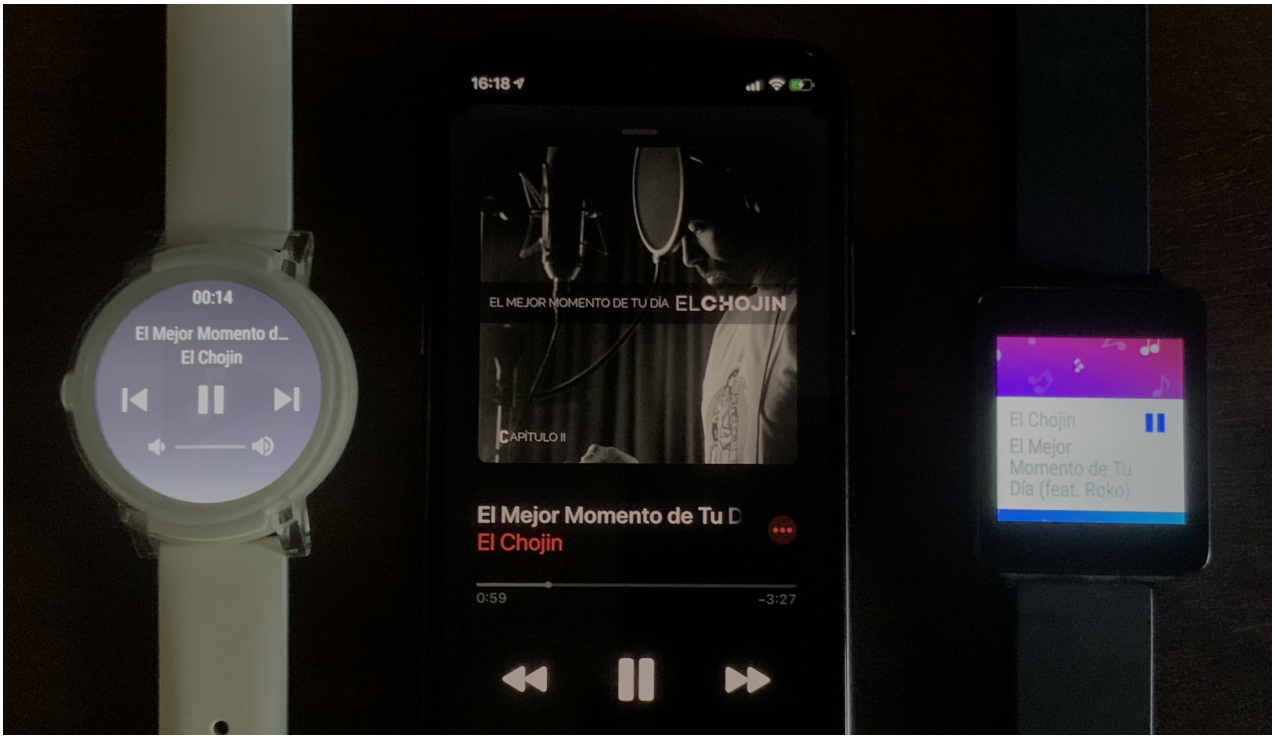

Figura 5.8: *Ejemplo de reproducción. De izquierda a derecha: Ticwatch, iPhone 11 Pro, G Watch.*

Se muestran correctamente los datos de título y artista, y se permite reproducir, pausar, avanzar pista, retroceder pista y controlar el volumen. En caso de tratarse de un título o nombre de artista largo este llega truncado, el servicio multimedia permitía pedir por separado los datos completos si estos llegaban truncados, en las pruebas esta funcionalidad no ha cumplido con la fiabilidad esperada llegando a mostrar datos incorrectos. Finalmente se ha decidido mostrar los datos truncados añadiendo "..." al final de estos.

