UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Uso de smartwatches en entornos educativos: arquitecturas alternativas y protocolos

Antonio Díaz Escudero

Tutor: Juan Carlos Torrado Vidal

Ponente: Germán Montoro Manrique

Julio 2017

Uso de smartwatches en entornos educativos: arquitecturas alternativas y protocolos

AUTOR: Antonio Díaz Escudero
TUTOR: Juan Carlos Torrado Vidal

AmIlab
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2017

Resumen

Uno de los retos a los que se enfrenta una persona con un Transtorno del Espectro Autista es el de la regulación emocional, para ellos es muy difícil recuperar el estado de calma después de que un estímulo les haya provocado una situación de crisis. Actualmente la asistencia emocional a estas personas es llevada por terapeutas que dependen de la observación constante para la detección de estos eventos.

Para mejorar esta circunstancia y además proveer a estas personas de una mayor independencia, el AmIlab ha desarrollado una herramienta para Smartphone y Android Wear que permite al terapeuta crear regulaciones digitales que se muestran en el Smartwatch de la persona con TEA cuando los sensores de ritmo cardiaco detectan una situación de crisis.

Este trabajo de fin de grado pretende crear un protocolo de transmisión de datos para que el Smartphone pueda comunicarse con el Smartwatch y, además, modificar la arquitectura de la aplicación para dar soporte multiusuario y así ayudar al terapeuta a personalizar diferentes regulaciones para cada chico.

Palabras clave

TEA, autismo, autoregulación emocional, control emocional, Smartphones, Smartwatches, multiusuario, comunicación de datos.

Abstract

One of the challenges that people with an Autistic Spectrum Disorder face is to regulate their emotions, for them it's very difficult to regain a calm state after an stimulus has provoked a crisis situation. Currently the emotional assistence is provided by therapists that rely upon observation to detect this kind of event.

To improve this condition and in addition to provide these people with a greater independency, the AmIlab has developed a tool for Smartphone and Android Wear that allows the therapist to create digital regulations that appear in the Smartwatch of the person with ASD when the cardiac rythm sensors detect a crisis situation.

This bachelor thesis intends to create a data transmission protocol so that the Smartphone can comunicate with the Smartwatch, and also, modify the aplication arquitecture to offer multi-user support and thus help the therapist to customize different regulations for each person.

Keywords

ASD, autism, emotional self-regulation, emotional control, mobile application, Smartphones, Smartwatches, multi-user, data communication.

Agradecimientos

En primer lugar quiero dar las gracias a mi amigo y tutor Juan Carlos, por ofrecerme la oportunidad de trabajar con él y estar ahí siempre que le he necesitado durante todo el proyecto. También agradezco a Javi y a Germán su ayuda y aceptación, los tres me habéis hecho sentir en todo momento parte del equipo y me da una pena enorme que estas líneas vayan a ser lo último que escriba en el AmIlab.

No puedo pasar sin mencionar a mis amigos, los que me han apoyado desde la distancia: Christian, Jordi, Guillermo y Robert; los que aunque empezaron conmigo y luego se fueron, siempre estuvieron ahí para unas cervezas, como Jorge; y por supuesto los que han estado junto a mi hasta el final: Edu y Raed, no cambiaría las horas de estudio con vosotros por nada del mundo.

Y por último, evidentemente, a mis padres, a mi hermana y en general a toda mi familia, que han confiado en mi incluso cuando yo mismo tenía mis dudas.

A todos, gracias.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 TRASTORNO DEL ESPECTRO AUTISTA

El trastorno del espectro autista (TEA) es una condición neurológica que afecta a la manera en la que una persona se comporta y se comunica. Se le conoce como "espectro" porque engloba diversas características cognitivas, psicológicas y conductuales.

Las personas con TEA presentan una dificultad en el control de su comportamiento y la capacidad de entender y regular sus emociones de manera adecuada. Esto, sumado a las dificultades con las que se enfrentan a la hora de comunicarse, hace que la detección y la asistencia en situaciones de crisis emocional sean cruciales para conseguir una mejora sustancial de su independencia.

1.2 TAIMUN-WATCH

Con objetivo de lograr una autonomía aceptable en las personas con TEA, el **AMILab** ha desarrollado una aplicación que se sirve de la tecnología de los Smartphones y los Smartwatches para mejorar la detección y la asistencia de situaciones de crisis emocional.

Taimun-Watch consta de dos partes: la parte Smartphone también llamada herramientas de autor, que permite a la persona de apoyo utilizar su móvil para crear regulaciones emocionales compuestas de texto e imágenes que ayudan a la persona con TEA a recuperar un estado de calma tras una crisis emocional; y la parte Smartwatch, que utiliza los diferentes sensores del reloj inteligente para monitorizar la actividad y detectar la situación de crisis tras la cual se mostrará la regulación creada con la herramienta de autor. Los datos monitorizados por el smartwatch, pueden visualizarse de manera gráfica en la herramienta de autor.

Figura 1.1 Taimun-Watch (Smartphone)

Figura 1.2 Taimun-Watch (Smartwatch)

Una regulación se trata de una secuenciación de estrategias de control emocional, estas, a su vez, son un conjunto de pasos que pueden ser imágenes o texto. A continuación se muestra un ejemplo de una regulación sencilla.

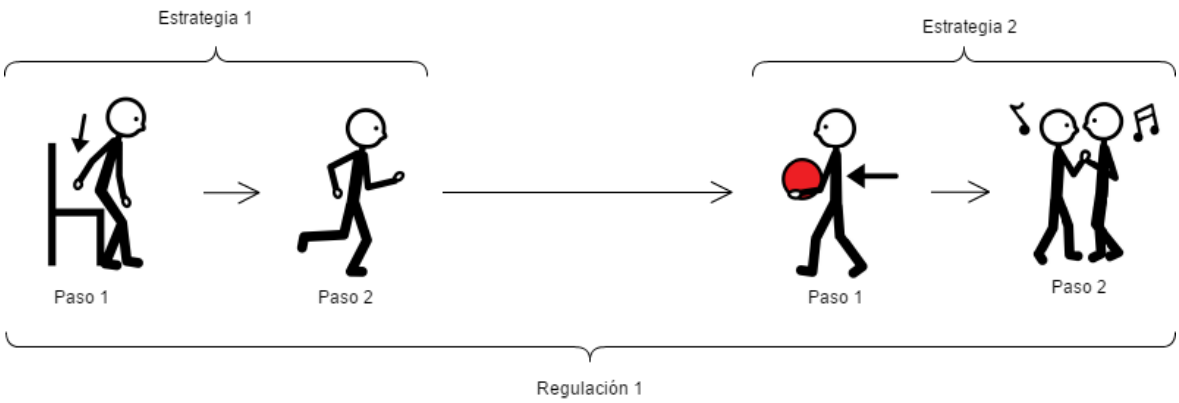


Figura 1.3 Regulación

Tanto las regulaciones del smartphone como los datos recogidos por el smartwatch se intercambian entre los dispositivos gracias al proceso de sincronización, en el que se abre una conexión *bluetooth* entre ambos para crear un canal de comunicación.

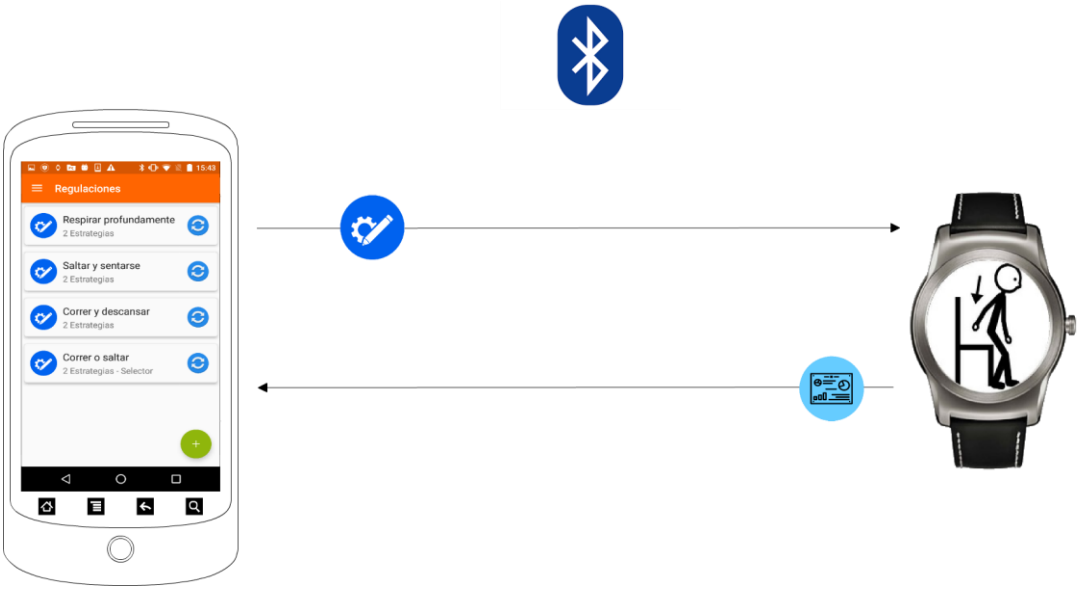


Figura 1.4 Conexión Taimun-Watch

Este proyecto se ha llevado a cabo con la colaboración del colegio de educación especial Alenta y ha sido seleccionado entre más de 150 propuestas recibidas por la Fundación Orange, en su convocatoria “Soluciones tencológicas aplicadas al autismo 2015”.

1.3 MOTIVACIÓN

Este proyecto parte del trabajo previo de desarrollo de las herramientas de autor y la parte de smartwatch de Taimun-Watch. Responde a la necesidad de integrar la comunicación entre ambas partes, para el envío de los datos correspondientes a las regulaciones y los ficheros que contienen la información relacionada con la monitorización del alumno.

Además, debido al modelo de aula que se sigue en el colegio Alenta, en el que un profesor se encarga de un grupo reducido de alumnos, nace la necesidad de que la herramienta de soporte a varios usuarios, ofreciendo así la posibilidad de que se pueda sincronizar diferente contenido de un solo smartphone con varios smartwatches.

Se propone realizar una modificación horizontal de la herramienta, integrando un protocolo de comunicación asimétrico para el envío y la recepción de datos, así como una modificación vertical del sistema que aumente su escalabilidad, realizando los cambios necesarios en la arquitectura para ofrecer soporte multiusuario.

1.4 OBJETIVOS

Con el fin de conseguir una comunicación robusta entre el Smartphone y el Smartwatch, y proveer a la herramienta Taimun-Watch de la funcionalidad multiusuario, se han establecido lo siguientes objetivos:

1. **Envío y recepción de regulaciones:** la aplicación debe almacenar de forma temporal las regulaciones creadas en diferentes ficheros para enviarlos posteriormente al Smartwatch.

2. **Envío y recepción de datos biométricos:** el smartwatch debe de ser capaz de transmitir todos los datos recogidos durante la sesión activa de la aplicación.

3. **Gestión de los usuarios:** el sistema debe ofrecer herramienta de creación, modificación y borrado de usuarios de una manera intuitiva.

4. **Selección de usuario activo:** debe poder elegirse el usuario activo de la aplicación de manera que este será con el que se realicen las sincronizaciones.

5. **Persistencia de la información de los usuarios:** la aplicación debe guardar la información relacionada con los usuarios en su base de datos.

6. **Información personalizada de cada usuario:** se debe mostrar la información biométrica recogida del usuario activo en ese momento.

1.5 ORGANIZACIÓN DE LA MEMORIA

La memoria consta de los siguientes capítulos:

1. Introducción: en este capítulo se da una visión general de los problemas que han motivado la realización de este trabajo, se resume el trabajo previo sobre el que se basa este proyecto y se lista los objetivos buscados.

2. Tecnología a utilizar: este apartado contiene información sobre la tecnología con la que se ha desarrollado el proyecto.

3. Diseño: en este bloque se dan detalles sobre la arquitectura de la aplicación, mostrando los diagramas tanto de clases como relacionales para la base de datos.

4. Desarrollo: en esta sección se detallan aspectos de la implementación de las distintas funcionalidades de la aplicación.

5. Pruebas: en este apartado se describen las pruebas realizadas para la evaluación de la funcionalidad de la aplicación.

6. Conclusiones y trabajo futuro: se tratará de dar una conclusión general del proyecto y se señalarán las posibles vías que se pueden tomar para seguir ampliando el proyecto.

2 TECNOLOGÍA A UTILIZAR

2.1 DISPOSITIVOS WEARABLE Y SMARTWATCH

Para el desarrollo de Taimun-Watch era importante elegir un dispositivo con sensores capaces de registrar los datos biométricos del usuario.

Un *wearable*, es un dispositivo inteligente que se lleva sobre, por debajo o incluido en la ropa como si se tratase de un complemento. Estos dispositivos deben de ser capaces de funcionar normalmente sin interferir en la actividad ordinaria del usuario. Pueden estar recogiendo información de manera continuada y funcionar independientemente, aunque lo normal es que se comuniquen con un smartphone enviando y recibiendo datos.

Cuando hablamos de *wearables*, es importante distinguir entre dos tipos: los no comerciales y los comerciales. Los dispositivos **no comerciales** son aquellos creados en un laboratorio para desempeñar una tarea específica. Estos *wearables* suelen tener características que los hacen destacar, como su tamaño y su forma. Por otro lado, los dispositivos **comerciales**, son aquellos diseñados para alcanzar el máximo número de consumidores posibles. A la larga, esto hace que nos acostumbremos a su uso y no lo veamos como algo fuera de lo común.

A la hora de diseñar herramientas para el soporte de personas con necesidades especiales, hay que tener en cuenta que esta no cause diferenciación y estigmatización, ya que eso puede provocar un rechazo de la persona hacia el dispositivo.

El componente normalizador de los *smartwatches* comerciales es el motivo por el que fueron elegidos para el desarrollo de Taimun-Watch.

2.1.1 Android Wear

Android Wear es la versión oficial del sistema operativo Android de Google diseñada para *smartwatches* y otros *wearable*. Además de tener compatibilidad con un gran número de dispositivos, tanto Android como iOS, es soportado por muchos fabricantes de tecnología, entre ellos: Samsung, Motorola, LG, HTC, ASUS, Broadcom, Intel, Broadcom… Esto hace que haya un gran rango de precios entre los que elegir, resultando más económicos que dispositivos de otras marcas, como por ejemplo Apple Watch.

