UNIVERSIDAD CARLOS III DE MADRID

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO:

Diseño y desarrollo del sistema SimilDroid para evaluar con usuarios un algoritmo de estimación de la similitud entre aplicaciones Android de una WBAN

Autor: José Ignacio Coig-O'Donnell Velasco
Tutora: Ana Isabel González-Tablas Ferreres

“Cada (tic-tac) es un segundo de la vida que pasa, huye, y no se repite. Y hay en ella tanta intensidad, tanto interés, que el problema es sólo saberla vivir. Que cada uno lo resuelva como pueda.”

- Frida Kahlo

Índice de Contenidos

Índice de Ilustraciones

Índice de Tablas

Agradecimientos

Me gustaría aprovechar esta sección del trabajo para agradecer a mi profesora del trabajo de fin de grado, Anabel, el gran apoyo que me ha brindado desde el primer día en el que me embarqué en este proyecto. Muchas veces he escuchado la frase: "*En el TFG vas a estar totalmente solo, tu tutor/a pasará de ti*". Quiero dejar claro que en este caso ha sido totalmente lo opuesto, de no ser por el apoyo, el empeño de Anabel, seguramente, tanto el alcance, como la calidad de mi Trabajo de Fin de Grado habría sido muchísimo menor. Gracias Anabel.

Por otra parte, no quiero olvidarme de mi señora Madre. ¿Qué decir de ella? No se trata de una madre cualquiera, se trata de MI MADRE. Gracias a ella hoy puedo luchar por mis sueños. Gracias a ella hoy puedo decir que presento mi Trabajo de Fin de Grado. Hace cuatro años, esto era algo totalmente impensable. De no ser por su apoyo incondicional, y por sus frases típicas como: "Nacho, tú puedes con todo lo que te propongas"; "Nacho, no te rindas, puedes con ello", etc., seguramente no estaría escribiendo estas líneas de agradecimiento (que por cierto, tanto me están costando escribir). Gracias mamá.

Podría estar escribiendo líneas y líneas de agradecimiento a todo aquel que me echó una mano cuando más lo necesitaba durante estos cuatro años. Pero sin duda quiero agradecer a mi magnífico compañero de prácticas Rubén García Sánchez, que me ha ayudado a comprender muchos de los conceptos que se han explicado a lo largo de este grado. Seguramente sin él, todo esto habría sido, al menos, diferente. Gracias amigo.

Por último agradecer a la Universidad Carlos III el hecho de haberme dado la oportunidad de formarme en una escuela como esta. Una escuela donde lo importante es aprender (y no siempre aprobar). Si pudiera viajar en el tiempo, seguramente volvería a estudiar la misma carrera, en el mismo lugar, y al ser posible con la misma compañía.

1. Introducción

En los últimos años, los dispositivos móviles se han convertido en un elemento imprescindible para la mayoría de las personas. Tanto es así, que según el Instituto Nacional de Estadística (INE), en el 96.7% de los hogares españoles existe al menos un teléfono móvil [1], muchos de los cuales están incluidos dentro de lo que conocemos como *Smartphones*. En el año 2016, un 88.3% de los usuarios de Internet en España, acceden a la red a través de un dispositivo inteligente, o Smartphone [9].

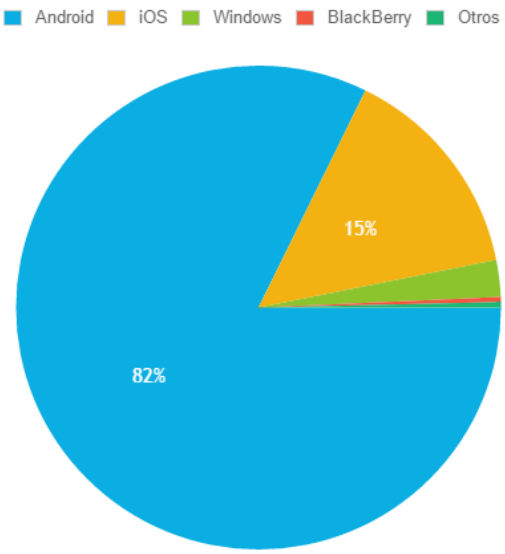

Ilustración 1 - Participación de mercado por Sistema Operativo en 2015 [10]

Y si además, tenemos en cuenta que en el año 2015, más de un 80% de los Smartphones comercializados, cuentan con sistema operativo Android [10], es fácil darse cuenta que Android es uno de los sistemas software más usados día a día.

Existe un problema grave con el sistema operativo Android. Según un estudio realizado por la universidad de Cambridge, más del 80% de los terminales con sistema operativo Android, son considerados "terminales inseguros" [2]. Con el calificativo "inseguros", nos referimos a que se tratan de dispositivos con una gran probabilidad de contener algún tipo de malware (software maligno) en sus entrañas, como lo son los troyanos, spyware, o simplemente lo que conocemos como virus común. El motivo es muy variado,

desde una falta de actualización del software del dispositivo, hasta una mala administración de las políticas de seguridad del terminal.

Es por todo esto, que la seguridad y la privacidad en Android es un problema que es necesario abordar de alguna manera. Este campo de investigación está, hoy en día, muy activo, existiendo multitud de estudios que pretenden mejorar la seguridad de los terminales móviles, además de asegurar la privacidad en los mismos. Es importante tener en cuenta que, a medida que el número de vulnerabilidades (y por tanto, de ataques) aumenta, existe también, un aumento del número de soluciones de seguridad ofrecidas por parte de investigadores [2].

Además, las aplicaciones ligadas a la computación móvil, se están viendo aumentadas de manera drástica en los últimos tiempos, lo que conlleva, además, a un aumento de la información cedida a proveedores de servicios, muchas veces sin el conocimiento del usuario [4].

Por otra parte, los avances tecnológicos de los últimos años en tecnologías biomédicas, como son los biosensores, junto al creciente desarrollo de tecnologías inalámbricas basadas en sistemas empotrados para "llevar puestos" (lo que conocemos hoy en día como dispositivo Wearable), está permitiendo el diseño, desarrollo e implementación de redes de área Corporal (conocido como WBAN Wiresless Body Area Network) [5].

Una red de área corporal (o como lo llamaremos a partir de ahora, WBAN), no es más que una red de comunicación inalámbrica entre dispositivos electrónicos. Estos dispositivos, como los relojes electrónicos, los Smartphone, o similares, trabajan en conjunto para poder facilitar la vida al usuario, en ocasiones, ofreciendo una forma de poder monitorizar elementos tan importantes, como por ejemplo la salud.

1.1 Motivación

El grupo de investigación COSEC Lab del Departamento de Informática de la UC3M está llevando a cabo el proyecto SPINY: *Security and Privacy in the Internet-of-You*[1]. Este proyecto utiliza el término "Internet of You" o IoY, para referirse a aquellas redes que, monitorizadas por los teléfonos inteligentes, son capaces de obtener información del usuario. Algunos ejemplos que el proyecto proporciona son*:*

[1] http://www.seg.inf.uc3m.es/spiny/

> *"Camisetas que proporcionan información en tiempo real para el usuario; pastilleros inteligentes que recuerdan a un paciente cuando es el momento de tomar la medicación y registrar cuando lo hace; y una nueva generación de dispositivos médicos inteligentes con capacidad de ser implantados, como marcapasos, bombas de insulina y neuroestimuladores".*

Las redes de este tipo, tienen problemas de seguridad y de privacidad más grandes que en la computación tradicional, ya que, este tipo de redes contienen dispositivos con multitud de sensores que pueden llegar a producir fugas de información sensible. El objetivo del proyecto *SPINY* es abordar estos problemas relacionados con la seguridad y privacidad de estas redes.

La tutora de este TFG está investigando en el marco del proyecto SPINY uno de los muchos problemas de privacidad que existen en redes de tipo IoY, como lo son las redes WBAN; "*Abordamos el problema de cómo ayudar a los usuarios a inicializar las políticas de seguridad y privacidad para aplicaciones de nueva implantación en redes inalámbricas de área corporal (WBAN)".* En el artículo titulado *Bootstrapping Security Policies for Wearable Apps using Attributed Structural Graphs* y publicado en [3], se expone la aproximación propuesta.

El objetivo de la investigación es por tanto ayudar a los usuarios a gestionar las políticas de privacidad de las aplicaciones instaladas en los dispositivos de su WBAN. Para ello, se hace uso de grafos generados para cada usuario a partir de la estructura de su red WBAN, de las aplicaciones instaladas en cada dispositivo, y de ciertos atributos asociados a los elementos de la WBAN. Se consideran dos grupos de atributos. Por un lado tendremos los atributos subjetivos, que serán definidos por el usuario, y por otro lado, atributos objetivos, que son inherentes a una red WBAN como la descrita.

El artículo demuestra de manera teórica que el uso de estos grafos permite elicitar agrupaciones de aplicaciones similares, teniendo en cuenta para la estimación de la similitud tanto la estructura de la WBAN, como los atributos objetivos y subjetivos especificados. La utilidad de estas agrupaciones es que, en caso de introducir una nueva aplicación en dicha red (en cualquiera de sus respectivos terminales / dispositivos), se facilite la búsqueda de la agrupación más afín a ella, de manera que se puedan reutilizar o adaptar las políticas de privacidad de dicha agrupación para la nueva aplicación.

Para poder demostrar que verdaderamente esta propuesta es además de factible teóricamente, es factible prácticamente y útil para los usuarios, es necesario poder

realizar un estudio real con aplicaciones, terminales, situaciones y sobre todo, con usuarios reales.

Para ello, la autora del artículo (a quien a partir de ahora llamaremos cliente), ha desarrollado un algoritmo en lenguaje de programación R, que dados ciertos grafos conteniendo la información mencionada previamente (estructura y atributos), separa las aplicaciones de una red WBAN en distintos grupos que contienen elementos similares, habiendo sido calculada dicha similitud usando los grafos. El cliente necesita que se desarrolle un sistema que permita evaluar el algoritmo.

1.2 Objetivos

El objetivo primordial de este Trabajo de Fin de Grado, es diseñar y desarrollar un sistema capaz de proporcionar a usuarios externos un mecanismo para representar su propia WBAN, de manera que esta representación pueda utilizarse para estimar con el algoritmo del cliente la similitud entre las aplicaciones de la WBAN y los grupos de aplicaciones óptimas para dicho usuario. Además, el sistema debe permitir que los usuarios evalúen los grupos de aplicaciones generadas por el algoritmo.

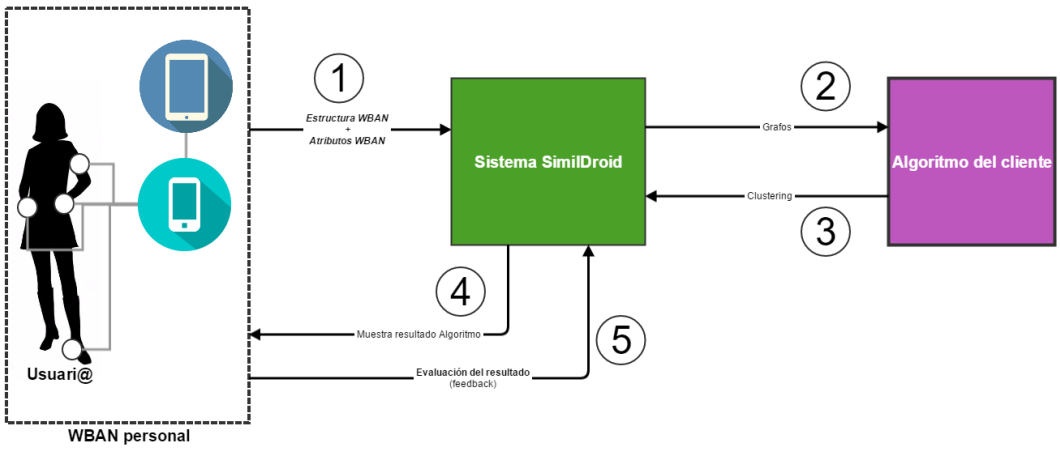

Ilustración 2 - Esquema básico de funcionamiento del sistema

La *Ilustración 2* es un pequeño esquema para poder entender el funcionamiento de *SimilDroid*.

La idea es que, en primer lugar, se extrae la estructura y algunos atributos pertenecientes a ella. El sistema *SimilDroid* es el encargado de recoger estos datos, y transformarlos para poder ser estudiados por el algoritmo del cliente. Cuando el algoritmo termina, devuelve los resultados que serán mostrados al usuario, quien será el encargado de realizar la evaluación del algoritmo (evaluando la salida del mismo).

Por tanto, una vez obtenido el resultado, es decir, los grupos de aplicaciones, el usuario pueda evaluar, de manera totalmente subjetiva, si el resultado obtenido por el sistema *SimilDroid* (que, como ya hemos comentado, se encarga de ejecutar el algoritmo del cliente), es correcto o no.

La evaluación que el usuario haga sobre el resultado obtenido, servirá al cliente para determinar si su algoritmo es eficaz, o si necesita algún tipo de ajuste para poder obtener, en un futuro, un resultado que satisfaga al usuario.

El objetivo que persigue este documento es mostrar el sistema desarrollado por el alumno, y que este sistema es de utilidad para la tutora del presente TFG (cliente) para poder evaluar, y en caso necesario ajustar, el algoritmo desarrollado.

1.3 Alcance del Sistema

Se debe matizar que una WBAN contendría un conjunto de dispositivos de muy diversa índole y naturaleza. Sin embargo, dado que el proyecto debe poder utilizarse con redes de dispositivos de usuario que puedan encontrarse con relativa facilidad en la actualidad, y que sea factible realizar la extracción de datos necesaria, los dispositivos que se considerarán en este Trabajo Fin de Grado son exclusivamente dispositivos Smartphone o Tablets con sistema operativo Android y dispositivos de tipo Wearable asociados a dichos dispositivos.

1.4 Estructura del documento

En este presente apartado, se realizará una breve descripción sobre la estructura que sigue el actual documento.

- **Introducción:** se trata del primer apartado del documento. En él, se pretende realizar una presentación del Trabajo de Fin de Grado, dando a conocer aspectos como el contexto, la motivación y los objetivos.

- **Estado del Arte:** en el desarrollo del estado del arte, o estado de la cuestión, se analiza el artículo escrito por la tutora del trabajo. Artículo que dará pie a la creación de la herramienta que se presenta en este documento.

- **Análisis del Sistema:** durante el análisis del sistema, se realizará un estudio detallado de las funcionalidades del sistema, mediante la definición de requisitos (tanto de usuario como de software), además de los casos de uso. Además, se presentarán las diferentes posibles estructuras (arquitecturas) del sistema, indicando cuál de ellas es la que tomará finalmente la herramienta *SimilDroid*.

- **Diseño del Sistema:** una vez determinada la arquitectura, durante la etapa de diseño del sistema, se concretarán aspectos relacionados con el diseño de los módulos del sistema.

- **Implementación:** durante el desarrollo de este capítulo, se analizarán algunos detalles relacionados con la etapa de desarrollo o implementación del código del sistema *SimilDroid*.

- **Evaluación del Sistema:** en el apartado de evaluación, se realizará la especificación del plan de pruebas del sistema. Además, se harán pruebas con usuarios reales para comprobar el funcionamiento del sistema.