5.1.3. Batería

En la Fig. 5.9 se puede ver que la información de la batería del dispositivo iOS es mostrada correctamente en los dispositivos Wear OS. Al ocurrir cambios en el nivel de la batería, estos cambios son mostrados correctamente en la interfaz de los dispositivos Wear OS.

Figura 5.9: *Nivel de batería. De izquierda a derecha: Moto 360, Ticwatch, iPhone 11 Pro, G Watch, SmartWatch 3.*

Se ha comprobado que al bajar el nivel de batería del 20 % se muestra una notificación en el *smartwatch* enlazado, además esta notificación viene acompañada de una vibración al ser el nivel de batería exactamente 20 %, 15 %, 10 % o 5 % siempre y cuando el nivel anterior haya sido superior, es decir, esta vibración no se dará en el caso de que el dispositivo este cargándose. Por último, la notificación desaparecerá al superar el nivel de batería el 20 %.

5.1.4. Control remoto de cámara

En la Fig. 5.10 se puede ver la interfaz de la aplicación Wear OS de control remoto de cámara antes y después de tomar una foto.

Figura 5.10: *Control remoto de cámara. De izquierda a derecha: Ticwatch, iPhone 11 Pro, G Watch.*

Al tomar una foto desde cualquier de los dispositivos Wear OS esta es recibida en ambos, la aplicación Aerlink en iOS no diferencia quien ha iniciado la acción de tomar foto, toma la foto y esta es enviada a todos los dispositivos conectados.

La calidad de la foto recibida en el *smartwatch* es bastante baja pero suficiente para cumplir el cometido de comprobar si el encuadre de la foto es correcto o si el objetivo de la foto ha salido correctamente. Esta funcionalidad se inspiró en la funcionalidad del Apple Watch que permite ver en tiempo real lo que está capturando la cámara del iPhone, en un principio se intentó replicar esta funcionalidad aunque fuese recibiendo imágenes cada X segundos; tras probar distintas soluciones se llegó a la conclusión de que requería un flujo de datos mucho mayor al que podía proporcionar BLE y se optó por recibir la imagen únicamente tras haberse completado la captura correctamente.

5.1.5. Recordatorios

En la Fig. 5.11 se puede ver la interfaz de la aplicación Wear OS de gestión de recordatorios. En el dispositivo Wear OS de la izquierda se observa el listado de listas replicando los datos del dispositivo iOS, en el de la derecha se observa el contenido de la lista "Compra".

Figura 5.11: *Gestión de recordatorios. De izquierda a derecha: Ticwatch, iPhone 11 Pro, G Watch.*

Se ha comprado que los colores de los listados mostrados en los dispositivos Wear OS son los mismos que se muestran en la aplicación Recordatorios de iOS. También se ha probado que funciona el modificar el estado de un recordatorio desde el *smartwatch*, este cambio se ve reflejado en tiempo real en la aplicación Recordatorios de iOS.

En caso de no tener una conexión establecida, tanto está aplicación como la de control remoto de cámara muestran un mensaje informando al usuario que necesitan establecer una conexión antes de poder continuar.

5.1.6. Conexión

Este es el apartado donde más pruebas se han realizado, siendo estas también las que más variabilidad han tenido en sus resultados. Se ha logrado establecer una conexión plena con todos los dispositivos Wear OS probados, pero también es cierto que el tiempo de conexión puede variar mucho entre un dispositivo Wear OS y otro, ya no solo por la versión de Wear OS si no también por el fabricante de este. Por ejemplo, algunos dispositivos pueden tener problemas al establecer la conexión llegando a requerir un reinicio de las funcionalidades Bluetooth del dispositivo, Aerlink realiza automáticamente este tipo de reinicio, pero esto llega a tener un impacto en el tiempo de conexión.

Para realizar un enlace BLE con otro dispositivo, iOS requiere que la aplicación Aerlink esté en primer plano. Mientras no tenga una conexión establecida, el dispositivo Wear OS informa al usuario que tiene que abrir la aplicación Aerlink en su dispositivo iOS para poder establecerla. Esto puede ser una molestia para el usuario pero no hay forma de evitarlo.