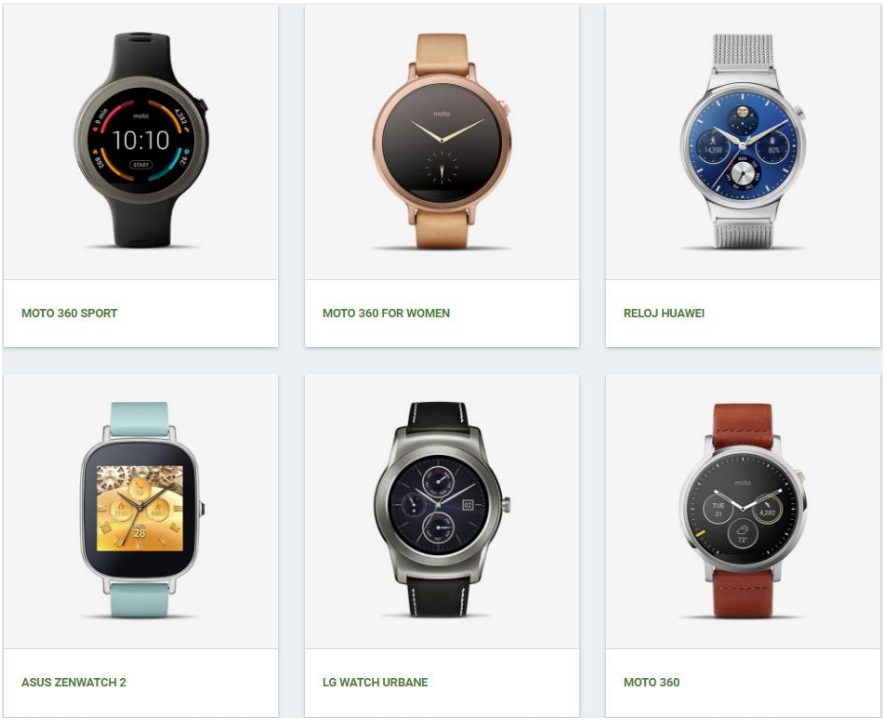


Figura 2.1 Smartwatches Android Wear

2.2 GOOGLE PLAY SERVICES

Google Play Services es un paquete de servicios de Google que provee a las aplicaciones de diferentes funcionalidades. Permite aprovechar las propiedades de *Maps*, Juegos, Google+, *Google Cloud Messaging* y *Android Wear* entre otros y también ofrece el uso de características a nivel de sistema, como un proveedor de seguridad dinámica que permite implementar OpenSSL, ayudando así a las aplicaciones a beneficiarse de parches regulares de seguridad independientes de las actualizaciones de Android OS.

La plataforma de Android es la que nos va a ofrecer las APIs necesarias para implementar la conexión entre dispositivos.

2.2.1 Android Wear Network

Dentro de los servicios de Google y en particular de la plataforma Android Wear, se utiliza el concepto de Android Wear Network para referirse a la red de nodos que incluye tanto dispositivos *wearables* como *smartphone*. Estos dispositivos “aparecen” y “desaparecen” de la red de manera dinámica a medida que se conectan y desconectan entre ellos. Al ser dispositivos portátiles, a diferencia de los ordenadores que están conectados de manera estática en redes cableadas, sus conexiones son más frágiles y están ligadas a factores como la batería o la distancia entre los dispositivos.

Por ejemplo, si llevas puesto un *smartwatch* y te alejas del escritorio en el que se encuentra tu *smartphone* es probable que la conectividad entre ambos se pierda a partir de cierto rango. Esto hace que sea necesario ofrecer a los nodos funcionalidades para la detección de otros nodos, el envío de mensajes y el intercambio de datos.

2.2.2 Wearable Data Layer

La Wearable Data Layer, también conocida como Android Wear API es uno de los servicios que ofrece Google a los desarrolladores en su paquete de Google Services. Provee de un canal de comunicación a las aplicaciones entre los dispositivos smartphone y smartwatch así como funcionalidades para el intercambio de información.

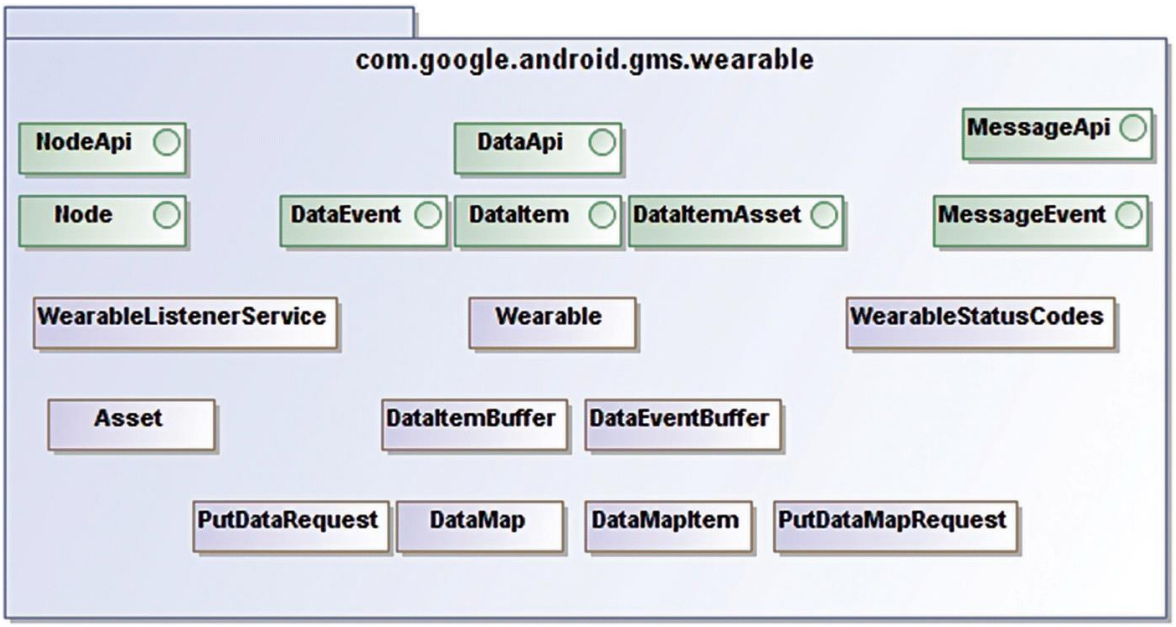


Figura 2.2 Visión general del paquete Wearable de la API Android Wear.

A alto nivel la Data Layer está formada por otras tres APIs: la Message API, la Node API y la Data API.

- Node API: da soporte a la detección de los nodos según se conectan y desconectan de la Android Wear Network. Cuando se detecta alguno de estos eventos, informa a la aplicación Wear que haya registrado los escuchadores de esta interfaz.

- **Data API:** esta interfaz provee a la aplicación de la habilidad de sincronizar datos a través de todos los dispositivos de una red Android Wear. Fundamentalmente, provee a la aplicación de las funcionalidades de *put* (insertar) y *get* (recibir) para leer y escribir datos a la Android Wear Network.

- **Message API:** da soporte al envío y recepción de mensajes cortos entre instancias de la misma aplicación que residen en diferentes nodos de la Android Wear Network. Un mensaje es privado de la aplicación que lo genera, y solo puede ser recibido por otra instancia de la misma aplicación.

A su vez, estas APIs contienen un set de objetos de datos que el sistema puede sincronizar. Además, también consta de escuchadores que notifican a la aplicación cuando ocurren ciertos sucesos.

3 ANÁLISIS

3.1 INTRODUCCIÓN

Este proyecto se ha llevado a cabo con el apoyo y la aprobación del colegio de educación especial Alenta. Principalmente, se han mantenido reuniones con su directora, pero también han participado otros educadores.

Estos profesionales, a la hora de utilizar la aplicación Taimun-Watch, expresaron la necesidad de poder comunicar su smartphone con varios smartwatches para la gestión y personalización de las regulaciones de varios de sus alumnos.

Gracias a estas reuniones, se han ido educiendo los requisitos de la aplicación de forma iterativa. Por lo tanto, este proyecto sigue un ciclo de vida iterativo e incremental, y debido a que va a ser utilizado por personal no técnico, con un diseño de software centrado en el usuario.

3.2 REQUISITOS

Para conseguir realizar la integración de las dos partes de la aplicación y la comunicación uno a muchos a través de un protocolo asimétrico se tienen que cumplir los siguientes requisitos:

3.2.1 Requisitos funcionales

- **RF 1. Sistema operativo:** la nueva versión de la aplicación debe de ser compatible para smartphones y smartwatches con sistema operativo Android con la versión de SDK 20 y sensor de ritmo cardiaco.

- **RF 2. Sincronización de regulaciones:** debe permitir el envío de una regulación creada con el smartphone al smartwatch del usuario activo de la aplicación.

- **RF 3. Sincronización de registros:** el smartphone debe de poder recibir los datos biométricos del usuario activo recogidos por el smartwatch.

- **RF 4. Creación de usuarios:** se deberá proporcionar una funcionalidad para añadir usuarios nuevos al sistema.

- **RF 5. Guardado de usuarios:** la aplicación debe de ofrecer la opción de guardar los usuarios creados.

- **RF 6. Borrado de usuarios:** debe de existir la opción de eliminar un usuario almacenado en la aplicación.

- **RF 7. Nombrado de usuarios:** el sistema debe de permitir elegir un nombre que sirva de identificación a un usuario dentro de la aplicación.

- **RF 8. Lista de smartwatches:** debe de mostrarse una lista con los dispositivos Wear conectados en ese momento a la Android Wear Network.

- **RF 9. Selección de smartwatch:** tiene que existir una funcionalidad para emparejar un usuario creado con un dispositivo conectado a la Android Wear Network.

- **RF 10. Edición de usuario:** la aplicación debe ofrecer la posibilidad de editar los datos de un usuario creado con anterioridad.

- **RF 11. Selección del usuario activo:** debe de poder elegirse el usuario con el que se quiere sincronizar el contenido.

- **RF 12. Mostrado de usuario activo:** la aplicación debe de mostrar en todo momento el usuario activo.

- **RF 13. Listado de usuarios:** la herramienta ofrecerá la opción de visualización de todos los usuarios existentes.

- **RF 14. Guardado al salir:** el sistema debe de preguntar si se desea guardar un usuario al salir del menú de edición.

- **RF 15. Confirmación de borrado:** la aplicación debe de mostrar un mensaje de confirmación de borrado antes de eliminar al usuario de manera definitiva.

3.2.2 Requisitos no funcionales

- **RNF 1. Conversión de regulaciones:** la herramienta debe de tener un sistema que convierta las regulaciones en distintos ficheros para facilitar su envío al Smartwatch.

- **RNF 2. Idioma:** las nuevas funcionalidades de esta versión de la aplicación deben de estar disponibles en inglés y catalán aparte de en castellano.

- **RNF 3. Mensajes de información:** la aplicación debe de mostrar mensajes que indiquen que las operaciones de sincronización se están realizando.

- **RNF 4. Interfaz:** las nuevas funcionalidades contarán con interfaces gráficas bien formadas y coherentes con el diseño del resto de la aplicación.

- **RNF 5. Operatividad:** la nueva versión de la aplicación tiene que seguir siendo fácil de comprender y manejar para el personal no familiarizado.

- **RNF 6. Escalabilidad:** la arquitectura multiusuario debe de estar construida de manera que en un futuro puedan añadirse con facilidad nuevos requerimientos y funcionalidades.

4 DISEÑO

A la hora de realizar el diseño de la aplicación se ha separado el trabajo en dos partes: la primera detalla los cambios que se han hecho a la arquitectura de la versión anterior para cumplir todos los requisitos de la funcionalidad multiusuario, mientras que la segunda se centra en los detalles del diseño del nuevo protocolo de comunicación frente al estándar de Android Wear. La mayor parte de las modificaciones de diseño se han hecho en la herramienta de autor, por lo que este capítulo se centra en esa parte de la aplicación.

4.1 MULTIUSUARIO

En la anterior versión de la aplicación Taimun-Watch existía el concepto de usuario, pero hacía referencia a la persona que usaba la herramienta de autor (la parte smartphone de la aplicación) en vez de a la persona que llevaba puesto el smartwatch. El usuario tenía la opción de configurar su nombre y el tipo de reloj, así como añadir una fotografía para que luego se mostrase en la pantalla de perfil.

Tras varias reuniones con el personal docente del colegio Alenta, se tomó la decisión de que no era necesario representar en la aplicación al usuario de la herramienta de autor, sin embargo, sí que debía de existir una representación del chico que llevaba puesto el smartwatch.

Estos nuevos requisitos se añadieron a la hora de implementar la funcionalidad de multiusuario dando lugar a diferentes cambios en la arquitectura existente.

4.1.1 Diagrama de clases

Con respecto al modelo de datos de la aplicación, uno de los cambios significativo es la modificación de la clase **Usuario** para que cumpla los nuevos requisitos. En la siguiente figura se muestra una comparativa de ambas versiones.

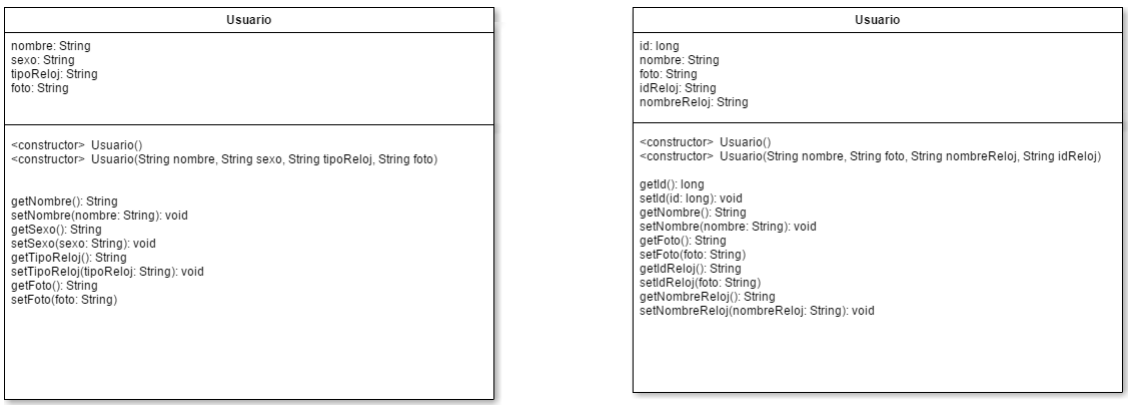


Figura 4.1 Comparación de la clase Usuario de la primera versión (izquierda) con respecto a la nueva (derecha).

En la nueva versión se han eliminado los atributos correspondientes a la configuración del sexo y el tipo del reloj, se ha añadido un atributo **id** que almacena un identificador único del usuario y dos atributos **idReloj** y **nombreReloj** que sirven para identificar en la Android Wear Network el dispositivo smartwatch asociado con el usuario. El conjunto de funciones de la clase ha sido modificado únicamente para eliminar y añadir los *getters* y *setters* correspondientes a los atributos modificados.

La clase **Regulacion** también se ha modificado para añadir las funciones y atributos necesarios para convertir una regulación en ficheros transmisibles por el canal de comunicación. Los cambios realizados pueden verse a continuación.