- **Gestión del Proyecto:** este apartado mostrará la gestión del proyecto, teniendo en cuenta tanto la planificación realizada, como el coste del proyecto.

- **Conclusiones y líneas futuras:** este último apartado contendrá los resultados obtenidos tras la implementación del sistema *SimilDroid*, realizando comentarios al respecto y definiendo posibles futuras mejoras del sistema.

Al final del documento se podrá encontrar un apartado denominado *Bibliografía*, que contiene todas las fuentes y referencias consultadas para la escritura del presente documento.

Tras este apartado, como anexo, se ha añadido un *resumen en inglés* del documento que se presenta.

1.5 Definiciones y acrónimos

Con la idea de facilitar la comprensión lectora del documento, se añadirán en este apartado todas aquellas palabras o acrónimos que se considere, sean necesario su definición.

- ***Ad hoc:*** que es apropiado o adecuado para un determinado fin. En
- ***WBAN:*** Wiresless Body Area Network.
- ***Wearable:*** se refiere a los dispositivos "ponibles", es decir, que se pueden llevar puestos. Estos dispositivos incluye a todo tipo de sensores, relojes de última generación (iWatch, Gear, etc), pulseras inteligentes (Xiaomi Mi Band), etc.
- ***API:*** Application Programming Interface.
- ***SGBD*****:** Sistema Gestor de Base de Datos.

2. Estado de la cuestión

Tal y como se ha comentado anteriormente, la tutora del presente Trabajo de Fin de Grado, ha desarrollado un algoritmo que podrá ser evaluado utilizando el sistema que se describe en este documento. Este algoritmo, propuesto en el artículo *Bootstrapping Security Policies for Wearable Apps using Attributed Structural Graphs*, es el que se describirá a continuación.

En este capítulo de estado de la cuestión, no existe intención de mostrar otro tipo de tecnologías que se ajusten a la necesidad del cliente, ya que se trata de un sistema completamente ***ad hoc*** para un cliente determinado. Aunque existen otras propuestas, fundamentalmente académicas, que permiten estimar la similitud de aplicaciones Android, ninguna de ellas lo realiza a partir de los grafos que se utilizan en el algoritmo del cliente. Aquellos lectores interesados en estas otras propuestas, pueden consultar la sección de trabajos relacionados del artículo *Bootstrapping Security Policies for Wearable Apps using Attributed Structural Graphs*.

2.1 Introducción

En el artículo *Bootstrapping Security Policies for Wearable Apps using Attributed Structural Graphs*, se trata de abordar el problema de determinar qué política de seguridad y privacidad de las existentes, o una adaptación de ésta, es la mejor para inicializar la de una aplicación recién insertada en una red WBAN. Este tipo de redes, están compuestas por dispositivos inteligentes como *Smartphones, pulseras inteligentes, gafas inteligentes* y otro tipo de dispositivos, como lo son, aquellos ligados a atención médica.

Este paso de realizar la inicialización de la política de seguridad y privacidad (es decir, identificar de manera automática o, al menos guiada, las preferencias de privacidad del usuario), es, hoy en día, una necesidad para abordar el reto de conseguir que una red WBAN goce de una seguridad lo más personalizada posible.

Para entender la idea que se persigue en este artículo, se propone el siguiente ejemplo:

> *Consideremos, por ejemplo, un usuario con una red WBAN donde, varios de sus dispositivos tienen aplicaciones médicas. Así, por ejemplo, una de ellas tiene la capacidad de leer el pulso cardiaco del usuario. La idea es que, gracias a la*

correcta definición de políticas de seguridad en esta red, ninguna aplicación con fines no médicos pueda acceder a estos datos.

En el artículo se asume como hipótesis que aplicaciones similares según el usuario, es probable que tengan asociada una política de seguridad y privacidad similar. Por tanto, encontrar el grupo de aplicaciones similares a una nueva serviría para identificar de forma automática una posible política candidata para inicializar la aplicación nueva. El artículo *Bootstrapping Security Policies for Wearable Apps using Attributed Structural Graphs* se centra en estimar la similitud entre aplicaciones utilizando los grafos mencionados.

2.2 Modelado de la WBAN

El artículo parte de la suposición de que el usuario ya tiene aplicaciones instaladas y desplegadas en su red WBAN.

En primer lugar, en el artículo se propone modelar las redes WBAN como un Grafo No Dirigido al que denominaremos (G_a), cuyos vértices se corresponden con elementos de la WBAN, es decir, aplicaciones y dispositivos[2].

Las aristas de G_a, señalan únicamente las relaciones que existen entre cada uno de los elementos, sin ningún tipo de peso (únicamente muestran la estructura de la red). Si son entre dispositivos, son enlaces de comunicaciones, si son entre aplicaciones se refiere a despliegue en un dispositivo o a la relación existente entre aplicaciones asociadas pero desplegados sus componentes en varios dispositivos (esta relación generalmente debe existir en la actualidad en el caso de las aplicaciones instaladas en dispositivos Wearable).

[2] En el artículo también se consideran elementos de tipo recurso pero no se considerarán en este Trabajo Fin de Grado por la dificultad de obtener información respecto a este tipo de elementos.

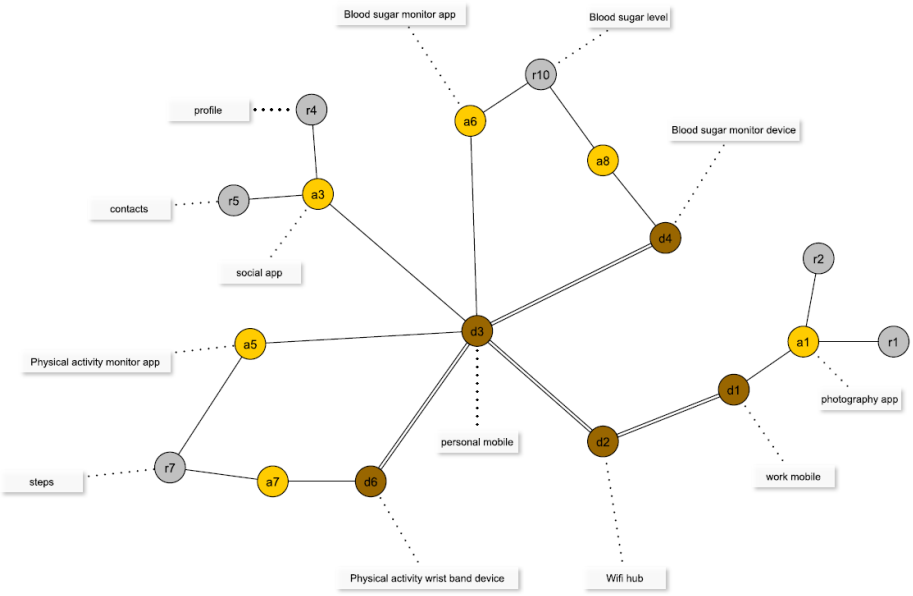

Ilustración 3 - Ejemplo de grafo Ga [3] (Imagen reproducida con el permiso de los autores)

Una vez construido este grafo, se realiza un aumento de éste añadiendo características esenciales a cada uno de los elementos (*atributos objetivos*, inherentes a cada elemento de G_a, como lo son el tipo, el autor, la categoría, los permisos, etc.). Los nodos que se añaden al grafo representan un valor concreto de un determinado atributo objetivo (e.g., considerando el atributo objetivo del nombre de la aplicación, se añadirían al grafo nodos "Facebook" y "Whatsapp").

Además de los atributos objetivos, se unirán *atributos subjetivos* que el usuario definirá en forma de etiquetas (o *tag*). En este caso los nodos que se añaden al grafo representarán una etiqueta concreta, que se considerará como su propio tipo de atributo objetivo.

El grafo resultante de añadir los nodos con los valores de los atributos objetivos y subjetivos se denominará como ***grafo estructural con atributos (G_b).***

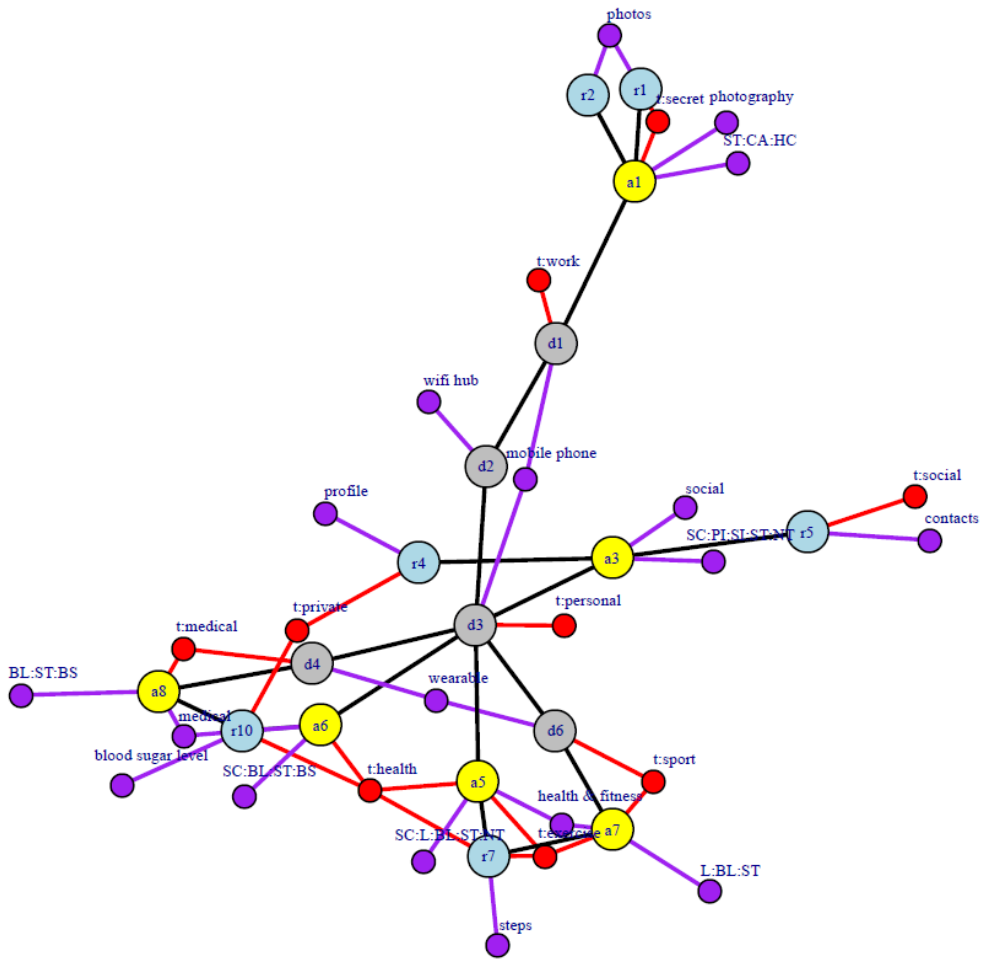

Ilustración 4 - Ejemplo de grafo Gb [3] (Imagen reproducida con el permiso de los autores)

A continuación, sobre este grafo G_b se vuelve a introducir información adicional, con el fin de enriquecer la estimación de similitud. Se incorpora al grafo la información de similitud entre valores de atributo correspondientes a un mismo atributo.

Para los nodos de tipo atributo que existen en G_b, para cada atributo (e.g., el nombre de la aplicación), se calcula la similitud entre todos los posibles valores de ese atributo existentes en el grafo G_b (e.g., entre "Facebook" y "Whatsapp"). En el caso de las etiquetas, se calcula la similitud entre todas las etiquetas por parejas. Esta similitud se incorpora como el peso de aristas que unen por parejas los nodos que representan valores diferentes de un mismo atributo o de etiquetas distintas. El grafo que contiene esta información de similitudes se denominará ***grafo de similitudes G_c***.

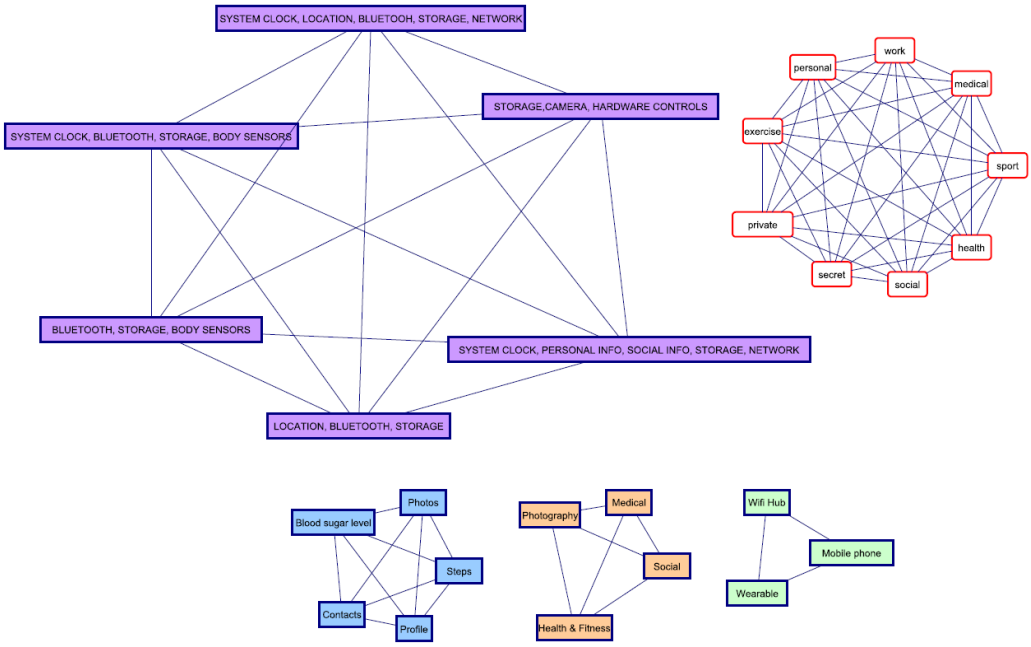

Ilustración 5 - Ejemplo de grafo Gc [3] (Imagen reproducida con el permiso de los autores)

Finalmente, el grafo G_b y G_c se unen para formar el grafo denominado G_u, ***grafo estructural con atributos y similitudes***. Este es el grafo que se utiliza para estimar la similitud entre todos los nodos del grafo y en particular entre los nodos de tipo aplicación.