Para probar la reconexión se ha deshabilitado el Bluetooth en el dispositivo iOS, al rehabilitar el Bluetooth todos los dispositivos Wear OS han vuelto a encontrar al dispositivo iOS e intentado restablecer la conexión. También se ha probado a encerrar el dispositivo iOS en un microondas para simular una desconexión en la que el Bluetooth no es deshabilitado, en este caso ocurre lo mismo, al sacar el iPhone del microondas los dispositivos Wear OS son capaces de encontrarlo e intentan restablecer la conexión perdida. El éxito de está reconexión es similar al de la conexión normal, variando también de dispositivo a dispositivo.

Capítulo 6

Conclusiones y líneas futuras

> How does a project get to be a year late?... One day at a time.
>
> *Fred Brooks*

Aerlink es sin duda uno de los proyectos que más me ha aportado, he disfrutado mucho aprendiendo a trabajar con la tecnología Bluetooth a bajo nivel, intentando solucionar problemas de conexión en muchos dispositivos diferentes, teniendo como competencia directa una aplicación desarrollada por Google, trabajando en una interfaz que recuerde al Apple Watch pero que no se sienta fuera de lugar en Wear OS, viendo como la gente disfruta de un trabajo realizado por mí...

Por un lado, estoy satisfecho con el trabajo realizado. Se han cumplido los objetivos propuestos en la sección 1.2, Aerlink es capaz de establecer una conexión entre un dispositivo iOS y un dispositivo Wear OS, utilizando únicamente la tecnología BLE, ofreciendo las funcionalidades deseadas. Se cubren las funcionalidades básicas, como son la gestión de notificaciones recibidas o el control remoto de medios multimedia, y las funcionalidades exclusivas de Aerlink demuestran que, a pesar de los problemas planteados, es posible aumentar la utilidad de un dispositivo Wear OS emparejado con un dispositivo iOS.

Por otro lado, sé que este trabajo está lejos de ser perfecto. Es un poco descorazonador el estado del desarrollo BLE para plataformas Android e iOS. Como se ha mencionado previamente, el Bluetooth stack de Android es un desastre, con funcionalidades mal imple-

mentadas o códigos de error sin documentar. En iOS, las restricciones de Apple dificultan mucho el desarrollo de una aplicación como Aerlink.

Las faltas de iOS en este aspecto se hacen patentes en la aplicación de *Wear OS by Google*, demostrando que no importa si eres un gigante tecnológico como Google o un desarrollador solitario madrileño, te vas a enfrentar a las mismas dificultades. No cabe duda de que tener herramientas que no ofrece a sus competidores puede beneficiar a Apple, al no poder estos ofrecer funcionalidades similares a las de un Apple Watch; pero no queda tan claro que esto no perjudique al mercado de accesorios *wearable* en general, ya que alguien que esté desarrollando un accesorio tendrá que, o bien ignorar la plataforma iOS, o bien ofrecer un set de funcionalidades muy limitado, siendo ninguna de estas opciones ideal.

El futuro de las aplicaciones que añaden funcionalidad a los dispositivos móviles más allá de una interfaz visual es demasiado incierto. Android, la plataforma que tradicionalmente ha dado más libertad a los desarrolladores, se parece cada vez más a iOS, añadiendo limitaciones a los desarrolladores para proteger la seguridad y rendimiento de sus dispositivos. iOS a su vez se ha ido abriendo, ofreciendo herramientas hace unos años impensables, pero lejos queda aún la ejecución de servicios en segundo plano sin restricciones. Claramente, la seguridad y rendimiento de nuestros dispositivos es algo que nos interesa a todos, pero también está claro que, tanto usuarios como desarrolladores de software, no queremos un futuro de la computación dictado en exclusiva por los proveedores de sistemas operativos, un futuro en el que la única forma de obtener funcionalidad extra en tu dispositivo dependa de quien te lo ha vendido o en el que las únicas aplicaciones que podamos desarrollar sean para gestionar datos en un servidor.