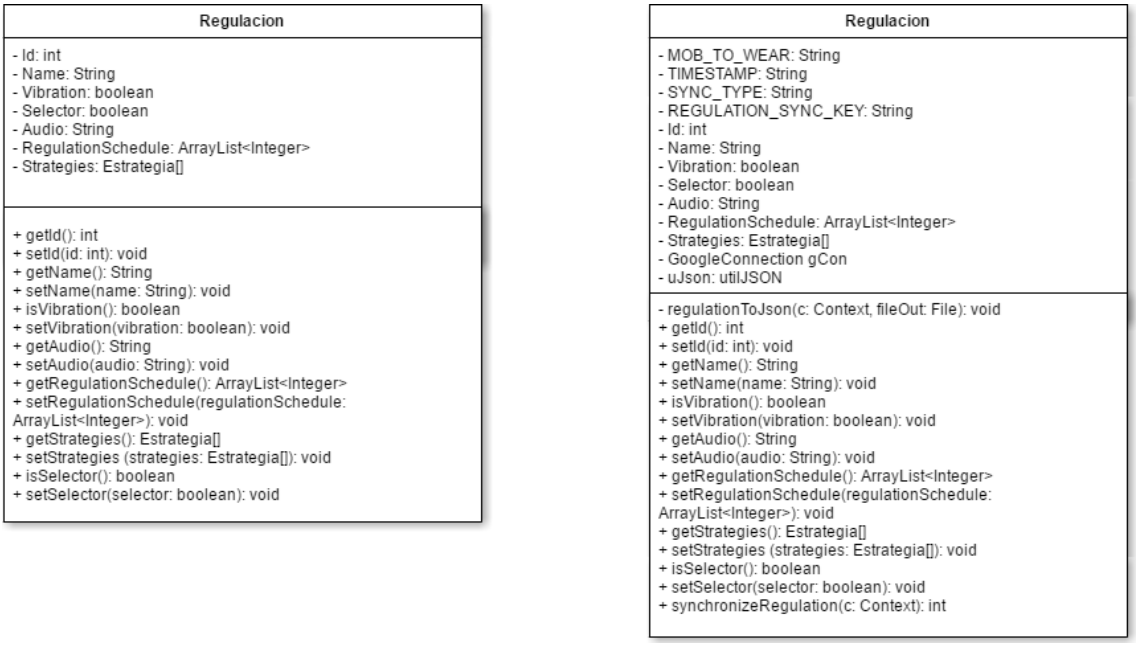


Figura 4.2 Comparación de la clase Regulación de la primera versión (izquierda) con respecto a la nueva (derecha).

Además, se han añadido nuevas clases para representar las actividades y los fragmentos de la lógica de la funcionalidad multiusuario y para la implementación del protocolo de comunicación. La relación de estas queda reflejada en las figuras 4.3 y 4.4.

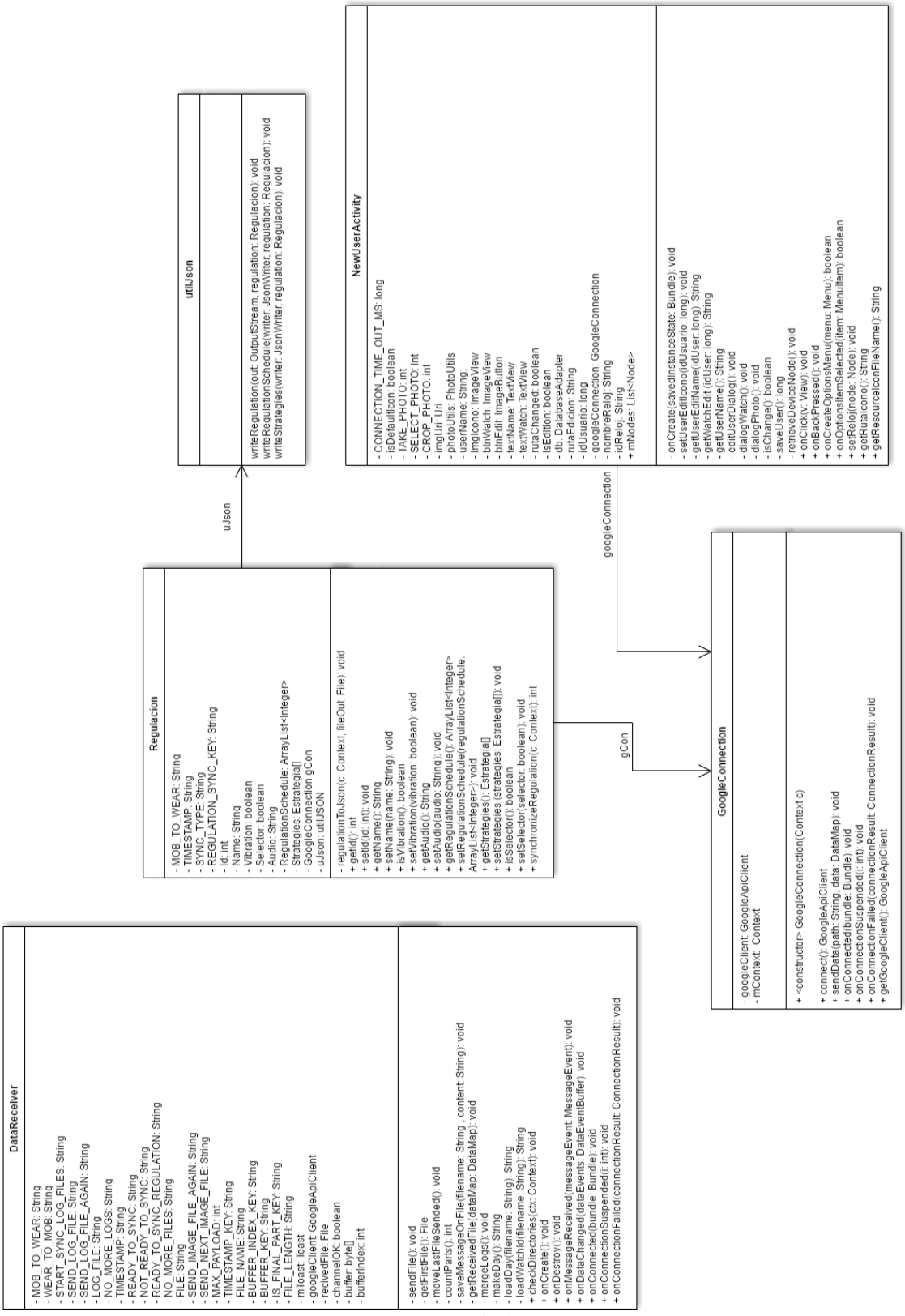


Figura 4.3 Diagrama de clases. Parte I

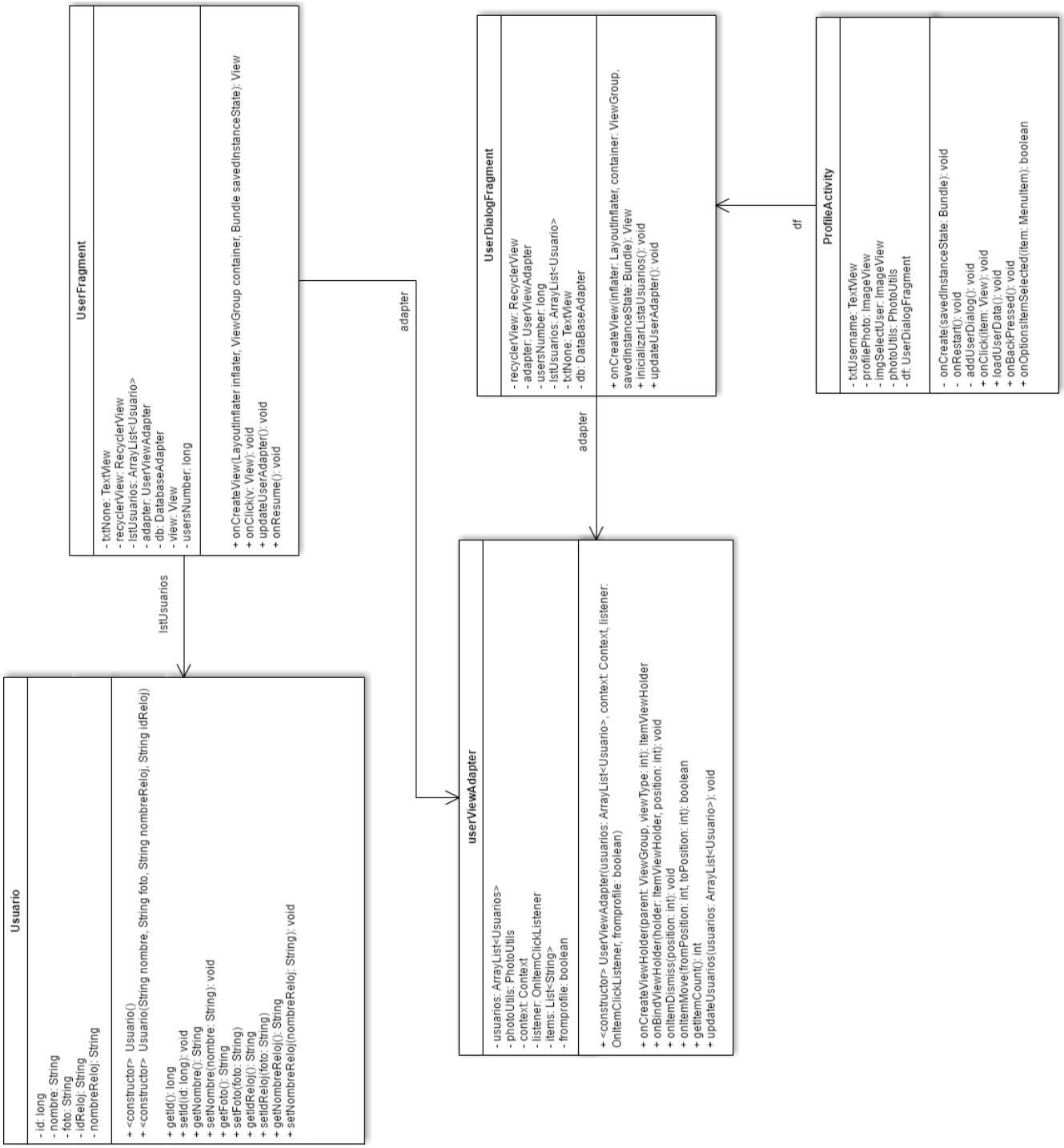


Figura 4.4 Diagrama de clases. Parte II

- **DataReceiver:** esta clase implementará las funciones y los Listeners necesarios de la API de Android Wear para permitir la comunicación entre dispositivos. Permitirá enviar y recibir datos y mensajes de la DataLayer.

- **GoogleConnection:** esta clase contendrá las funciones necesarias para realizar la conexión con las APIs de Google, en particular con la API de Android Wear.

- **utilJson:** implementará las funciones de conversión de una Regulación a un fichero JSON. Será referenciada por la clase Regulacion.

- **NewUserActivity:** esta clase contiene la lógica de la actividad de nuevo usuario que permitirá crear y editar un usuario dentro de la aplicación. Incluirá un atributo de la clase GoogleConnection para conectarse a la API y poder mostrar los dispositivos conectados a la Android Wear Network.

- **UserViewAdapter:** esta clase permitirá crear una vista con el listado de usuarios creados.

- **UserFragment:** contendrá la lógica del fragmento de Usuarios. Incluirá un atributo de tipo UserViewAdapter que le permitirá mostrar los usuarios existentes en una lista. También implementará la funcionalidad de borrado de usuarios.

- **UserDialogFragment:** permitirá mostrar un diálogo con la lista de usuarios creados.

- **ProfileActivity:** esta clase contendrá la lógica de la actividad de usuario activo, referenciará a la clase UserDialogFragment para poder mostrar los usuarios en una ventana de diálogo.

4.1.2 Prototipos

A continuación, se muestran los prototipos de las ventanas correspondientes a las actividades y fragmentos añadidos a la nueva versión de la aplicación.

- **Drawer Menu:** en el menú desplegable de la aplicación se requiere añadir la sección de usuarios, así como cambiar el botón de perfil de usuario para adaptarlo a la selección de usuario activo para cumplir con el requisito **RF 12**. En la figura 4.5 se muestra una comparativa del nuevo prototipo con respecto a la ventana de la primera versión.

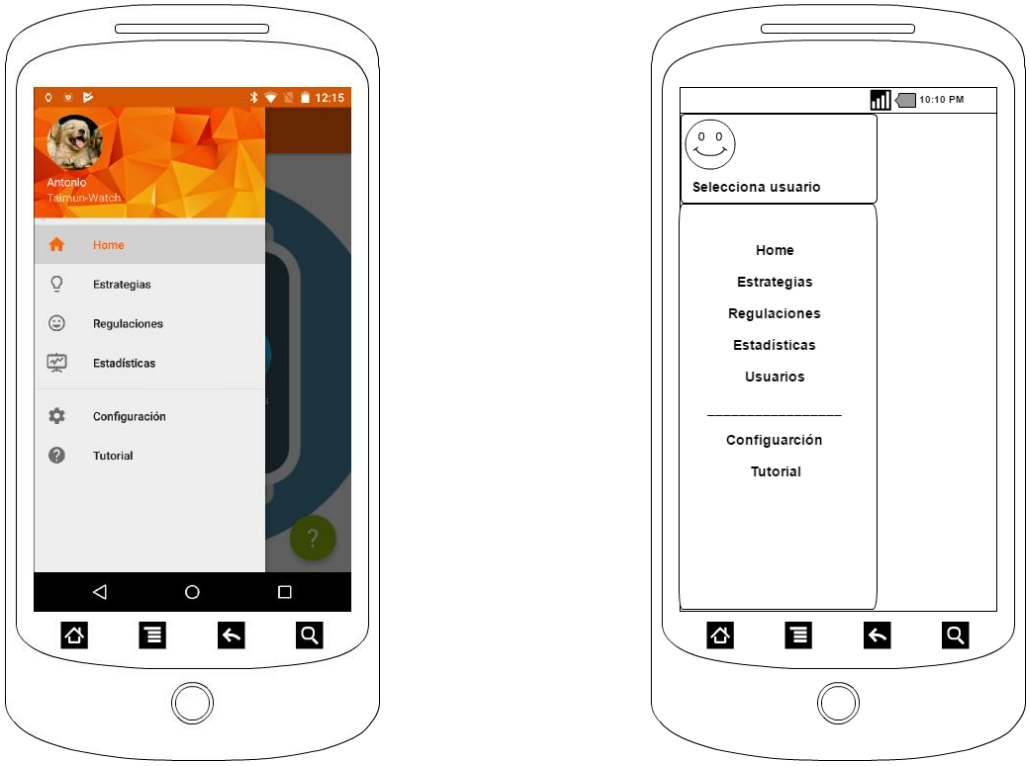


Figura 4.5 Comparativa Drawer Menu

- **Usuarios:** esta ventana mostrará tarjetas de los usuarios creados también permitirá borrarlos arrastrando las tarjetas fuera de la pantalla satisfaciendo los requisitos **RF 13 y RF 6**.