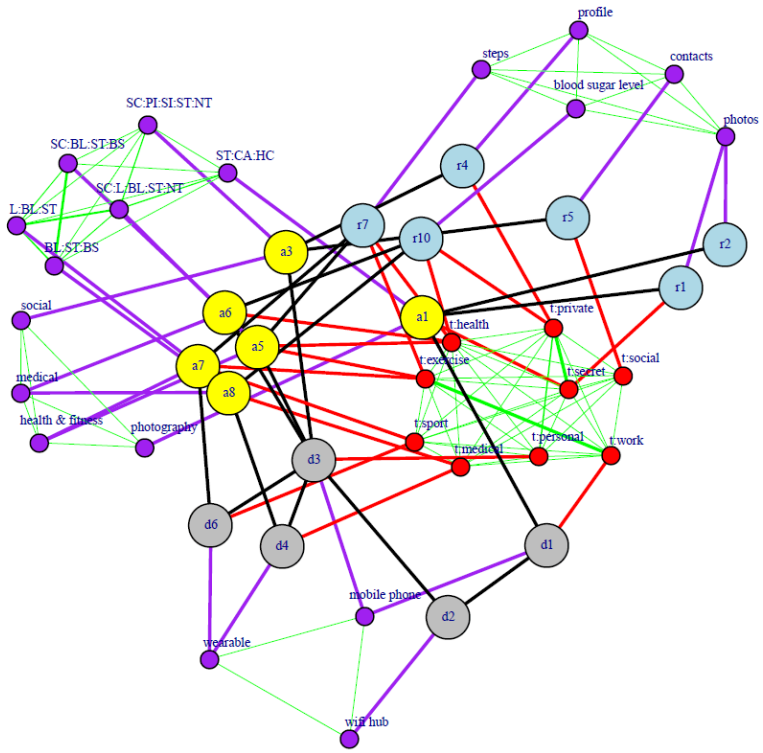

Ilustración 6 - Ejemplo de grafo Gu [3] (Imagen reproducida con el permiso de los autores)

2.3 Estimación de la similitud y obtención de los grupos de aplicaciones

Con G_u creado, se obtendrá la similitud de todos y cada uno de los nodos de la red, creando lo que llamaremos *Matriz de Similitud* (*MS*). Dicha matriz, como se ha comentado anteriormente, contiene la similitud entre todos y cada uno de los nodos pertenecientes al grafo G_c (absolutamente todos).

Para calcular la similitud se utiliza un algoritmo de "camino aleatorio" sobre el grafo (*Random Walk*) con una longitud determinada.

De la matriz *MS* se extrae la submatriz referente a las aplicaciones, y sobre ésta se aplica el algoritmo de clustering denominado PAM (*Partitioning Around Medoids*) estimando antes el número de clusters (o grupos) óptimo.

La idea a continuación, es repetir el cálculo de la matriz MS pero ajustando los pesos de los enlaces que existen entre nodos estructurales y nodos atributo[3]. Para la primera iteración, estos pesos son 1. El ajuste de pesos se estima a través de votaciones que destacan los atributos más influyentes en la homogeneidad de los clusters. Para ello, se calcula el número de coincidencias de valor para un mismo atributo entre los elementos de los clusters y sus medoids. El peso de los enlaces que hacen referencia a los atributos con más votaciones se incrementa, disminuyendo los pesos de aquellos atributos que tienen menos votos.

Al final, tras un número de iteraciones, la matriz MS debería mantenerse estable y el clustering obtenido debería converger a uno determinado. Este es el clustering que se mostraría al usuario como sugerencia de grupos de aplicaciones.

[3] En el artículo este proceso de ajuste de pesos no se describe en profundidad.

3. Análisis del Sistema

Durante el presente apartado de análisis se obtendrá una especificación detallada del sistema que se va a construir, que, más tarde, servirá como punto de partida para realizar el diseño del sistema. Por último, al final de este capítulo se estudiará el marco legal que envuelve a este Trabajo de Fin de Grado.

3.1 Definición del Sistema

El proyecto pretende realizar el siguiente funcionamiento. En primer lugar, se quiere obtener, tanto la estructura de la WBAN del usuario, como los atributos pertenecientes a la misma. Estos atributos se diferencian en: atributos objetivos (aquellos atributos que son inherentes a la arquitectura concreta del usuario; como el nombre de la aplicación, el nombre del paquete, los permisos, etc), y atributos subjetivos (que a diferencia de los anteriores, son definidos por el usuario; en este sistema serán las etiquetas).

Una vez obtenidos estos elementos, se enviarán al sistema central de *SimilDroid*, donde se terminarán de obtener datos necesarios de la WBAN (se obtendrán nuevos datos a raíz de los ya obtenidos; pe: categorías de las aplicaciones).

Cuando se tiene toda la información necesaria, se realizará la construcción de los grafos. Es importante señalar que el algoritmo cliente solicita dos grafos distintos:

- *Grafo estructural con atributos*: grafo no dirigido, que contiene las conexiones existentes en todos los elementos de la WBAN. Estas conexiones tienen en cuenta, tanto el esquema físico de la WBAN (conexiones físicas entre dispositivos), como conexiones lógicas entre estos y sus atributos (por ejemplo; si un dispositivo está etiquetado por el usuario como “Trabajo”, existirá una conexión entre el nodo relativo a ese terminal, y el nodo atributo “Trabajo”. Las conexiones entre nodos no tienen ningún tipo de peso.
- *Grafo de similitudes*: grafo no dirigido, que contiene la similitud entre atributos del mismo tipo. Así pues, por ejemplo, para cada categoría, existe una conexión con todas las demás categorías, indicando la similitud entre ellas. Es importante señalar que la similitud de este grafo estará normalizada, es decir, que la similitud entre dos nodos es [0,1].

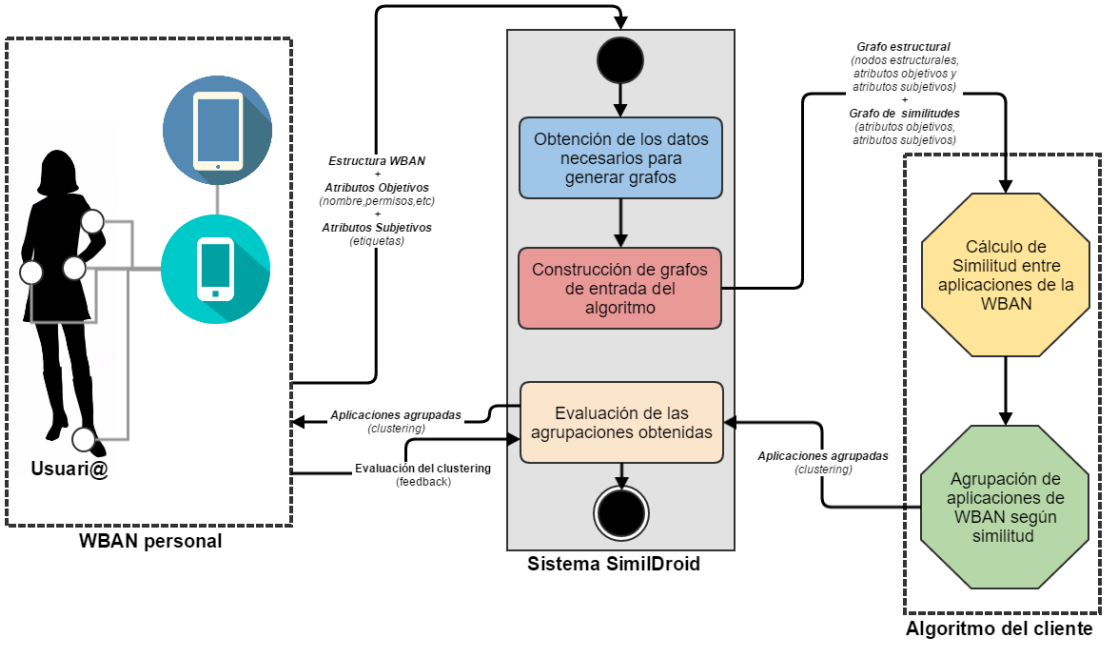

Ilustración 7 - Esquema de funcionamiento de SimilDroid

Tras realizar los grafos, el sistema los enviará al algoritmo cliente, encargado de realizar la similitud entre las aplicaciones incluidas dentro de la WBAN. Una vez calculada la similitud entre las aplicaciones, generará un *clustering* o agrupación de estas según corresponda. Esta agrupación se devolverá al sistema *SimilDroid*, el cual será el encargado de mostrarlo de forma adecuada al usuario.

Una vez que el usuario vea el resultado obtenido de la similitud, podrá realizar una evaluación de dicha salida, simplemente, reordenando las agrupaciones de aplicaciones. Si el usuario considera que un grupo no está lo suficientemente ordenado, o si en un grupo existe una o más aplicaciones que no deberían de estar ahí, sino en otro grupo, podrá moverlas donde considere.

El resultado será almacenado en el sistema para la posterior revisión del cliente. Esto servirá para poder determinar si el funcionamiento del algoritmo es correcto o necesita ajustarse.

Es importante dejar claro cuál es el alcance del sistema. Este sistema está orientado a evaluar el funcionamiento del cálculo de similitud entre aplicaciones que se desarrolla en el proyecto universitario del que ya se ha hablado anteriormente.

Para poder llevar a cabo dicha actividad, será necesario obtener un listado de todas las aplicaciones que el usuario evaluador tiene instalado en uno o más de sus dispositivos (pertenecientes a su propia WBAN), además de las estructuras que existe entre los dispositivos de la red.

Además, será necesario obtener ciertos atributos inherentes a cada una de las aplicaciones, lo que llamaremos *atributos objetivos*, como lo son; el nombre, la categoría de la aplicación y los permisos. Por último, será necesario otro tipo de atributos, pero, que a diferencia de los anteriores, serán definidos por el usuario, lo que llamaremos *atributos subjetivos*. Es importante detallar, que estos últimos atributos, los atributos adjetivos, en el artículo se definen como etiquetas que el usuario podrá poner a varios elementos de la red WBAN (aplicaciones y / o terminales).

El usuario evaluador, por lo tanto, podrá solicitar que se realice un análisis de similitud sobre las aplicaciones que tiene instaladas en los dispositivos que pertenecen a su WBAN (siempre aquellos con sistema operativo Android).

Así pues, el sistema se encargará realizar la ejecución del algoritmo desarrollado por el cliente, cuya salida será evaluada posteriormente por el mismo usuario.

El algoritmo a evaluar necesita varios elementos de entrada, que, el sistema *SimilDroid* será el encargado de generar:

Elementos de entrada del algoritmo cliente

- *Grafo estructural:* el grafo estructural no es más que un grafo no dirigido, que almacena la estructura de la red WBAN, junto a los atributos que lo describen. Así, existen diferentes tipos de nodos en este grafo:
  - Nodo estructural: todos aquellos nodos que representan un dispositivo (hub, móvil, wearable) o una app.
  - Nodo atributo objetivo: nombre de la aplicación (infinitas posibilidades), autor de la aplicación (infinitas posibilidades), categoría (grupo cerrado) y grupos de permisos (existen únicamente $2^{(\text{número de permisos de Android})}$ grupos de permisos).

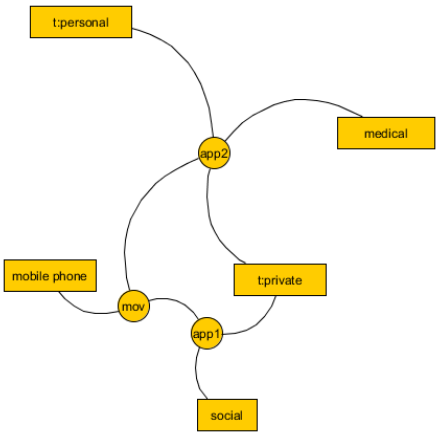

Ilustración 8 - Ejemplo grafo estructural

- Nodo atributo subjetivo: etiquetas definidas por el usuario. Estas etiquetas únicamente tendrán un solo término, es decir, una sola palabra, y nunca más de una.

- *Grafo de similitud:* este grafo realmente es un conjunto de subgrafos no conectados entre sí. Cada uno de los grafos se refiere a un tipo de atributo, de manera que, por cada atributo, todos sus posibles valores estarán conectados entre sí, donde el peso de la arista que los une, mostrará la similitud entre ellos. Así pues, existiran los siguientes subgrafos.
  - *Grafo de similitud de **permisos**:* el cliente ha definido que la similitud calculada para los permisos, debe seguir la Similitud Jaccard [7].
  - *Grafo de similitud* de ***nombres**:* en este caso, el cliente ha solicitado que la similitud entre nombres, se haga mediante la distancia de **Levenshtein** [7].
  - *Grafo de similitud de **etiquetas**:* en este caso, para la similitud entre etiquetas será necesario buscar alguna herramienta que encuentre la similitud semántica normalizada (similitud entre [0 y 1]) entre dos palabras cualesquiera.
  - *Grafo de similitud de **categorías**:* en el caso de las categorías, tendremos que hacer uso de la misma herramienta que las etiquetas.
  - *Grafo de similitud de **autores**:* según el cliente, la similitud entre autores será **1** si es el mismo autor, o será **0** si no lo es.

La salida que devuelve el algoritmo, junto a la evaluación que realiza el usuario a dicha salida, será lo que servirá al cliente como ayuda para poder valorar el buen o mal funcionamiento del algoritmo a valorar.

3.2 Requisitos de Usuario

Para que el funcionamiento del sistema sea correcto, y sobre todo, sea lo que el cliente quiere, se han definido lo que se conoce como **Requisitos de Usuario**, que no son más que requisitos que, durante varias reuniones, se han ido definiendo por el cliente junto al desarrollador del proyecto.

Cada uno de los requisitos definidos vendrá explicado mediante una tabla con el siguiente formato:

RU-XX			
Nombre			
Prioridad		**Necesidad**	
Descripción			

Tabla 1 - Modelo requisito de usuario

Donde:

- ***RU-XX:*** Servirá para identificar el requisito de Usuario de manera unívoca.
  - RU: Indica que es un requisito de Usuario.
  - XX: Número Comprendido entre 01 y 99 que identifica al requisito dentro de su tipo.

- ***Prioridad:*** Indica el valor de prioridad para facilitar la planificación del programador.

- ***Necesidad:*** El nivel de necesidad indicará si el requisito es negociable o no con el cliente.

- ***Descripción:*** Definición detallada del requisito.

RU-01			
Nombre	Obtener de cada terminal las aplicaciones instaladas.		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema tendrá que ser capaz de obtener una lista de todas las aplicaciones que están instaladas en cada uno de los terminales asociados a la WBAN del usuario.		

Tabla 2 - Requisito de usuario RU-01

RU-02			
Nombre	Captar atributos objetivos por cada aplicación obtenida.		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Por cada aplicación que se obtenga de la red WBAN, será necesario además, captar todos aquellos atributos que son		

	inherentes a la aplicación (atributos objetivos). Estos atributos son los siguientes: ○ Permisos de la aplicación. ○ Nombre de la aplicación. ○ Autor de la aplicación. ○ Nombre del paquete Android al que pertenece la aplicación. ○ Categoría a la que pertenece la aplicación.

Tabla 3 - Requisito de usuario RU-02

RU-03			
Nombre	Captar atributos subjetivos por cada aplicación y dispositivo obtenidos.		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Por cada aplicación y dispositivo que se obtenga de la red WBAN, será necesario además, captar todos aquellos atributos que son, necesariamente definidos por el usuario. Es lo que conocemos como etiquetas o tags (atributos objetivos).		

Tabla 4 - Requisito de usuario RU-03

RU-04			
Nombre	Generar grafo estructural con atributos		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema tendrá que ser capaz de generar el grafo estructural para poder enviarlo posteriormente al algoritmo proporcionado por el cliente.		

Tabla 5 - Requisito de usuario RU-04

RU-05			
Nombre	Generar grafo de similitudes		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema tendrá que ser capaz de generar el grafo de similitudes para poder enviarlo posteriormente al algoritmo proporcionado por el cliente.		