A pesar de todo, espero que este proyecto le pueda resultar útil a alguien iniciándose en el desarrollo de funcionalidades BLE. Creo que merece la pena seguir trabajando con esta tecnología, ya sea en plataformas móviles o en cualquier otro tipo de dispositivo; es una tecnología con mucho futuro y aún estamos muy lejos de llegar a ver su verdadero potencial.

Por último, tengo muchas ganas de continuar mi trabajo en Aerlink. Sé que hay mucha

gente que ha elegido Aerlink como la forma principal de conexión con su *smartwatch* y sé que hay muchas más funcionalidades en las que se pueden trabajar para mejorar, aún más, el funcionamiento de este con un dispositivo iOS. Creo que el tracking de actividad física y salud puede generar mucho interés en los usuarios, así que, aprovechando que muchos relojes Wear OS en la actualidad vienen con un sensor de ritmo cardíaco y podómetro, el siguiente paso para Aerlink será conseguir obtener información de estos sensores y sincronizarlos con la aplicación Salud en el dispositivo iOS.

Capítulo 6

Conclusions and future lines

> How does a project get to be a year late?... One day at a time.
>
> *Fred Brooks*

Aerlink is without a doubt one of the projects that has given me more, I have enjoyed a lot learning how to deal with Bluetooth technology at a low level, trying to fix connection problems in a lot of different devices, having an app developed by Google as a direct competitor, working on an interface reminiscent of the Apple Watch but that does not feels out of place in Wear OS, seeing how people enjoy a product of my work...

On one side, I am satisfied with the job done. The goals proposed in section 1.2 have been reached, Aerlink is capable of establishing a connection, using only BLE technology, offering the desired functionalities. Basic functionalities, like notification management or remote media control, are covered, and Aerlink's exclusive functionalities prove that, despite of the problems raised, improving the utility of a Wear OS device paired with an iOS device is possible.

On the other side, I know the job is far from perfect. The state of BLE development for Android and iOS is a little disheartening. As previously mentioned, Android's Bluetooth stack is a mess, with badly implemented functionalities or undocumented error codes. On iOS, Apple's restrictions make the development of an application like Aerlink really hard.

iOS' faults in this regard are made patent on the *Wear OS by Google* application, showing that it does not matter if you are a tech giant like Google or a single developer from Madrid,

you will face the same difficulties. There is no doubt that having tools at your disposal that your competitors cannot have may benefit Apple, as the former will not be able to offer similar functionalities as an Apple Watch; but it is not so clear that this does not also harm the general market of wearable accessories, as someone developing an accessory will have to either ignore iOS platform, or offer a very limited set of functionalities, being neither of these ideal.

The future of applications that add functionality to mobile devices aside from a visual interface is unclear. Android, the platform that has traditionally given more freedom to developers, is looking more and more like iOS, adding limitations to their developers to protect the security and performance of their devices. iOS in its turn has been opening up, offering tools unthought-of years ago, but it is still far away from allowing background services unrestricted. Clearly, the security and performance of our devices is something that we all care about, but it is also clear that we, as users and as software developers, do not want a future of computing dictated exclusively by operating system providers, a future in which the only way of gaining functionality on your device depends on the one who sold it to you or in which the only applications that we can develop are to manage data from a server.

In spite of everything, I hope that this project can be useful to someone getting into the development of BLE functionalities. I think it is worth it to keep working with this technology, on mobile platforms or any other kind of device; it is a technology with a bright future and we are still far from seeing its true potential.