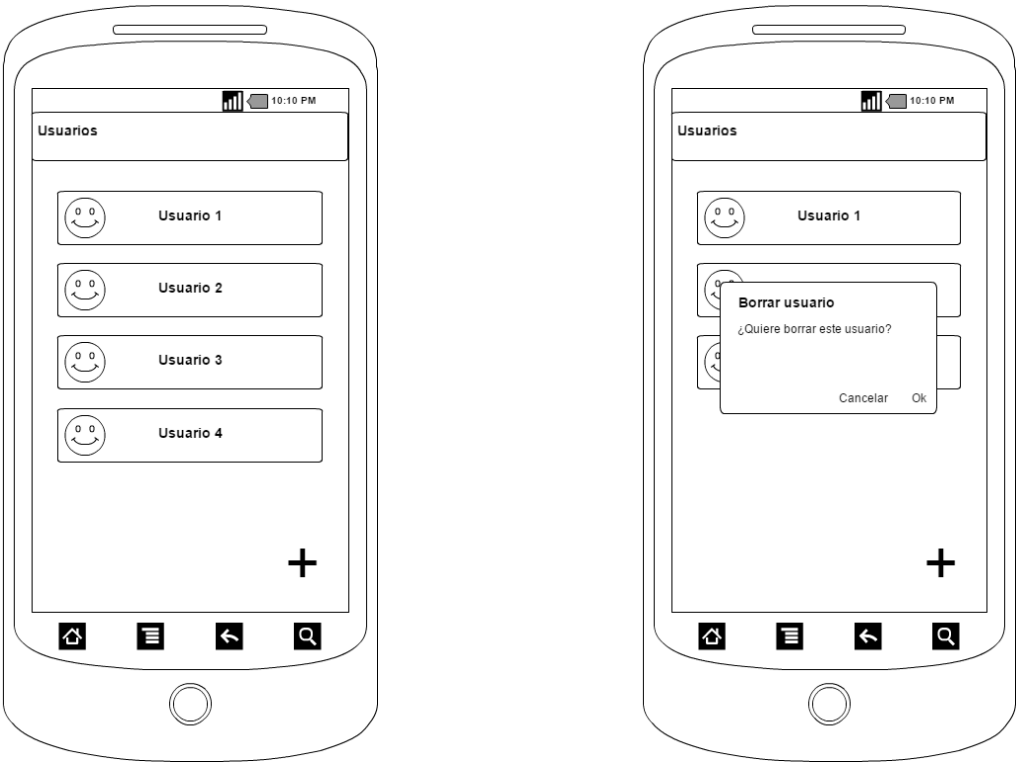


Figura 4.6 Prototipo Lista y Borrado de usuarios

- **Nuevo usuario:** esta ventana debe permitir crear un usuario nuevo con todos los parámetros necesarios para satisfacer los requisitos **RF 4**, **RF 7** y **RF 9.**

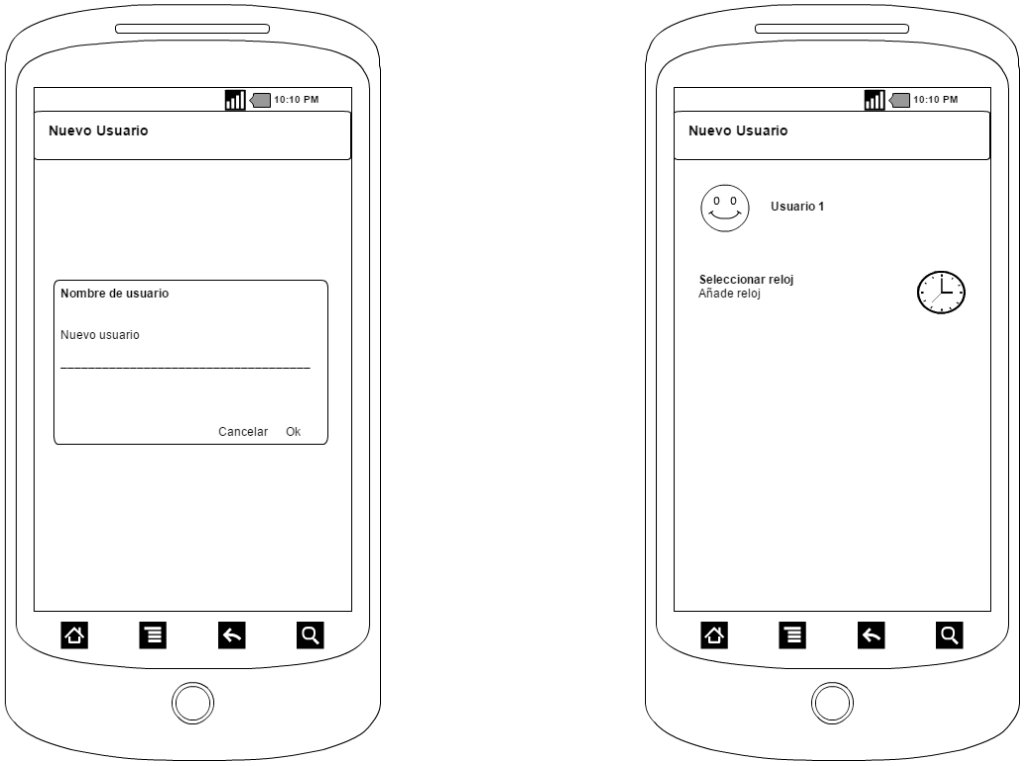


Figura 4.7 Prototipo Nuevo usuario

- **Editar usuario:** si se pulsa sobre una de las tarjetas de la lista, se abrirá la pantalla de edición de usuario, que permite modificar la configuración de un usuario existente cumpliendo con el requisito **RF 10**.

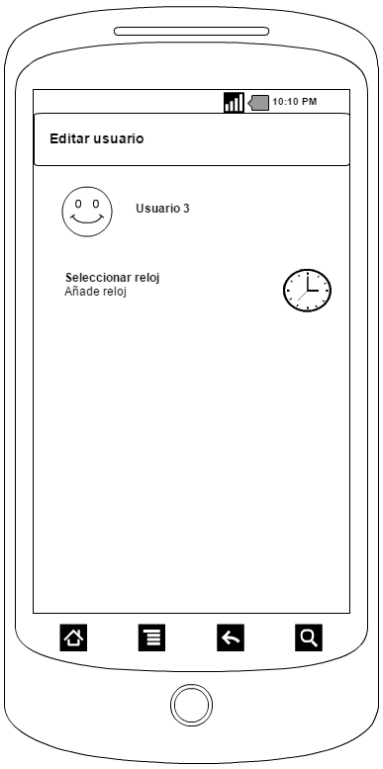


Figura 4.8 Prototipo Editar usuario

- **Lista Smartwatches:** como se especificaba en el requisito **RF 8**, a la hora de crear el usuario se debe de mostrar una lista con los dispositivos wearable conectados en ese momento a la Android Wear Network. En la siguiente figura se muestra un prototipo de esa ventana.

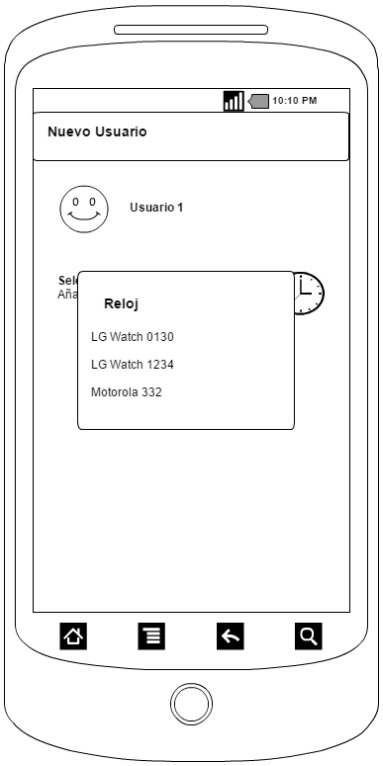


Figura 4.9 Prototipo Lista Smartwatches

- **Usuario activo:** esta pantalla sustituirá a la pantalla de perfil de la versión anterior de la aplicación. Debe mostrar al usuario activo de la aplicación así como ofrecer la posibilidad de seleccionar a otro usuario para satisfacer el requisito **RF 11.** En la siguiente figura se muestra una comparativa entre las dos versiones.

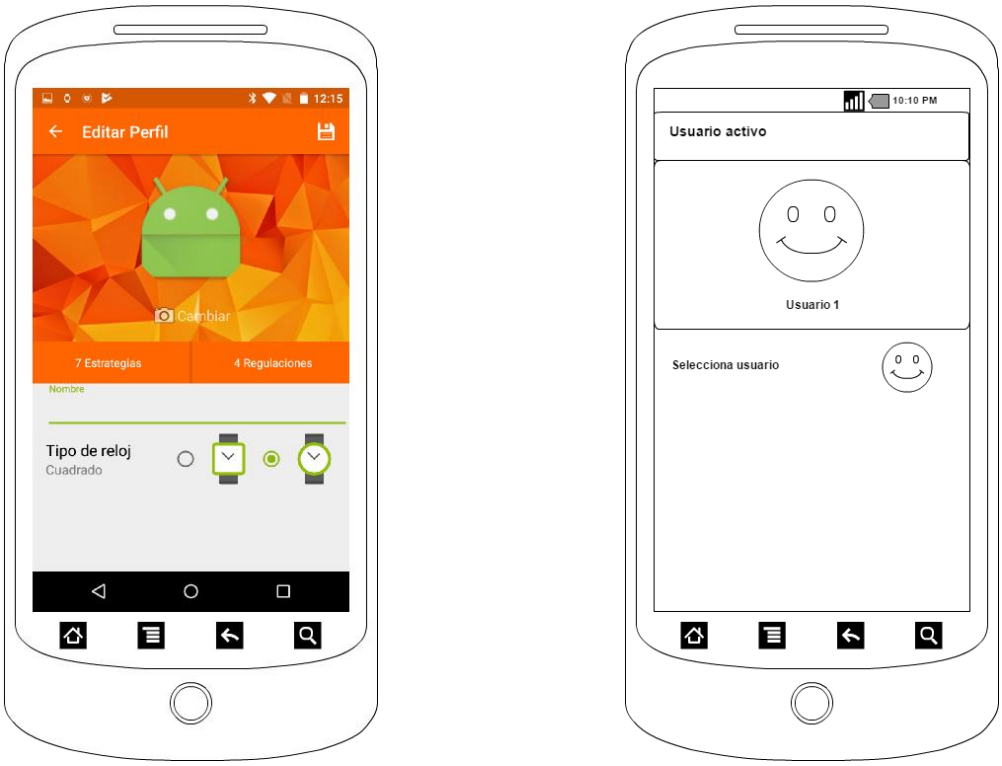


Figura 4.10 Comparativa selección de usuario

- **Lista de usuarios:** en la pantalla anterior debe de mostrarse un diálogo que te permita seleccionar uno de los usuarios creados como se muestra en el siguiente prototipo.

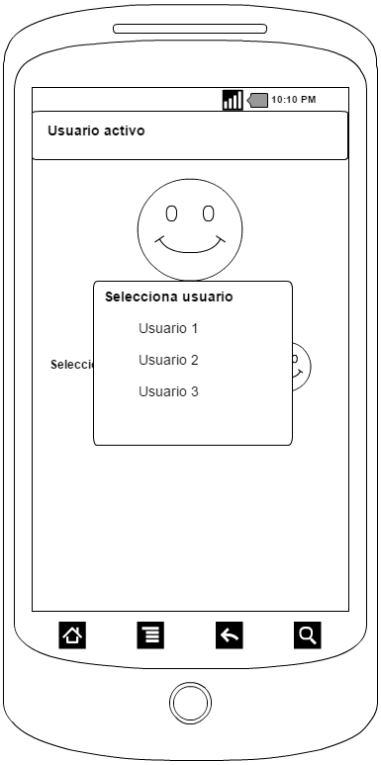


Figura 4.11 Prototipo Lista de usuarios

En la página siguiente se muestra un diagrama de navegación de las nuevas funcionalidades de la aplicación:

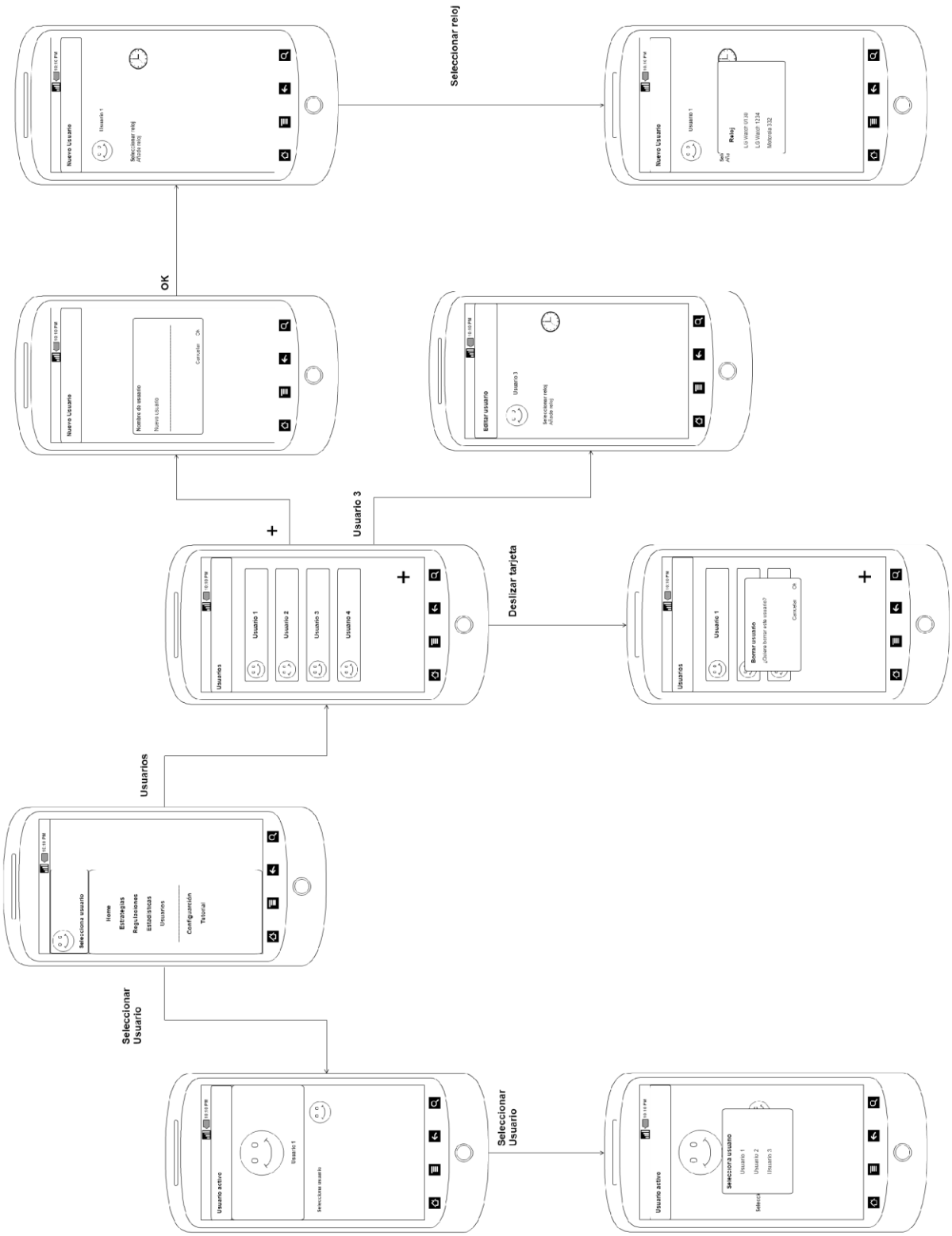


Figura 4.12 Diagrama de navegación multiusuario

4.1.3 Base de datos

No hay cambios significativos en la base de datos excepto la modificación de la tabla de usuarios para añadir las nuevas tuplas y eliminar aquellas que ya no son necesarias. A continuación, se muestra una comparativa de las dos versiones.