Tabla 6 - Requisito de usuario RU-05

RU-06			
Nombre	Ejecutar algoritmo del cliente		
Prioridad	Alta	**Necesidad**	Esencial

Descripción	El sistema tendrá que ser capaz de ejecutar el algoritmo del cliente.

Tabla 7 - Requisito de usuario RU-06

RU-07			
Nombre	Mostrar resultados al usuario		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema tendrá que ser capaz de mostrar los resultados del algoritmo al cliente. El algoritmo devolverá clusters de aplicaciones ordenados según su similitud.		

Tabla 8 - Requisito de usuario RU-07

RU-08			
Nombre	Obtener feedback del cliente		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema tendrá que ser capaz de obtener el feedback del usuario. Este feedback será en forma de reordenamiento del resultado del algoritmo según él vea conveniente.		

Tabla 9 - Requisito de usuario RU-08

RU-09			
Nombre	Formato .graphml de los grafos		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El formato de los grafos generados por el sistema, ha de ser .graphml. Los nodos del *graphml* deben llevar un conjunto de metadatos requeridos por el algoritmo cliente en forma de atributos de los nodos. Nótese que estos atributos no son los mismos que los nodos atributos que permiten completar el modelo de la WBAN.		

Tabla 10 - Requisito de usuario RU-09

3.3 Arquitectura del Sistema

En este sub-apartado, se pretende mostrar una visión global de los componentes del sistema, estudiando las posibles arquitecturas, y, definiendo la arquitectura final, según los requisitos que ha definido el cliente.

Existen varias organizaciones o arquitecturas posibles para realizar lo que el cliente necesita:

Arquitectura "Solo Android"

Una de las opciones posibles para realizar el proyecto, sería hacer uso únicamente de una aplicación Android que, obtenga todas las aplicaciones de la red WBAN, además de todos los atributos necesarios, y ejecute el algoritmo del cliente en alguno de los terminales de la red WBAN del usuario (terminal maestro).

Esta arquitectura tiene un problema muy grave y es la necesidad de utilizar el lenguaje R para poder ejecutar el programa del cliente. Se podría solucionar traduciendo el código a Java (para poder ejecutarlo bajo una .apk), pero existirían problemas con la eficiencia del programa, y la ejecución para un solo cliente podría llegar a hacerse muy larga (a parte del consumo de batería que supondría la ejecución del algoritmo durante tanto tiempo).

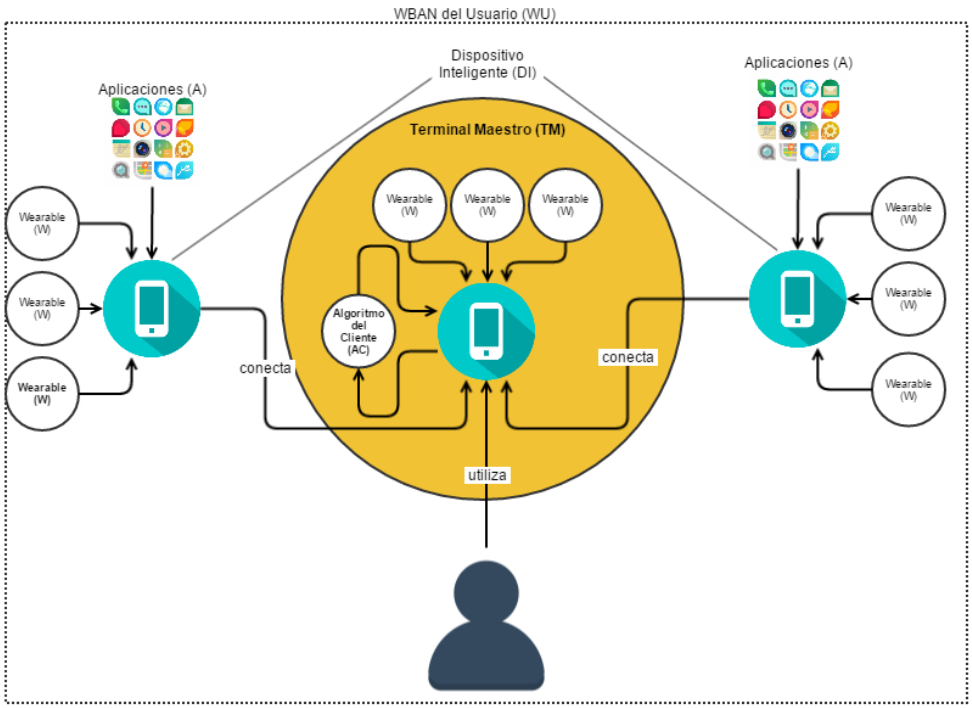

Ilustración 9 - Esquema Arquitectura "Solo Android"

Arquitectura "Android + Servidor con R"

Esta arquitectura, en principio, parece que es la mejor idea. La aplicación Android realiza todo el trabajo necesario, generando los grafos necesarios y luego enviándolos

al servidor. Más tarde, el servidor se encargará de ejecutar el código en R, y devolver el resultado para que la aplicación Android la muestre.

A primera vista, esta es una buena manera de solventar el problema del cliente. Pero, existen varios impedimentos que hacen que esta arquitectura no sea la adecuada.

En primer lugar, no podría almacenarse en una base de datos general, la información de todos los usuarios del sistema, por lo que no existiría un control total de la información del sistema por parte del administrador del mismo.

Además, en segundo lugar, los resultados obtenidos por el algoritmo programado en R, devolverá información que, preferiblemente debería de visualizarse en una pantalla relativamente grande, algo que, no se cumple para los Smartphone comunes.

Por último, el gran inconveniente de esta arquitectura es, la necesidad de tener un conocimiento elevado sobre Java y sobre todo, en programación Android, requisito, que no cumple el desarrollador del proyecto, por lo que será necesario encontrar una alternativa.

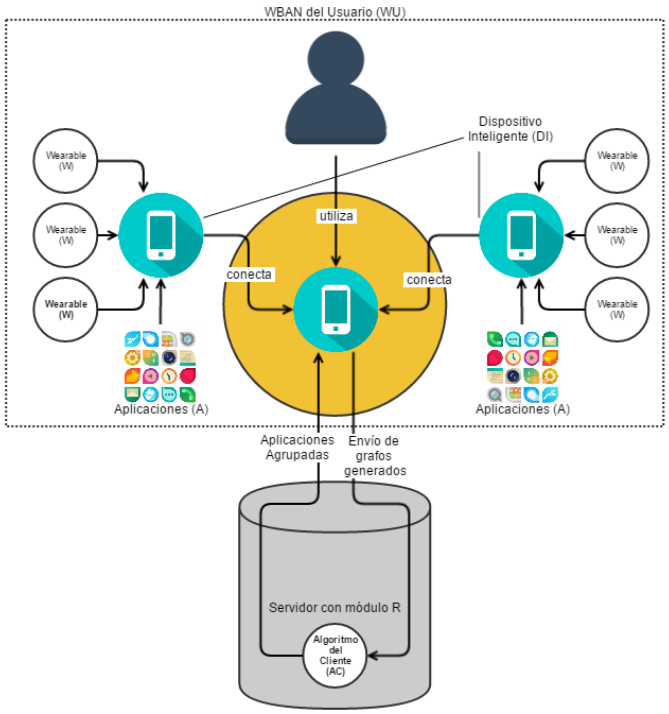

Ilustración 10 - Esquema Arquitectura "Android + Servidor con R"

Arquitectura "Android + Servidor Web PHP"

El elevado conocimiento del programador del proyecto hace, que el uso del lenguaje de programación PHP sea una gran ventaja.

A diferencia que en la arquitectura anterior, la aplicación Android genera únicamente una lista de aplicaciones que enviará posteriormente al Servidor Web. El mencionado servidor, será el encargado de almacenar toda la información relevante en su base de datos.

El usuario tendrá que registrar uno a uno todos los dispositivos de su red WBAN en el servidor, haciendo uso de la aplicación Android. Una vez registrados todos y cada uno de los terminales (Wearable incluidos), las gestiones se llevarán a cabo a través del Servidor Web.

En parte, esta arquitectura hace que el trabajo sea más incómodo, ya que, el usuario tendrá que usar, por una parte la aplicación *SimilDroid.apk*, y por otra, el Servidor Web, pero este problema se compensa con las ventajas que proporciona. Al realizarse en un entorno cómodo para el programador del proyecto, hará que el desarrollo sea mucho más rápido y satisfactorio.

Por lo tanto, esta arquitectura será la seleccionada para el sistema de *SimilDroid*.

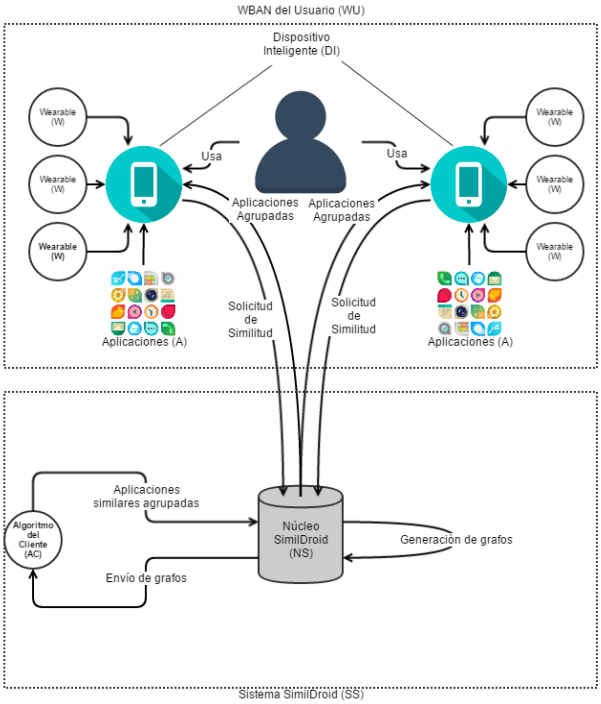

Ilustración 11 - Esquema Arquitectura "Android + Servidor Web PHP"

El sistema ***SimilDroid***, por lo tanto, estará compuesto por los siguientes componentes básicos:

- **WBAN del Usuario (WU):** se refiere al conjunto de dispositivos asociados a un mismo usuario.
- **Dispositivo Inteligente (DI):** dispositivo inteligente capaz de soportar el sistema operativo Android, además de poder instalar aplicaciones en él. Normalmente, este dispositivo se corresponderá con un Smartphone, Tablet o similar.
- **Wearable (W):** dispositivo electrónico que "se puede llevar", o que "es ponible". Este dispositivo estará conectado a una o más aplicaciones de alguno o algunos de los DI de la WBAN del usuario.
- **Aplicaciones (A):** las aplicaciones instaladas en cada uno de los DI.
- **Núcleo de SimilDroid (NS):** se refiere al conjunto de elementos que realizaran las pertinentes funciones para obtener los grafos solicitados por el algoritmo del cliente.
- **Algoritmo del cliente (AC):** se refiere al algoritmo que el cliente aporta para su posterior evaluación.

3.4 Definición de los casos de uso

Durante el desarrollo de esta cuestión, se presenta, de manera general, las posibles interacciones que existirán en el sistema entre éste, y el usuario. Estos casos de uso también ayudarán a definir posteriormente tanto los Requisitos Funcionales, como los No Funcionales, ya que, de una manera u otra, estos dos conceptos están muy ligados.

Para exponer los casos de uso se empleará una tabla a modo de plantilla con la siguiente estructura:

CU-XX	
Nombre	
Descripción	
Precondiciones	
Postcondiciones	
Condiciones de fallo	
Prioridad	

Tabla 11 - Formato de caso de uso

Donde;

- ***CU-XX***: identifica unívocamente al caso de uso. Se define como CU-XX, donde XX es un número que representa el caso de uso.
- ***Nombre:*** nombre que se le da al caso de uso.
- ***Descripción:*** breve explicación del proceso llevado a cabo en el caso de uso.
- ***Pre-condiciones:*** son aquellos hechos que deben ser previamente ciertos para poder llevar a cabo el caso de uso.
- ***Post-condiciones:*** son aquellos hechos que la correcta ejecución del caso de uso los hacen posibles.
- ***Condiciones de fallo:*** descripción de los posibles fallos que pueden sucederse en el presente caso de uso, y la respuesta.
- ***Prioridad:*** nivel de prioridad que tiene el caso de uso con respecto al resto.

CU-01	
Nombre	Registro en el Sistema de *SimilDroid*
Descripción	Para poder hacer uso de la herramienta *SimilDroid*, el usuario deberá registrarse en el Sistema, dando los siguientes datos: a. Nombre de Usuario. b. Contraseña. c. Correo Electrónico.
Prioridad	Alta
Precondiciones	El usuario necesita conexión a internet para poder acceder al servicio web de *SimilDroid*
Postcondiciones	El usuario queda registrado en la base de datos de SimilDroid. A partir de aquí, el usuario podrá loguearse.
Condiciones de fallo	Nombre de usuario erróneo. Contraseña y confirmación de contraseña diferentes. Correo electrónico ya asociado a otro usuario.

Tabla 12 - Caso de uso CU-01

CU-02	
Nombre	Login en el Sistema de *SimilDroid*
Descripción	Para poder hacer uso de la herramienta *SimilDroid*, el usuario deberá loguearse.
Prioridad	Alta
Precondiciones	El usuario necesita conexión a internet para poder acceder al servicio web de *SimilDroid*, además de ya estar registrado en el sistema.
Postcondiciones	El usuario queda logueado en el sistema de SimilDroid.
Condiciones de fallo	Nombre de usuario erróneo. Contraseña errónea.

Tabla 13 - Caso de uso CU-02

CU-03	
Nombre	Obtener aplicaciones instaladas en el terminal
Descripción	El sistema necesita que el usuario envíe una lista de las aplicaciones que tiene instaladas en su terminal. Para ello, el usuario hará uso de la App SimilDroid que se ha desarrollado en este TFG.
Prioridad	Alta

Precondiciones	Una vez el usuario tiene asignado un nombre de Usuario (nickname) y una contraseña; deberá descargar la aplicación SimilDroid en el terminal que quiera evaluar. Para ello tendrá que acceder a la página web del Sistema y descargar el fichero ".apk". Una vez instalada, deberá loguearse.
Postcondiciones	La base de datos se actualiza, y quedan almacenadas las aplicaciones que tiene el usuario instaladas en el terminal, además de atributos como los permisos y el nombre del paquete de cada aplicación. También, y de forma simultánea, queda almacenado el terminal sobre el que se ha realizado el listado de aplicaciones.
Condiciones de fallo	Fallo de conexión con el servidor (terminal sin internet).

Tabla 14 - Caso de uso CU-03

CU-04	
Nombre	Registrar nuevo dispositivo Wearable
Descripción	El sistema está preparado para añadir dispositivos Wearable al estudio del análisis. Así pues, el hecho de añadir un elemento Wearable, puede influir en la salida del análisis de Similitud que posteriormente se llevará a cabo.
Prioridad	Media - Alta
Precondiciones	Será necesario que al menos se haya realizado el listado de aplicaciones de un terminal para así poder asociar el dispositivo Wearable a una (o más) de las aplicaciones que tiene instaladas el usuario en alguno de sus terminales.
Postcondiciones	Una vez hecho esto, se almacenará en el sistema el nuevo terminal, indicando cuales son las aplicaciones a las que está asociado.
Condiciones de fallo	Creación de un terminal Wearable sin asignar ninguna aplicación asociada.