Finally, I am eager to keep working on Aerlink. I know there are many people that have chosen Aerlink as their main way of establishing a connection with their smartwatch and I know there are still many functionalities on which I can work on that would improve how it works with an iOS device. I think fitness and health tracking can generate a lot of interest, so, taking advantage of the fact that many Wear OS watches come equipped with a heart rate sensor and a pedometer, the next step for Aerlink will be obtaining information from

those sensors and syncing them with the Health app on iOS devices.

Glosario

BLE Bluetooth Low Energy. 10, 11, 14, 43, 45–47, 49, 50

GAP Generic Access Profile. 10

GATT Generic Attribute Profile. 10, 11, 24, 25

kbps kilobit por segundo. 14

Mbps Megabit por segundo. 14

MFi El programa Made For iPhone es un programa de certificación de Apple. Los accesorios deberán demostrar que han sido diseñados especificamente para iPhone y certificados por el desarrollador para cumplir los estándares de rendimiento de Apple.. 10, 14

MTU Maximum Transmission Unit o Unidad Máxima de Transferencia. 23, 26

UUID Universally Unique Identifier o Identificador Único Universal. 11, 24, 25

Bibliografía

[1] Jason Tsai. Global Smartwatch Shipments Projected to Reach 80.55 Million Units in 2020. *TrendForce*, 2019. URL https://www.trendforce.com/presscenter/news/20190925-10187.html.

[2] Jack Purcher. Apple Watch Dominates Global Smartwatch Market Share in Calendar Q1 by Delivering 4X Samsung Sales. *Patently Apple*, 2020. URL https://www.patentlyapple.com/patently-apple/2020/05/apple-watch-dominates-global-smartwatch-market-share-in-calendar-q1-by-delivering-4x-samsung-sales.html.

[3] Chaim Gartenberg. Google buys Fitbit for $2.1 billion. *The Verge*, 2019. URL https://www.theverge.com/2019/11/1/20943318/google-fitbit-acquisition-fitness-tracker-announcement.

[4] Google. Wear OS by Google. *App Store*, 2015. URL https://apps.apple.com/es/app/wear-os-by-google-smartwatch/id986496028.

[5] Especificación de Requisitos según el estándar de IEEE 830. URL https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf.

[6] Android Developers. *Service*, . URL https://developer.android.com/reference/android/app/Service.

[7] Android Developers. *BroadcastReceiver*, . URL https://developer.android.com/reference/android/content/BroadcastReceiver.

[8] Apple Developers. *CBPeripheralManager*, . URL https://developer.apple.com/documentation/corebluetooth/cbperipheralmanager.

[9] Apple. Apple Notification Center Service (ANCS) Specification. *Developer Documentation Archive*, . URL https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleNotificationCenterServiceSpecification/Introduction/Introduction.html.

[10] Apple. Apple Media Service Reference. *Developer Documentation Archive*, . URL https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleMediaService_Reference/Introduction/Introduction.html.

[11] Battery Service. *Bluetooth Technology Website*. URL https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Services/org.bluetooth.service.battery_service.xml.

[12] MFi Program. *Apple Developer*. URL https://developer.apple.com/programs/mfi/.

[13] Bluetooth Low Energy. *Bluetooth Technology Website*. URL https://web.archive.org/web/20170310111443/https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy.

[14] Robert Davidson, Akiba, Carles Cufí, Kevin Townsend. Getting started with bluetooth low energy. chapter 4. gatt (services and characteristics). *O'Reilly*. URL https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html.

[15] Jesús Macias. Bluetooth BLE: el conocido desconocido. *Solid Gear Group*. URL https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido/.

[16] Kevin Townsend. Introduction to Bluetooth Low Energy. *Adafruit Learning System*. URL https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt.

[17] Quotes for Software Engineers. URL https://www.comp.nus.edu.sg/~damithch/pages/SE-quotes.htm.

[18] Apple Developer. *Documentation*. URL https://developer.apple.com/documentation/.

[19] Android Developers. *Documentation*, . URL https://developer.android.com/docs.