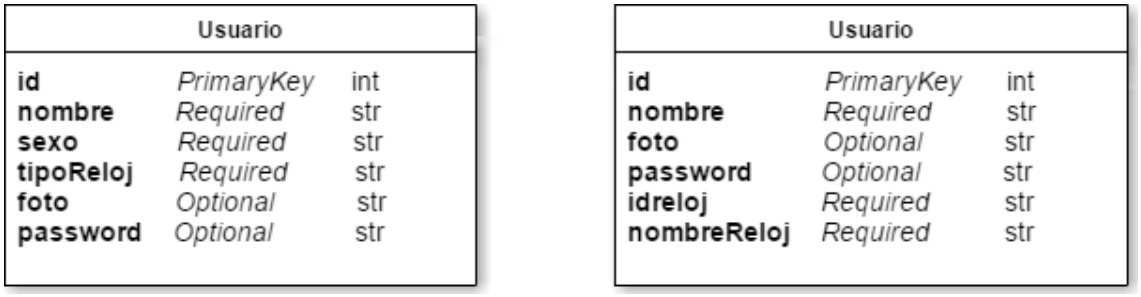


Figura 4.13 Comparativa Tabla Usuario de la primera versión (izquierda) con respecto a la nueva (derecha)

Se ha añadido la tupla **idreloj** para almacenar la id de nodo, que reconoce al Smartwatch asociado al usuario dentro de la Android Wear Network y la tupla **nombreReloj** que permite identificar el smartwatch con un nombre legible dentro de la aplicación.

4.2 TRANSMISIÓN DE DATOS

Para poder satisfacer los requisitos relacionados con la transmisión de datos entre dispositivos necesitamos dos sistemas principales:

El primero servirá para convertir una regulación de la herramienta de autor en diferentes ficheros. Esto facilitará su envío por partes al smartwatch, una vez se encuentren en el dispositivo serán reconstruidos siguiendo unas pautas.

Por otro lado, requerimos un protocolo de comunicación que permita a los dispositivos enviarse paquetes de datos, reconocer los mensajes que están destinados a ellos y descartar los que no.

4.2.1 Conversión de la regulación

Una regulación es una secuencia de estrategias, que a su vez están compuestas de pasos de imágenes o texto. Los pasos dentro de las estrategias comparten configuración, por ejemplo, pueden ser por tiempo: un paso se mostrará durante los segundos especificados antes de pasar al siguiente, o por tacto: el paso se mostrará hasta que el usuario pulse en la pantalla del reloj.

A continuación, se muestra un ejemplo de una regulación sencilla formada por una estrategia de tiempo y otra táctil:

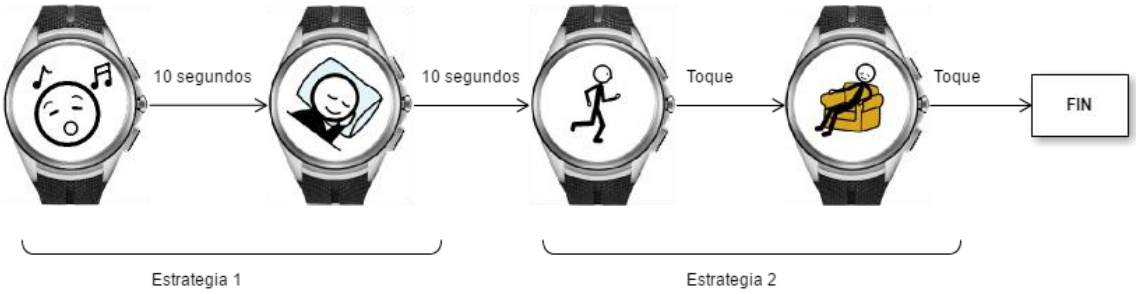


Figura 4.14 Regulación básica

En reuniones previas al comienzo del desarrollo de este proyecto, se acordó que para la transmisión de una regulación por el canal de comunicación se utilizaría un protocolo basado en JSON. Una regulación transformará todos sus pasos en ficheros JPG y se creará un fichero JSON que contenga campos con las características principales de la regulación, además de instrucciones de cómo se deben agrupar los pasos en las distintas estrategias. En el Anexo A se puede ver un ejemplo del fichero JSON de una regulación típica.

En la siguiente imagen se muestra la conversión de la regulación usada anteriormente como ejemplo.

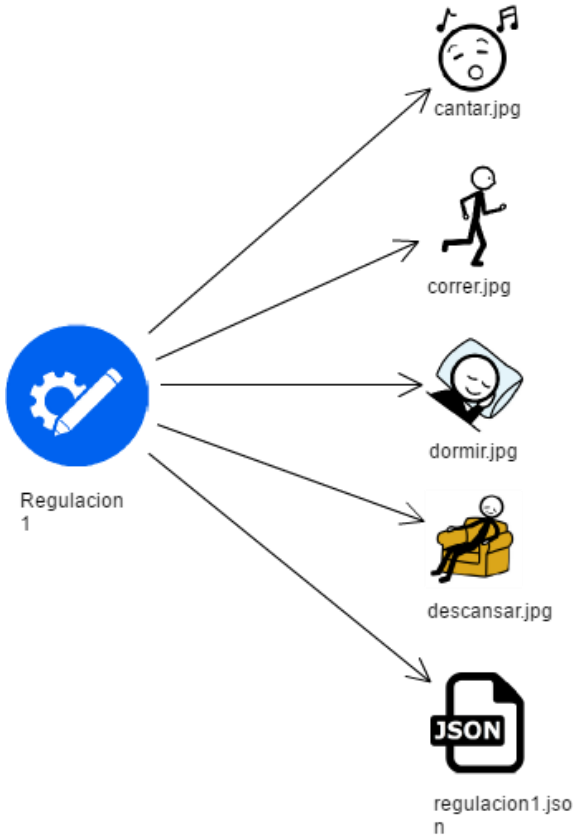


Figura 4.15 Conversión de regulación

4.2.2 Protocolo de comunicación

El funcionamiento estándar de la comunicación entre dispositivos de la Android Wear Network es de la siguiente manera: el dispositivo que envía los datos los sube a la Data Layer, el resto de dispositivos comprueban si hay datos nuevos disponibles y si es así se lo descargan. De esta manera la información queda replicada entre todos los smartwatches. Se podría decir que, en este caso, el teléfono envía información en difusión al resto de dispositivos.

Esta implementación no es válida para este proyecto, ya que de esta manera el teléfono solo podría sincronizar una única regulación con todos los relojes. Los requisitos educidos durante la fase de análisis especifican que se debe poder sincronizar diferentes regulaciones con cada smartwatch conectado a la red, esta funcionalidad queda reflejada en la figura 4.16.

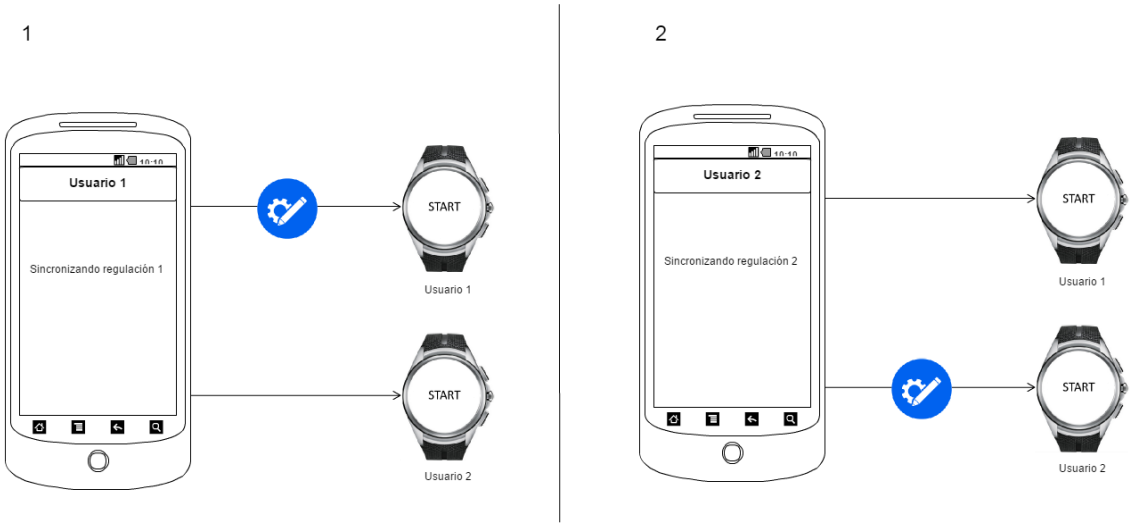


Figura 4.16 Comunicación multiusuario (regulaciones)

La transmisión de los logs con los datos biométricos de los usuarios recogidos por los smartwatches se realizará de la misma manera.

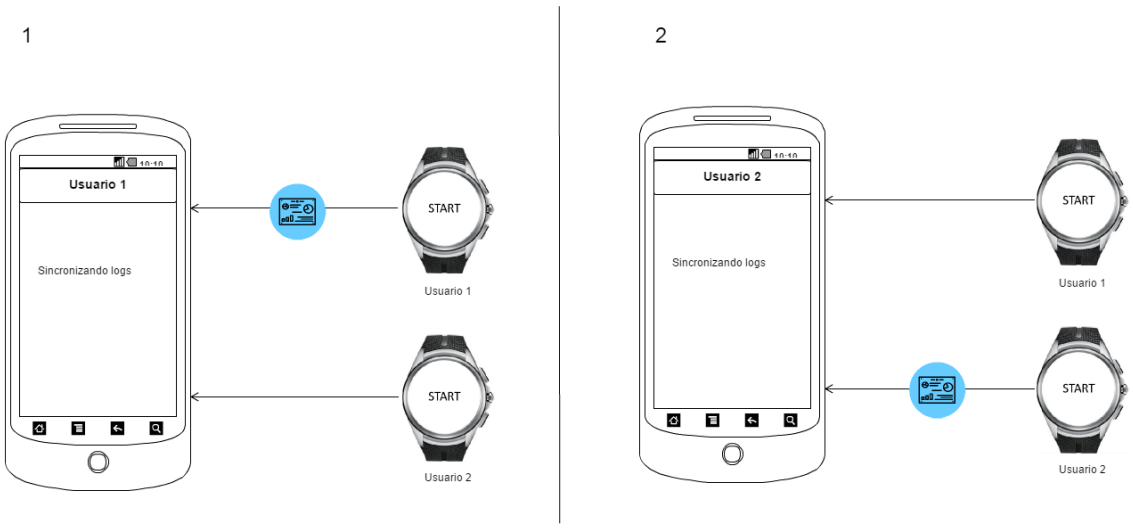


Figura 4.17 Comunicación multiusuario (logs de usuario)

5 DESARROLLO

En este apartado se detalla la implementación de las distintas partes del proyecto. Al igual que la sección de diseño, se ha dividido el desarrollo en dos partes: la primera centrada en la arquitectura multiusuario de la aplicación y la segunda en la implementación del protocolo de transmisión de datos.

5.1 MULTIUSUARIO

A continuación se hace una descripción de la implementación de las nuevas pantallas de la aplicación que dan soporte a la funcionalidad multiusuario. Además de la lógica de estas, se detalla la implementación en XML de la interfaz de usuario cuyo diseño se ha ajustado a los requisitos de usabilidad y se puede ver en el Anexo B.

5.1.1 Menú lateral

El Drawer menu o menú lateral se utiliza para navegar entre todas las secciones de la aplicación. Para acceder al apartado de usuarios se ha tenido que añadir una nueva opción, además, se ha cambiado la lógica del perfil, para que se muestre la información del usuario activo en ese momento.

5.1.1.1 Interfaz

La única modificación realizada en la interfaz con respecto a la versión anterior, ha sido añadir el botón correspondiente a la sección de usuarios en el archivo main_activity_drawer.xml. El ítem tiene tres atributos: la id “nav_users”, el icono “ic_users” que se ha obtenido del banco de imágenes vectoriales de Android Studio, y por último el título “Usuarios”.

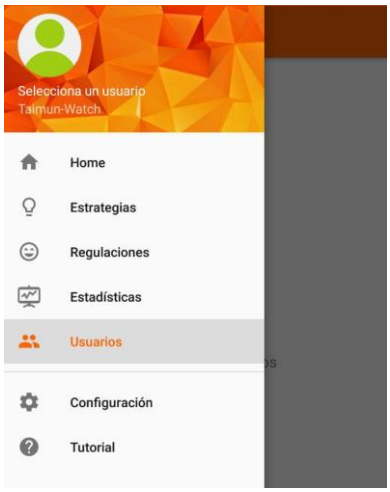


Figura 5.1 Menú lateral

5.1.1.2 Lógica

Para conseguir la funcionalidad descrita, se ha modificado el código de la clase MainActivity. En la función "inicializarDrawer()", que se encargaba de cargar la información del perfil del usuario del teléfono además de inicializar los botones de las diferentes secciones, ahora se obtiene la id del usuario activo a través de una llamada a las preferencias de la aplicación. Con la id del usuario, se consigue el resto de la información necesaria haciendo una llamada a "getUser()" de la clase del adaptador de la base de datos. Una vez se obtiene el nombre del usuario y la imagen de perfil se cargan en los diferentes componentes de la interfaz. En el caso de que no haya ningún usuario seleccionado, se mostrará una imagen por defecto y el texto "Selecciona un usuario". Se ha modificado el Listener "onClickListener()" de la imagen de perfil para que, al pulsar encima, se inicie la actividad de selección de usuario activo.