Tabla 15 - Caso de uso CU-04

CU-05	
Nombre	Solicitar nuevo análisis de similitud entre aplicaciones
Descripción	El usuario será el encargado de evaluar que el algoritmo funcione correctamente. Es por ello que es necesario permitir

	al usuario que sea él mismo quien pida realizar el análisis de similitud.
Prioridad	Media - Alta
Precondiciones	Será necesario que se haya introducido al menos un terminal con una cantidad razonable de aplicaciones (al menos ocho) para poder solicitar el análisis de similitud. Será necesario haber realizado el etiquetado de aplicaciones para que el análisis surta efecto.
Postcondiciones	Una vez solicitado el análisis, se creará una entrada en la base de datos indicando que el usuario ha gestionado la solicitud de análisis de similitud, por lo que cuando el sistema sea capaz, comenzará con el análisis.
Condiciones de fallo	No existe ningún terminal ni aplicación asociados al usuario.

Tabla 16 - Caso de uso CU-05

CU-06	
Nombre	Realizar Evaluación de Similitud
Descripción	Una vez terminado el análisis de similitud entre las aplicaciones del usuario, éste será el encargado de realizar una evaluación del análisis.
Prioridad	Alta
Precondiciones	El usuario recibirá un correo electrónico a su correo personal, indicando que ya ha terminado el análisis, por lo que ya podrá acceder a realizar la evaluación.
Postcondiciones	Una vez realizada la evaluación, se almacenará el resultado en la base de datos para que posteriormente el Administrador del Sistema pueda comparar los valores devueltos por el Algoritmo y los devueltos por el usuario.
Condiciones de fallo	N / A

Tabla 17 - Caso de uso CU-06

CU-07	
Nombre	Añadir etiqueta a nodo (terminal o aplicación)
Descripción	Todas y cada una de las aplicaciones y de los terminales podrán ser etiquetadas por el usuario. Estas etiquetas son personales,

	de manera que cada usuario podrá tener su propio dominio de etiquetas.
Prioridad	Alta
Precondiciones	Para poder etiquetar una aplicación o a un terminal, en primer lugar habrá que añadir la etiqueta al dominio del usuario. Una vez añadida, podrá realizarse el etiquetado.
Postcondiciones	Una vez realizado el etiquetado, quedará almacenada dicha etiqueta en el sistema, junto a la unión entre la etiqueta y el elemento etiquetado.
Condiciones de fallo	N / A

Tabla 18 - Caso de uso CU-07

3.5 Definición de requisitos de software.

En el presente apartado se muestran los requisitos de software, que no son más que la descripción de las características necesarias para el desarrollo del sistema. Estos requisitos se han obtenido, por una parte, de los Requisitos de Usuario *(véase apartado 2.2)*. También, por la otra parte, existe una definición de requisitos que, por la naturaleza del sistema, necesitan ser declarados.

Cada uno de los requisitos definidos vendrá explicado mediante una tabla con el siguiente formato:

RY-XX			
Nombre			
Prioridad		**Necesidad**	
Descripción			

Tabla 19 - Formato de requisito de software

Donde:

- ***RY-XX:*** Servirá para identificar el requisito de manera unívoca.
  - R: Indica que es un requisito.
  - Y: Indica el tipo de requisito, y admite los valores:
    - F: de manera que el requisito será considerado Funcional.

        - NF: de manera que el requisito será considerado No Funcional.
    - XX: Número Comprendido entre 01 y 99 que identifica al requisito dentro de su tipo.

- ***Prioridad:*** Indica el valor de prioridad para facilitar la planificación del programador.

- ***Necesidad:*** El nivel de necesidad indicará si el requisito es negociable o no con el cliente.

- ***Descripción:*** Definición detallada del requisito.

Por lo tanto, así se definen los requisitos del sistema:

3.5.1 Requisitos Funcionales

Los Requisitos Funcionales no son más que la definición de qué ha de hacer el sistema para satisfacer las necesidades del cliente y el usuario.

RF-01			
Nombre	Registro del usuario		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema permitirá a un usuario cualquiera registrarse en el sistema de SimilDroid.		

Tabla 20 - Requisito funcional RF-01

RF-02			
Nombre	Login del usuario		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Para poder hacer uso del sistema de SimilDroid, se permitirá que el usuario se loguee en el sistema.		

Tabla 21 - Requisito funcional RF-02

RF-03			
Nombre	Añadir terminales al sistema		
Prioridad	Alta	**Necesidad**	Esencial

Descripción	El usuario podrá añadir terminales al sistema, de los cuales posteriormente se obtendrán las aplicaciones.

Tabla 22 - Requisito funcional RF-03

RF-04			
Nombre	Almacenar aplicaciones instaladas		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema podrá realizar una búsqueda de las aplicaciones que tiene instaladas el usuario en cada uno de sus dispositivos móviles.		

Tabla 23 - Requisito funcional RF-04

RF-05			
Nombre	Visualizar terminales y aplicaciones instaladas.		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El usuario podrá visualizar las aplicaciones que tiene instaladas en cada uno de sus terminales.		

Tabla 24 - Requisito funcional RF-05

RF-06			
Nombre	Etiquetar aplicaciones		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El usuario podrá etiquetar de forma individual todas y cada una de sus aplicaciones.		

Tabla 25 - Requisito funcional RF-06

RF-07			
Nombre	Etiquetar terminales (incluido wearables)		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El usuario podrá etiquetar de forma individual todos y cada uno de sus terminales.		

Tabla 26 - Requisito funcional RF-07

RF-08			
Nombre	Asociar dispositivo Wearable		
Prioridad	Alta	**Necesidad**	Media
Descripción	El usuario podrá asociar un dispositivo Wearable a una, o más, aplicaciones de sus terminales. Importante detallar, que en el caso de esta herramienta, los Wearable son ficticios, es decir, que el sistema no tomará la información relativa de este dispositivo de manera automática. Por lo que tendrá que ser el		

	propio usuario el que indique que tiene un Wearable, y que está asociado a X grupo de aplicaciones.

Tabla 27 - Requisito funcional RF-08

RF-09			
Nombre	Solicitar análisis de similitud		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El usuario podrá solicitar que se realice un análisis de similitud entre las aplicaciones que ha introducido en el sisma SimilDroid.		

Tabla 28 - Requisito funcional RF-09

RF-10			
Nombre	Evaluar análisis de similitud		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El usuario podrá evaluar la salida que el sistema proporcione a la similitud entre aplicaciones.		

Tabla 29 - Requisito funcional RF-10

RF-11			
Nombre	Cálculo de similitud normalizada		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Para la creación del grafo de similitudes, será necesario calcular la similitud existente entre todos y cada uno de los atributos pertenecientes a la WBAN, siempre de manera normalizada (similitud entre 0 y 1).		

Tabla 30 - Requisito funcional RF-11

3.5.2 Requisitos No Funcionales

Los Requisitos No Funcionales son aquellos que derivan de los requisitos de usuario de restricción, es decir, que definen ciertas restricciones que se aplicarán al sistema.

RNF-01			
Nombre	Disponibilidad del sistema		
Prioridad	Alta	**Necesidad**	Esencial

Descripción	El sistema SimilDroid deberá de ofrecer el servicio que promete siempre y cuando el servidor sobre el que está montado esté en funcionamiento.

Tabla 31 - Requisito no funcional RNF-01

RNF-02			
Nombre	SimilDroid en Android		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema SimilDroid incorporará una aplicación Android (SimilDroid.apk), que el usuario tendrá que instalar en su dispositivo para poder usar la herramienta.		

Tabla 32 - Requisito no funcional RNF-02

RNF-03			
Nombre	SimilDroid en la Web		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema SimilDroid proporciona una interfaz web para que los usuarios puedan visualizar los terminales, las aplicaciones y permisos instalados. Además podrán solicitar que se realice el análisis mediante esta herramienta.		

Tabla 33 - Requisito no funcional RNF-03

RNF-04			
Nombre	Acceso al sistema		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Cualquier usuario con uno o más terminales Android, podrá acceder al contenido del sistema, siempre y cuando se registre y loguee primeramente en el sistema.		

Tabla 34 - Requisito no funcional RNF-04

RNF-05			
Nombre	Aviso de fin de análisis del sistema		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El sistema SimilDroid avisará por correo electrónico al usuario de que el análisis solicitado se ha realizado correctamente.		

Tabla 35 - Requisito no funcional RNF-05

RNF-06			
Nombre	Seguridad en la comunicación		
Prioridad	Alta	**Necesidad**	Esencial

Descripción	Todas las comunicaciones internas entre las partes del sistema, se realizarán mediante seguridad SSL/TSL.

Tabla 36 - Requisito no funcional RNF-06

RNF-07			
Nombre	Datos sensibles del usuario		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Toda la información sensible del usuario se almacenará en la base de datos de manera que se cumpla la Ley Orgánica de Protección de Datos (LOPD)		

Tabla 37 - Requisito no funcional RNF-07

RNF-08			
Nombre	Tiempo de espera de cálculo de Similitud		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	El tiempo de espera entre que el usuario solicita la similitud, hasta que recibe el resultado, puede ir desde 2, hasta 72 horas.		

Tabla 38 - Requisito no funcional RNF-08

RNF-09			
Nombre	Obtención de atributos objetivos no almacenados en el terminal		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Algunos atributos objetivos necesarios para la creación de los grafos, no pueden obtenerse directamente desde la aplicación (como la categoría). Por ello será necesario utilizar alguna API externa que resuelva el problema.		

Tabla 39 - Requisito no funcional RNF-09

RNF-10			
Nombre	Comunicación con API's externas		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Las comunicaciones con las API's externas se realizaran haciendo uso del formato de texto ligero JSON.		

Tabla 40 - Requisito no funcional RNF-10

RNF-11			
Nombre	Almacén de contraseñas		
Prioridad	Alta	**Necesidad**	Esencial

Descripción	La contraseña del usuario en claro no quedará almacenada (en ningún momento) en el sistema. A cambio quedará almacenado su resumen.

Tabla 41 - Requisito no funcional RNF-11

RNF-12			
Nombre	Dos usuarios distintos, nicks distintos		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	En el sistema no pueden existir dos usuarios con el mismo nombre de usuario (o Nick).		

Tabla 42 - Requisito no funcional RNF-12

RNF-13			
Nombre	Inyección de código en la base de datos		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Se intenta evitar la inyección de sentencias MySQL mediante el uso de elementos preparativos de sentencias (prepare statement).		

Tabla 43 - Requisito no funcional RNF-13

RNF-14			
Nombre	Similitud entre categorías ya precargado		
Prioridad	Alta	**Necesidad**	Esencial
Descripción	Al tratarse de un grupo cerrado (y no muy grande), se ha decidido realizar el precalculado de la similitud entre todas y cada una de las categorías.		

Tabla 44 - Requisito no funcional RNF-14

3.5.3 Matrices de trazabilidad

En este apartado se realizaran varias tablas de trazabilidad donde se puede ver la coherencia entre los requisitos de software y los casos de uso.

- **Requisitos Funcionales / Casos de uso:**

	CU-01	*CU-02*	*CU-03*	*CU-04*	*CU-05*	*CU-06*	*CU07*
RF-01							
RF-02							
RF-03							
RF-04							
RF-05							
RF-06							
RF-07							
RF-08							
RF-09							
RF-10							
RF-11							

Tabla 45 - Trazabilidad requisitos funcionales y casos de uso

- **Requisitos No Funcionales / Casos de uso:**

	CU-01	*CU-02*	*CU-03*	*CU-04*	*CU-05*	*CU-06*	*CU-07*
RNF-01							
RNF-02							
RNF-03							
RNF-04							
RNF-05							
RNF-06							
RNF-07							
RNF-08							
RNF-09							
RNF-10							
RNF-11							
RNF-12							
RNF-13							
RNF-14							

Tabla 46 - Trazabilidad requisitos no funcionales y casos de uso

3.6 Análisis de las Tecnologías

En el presente sub-apartado, se van a analizar las posibles tecnologías que se pueden usar para los diferentes módulos del sistema.

3.6.1 Tecnologías para el gestor de la Base de Datos

Siempre es importante que en cualquier sistema exista un gestor de base de datos rápida, fácil de usar, y que ocasione los menos problemas posibles. Pero en este caso, en nuestro sistema, lo más importante es que sea sencillo de usar, de implementar, y que ocasione los mínimos errores posibles.

Opción 1 - MySQL (Elegida)

Se trata de un sistema que gestiona bases de datos relacionales, multihilo y multiusuario. La particularidad de este gestor es que funciona sobre múltiples plataformas, entre ellas Mac OS X, GNU/Linux y Windows. Destaca por su robustez, su escalabilidad, disponibilidad y fácil despliegue, siendo muy utilizado en aplicaciones web como Joomla, sistemas programados con PHP o plataformas Linux/Windows-Apache-MySQL-Python. Algunos de sus beneficios son:

- Gestor Multiplataforma.
- Soporta gran manejo de datos.
- Conexión segura.
- Multihilo.
- **Licencia Libre (GNU)**

Opción 2 - Oracle

Oracle se trata de un **SGBD**, que trabaja bajo el modelo objeto-relacional. Oracle Database, se considera como uno de los sistemas de gestión de base de datos más completos del mercado. Tanto es así, que hasta hace relativamente poco tiempo, ha sido líder indiscutible en este sector. Algunas de sus ventajas son:

- Estabilidad.
- Escalabilidad.

- Multiplataforma.
- Durabilidad.

Finalmente, el SGBD elegido es MySQL. El motivo es muy sencillo, MySQL es lo suficientemente potente como para soportar una base de datos definida para un sistema como SimilDroid, ya que no va a tener una carga muy elevada de información (aunque lo soporte). Además, se trata de un gestor totalmente gratuito.

3.6.2 Programación de SimilDroid

Existen una multitud de lenguajes de programación especializados en Web. Desde Java, ASP, Python, Ruby, PHP, etc. Todos tienen puntos fuertes que pueden ayudar a que el trabajo sea más sencillo, y puntos no tan fuertes, que pueden llegar a ocasionar retrasos en el proyecto.

En este caso, el lenguaje que se ha decidido utilizar es PHP por muchos motivos.

- Lenguaje antiguo pero en continuo crecimiento, lo que hace que la comunidad sea muy grande y experimentada. Incluso el mismo grupo desarrollador de PHP proporciona un manual completo de su lenguaje de programación en la página oficial **http://php.net/manual/es.**
- Lenguaje multiplataforma.
- Lenguaje con gran cantidad de extensiones que permiten conectar el mismo sistema de PHP con otras plataformas como JAVA, Unix, MySQL, etc.

Por otro lado, *SimilDroid* está compuesto por una parte móvil, es decir, una parte de la programación de *SimilDroid* será orientada a los dispositivos móviles. Más concretamente Android, ya que el algoritmo que se ha desarrollado en [3], estudia las aplicaciones de este Sistema Operativo.

3.6.3 Servidor de SimilDroid

Para la puesta en marcha del servidor Web, se va a utilizar el servidor HTTP Apache2, ya que se trata de servidor de código abierto que funciona bajo cualquier plataforma (Windows, Linux). Además, se trata de un servidor que lleva en desarrollo desde 1995, lo que hace que ya exista mucha documentación respecto a la configuración del mismo.