También se ha modificado el Listener "onNavigationItemSelected()" que comprueba la id del elemento pulsado en el menú y llama a la función auxiliar "selectFragment()" la cual se encarga de realizar la transacción del fragmento Home por el fragmento de la sección seleccionada. Se ha añadido el caso de selección de R.id.nav_users que corresponde con la sección de usuarios.

5.1.2 Fragmento de usuarios

El fragmento UserFragment muestra la lista de usuarios creados de la aplicación. Desde aquí se puede tanto crear usuarios nuevos, como editar o borrar los ya creados. Se ha seguido el diseño de tarjetas que se utilizó en la versión anterior para las secciones de Regulaciones y Estrategias.

5.1.2.1 Interfaz

La interfaz de este fragmento está formada por tres componentes agrupados en un CordinatorLayout: El RecyclerView encargado de mostrar los usuarios, una vista de texto que aparece cuando no hay usuarios en la base de datos y que muestra el mensaje: "No hay usuarios creados", y un botón flotante en el extremo inferior izquierdo de la pantalla con el símbolo "+" para añadir nuevos usuarios.

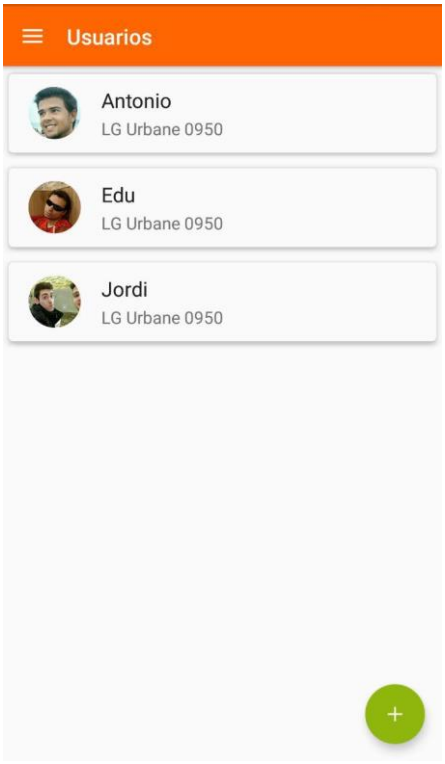


Figura 5.2 Fragmento Usuarios

5.1.2.2 Lógica

En la función principal de la clase se inicializan los diferentes elementos de la interfaz.

Esta actividad principalmente contiene un RecyclerView, un widget de Android que permite mostrar grandes conjuntos de datos que se pueden desplazar de manera muy eficiente al mantener una cantidad limitada de vistas. El widget se sirve de un adaptador que se encarga de enlazar cada ítem del set de datos, en este caso cada usuario, con una tarjeta del RecyclerView. Las clases auxiliares “inicializarListaUsuarios()” y “updateUserAdapter()” inicializan y mantienen actualizado el set de datos. Estas funciones contienen llamadas al adaptador de la base de datos para obtener el set de usuarios almacenados, y una llamada al constructor del adaptador que se guarda en un atributo de la clase para luego enlazarlo con el RecyclerView.

Otro componente importante es el botón de añadir usuario, al que se le asigna un Listener que escucha las pulsaciones e inicia un cambio de actividad con un Intent a Nuevo Usuario.

5.1.3 Adaptador de la vista de usuarios

5.1.3.1 Interfaz

En el caso de las tarjetas, la interfaz se compone de un widget de tipo CardView que agrupa los siguientes elementos principales: un CircleImageView, para mostrar la imagen de perfil del

usuario; y dos TextViews, alineados horizontalmente, que muestran el nombre de usuario y el del reloj asociado.

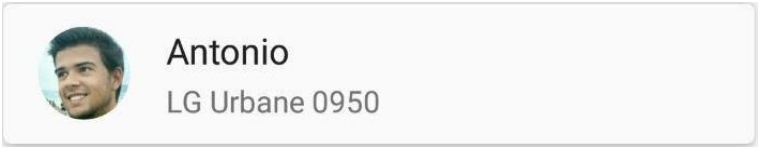


Figura 5.3 Tarjeta usuario

5.1.3.2 Lógica

Como se mencionaba anteriormente, el adaptador de la vista, implementado en la clase UserViewAdapter, permite enlazar cada usuario con una tarjeta mostrando la información correspondiente. Además de eso, también se pueden añadir Listeners que responden a diferentes interacciones con las tarjetas, como pulsaciones o desplazamientos.

A continuación se muestra el constructor del adaptador de la vista:

```
public UserViewAdapter(ArrayList<Usuario> usuarios, Context context,
                       final OnItemClickListener listener, boolean fromprofile);
```

Figura 5.4 Constructor del adaptador de la vista de usuario.

En el argumento usuarios se le pasa la tabla de usuarios obtenida de la base de datos. Además también se pasa el contexto de la aplicación y el Listener de la pulsación en cada tarjeta, en este caso una función que inicia un Intent de cambio a la actividad de Nuevo Usuario, este Intent lleva consigo un atributo con la id del usuario cuya tarjeta se ha pulsado, este atributo informa a la actividad de Nuevo Usuario de que se trata de una edición y esta se encarga de extraer la información necesaria de la base de datos. El último argumento, “fromprofile”, permite adaptar el aspecto de la vista dependiendo de si se está llamando al adaptador desde el fragmento de usuarios, o desde la actividad de selección de usuario activo.

El adaptador implementa una subclase llamada ItemViewHolder, que es el modelo para cada tarjeta de la lista. Contiene los atributos correspondientes a las vistas de nombre de usuario, imagen de usuario y nombre del reloj que se muestran en cada tarjeta. Además, implementa la función “bind()” que se encarga de asignar el Listener a la vista.

```
public static class ItemViewHolder extends RecyclerView.ViewHolder {

    /*Elementos dentro de la vista de la tarjeta.*/
    public ImageView imgIcon;
    public TextView nombreUsuario, nombreReloj;

    public ItemViewHolder(View itemView) {
        super(itemView);
        imgIcon = (ImageView) itemView.findViewById(R.id.user_icon);
        nombreUsuario = (TextView) itemView.findViewById(R.id.name_user);
        nombreReloj = (TextView) itemView.findViewById(R.id.watch_name);
    }
```

Figura 5.5 Clase ItemViewHolder

Además, en el adaptador se implementa un Listener que escucha la acción de desplazar la tarjeta hacia los extremos de la pantalla. Se ha implementado para que, al hacerlo, muestre un diálogo de confirmación de borrado, en el caso afirmativo, se hace una llamada a la función "deleteUser(idUsuario)" del adaptador de la base de datos, que se encarga de borrar el usuario seleccionado. Después de esto, se hace una llamada a la función "notifyItemChanged(position)" que se encarga de notificar que el set de datos ha cambiado y realiza las modificaciones adecuadas en la interfaz después del borrado del usuario.

5.1.4 Nuevo Usuario

Esta actividad, implementada en la clase NewUserActivity, contiene la lógica principal de creación y edición de usuarios, también se encarga de hacer las llamadas necesarias a la clase de conexión con la API de Android Wear para obtener la lista de Smartwatch conectados a la Android Wear Network. Además, se ha implementado la lógica de guardado de usuario, cumpliendo con los requisitos **RF 5** y **RF 14**.

5.1.4.1 Interfaz

La interfaz de esta actividad está formada por dos LinearLayouts. El primero contiene alineados de forma horizontal un CircleImageView para la imagen de perfil, un TextView para el nombre de usuario y un ImageButton con el icono del lápiz que sirve para editar el nombre de usuario. El segundo LinearLayout contiene alineados en vertical dos TextViews, uno de ellos con el texto "Selecciona reloj" y el otro con el nombre del reloj seleccionado o el texto "Añade reloj" en caso de que no se haya seleccionado ninguno. Alineado en horizontal con ambos elementos hay un ImageView con el icono del reloj, que sirve para seleccionar un smartwatch conectado a la red de Android Wear y enlazarlo con el usuario.

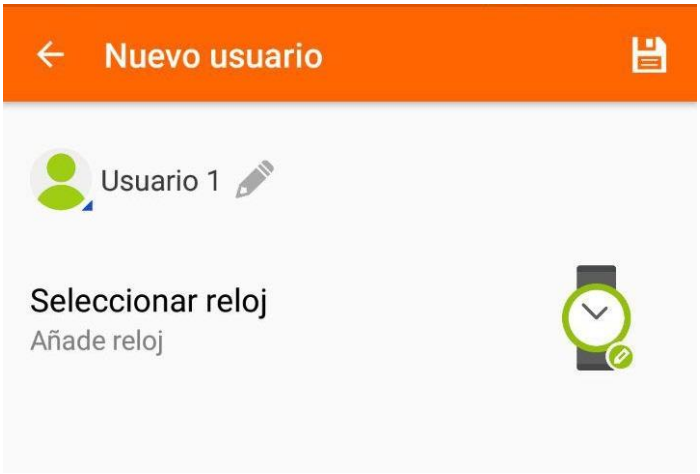


Figura 5.6 Actividad de Nuevo Usuario

5.1.4.2 Lógica

Lo primero que se comprueba en esta clase es si el Intent con el que se ha cambiado la actividad tiene la información extra de una id de usuario, si es así significa que se ha accedido a la actividad a través de una pulsación encima de una tarjeta de usuario creado en vez de una pulsación encima del botón de nuevo usuario. En ese caso se obtiene el usuario de la base de la base de datos a través de la función del adaptador "getUser(idUsuario)" y se carga su información (nombre, imagen y smartwatch asociado) en los elementos correspondientes de la interfaz, además, se cambia el título de la ventana a "Editar Usuario". Si no existe una id de usuario en el Intent, se asume que se quiere crear uno nuevo. En ese caso, se hace una llamada a la función "editUserDialog()", que crea un diálogo Popup en el que se pide introducir un nombre de usuario, por defecto, se obtiene el número de usuarios creados, y se pone el nombre "Usuario X" donde X es la posición que va a ocupar este nuevo usuario en la tabla de la base de datos.

```
//Comprobamos si es creación o edición de usuario
Intent intent = getIntent();
idUsuario = intent.getLongExtra("userId", -1);
if (idUsuario > 0) {
    isEdition = true;
    db.open();
    nombreReloj = db.getUser(idUsuario).getNombreReloj();
    idReloj = db.getUser(idUsuario).getIdReloj();
    db.close();
    getSupportActionBar().setTitle(R.string.edit_user);
} else {
    getSupportActionBar().setTitle(R.string.new_user);
}
```

Figura 5.7 Comprobación del Intent de Nuevo Usuario

En la función principal, se establecen tres Listeners para escuchar las pulsaciones en los tres elementos principales de la actividad. Si se pulsa encima del elemento con id "new_user_icon", es decir, el icono de usuario, se llama a la función "dialogPhoto()" que crea un diálogo Popup donde se da la opción de añadir imágenes a través de la cámara o de la galería, para ello utiliza funciones

de la clase auxiliar PhotoUtils. Si el elemento pulsado tiene la id “user_edit”, el icono del lápiz al lado del nombre de usuario, se llama a la función “editUserDialog()” explicada en el párrafo anterior. Por último, si se pulsa encima del icono del reloj, cuya id es “btnWatch”, se crea un diálogo con la lista de dispositivos conectados a la Android Wear Network. Esta lista se obtiene llamando a la función “retrieveDeviceNode()”, que utiliza la clase GoogleConnection para establecer una conexión con la API de Nodos de Google y hacer una solicitud de los nodos conectados a la red, a continuación se puede ver esta función.

```
private void retrieveDeviceNode() {
    final GoogleApiClient client = googleConnection.getGoogleClient();
    new Thread(new Runnable() {

        @Override
        public void run() {
            client.blockingConnect(CONNECTION_TIME_OUT_MS, TimeUnit.MILLISECONDS);
            NodeApi.GetConnectedNodesResult result =
                    Wearable.NodeApi.getConnectedNodes(client).await();

            mNodes = result.getNodes();

            client.disconnect();
        }
    }).start();
}
```

Figura 5.8 Función retrieveDeviceNode()

Se ha sobrescrito la función “onOptionItemSelected()”, un Callback que se llama cada vez que se pulsa sobre un elemento de la barra superior, para que cuando se pulse sobre el icono de guardado se llame a la función “saveUser()”, esta función se encarga de recoger todos los datos introducidos a través de la interfaz y guardarlas en la base de datos usando las funciones del adaptador. Si se trata de una edición de usuario, modifica los campos de ese usuario en la base de datos, si en cambio se trata de una creación, introduce el nuevo elemento con los atributos correspondientes.

Cumpliendo con el requisito de guardado automático, se ha sobrescrito la función “onBackPressed()”, que se llama automáticamente cuando se pulsa sobre el botón de atrás, para que muestre un diálogo de guardado cuando se detecta que se han realizado cambios sobre el usuario.

5.1.5 Seleccionar usuario activo

Esta actividad, permite seleccionar de entre la lista de usuarios de la aplicación aquel con el que se quiere establecer la conexión para enviar y recibir datos de su smartwatch. Esta actividad está implementada en la clase ProfileActivity y sustituye a la actividad de perfil de la versión anterior de la aplicación.

5.1.5.1 Interfaz

La interfaz de esta actividad se divide en dos bloques alineados en vertical y contenidos en un LinearLayout: la cabecera y la zona de selección de usuario. La primera contiene en un LinearLayout un CircleImageView con la imagen de perfil del usuario activo y un TextView con su nombre alineados verticalmente. La segunda contiene otro LinearLayout con un TextView con el texto “Selecciona un usuario” y un ImageView con el icono de una silueta que sirve para abrir el dialogo de selección de usuario. Este diálogo está compuesto por tarjetas similares a las del fragmento de usuarios.

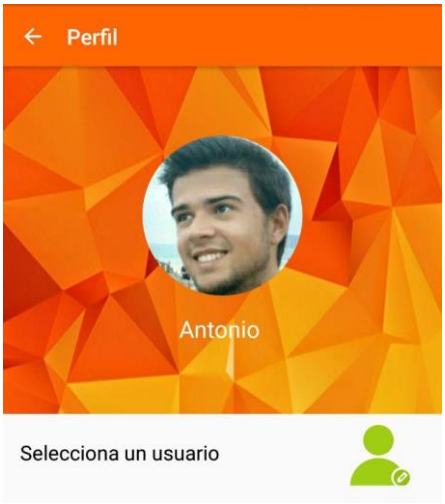


Figura 5.9 Actividad seleccionar usuario.

5.1.5.2 Lógica

En la función “onCreate()” de esta actividad se llama a la función auxiliar “loadUserData()”. Esta función se encarga de obtener la id del usuario activo de las preferencias de la aplicación con la función Preferences.getUserId(), de extraer el usuario correspondiente de la base de datos a través de la función del adaptador “getUser(idUsuario)” y por último de establecer la información de este en los elementos de la interfaz correspondientes. Si la función “getUserId()” devuelve el valor por defecto -1 cuando no se ha seleccionado ningún usuario activo, entonces se establece la información por defecto en los elementos de la interfaz.

La función principal también enlaza el icono de selección de usuario con su Listener correspondiente. Cuando se produce una pulsación sobre el icono, se llama a la función

"addUserDialog()" que muestra un diálogo Popup con tarjetas similares a las del fragmento de usuarios. Al igual que en esa clase, este fragmento de diálogo se sirve de un widget RecyclerView y de un adaptador para mostrar la lista de tarjetas. La lógica del adaptador es igual en ambos casos, excepto por el Listener de las tarjetas. En este caso, si se produce una pulsación sobre una tarjeta de usuario, se establece la id de ese usuario y la del reloj asociado en las preferencias de la aplicación con las funciones "Preferences.setUserId()" y "Preferences.setWatchId()", se cierra el Popup y se vuelve a llamar a la función "loadUserData()" para actualizar la información en la actividad.

5.2 TRANSMISIÓN DE DATOS

Como se explicaba en el apartado de diseño, para que se produzca el envío de una regulación del móvil al Smartwatch son necesarios dos componentes: un sistema de conversión, que transforme la regulación en imágenes y un fichero JSON que sirva de instrucciones para ensamblarlas en el otro extremo; y un protocolo de comunicación entre ambos dispositivos para poder enviar y recibir estos archivos. Este protocolo además, permite la recepción de ficheros con los datos biométricos recogidos por el reloj.

Para el envío de regulaciones se ha añadido a la interfaz de las tarjetas un botón de sincronización. Este botón tiene asociado un Listener que llama a la función "synchronizeRegulation()" que se encarga de realizar los pasos de conversión y envío. A continuación se hace una descripción de la implementación de estos pasos.

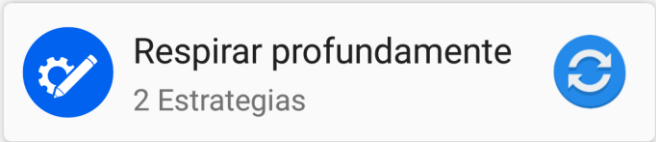


Figura 5.10 Tarjeta de regulación

5.2.1 Conversión de la regulación

El primer paso en la conversión de la regulación es la creación de una carpeta en el directorio raíz de la aplicación llamada "Regulation". En ella se guardará, de forma temporal, los archivos que conforman la regulación que se quiere sincronizar. A continuación, se crea el fichero JSON de la regulación con las funciones "writeRegulation()", "writeStrategy()" y "writeStep()" de la clase utilJson. Esta clase utiliza la librería de Android JsonWriter para facilitar la tarea de la conversión de campos.

Una vez se ha copiado el fichero JSON en la carpeta temporal, dos bucles anidados recorren los pasos de cada estrategia de la regulación, en este punto hay tres posibilidades:

1. **El paso es una imagen propia:** si el paso está compuesto por una imagen tomada de un recurso externo de la aplicación, se copia desde la carpeta en la que está almacenada al directorio temporal "Regulation". El fichero está identificado por su posición en la tabla de recursos propios de la base de datos.

2. **El paso es una imagen predeterminada:** si el paso está compuesto por una imagen predeterminada, se obtiene desde los recursos de la aplicación y se copia a la carpeta temporal “Regulation” gracias a la función auxiliar “resourceToPNG()”. El fichero está identificado por su id de recurso.
3. **El paso es un texto:** en este caso se convierte la vista de texto a una imagen PNG con la función de PhotoUtils “textToImage()”.

5.2.2 Protocolo de comunicación

El protocolo de comunicación se basa en el envío y la recepción de mensajes y paquetes de datos, utilizando la MessageAPI y la DataAPI.

Los mensajes sirven para enviar información de control al otro extremo, como por ejemplo el inicio de una transacción o una petición de reenvío de datos. En la clase DataReceiver se ha sobrescrito la función callback “onMessageReceive()” perteneciente a la Api de Android Wear. Esta función se llama automáticamente cuando detecta que se ha producido el envío de un mensaje a la capa de datos. Un mensaje se compone de un campo Path y del mensaje propiamente dicho. El Path actúa de identificador del tipo de mensaje, a continuación se listan los Path utilizados en el desarrollo de este protocolo.

- **WEAR_TO_MOB:** este es el mensaje inicial que se manda desde el Smartwatch al móvil cuando se inicia una regulación, dentro de este Path hay dos mensajes: NOT_READY_TO_SYNC y READY_TO_SYNC, que avisan a la herramienta de autor de si está preparado para recibir la regulación.
- **NO_MORE_LOGS:** este mensaje se envía desde el Smartwatch para anunciar al teléfono que se ha terminado de retransmitir los ficheros con los datos biométricos recogidos.
- **SEND_IMAGE_FILE_AGAIN:** esta petición hace que el móvil envíe de nuevo el último fichero de regulación enviado al Smartwatch.
- **SEND_NEXT_IMAGE_FILE:** esta petición se envía desde el Smartwatch cuando ha procesado un fichero y solicita el siguiente al teléfono.
- **MOB_TO_WEAR:** este mensaje se envía desde el teléfono al Smartwatch para avisar de que va a dar comienzo la sincronización de la regulación.
- **START_SYNC_LOG_FILES:** esta petición se envía desde el teléfono para informar al Smartwatch de que se solicitan los ficheros de datos biométricos. Es el mensaje que inicia la sincronización de logs de usuario.
- **SEND_LOG_FILE_AGAIN:** este mensaje se envía al Smartwatch para informar de que ha habido algún error en la retransmisión del último bloque de datos y solicita que se vuelva a enviar.

Los paquetes de datos se envían a través de un objeto llamado DataMap, que al igual que los mensajes está formado por dos atributos: el Path que sirve como identificador del tipo de datos que se está enviando, y el bloque de datos en forma de una array de bytes. Para escuchar si se ha producido algún envío de bytes a la capa de datos se ha sobrescrito la función Callback "onDataChange()" en la clase DataReceiver, que procesa la información dependiendo del Path que reciba. Dado que hay dos tipos de datos que se pueden enviar y recibir (regulaciones y logs de usuario) se han establecido los siguientes Paths para la comunicación:

- **LOG_FILE:** este Path indica al Smartphone que los datos que se están recibiendo forman parte de un fichero de log de usuario.
- **FILE:** este Path indica al Smartwatch que se está recibiendo un bloque de datos correspondiente a una regulación de la herramienta de autor.

El objeto DataMap tiene una restricción de carga de 100 KB por envío, por lo que se han desarrollado funciones de división y recomposición de ficheros. Una vez los datos de la regulación se han retransmitido al Smartwatch son procesados por la clase JSONToRegulation que se encarga de ensamblar todos los archivos en la regulación correspondiente siguiendo las directrices establecidas en el fichero JSON.

Para realizar el envío de mensajes y datos es necesario haber establecido una conexión con el cliente de Google, las funciones que se encargan de esto están implementadas en la clase GoogleConnection, además, en esta clase también están implementadas las clases "sendToDataLayerThread()" y "sendMessageThread()" que lanzan un hilo en el que se envían los datos y los mensajes respectivamente. En la figura 5.11 se puede ver un esquema de la implementación de la función de envío de datos.

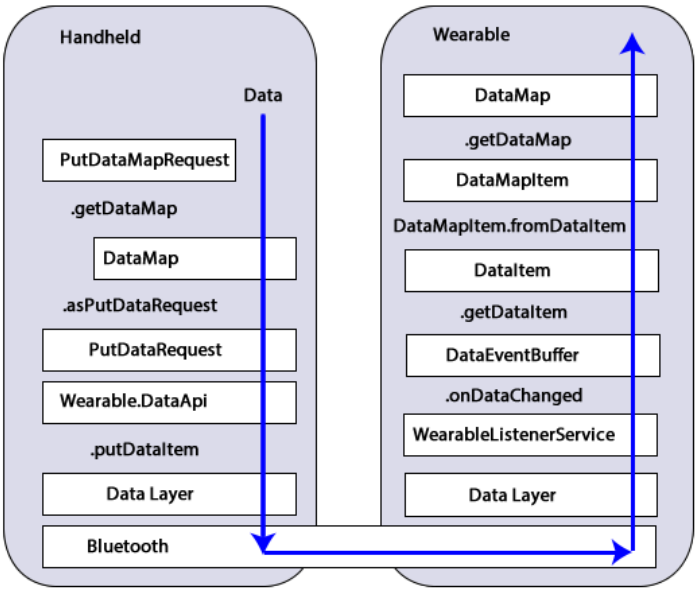


Figura 5.11 Implementación de envío de datos

En una implementación basada tan solo en los Paths para identificar los mensajes y los bloques de datos el envío funciona en difusión. Es decir, en el caso de tener conectados tres Smartwatch al mismo móvil, una sincronización de una regulación se haría al mismo tiempo en los tres dispositivos, quedando replicada la información. El comportamiento requerido es que la regulación solo se sincronice con el Smartwatch asociado al usuario activo. Para conseguirlo, al realizar envío de datos y de mensajes, se obtiene la id del reloj activo de las preferencias gracias la función "Preferences.getWatchId()" y se concatena con la cadena del Path. En el extremo del Smartwatch, se vuelven a separar las dos cadenas y se comprueba si la id coincide con la propia, si es así, el mensaje o el paquete de datos se procesa normalmente, si no, se ignora. Esta id viene dada por la API de nodos de Android Wear y es un identificador único para toda la red. A continuación se puede ver un diagrama de una comunicación típica entre un móvil con dos Smartwatch conectados.

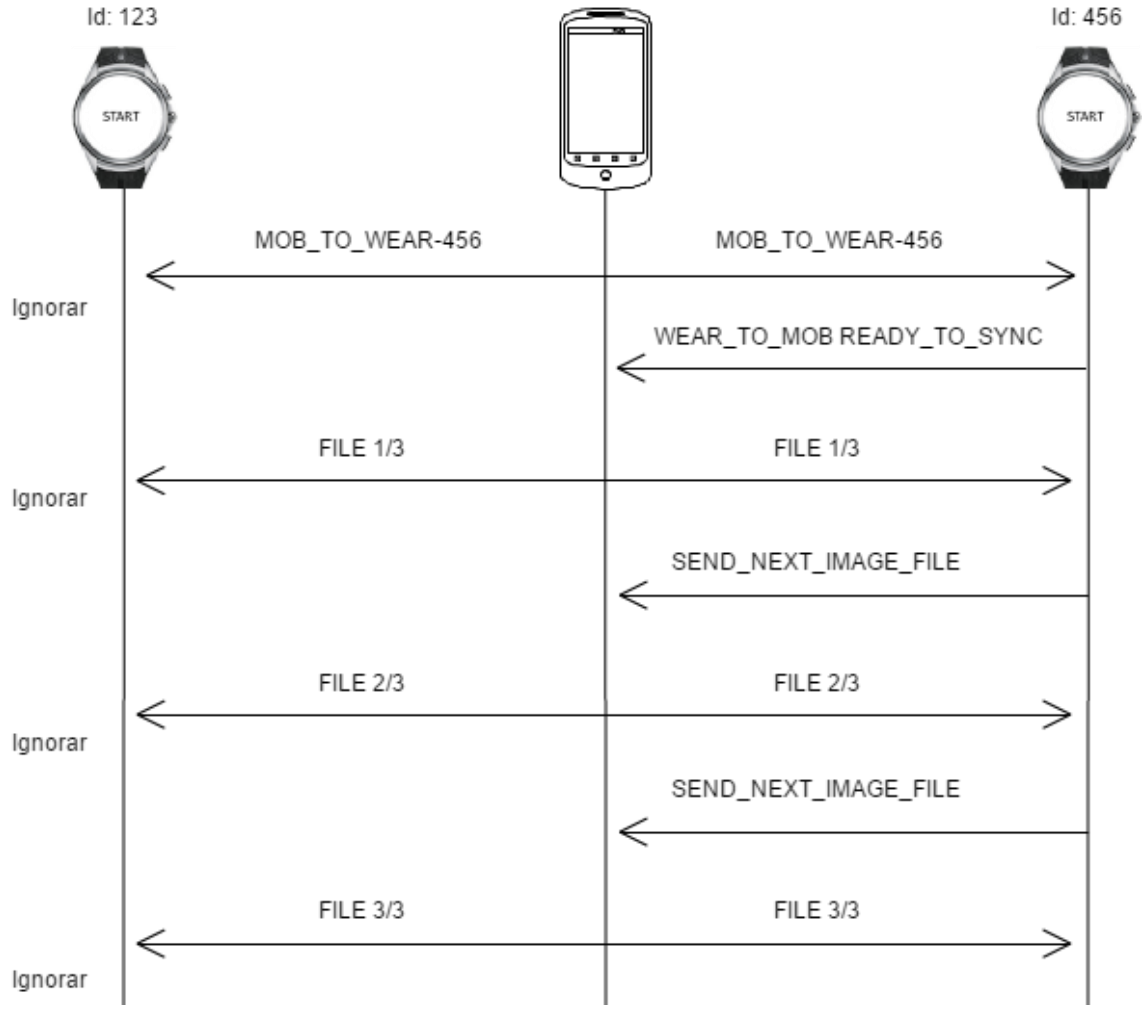


Figura 5.12 Diagrama de secuencia sincronización

6 PRUEBAS

Las pruebas de la aplicación se han realizado en dos bloques: por un lado se ha comprobado que todos los componentes de la parte multiusuario funcionan correctamente, y por otro lado se ha hecho una prueba de sincronización para comprobar que el protocolo de comunicaciones se comporta de la manera esperada.

6.1 PRUEBAS MULTIUSUARIO

6.1.1 Creación de usuarios

Se ha procedido a crear tres usuarios con nombres e imágenes diferentes, se le ha asignado a cada uno un Smartwatch de la lista y se ha comprobado que se muestran en el fragmento de Usuarios con la información correcta, comprobando así también el funcionamiento del guardado de usuarios.

6.1.2 Edición de usuarios

Para probar la edición de usuarios se ha editado el nombre y la imagen de dos usuarios y se ha comprobado que la nueva información se actualizaba en las tarjetas respectivas. En vez de pulsar sobre el botón de guardado, se ha pulsado sobre el botón “Atrás” y la aplicación ha mostrado un diálogo de guardado automático cumpliendo con el requisito **RF 14**.

6.1.3 Borrado de usuarios

Se comprobó que al desplazar las tarjetas en el fragmento de Usuarios mostrase efectivamente la confirmación de borrado. Al darle a “Sí” la tarjeta desapareció de la lista. A través de métodos de depuración se confirmó que el usuario había sido eliminado de la base de datos.

6.1.4 Seleccionar usuario activo

Se seleccionó uno de los tres usuarios de la lista en la actividad de Seleccionar usuario activo y se comprobó que se actualizaba el nombre y la imagen de perfil de ese usuario en el menú lateral de la aplicación.