Este servidor se montará sobre sistema Operativo Ubuntu Linux, ya que, como es sabido, se trata de un sistema operativo libre con una excelente comunidad.

3.6.4 Librerías y API's

En el desarrollo de los grafos, será necesario solventar el problema que existe con el cálculo de similitud entre categorías, y entre etiquetas. Es necesario encontrar una herramienta que, dadas dos palabras, devuelva la similitud semántica **normalizada** entre esos dos términos.

Existe una base de datos léxica en inglés, llamada ***Word NET*** [13] que agrupa términos que pueden ser: sustantivos, verbos, adjetivos o adverbios, según la similitud entre ellos. El problema es que es necesario "crear una función" que dadas las dos palabras, realice la consulta necesaria a esta base de datos. Para ello ya existe la herramienta **Semilar** [12].

Semilar es una aplicación programada en lenguaje de programación JAVA que dada una lista de pares de palabras, devuelve la misma lista añadiendo la similitud entre esas dos palabras. Por tanto, haremos uso de esta herramienta para calcular la similitud entre etiquetas y categorías.

Con el uso de Semilar, aparece un nuevo problema, y es que ***WORD NET 3.0*** trabaja únicamente con palabras en inglés. Realmente esto no es un gran problema, pero sí que puede dificultar el entendimiento del usuario, por lo que será necesario buscar una solución a este problema.

Google Translate ofrece un API para poder realizar consultas a la base de datos del mismo. Haremos uso de la API de Google Translate para poder traducir las categorías y las etiquetas de cada usuario y así poder hacer uso de Semilar sin problema alguno.

Por último, se hará uso de la API ***42matters*** [15]. Esta API, recibe el nombre de un paquete Android, y devuelve toda la información relativa a él (y por tanto a la aplicación que contiene el paquete). Así, realizando peticiones a esta API, podremos obtener varios atributos objetivos para la posterior creación de nuestros grafos. Los elementos que obtenemos de esta API son el *autor* y la *categoría* a la que pertenece la aplicación.

3.7 Marco legal del proyecto

Este proyecto se basa en capturar información relevante acerca de usuarios reales, con terminales y datos reales. Es por ello, que durante el desarrollo de todo el proyecto se evitará almacenar información importante del usuario.

Según la *Ley Orgánica de Protección de Datos* (LOPD) [6], el usuario tiene derecho a conocer qué datos van a ser recogidos por el sistema, y está en su derecho de no dar información que no desea. Es por ello que, cada vez que se requiere un dato relevante del usuario, se le hará saber (ej: contraseña, email).

El artículo 7 de la LOPD determina que existen datos especialmente protegidos. Estos datos son aquellos asociados a la ideología, religión y creencias del usuario. En SimilDroid, no se requiere ninguno de estos datos, por lo que queda exento de incumplir este artículo.

Por otra parte, el artículo 9 de la LOPD, titulado: Seguridad de los datos, determina que es el encargado del tratamiento de la información del sistema quien tiene que asegurarse de que la información está correctamente asegurada, de manera que terceras personas no puedan acceder a dicha información. Para ello, durante el proceso de análisis, diseño e implementación de Simildroid, se colocarán "barreras" de seguridad para evitar que se incumpla este artículo.

Existen varios datos importantes que es mejor no almacenar en el sistema por temas de seguridad. Estos datos son los siguientes:

- *Contraseña del usuario:* por motivos de seguridad, y para evitar posibles problemas legales, se ha decidido no almacenar la contraseña del usuario en ninguna parte del sistema de SimilDroid. En su lugar, se almacena la función resumen de la contraseña.
- *IMEI de los dispositivos:* para evitar cualquier tipo de vinculación entre la persona física real, y los datos almacenados en nuestro sistema, el identificador utilizado para cada terminal dependerá no de su IMEI, sino de la posición en la que se ha introducido en el sistema. Así pues, el primer terminal introducido tiene id = 1, el segundo tendrá id = 2, etc.

La comunicación de la poca información sensible que existe en *SimilDroid*, se hará siempre a través de un canal seguro, como lo es SSL/TLS.

SimilDroid, tampoco dará a conocer los datos de los usuarios a terceras partes. Se trata de un sistema con intenciones puramente de investigación, por lo que no se pretende realizar ningún tipo de gestión económica con los datos almacenados en la base de datos.

4. Diseño del Sistema

A lo largo de este apartado de Diseño, se obtendrá una estructura del sistema haciendo uso del análisis realizado en el apartado anterior *(Apartado 3 – Análisis del Sistema)*. Se presentará, entonces, la arquitectura seleccionada para llevar a cabo el proyecto, además de todas aquellas decisiones de diseño tomadas.

4.1 Diseño de la Arquitectura del Sistema

Durante el desarrollo del siguiente apartado, se determinaran los diferentes componentes o módulos que componen el sistema. La arquitectura de software que toma el sistema que se va a desarrollar, toma el patrón de modelo-vista-controlador, también conocido por sus siglas como MVC. Esta disposición es resultado de la separación de la interfaz, del control del sistema y de la lógica de datos.

Por lo tanto:

- ***Modelo:*** Se refiere al conjunto de componentes relacionados con los datos y la lógica que los controla. Esta capa, será la encargada de proporcionar la información necesaria al sistema en el momento en el que este lo solicite, además de dar funcionalidad para poder añadir y modificar información.

- ***Vista:*** Se refiere al conjunto de módulos o componentes relacionados con la interfaz de usuario. Será la capa encargada de mostrar la información obtenida por la capa Modelo al usuario del sistema, para una correcta interpretación.

- ***Controlador:*** Se refiere al conjunto de módulos o componentes que contienen toda la lógica operacional del sistema. Es considerada la capa *cerebro* del sistema SimilDroid.

4.1.1 Modelo

Contiene la representación de la información con la cual el sistema va a operar, por lo que es la encargada de realizar las operaciones relacionadas con el tratamiento de datos (consulta, modificación e inserción) sobre las bases de datos.

Las peticiones de manipulación de la base de datos llegan a esta capa a través de la capa *Controlador*.

En el terminal móvil (es decir, en el módulo Android), no existe ningún tipo de almacenamiento, por lo que no aparecerá en la capa de modelo.

En el caso de *SimilDroid*, la capa modelo estará compuesta por:

- ***MySQL:*** se trata de un gestor de base de datos que almacenará y gestionará la base de datos creada para el sistema. Se usará, para ello, la API original de MySQL.

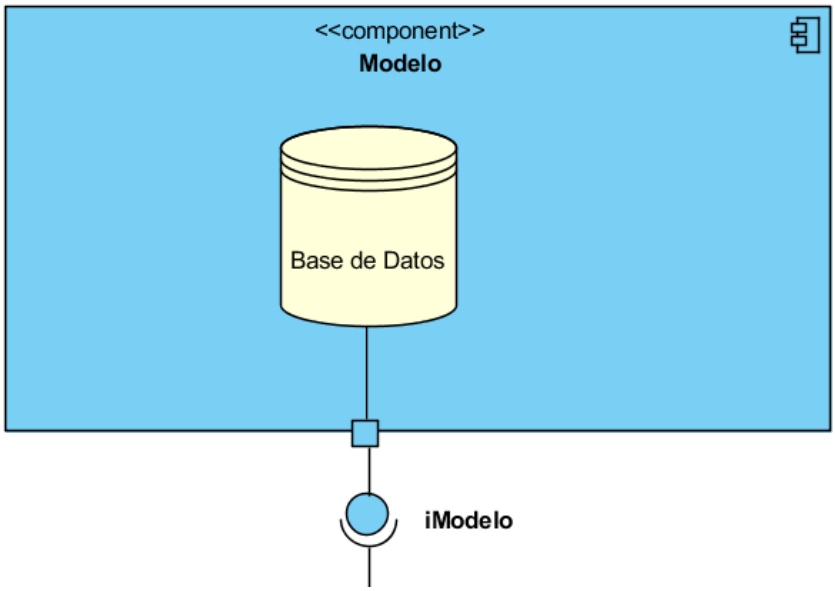

Ilustración 12 - Modelo del sistema SimilDroid

4.1.2 Vista

Se trata de la capa encargada de mostrar la información contenida en la capa modelo al usuario (según corresponda), de forma que el usuario sea capaz de poder interpretar la información. El usuario podrá, por tanto, interactuar con los elementos de esta capa, pudiendo así, obtener unos resultados u otros de la capa modelo.

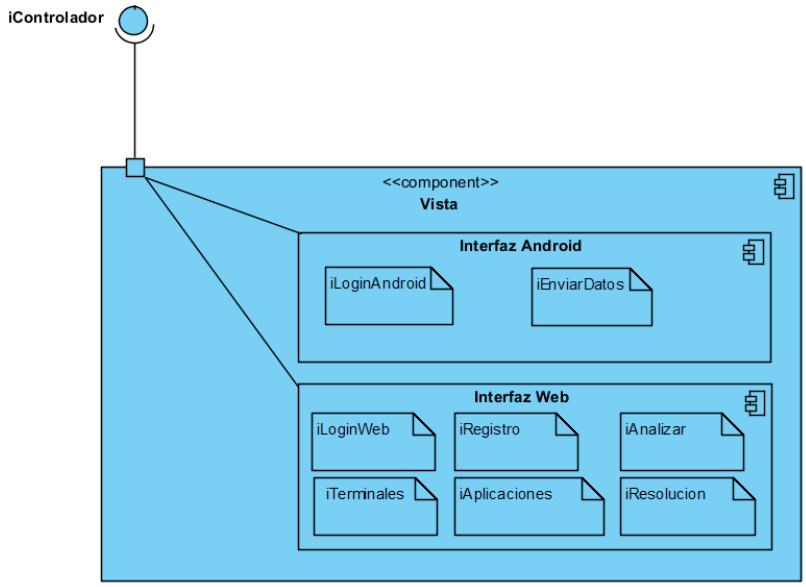

Ilustración 13 - Vista del sistema SimilDroid

En el sistema de SimilDroid, la capa Vista está, a su vez, subdividida en dos partes. Por un lado tenemos la parte relacionada con la aplicación Android (Interfaz Android), y por otra parte, tenemos aquella relacionada con la vista del navegador (Interfaz Web).

Como se puede ver en la imagen anterior, el componente Vista, utiliza los recursos que obtiene del componente Controlador.

Las interfaces con las que cuenta este componente son las siguientes:

- ***iLoginAndroid:*** componente de la aplicación Android que permite al usuario poder autenticarse en el sistema.
- ***iEnviarDatos:*** componente de la aplicación Android que permite listar las aplicaciones encontradas, y enviarlas al servidor para almacenarlas.
- ***iLoginWeb:*** componente web, que permite al usuario poder autenticarse en el sistema.
- ***iRegistro:*** componente web que permite al usuario poder registrarse en el sistema.

- ***iAnalizar:*** componente web que permite solicitar análisis de similitud.
- ***iTerminales:*** componente web que permite visualizar y añadir nuevos terminales en el sistema.
- ***iAplicaciones:*** componente web que permite visualizar las aplicaciones que tiene un terminal en concreto.
- ***iResolucion:*** componente web que permite realizar la evaluación de la similitud entre aplicaciones, una vez terminado el proceso de análisis.

4.1.3 Controlador

El controlador, tal y como se ha comentado anteriormente, se trata de la capa "cerebro" del sistema. Responde a eventos, que, normalmente se corresponden con acciones del usuario e invoca peticiones a la capa de Modelo. También puede enviar órdenes a la Vista para poder mostrar la información que el Modelo le ha devuelto. En definitiva, esta capa realiza de mediador entre la capa Vista y la capa Modelo.

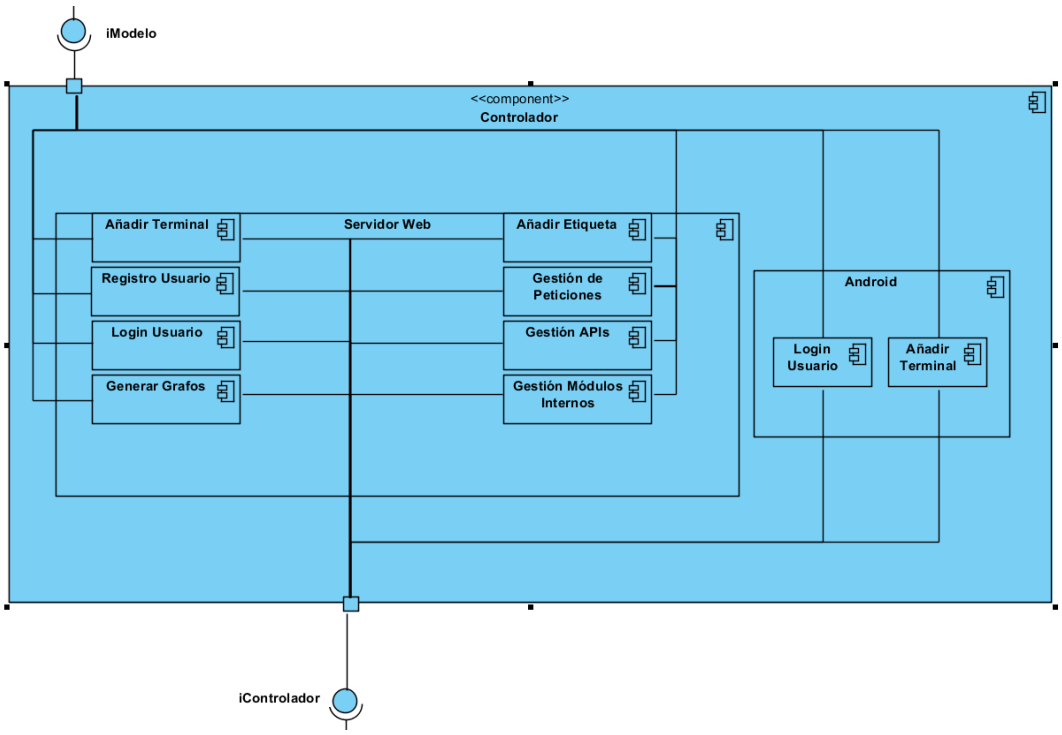

Ilustración 14 - Controlador del Sistema SimilDroid

De la misma manera que anteriormente, la capa Controlador está subdivida en dos partes que, coinciden con las dos partes de la Vista (Android, y Servidor Web). Estos son los componentes de la capa Controlador:

- ***Añadir Terminal:*** este componente gestiona la funcionalidad de añadir un nuevo terminal al sistema. Mediante web únicamente se pueden añadir terminales Wearable.
- ***Registro Usuario:*** este componente gestiona la funcionalidad del registro del usuario en el sistema.
- ***Login Usuario:*** este componente gestiona la funcionalidad de la autenticación del usuario en el sistema.
- ***Generar Grafos:*** este componente gestiona la funcionalidad de la generación de grafos en formato .graphml para su posterior análisis.
- ***Añadir Etiqueta:*** este componente gestiona la funcionalidad de añadir nuevas etiquetas a las aplicaciones y terminales del usuario.
- ***Gestión de Peticiones:*** este componente gestiona la funcionalidad del control de peticiones de similitud del sistema, es decir, se encarga de llevar un control del número de peticiones que se han realizado para realizar el análisis de similitud. El servidor Web es capaz de atender varias peticiones a la vez, pero para evitar sobrecarga del sistema, los cálculos de similitud se realizarán de manera secuencial.
- ***Gestión de APIs:*** este componente gestiona la comunicación entre el servidor Web y las APIs externas usadas para la obtención de los grafos.
- ***Gestión Módulos Internos:*** este componente gestiona la comunicación entre el servidor Web y los módulos internos utilizados para la obtención de los grafos.
- ***Login Usuario (Android):*** este componente perteneciente a la aplicación Android, se encarga de gestionar la funcionalidad de autenticación del usuario en el sistema.
- ***Añadir Terminal (Android):*** este componente perteneciente a la aplicación Android, se encarga de gestionar la inserción de un nuevo terminal en la WBAN del usuario.

4.2 Diseño del módulo Android

Durante el desarrollo del presente subapartado, se va a describir el diseño elegido para el módulo Android. Es importante recordar, que el módulo Android, tal y como se ha presentado en la fase de análisis, únicamente es el encargado de recoger la información relativa a las aplicaciones instaladas en el terminal, y enviarlas al servidor web.

Como se puede observar en el desglose de la arquitectura, el módulo Android se encuentra tanto en la capa de Vista (evidentemente es necesario para poder interactuar

con el usuario) y la capa Controlador (ya que, una vez obtenida la información necesaria, ha de enviarse al módulo Web).

Para poder llegar a entender correctamente el funcionamiento del módulo Android, y teniendo en cuenta, que al tratarse de código implementado en JAVA, vamos a realizar un análisis de las clases que componen la aplicación *SimilDroid*, explicando, para cada una de ellas, su objetivo, además del funcionamiento de cada uno de sus métodos y el significado de cada uno de sus campos.

4.2.1 Clase Aplicación

La clase Aplicación, permite crear objetos que almacenan la información relativa a una aplicación.

	Nombre	Descripción
CAMPOS	*String nombre*	Variable que almacena el nombre de la aplicación.
	String permisos []	Array que almacena el conjunto de permisos de la aplicación.
	String paquete	Variable que almacena el paquete al que pertenece la aplicación.

Tabla 47 - Campos clase Aplicacion

NOTA: Cada uno de estos campos tiene su correspondiente método *get* y *set*, que no serán explicados.

Método *Aplicacion()*	
Argumentos	*permisos[], nombre, paquete*
Objetivo	Método constructor de la clase Aplicación. Se encarga de asignar valores iniciales al objeto instanciado.
Funcionamiento	Asigna a cada campo, el valor correspondiente obtenido por argumento.

Valor/es devuelto/s	No devuelve nada

Tabla 48 – Método Aplicacion de la clase Aplicacion

Método *getPermisosJSON()*	
Argumentos	*Ninguno*
Objetivo	Convierte el array de permisos de la aplicación, en array en formato JSON (para poder enviarlo posteriormente).
Funcionamiento	Obtiene los permisos actuales de la aplicación, haciendo uso del correspondiente método get (getPermisos()), para después introducirlos en un array en formato JSON.
Valor/es devuelto/s	Devuelve array en formato JSON.

Tabla 49 - Método getPermisosJSON de la clase Aplicacion

Método *toJSON()*	
Argumentos	*Ninguno*
Objetivo	Convierte el objeto instanciado *Aplicacion* en un objeto en formato JSON
Funcionamiento	Obtiene todos y cada uno de los campos de la aplicación, para después insertarlos en un objeto en formato JSON
Valor/es devuelto/s	Devuelve objeto *Aplicacion* en formato JSON.

Tabla 50 - Método toJSON de la clase Aplicacion

4.2.2 Clase ConexionServidor

La clase *ConexionServidor* almacena información relativa al servidor web de *SimilDroid*. Además, contiene métodos para enviar información al servidor.

	Nombre	**Descripción**
	String IP	Variable que almacena la *IP* del servidor web de SimilDroid.

CAMPOS	*Final MediaType JSON*	Constante utilizada para poder enviar información en formato JSON. (únicamente se define al principio)
	OkHttpClient client	Objeto de tipo OkHttpClient, utilizado para enviar información a un servidor mediante métodos get y post.

Tabla 51 - Campos de la clase ConexionServidor

NOTA: El campo *IP* tiene su correspondiente método *get*, que no serán explicado.

Método *Post()*	
Argumentos	*String relativeURL, String json*
Objetivo	Método utilizado para enviar información mediante el método POST (que envía la información mediante la entrada estándar STDIO) a la URL formada por (campo *IP)*+ (Argumento *relativeURL*).
Funcionamiento	Realiza las operaciones necesarias para enviar un mensaje mediante el método POST, a la URL formada por (campo *IP)*+ (Argumento *relativeURL*). Para ello hace uso del objeto *OkHttpClient*.
Valor/es devuelto/s	String respuesta (respuesta que brinda el servidor)

Tabla 52 - Método Post de la clase ConexionServidor

4.2.3 Clase Login

La clase *Login* se encarga de obtener el usuario y contraseña del usuario y enviarlos al servidor. El servidor será el encargado de decidir si las credenciales son correctas o no lo son.

	Nombre	Descripción
CAMPOS	*Button btnAcceder*	Objeto de tipo *Button,* que servirá para recoger el evento de pulsación de botón para autenticarse en el sistema.
	Button btnRegistro	Objeto de tipo *Button,* que servirá para recoger el evento de pulsación de botón para registrarse en el sistema.
	EditText user	Objeto de tipo *EditText,* que servirá para obtener el contenido insertado en el campo *usuario* correspondiente a la interfaz de usuario de la actividad de *Login*.
	EditText pass	Objeto de tipo *EditText,* que servirá para obtener el contenido insertado en el campo *contraseña* correspondiente a la interfaz de usuario de la actividad de *Login*.

Tabla 53 - Campos de la clase Login

Método *getHash()*	
Argumentos	*String pwd*
Objetivo	El objetivo de este método es obtener la función resumen de la palabra pasada por argumento (en este caso siempre se tratará de la contraseña del usuario).
Funcionamiento	Se recoge el argumento pwd, al cual se le aplicará la función resumen SHA-256.
Valor/es devuelto/s	Byte[] hash (conjunto de bytes que conforman el hash de pwd)

Tabla 54 - Método getHash de la clase Login

Método *bin2Hex ()*	
Argumentos	*Byte[] data*

Objetivo	El objetivo de este método es convertir un número binario (almacenado en un objeto de tipo byte), en un número hexadecimal (almacenado en un objeto de tipo String)
Funcionamiento	Se recoge el número binario, al cual se le aplican las oportunas operaciones para convertirlo a base Hexadecimal.
Valor/es devuelto/s	String dataHexadecimal

Tabla 55 - Método bin2Hex de la clase Login

Método *comprobarLogin ()*	
Argumentos	*String usr, String pwd*
Objetivo	El objetivo de este método es comprobar si las credenciales ofrecidas por el usuario sirven para poder autenticarse con el servidor web o no.
Funcionamiento	Crea un objeto de tipo conexionServidor (explicado anteriormente). Recoge el usuario y la contrsaseña (a esta última le aplica la función resumen explicada anteriormente) y los intenta enviar a la función correspondiente que realiza la comprobación en el servidor. Es importante saber que realiza el envío de un objeto de tipo Usuario (que explicaremos más adelante).
Valor/es devuelto/s	String respuestaServidor

Tabla 56 - Método comprobarLogin de la clase Login

Método *onCreate ()*	
Argumentos	*Bundle savedInstanceState*
Objetivo	Éste método se ejecuta cuando el usuario accede a la actividad *Login*, o lo que es lo mismo, se ejecuta cuando el usuario visualiza el login de la aplicación.
Funcionamiento	En primer lugar, comprueba si el usuario está ya logueado o no (visualizando el fichero SharedPreferences, donde únicamente se almacena si el usuario está o no está logueado. IMPORTANTE: no se almacena ningún tipo de contraseña en

	este fichero). Una vez comprobado, si el usuario no está logueado, se le solicitarán las credenciales. Cuando el usuario introduzca sus credenciales, se hará uso de los métodos anteriores para comprobar si realmente el usuario está o no está registrado en el sistema. Para ello mandará una petición POST al servidor con el nombre del usuario y su contraseña (mediante protocolo SSL/TLS). En caso de comprobar que, efectivamente está registrado en el sistema, se mostrará la siguiente pantalla, en caso contrario, se mostrará el error que ha devuelto el servidor.
Valor/es devuelto/s	String respuestaServidor

Tabla 57 - Método onCreate de la clase Login

4.2.4 Clase Usuario

	Nombre	**Descripción**
CAMPOS	*String Nombre*	Variable que almacena el nombre del usuario.
	String password	Variable que almacena el resumen de la contraseña del usuario.
	ArrayList<Aplicacion> aplicaciones	Lista que almacena todas y cada una de las aplicaciones instaladas en el terminal. Aplicaciones que quedarán asociadas a este Usuario (Nombre).
	String modeloTerminal	Esto servirá para determinar el modelo del terminal.

Tabla 58 - Campos de la clase Usuario

NOTA: Cada uno de estos campos tiene su correspondiente método *get* y *set*, que no serán explicados.

Método *getAplicacionesJSON()*	
Argumentos	*Ninguno*

Objetivo	Convierte la lista de aplicaciones del terminal, en un array en formato JSON (para poder enviarlo posteriormente).
Funcionamiento	Obtiene las aplicaciones actuales instaladas en el terminal, haciendo uso del correspondiente método get (getAplicaciones()), para después introducirlas en un array en formato JSON.
Valor/es devuelto/s	Devuelve array en formato JSON.

Tabla 59 - Método getAplicacionesJSON de la clase Usuario

Método *toJSON()*	
Argumentos	*Ninguno*
Objetivo	Convierte el objeto instanciado *Usuario* en un objeto en formato JSON
Funcionamiento	Obtiene todos y cada uno de los campos del Usuario, para después insertarlos en un objeto en formato JSON
Valor/es devuelto/s	Devuelve objeto *Usuario* en formato JSON.

Tabla 60 - Método toJSON de la clase Usuario

4.2.5 Clase ObtenerAplicaciones

	Nombre	**Descripción**
CAMPOS	*ListView aplicaciones*	Objeto de tipo *ListView,* que servirá para mostrar la lista completa de aplicaciones instaladas en el dispositivo Android.
	TextView nAplicaciones	Objeto de tipo *TextView,* que servirá para mostrar el número de aplicaciones que hay instaladas en el dispositivo Android.
	Button enviarInfo	Objeto de tipo *Button,* que servirá para recoger el evento de pulsación de botón para enviar información al servidor web.

Tabla 61 - Campos de la clase ObtenerAplicaciones

Método *getDeviceName ()*	
Argumentos	*Ninguno*
Objetivo	Obtiene el nombre del terminal que ejecuta la aplicación.
Funcionamiento	Realiza consultas para poder obtener el nombre del dispositivo
Valor/es devuelto/s	String nombreDispositivo

Tabla 62 - Método getDeviceName de la clase ObtenerAplicaciones

Método *PermissionsApps ()*	
Argumentos	*Ninguno*
Objetivo	Obtiene la lista completa de aplicaciones y sus correspondientes permisos.
Funcionamiento	Obtiene, en primer lugar, la lista de aplicaciones instaladas en el terminal. Una vez obtenida esta lista, obtiene además, todos los permisos asociados a cada una de las aplicaciones.
Valor/es devuelto/s	ArrayList <Aplicacion>

Tabla 63 - Método PermissionsApps de la clase ObtenerAplicaciones

Método *onCreate ()*	
Argumentos	*Bundle savedInstanceState*
Objetivo	Éste método se ejecuta cuando el usuario accede a la actividad *listaDeAplicaciones*, o lo que es lo mismo, se ejecuta cuando el usuario visualiza la lista de aplicaciones instaladas.
Funcionamiento	En primer lugar llama al método PermissionsApps para obtener toda la información relativa a las aplicaciones y los permisos asociados a cada aplicación. Cuando el usuario pulsa el botón *enviarInfo*, realiza el envío de una petición mediante el método POST al servidor web. La petición contendrá un objeto de tipo *Usuario*, dentro del cual se almacena la lista de aplicaciones instaladas, la lista de permisos asociados a cada

	aplicación. Una vez enviada la información, espera respuesta del servidor.
Valor/es devuelto/s	String respuestaServidor

Tabla 64 - Método onCreate de la calse ObtenerAplicaciones

4.3 Diseño del módulo Web

A diferencia del módulo Android, éste módulo se ha programado haciendo uso del lenguaje de programación PHP. Es por ello, que no es necesario definir ningún tipo de clase, y de hecho, en este módulo, únicamente existen dos clases. Sin embargo, sí que existen scripts PHP que al ser invocados, realizan funciones necesarias para el correcto funcionamiento del sistema. Por ello, en este apartado, vamos, a realizar una descripción de aquellos scripts que pertenecen a la capa de Modelo. Por último, las dos únicas clases que existen las tomaremos como si de scripts se trataran.

4.3.1 Script databaseHandler

Esta es una de las dos únicas clases que existen en el módulo web. Contiene toda la información necesaria para conectarse a la base de datos MySQL (nombre del host, nombre de la base de datos, nombre del usuario de la base de datos, y por último, la contraseña de la base de datos).

Esta clase contiene tantas funciones como distintos accesos se hacen en la base de datos. A continuación mostramos una lista de todos los métodos existentes en este script, que, consideramos no necesitan descripción, ya que los nombres son auto explicativos.

Esta clase se instancia en los scripts en los que se necesita, de alguna manera, acceder a la base de datos, ya sea para consultarla o modificarla. Se ha creado esta clase, no solo por seguridad (para evitar que la contraseña de la base de datos esté en todos y cada uno de los scripts), sino también para ahorrar tener que repetir código.

Métodos de la clase *databaseHandler*
__construct()
obtenerTerminalesPorUsuario($nick)
obtenerNombreTerminal($idTerminal)
obtenerAplicacionesPorTerminal($idTerminal)
obtenerPermisosPorAplicacion($idTerminal,$aplicacion)

obtenerEmailPorUsuario($nick)
obtenerEtiquetasPorAplicacion($idTerminal,$aplicacion)
almacenarEtiquetaPorAplicacion($idTerminal,$aplicacion,$etiqueta)
almacenarResolucionDeAlgoritmoCliente($nick, $resolucionInicial)
almacenarTarea($nick)
eliminarTarea($nick)
obtenerTareas()
almacenarEtiquetaPorTerminal($idTerminal,$etiqueta)
obtenerTodasAplicacionesPorUsuario($nick)
obtenerPaquetesSinCategoria()
guardarNuevaCategoria($aplicacion, $categoria)
obtenerTodasEtiquetasCreadas()
obtenerTodasEtiquetasDeAplicacionesPorUsuario($nick)
obtenerEtiquetasCreadasPorUsuario($nick)
obtenerCategoriaPorAplicacion($aplicacion)
asignarAutor($aplicacion , $autor)
insertarNuevoWearable($usuario, $nombre_terminal, $apps)
insertarNuevaEtiquetaDeUsuario($usuario, $etiqueta)
terminarConexion()

Tabla 65 - Métodos de la clase databaseHandler

4.3.2 Script graphmlObjects

Este es el segundo script que podría considerarse clase, aunque realmente tiene en su interior dos clases. Este script está definiendo dos tipos de clases que servirán para instanciar objetos que usaremos para generar la estructura de los grafos.