6.2 PRUEBAS DE SINCRONIZACIÓN

Para probar la correcta sincronización de las regulaciones se crearon tres usuarios diferentes, cada uno asignado a un Smartwatch. Se fueron eligiendo uno a uno los usuarios en la actividad de selección de usuario activo y se sincronizaron tres regulaciones predeterminadas diferentes de la aplicación. El resultado fue que la regulación que se mostró en cada reloj cuando detectó la situación de crisis fue la asignada a cada usuario, confirmando el correcto funcionamiento de la sincronización de regulaciones.

También se sincronizaron los logs de usuarios de cada uno de los Smartwatches. A la hora de acceder al apartado de estadísticas de la aplicación, se mostraron las gráficas correspondientes al usuario activo.

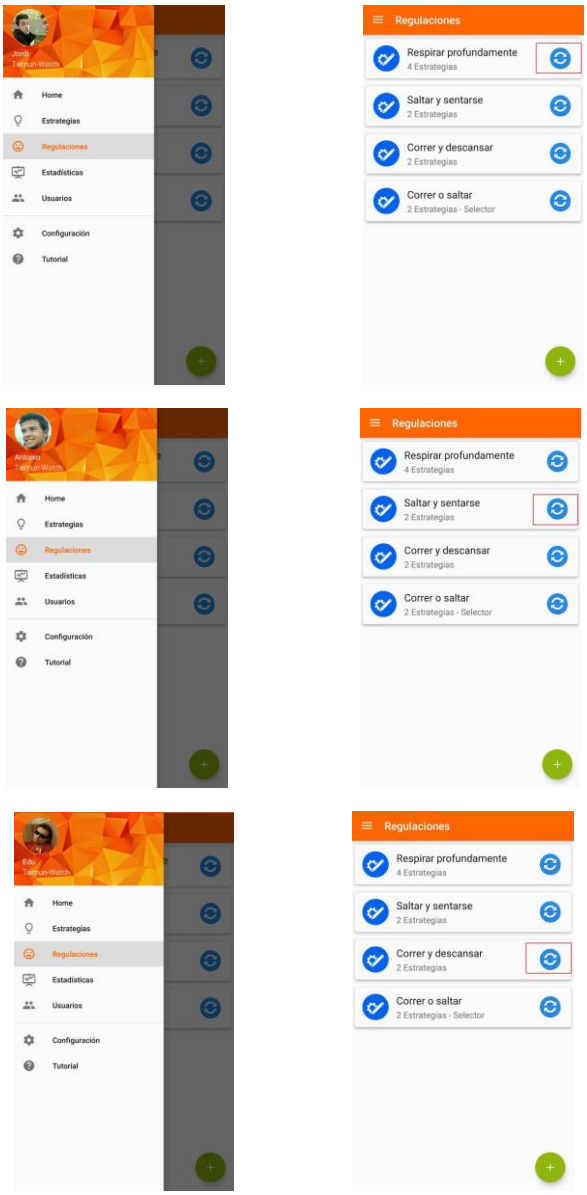


Figura 6.1 Pruebas de sincronización

7 CONCLUSIONES Y TRABAJO FUTURO

Tras las pruebas y los resultados obtenidos de la nueva versión de la aplicación podemos concluir que cumple con los objetivos acordados en la fase de análisis con los terapeutas profesionales.

Gracias a la flexibilidad de las herramientas que ofrece Google y a la tecnología de Android Wear hemos conseguido desarrollar un protocolo de comunicación que permite a la aplicación Taimun-Watch enviar y recibir datos desde su parte de Smartwatch a su parte de móvil. Además, respondiendo a una necesidad esencial por parte de los profesionales del colegio Alenta, se ha conseguido modificar la arquitectura de la aplicación para ampliar su escalabilidad, permitiendo la creación y gestión de varios perfiles de usuario para la personalización de las estrategias sincronizadas.

La respuesta ha sido generalmente positiva por parte de las entidades colaboradoras, lo que certifica la calidad obtenida.

Como trabajo futuro, hay algunas mejoras interesantes; por ejemplo, adaptar el protocolo de comunicación a las nuevas funcionalidades que ofrece la versión 2.0 de los dispositivos Android Wear para mejorar su rendimiento.

En cuanto a la aplicación en sí, se ha sugerido un aumento de la personalización de configuración de usuarios como, por ejemplo, añadir un umbral de pulsaciones máximas personalizado para cada caso, esto requeriría añadir nuevos mensajes de control al protocolo, pero no sería necesaria mucha restructuración gracias a que este se ha diseñado de una manera modular facilitando el añadido de nuevas funcionalidades.

Otra idea interesante sería la del envío de notificaciones por parte de los Smartwatches al móvil del terapeuta o incluso a su propio reloj cuando salta la regulación.

No hay duda que dentro de este campo hay espacio aún para mejorar, pero gracias a las tecnologías emergentes se espera poder ofrecer cada vez un mejor soporte sin sacrificar la independencia de la persona con TEA y este proyecto da un buen ejemplo de ello.

REFERENCIAS

[1] Andrea C. Samson, Jennifer M. Phillips, Karen J. Parker, Shweta Shah, James J. Gross, Antonio Y. Hardan, “Emotion Dysregulation and the Core Features of Autism Spectrum Disorder”, Journal of Autism and Developmental Disorders, Volumen 44, Número 7, 7, Diciembre 2013

[2] Juan C. Torrado, Germán Montoro and Javier Gomez, “The Potential of Smartwatches for Emotional Self-regulation of People with Autism Spectrum Disorder”, Conference: 9th International Conference on Health Informatics, 6, January 2016

[3] Tom Martin, Jennifer Healey, “2006’s Wearable Computing Advances and Fashions”, IEEE Pervasive Computing, Volumen 6, Número 1, 7, Enero 2006

[4] Sanjay M. Mishra, “Wearable Android”, 2015

[5] https://developer.android.com

GLOSARIO

API	Application Programming Interface
TEA	Transtorno del Espectro Autista
Smartwatch	Reloj inteligente
Node API	Interfaz de nodos de Android Wear
Data API	Interfaz de datos de Android Wear
Message API	Interfaz de mensajes de Android Wear
SDK	Software Development Kit
JSON	JavaScript Object Notation
JPG	Joint Photographics Experts Group
DataLayer	Capa de datos
XML	Extensible Markup Language

ANEXOS

A FICHERO JSON DE REGULACIÓN


```
{
    "Regulation": {
        "Id" : 15,
        "Name" : "Regulacion 20",
        "VibrationTime" : 2000,
        "Audio" : "null",
        "Strategies" : [
            {
                "Id" : 3,
                "Name" : "Estrategia sencilla",
                "DefaultTime" : 10,
                "DefaultGesture" : 0,
                "Selector" : 0,
                "StressCondition" : 1,
                "Steps" : [
                    {
                        "Id" : 3,
                        "NextStepId" : 7,
                        "StepType" : "Contenido",
                        "NumberOfSources" : 1,
                        "Text" : "Mira el numero",
                        "BackgroundColor" : "#2EFE2E",
                        "MaxStepTime" : -1,
                        "StepGesture" : -1,
                        "Sources" : [
                            {
                                "Id" : 7,
                                "Source" : "5",
                                "Text" : ""
                            }
                        ]
                    },
                    {
                        "Id" : 7,
                        "NextStepId" : -1,
                        "StepType" : "SelectorM",
                        "NumberOfSources" : 2,
                        "Text" : "Mira los pajaros",
                        "BackgroundColor" : "#2EFE2E",
                        "MaxStepTime" : -1,
                        "StepGesture" : -1,
                        "Sources" : [
                            {
                                "Id" : 8,
                                "Source" : "Pajaro1.png",
                                "Text" : "Gorrion"
                            },
                            {
                                "Id" : 9,
                                "Source" : "Pajaro2.png",
                                "Text" : "Aguila"
                            }
                        ]
                    }
                ],
                "NextStrategy" : [
                    -1
                ]
            }
        ]
    }
}
```

B CAPTURAS MULTIUSUARIO

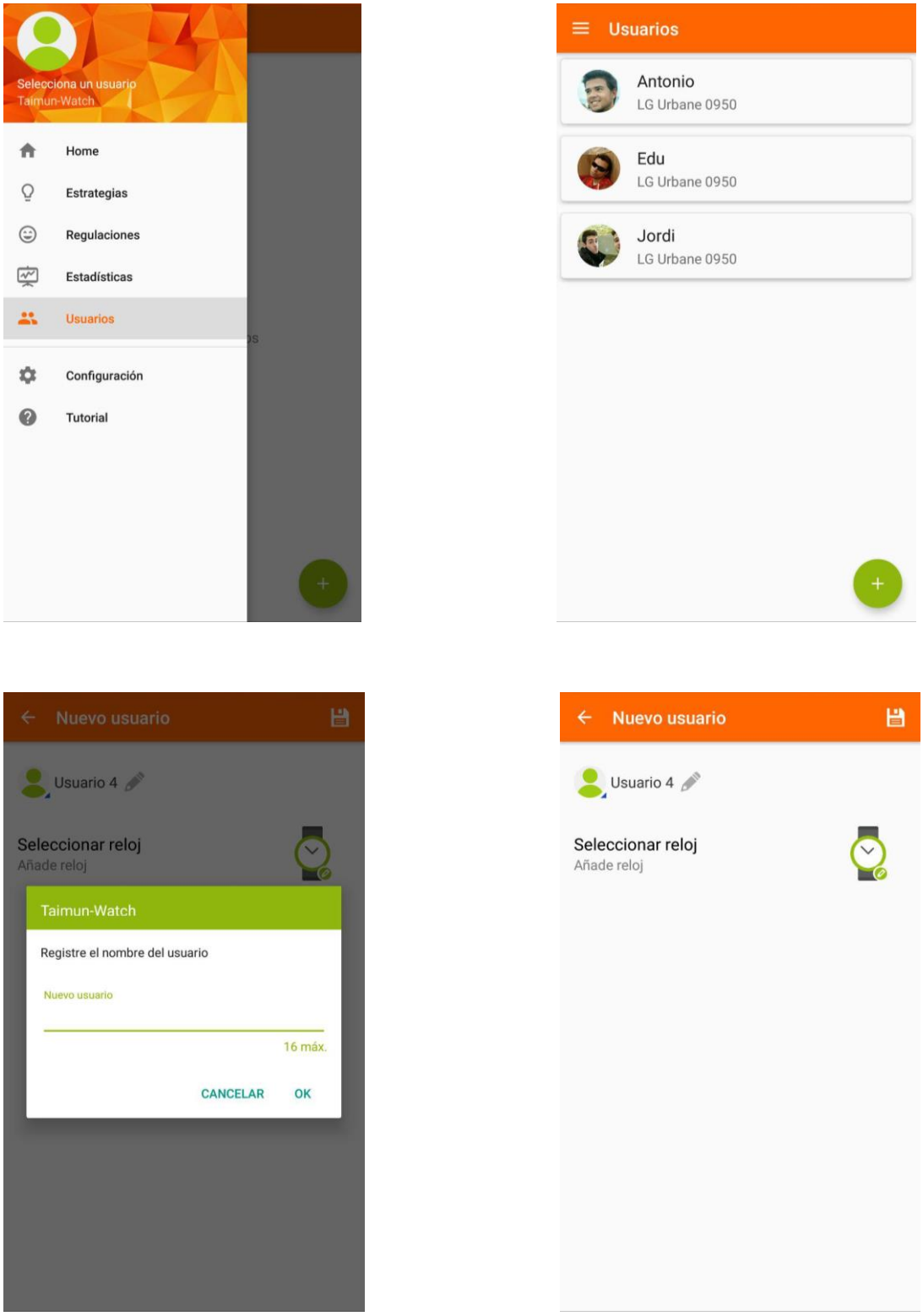
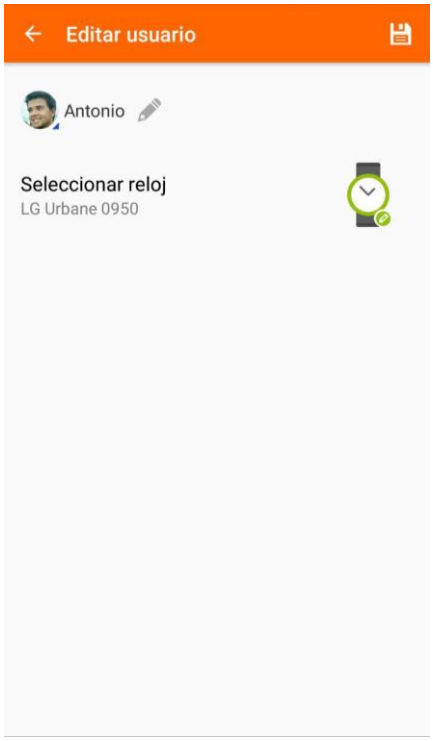
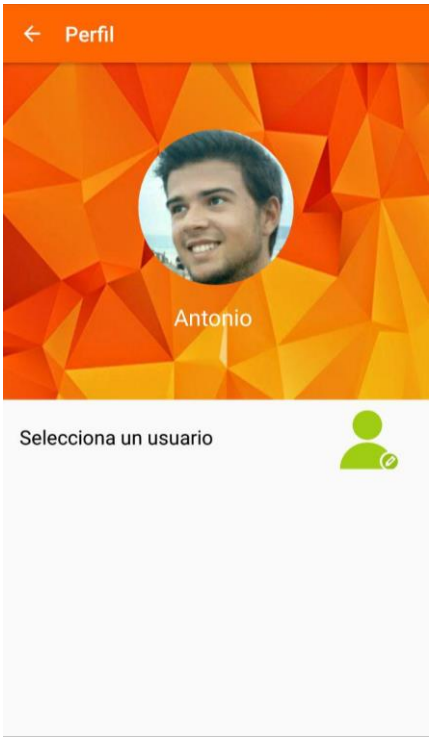


Figura B.1 Capturas multiusuario. Parte I.





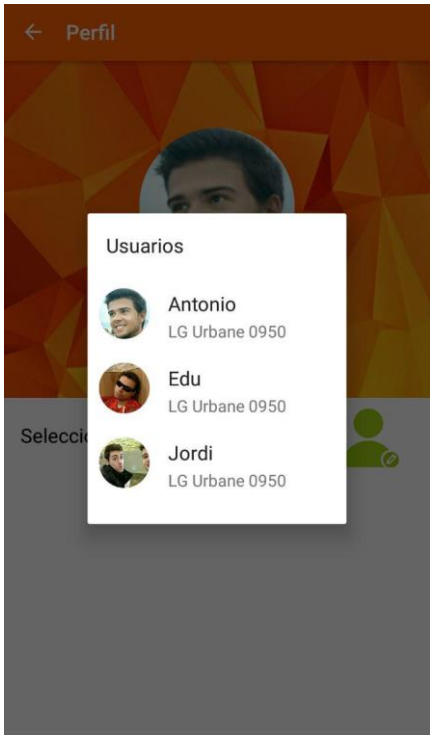


Figura B.2 Capturas multiusuario. Parte II.