Por un lado, la clase *Nodo*, que contiene información relativa al nodo, como el id del nodo, el nombre, y demás valores que únicamente son útiles para el algoritmo del cliente.

Por el otro lado, existe la clase *Arista*, que contiene toda la información necesaria para las aristas. Esta información únicamente es; *idNodoOrigen, idNodoDestino* (que al ser un grafo no dirigido, da igual el orden), y por último el *peso* o coste de la arista, que servirá para dar valor a la similitud entre dos nodos.

4.3.3 Script checkLogin.php y checkLoginAndroid

En este caso, estos dos scripts hemos decidido explicarlos juntos, ya que realmente hacen lo mismo pero para clientes distintos. Así pues, checkLogin se encarga de comprobar si las credenciales son correctas para los usuarios que acceden por Web, y checkLoginAndroid, recoge la información que envía el método *OnCreate* de la clase Login del módulo Android (explicado anteriormente) para realizar la misma

comprobación. Simplemente, estos dos scripts, realizan una consulta en la base de datos (haciendo uso de la clase DatabaseHandler explicada anteriormente – apartado 4.3.1), y comprueban si existe el usuario, y en caso afirmativo, si el hash recogido por POST, es el mismo que el hash almacenado en la base de datos. En caso de ser afirmativo, se iniciará la sesión php (haciendo session_start()).

4.3.4 Script checkRegistro

De manera muy parecida a la anterior, este script recoge los datos enviados por la web (nombre o Nick, contraseña, confirmación de contraseña y email). En primer lugar comprueba si existe ya algún usuario con ese Nick, en cuyo caso devolverá error (no puede haber dos usuarios con el mismo Nick). Una vez realizada esa comprobación, se comprobará si la contraseña coincide con la confirmación de la contraseña (esto es después de realizar el hash de las dos, por lo que realmente se comparan los resúmenes, esto es para evitar tener la contraseña lo máximo posible en el sistema), de manera que si no lo son, devolverá error. En caso de cumplir estas dos normas, se almacenará el nuevo usuario en el sistema.

4.3.5 Script checkAplicaciones

Este script es el encargado de recoger toda la información que envía el método OnCreate de la clase *ObtenerAplicaciones* del módulo Android. Toda la información que recoge, la almacena en la base de datos. Para ello, instancia un objeto de la clase DatabaseHandler cada vez que necesita acceder a la base de datos y lanzar una petición. Así pues, este script, en primera instancia, almacena el terminal (con todos sus elementos, id, nombre, dueño, etc), después, almacena toda la lista de aplicaciones asociadas al citado terminal, y por último, almacena todos los permisos asociados a cada una de las aplicaciones almacenadas.

4.3.6 Script nuevoDispositivoWearable

Tal y como se comentó anteriormente, los dispositivos Wearable tiene que introducirlos a mano el usuario, es por eso que, cuando en la web, el usuario inserta un nuevo Wearable, se ejecuta este código. Lo único que hace este código es, añadir un nuevo terminal, asignándole las aplicaciones asociadas que el usuario haya determinado.

4.3.7 Script incluirEtiquetas.php

Este script se encarga de incluir las etiquetas (o tags) en la base de datos. Así pues, cuando el usuario decide insertar una nueva etiqueta a un terminal o a una aplicación, esta función se encarga de llevar el proceso a cabo.

4.3.8 Script solicitudAnalisis

El usuario es el encargado de solicitar que se realice el análisis de similitud en su red WBAN. Pero, tal y como ya se ha comentado con anterioridad, no se puede realizar el análisis en tiempo real, ya que, por limitaciones de hardware, únicamente se va a llevar a cabo un cálculo de similitud a la vez. Por ello, este script, únicamente se encarga de almacenar una tarea en la tabla de tareas (ver apartado *4.5 – diseño del modelo físico de datos*).

4.3.9 Script consulta42matters

Este script es el encargado de realizar la petición de obtención de información a la API 42matters (como se comenta en el apartado *3.6.4 – Librerías y API's*), de esta manera, antes de realizar los grafos necesarios para el cálculo de la similitud, el sistema se asegura que se tiene toda la información necesaria. Simplemente realiza una consulta de información sobre el paquete Android que se necesita, y almacena la información en la base de datos (de esta API se extrae el Autor de la aplicación y la categoría a la que pertenece)

4.3.10 Script generarGraphML_estructural

Este script realiza una gran cantidad de operaciones para poder realizar el cálculo del grafo estructural con atributos. Realmente, necesitaríamos muchas páginas y mucho tiempo extra para poder explicar detalladamente todo lo que realiza este script. Pero en términos generales, este Script estudia la información almacenada en la base de datos por cada Usuario. Entonces;

- Por cada usuario, obtiene la lista de terminales. Por cada terminal, genera un nodo.
- Por cada terminal, obtiene la lista de aplicaciones. Por cada aplicación, genera un nodo y lo conecta al terminal al que pertenece.

- Por cada aplicación, obtiene la lista de permisos. Por cada permiso, genera un nodo y lo conecta al terminal al que pertenece (es relevante saber, que no existen dos nodos de tipo permiso idénticos, es decir, si dos aplicaciones tienen el mismo permiso, las dos aplicaciones estarán unidas al mismo permiso).
- Por cada aplicación, obtiene su categoría.
    - Si la categoría ya existe, simplemente conecta los dos nodos.
    - Si la categoría no existe, genera el nodo de la categoría, y los une.
- Por cada aplicación, obtiene su autor.
    - Si el autor ya existe, simplemente conecta los dos nodos.
    - Si el autor no existe, genera el nodo del autor y los une.
- Por cada aplicación, obtiene el nombre de la aplicación (no es lo mismo el nombre de la aplicación, ya que es un atributo de la misma, que la misma aplicación. El nombre no estará unido directamente a los demás atributos de la aplicación).
    - Si el nombre ya existe, simplemente, conecta los dos nodos.
    - Si el nombre no existe, genera el nodo del nombre y los une.
- Por cada terminal Wearable, se busca las aplicaciones asociadas a él, de tal manera que se duplican los nodos de dichas aplicaciones (ya que realmente, no son la misma aplicación), conectándolos a los mismos elementos a los que estaban conectados los nodos iniciales.
- Todos los terminales estarán unidos al nodo que indique el tipo de terminal que es. Por ejemplo, si un terminal es Wearable, estará conectado al nodo "Wearable", mientras que si no lo es, estará conectado al nodo Terminal.
- Por último, los nodos terminales, estarán todos unidos a un nodo central (desde el que parte la búsqueda). Este nodo lo denominamos HUB, y simboliza la conexión que existe entre todos los terminales (ya que al fin y al cabo todos estos terminales están en una misma red WBAN).

4.3.11 Script generarGraphML_similitud

De la misma manera que el script anterior*, generarGraphML_similitud.php* se trata de un script muy denso, que contiene una gran cantidad de operaciones. Por eso mismo, en este subapartado, haremos una breve síntesis de su funcionamiento.

En esta ocasión, el código es un poco más modular (por la naturaleza del problema). Así, se puede separar el cálculo en varias partes. Antes de comenzar a explicar cada subgrafo, es importante detallar que las similitudes calculadas están normalizadas, es decir, toman valores entre 0 y 1.

- *Cálculo de subgrafo de similitud correspondiente a las categorías.*

  Para realizar el cálculo de similitud entre categorías, en primer lugar se ha realizado un análisis de todas las posibles categorías que ofrece la ***API 42matters***. Una vez obtenidas todas las posibles categorías, se ha realizado el análisis de similitud semántico entre todas las categorías. Al tratarse de un dominio cerrado, se ha precalculado, de manera que se ha creado un fichero llamado (catSimilitud.csv) con la similitud entre categorías. Una vez leído este fichero, se generará un nodo por cada elemento del dominio de categorías, creando, por cada categoría, una conexión (arista) con todas las demás categorías.

- *Cálculo de subgrafo de similitud correspondiente a los permisos.*

  Para realizar el cálculo de similitud entre los permisos, en primer lugar se crea un array de bits de tamaño igual al número total de permisos almacenados en el sistema (número total de permisos Android, actualmente son 144). Este array de bits, se transforma a base 64. Una vez convertidos todos y cada uno de los grupos de permisos a base 64, se realiza el cálculo de la similitud entre ellos. Para realizar el cálculo de similitud entre grupos de permisos, se hace uso de la fórmula Jaccard . Una vez obtenidas las similitudes, se crean los nodos correspondientes a cada grupo de permisos (uno por permiso, es decir, no existen dos grupos de permisos iguales), y la conexión entre el grupo de permisos y todos los demás grupos de permisos.

- *Cálculo de subgrafo de similitud correspondiente a las nombres.*

  Para el cálculo de la similitud entre nombres, se ha utilizado una librería propia de *PHP* llamada *Levenshtein*. Al igual que anteriormente, se crea un nodo por cada nombre, y se conectan todos entre sí, dando un valor de peso correspondiente a la similitud entre cada par de nombres.

- *Cálculo de subgrafo de similitud correspondiente a las tags.*

  Por último, y quizá, la operación más pesada de todo el script, se encuentra el cálculo de similitud entre tags del usuario. En primer lugar se obtienen todos los tags que ha usado el usuario (y no los que ha creado, ya que el usuario ha podido crear tags que luego no usa). Una vez obtenida la lista completa, se traducen utilizando el ***API Google Translate***. Con la lista completa de tags utilizados por el usuario, se realiza el análisis de similitud haciendo uso de la librería ***SEMILAR***, implementada en Java. Esta librería, recibe una lista de pares de palabras en formato .csv, y devuelve la misma lista, añadiendo la similitud entre dichas dos palabras. Este proceso es bastante lento, cuantos más tags haya, más tarda en calcular la similitud (podríamos decir que tiene complejidad exponencial, ya que por cada nueva palabra, hay que calcular la similitud entre esa nueva palabra y todas las demás). Una vez calculadas todas las similitudes, se crean los nodos y las conexiones entre ellos (aristas).

4.3.12 Script daemon

Este último script se trata de un script demonio, o lo que es lo mismo, está en continua ejecución. Este script se ejecuta nada más encender el servidor. Es el encargado de comprobar si se ha realizado alguna solicitud de cálculo de similitud. Para ello, consulta la tabla Tarea (haciendo uso de la clase DatabaseHelper). Si encuentra algún usuario con la solicitud de similitud pendiente, genera los grafos correspondientes llamando a los scripts generarGraphML_estructural.php y generarGraphML_similitud.php.

También es el encargado de realizar la llamada al módulo R para realizar el análisis de similitud.

Por último, se encarga de enviar un correo electrónico al usuario cuando ha terminado de realizarse el cálculo de la similitud, para solicitar que realice la tarea de evaluar la salida del algoritmo.

4.4 Especificación de diagramas de secuencia

Para una mejor compresión del funcionamiento del sistema SimilDroid, en este subapartado se muestran los diagramas de secuencia correspondientes a cada uno de los casos de uso definidos en el apartado 3.4 – Casos de Uso.

CU-01: Registro en el Sistema de SimilDroid:

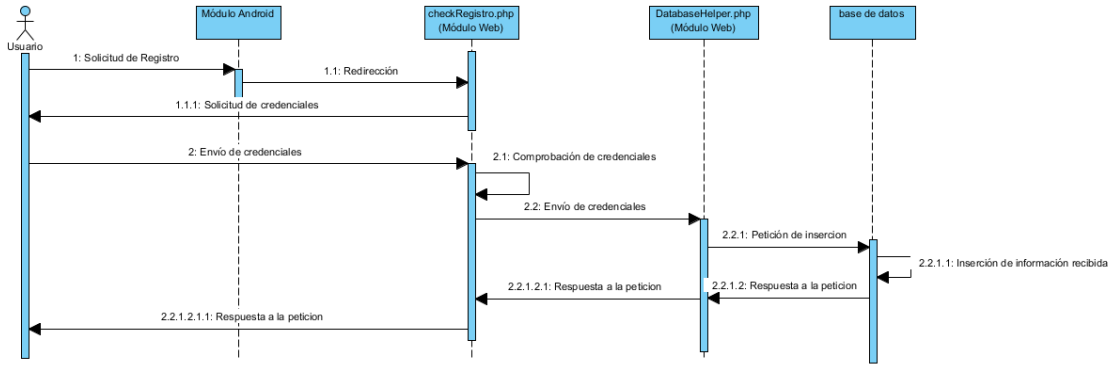

Ilustración 15 - Diagrama de secuencia 1; registro en SimilDroid

CU-02: Login en el Sistema de SimilDroid (desde Android):

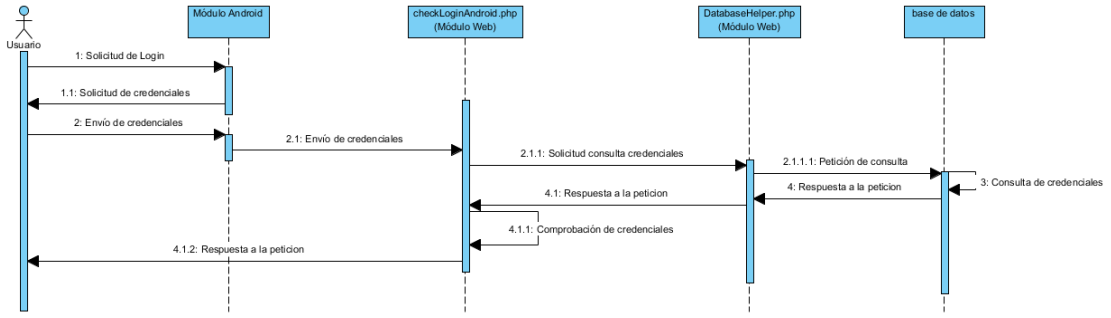

Ilustración 16 - Diagrama de secuencia 2; login en SimilDroid (Android)

CU-02: Login en el Sistema de SimilDroid (desde Web):

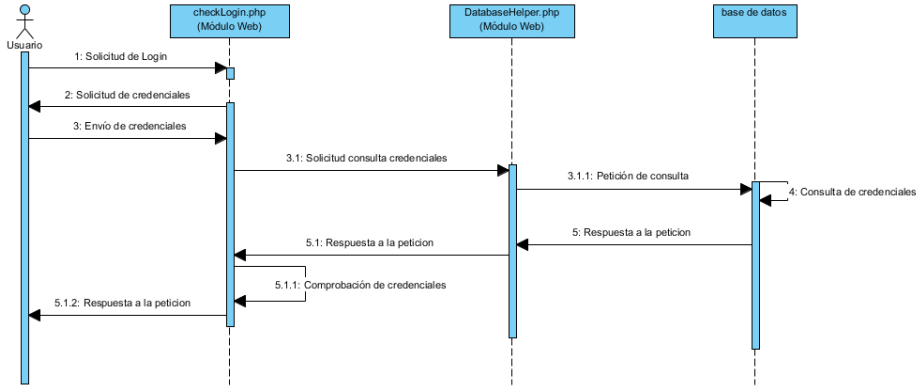

Ilustración 17 - Diagrama de secuencia 3; login en SimilDroid (Web)

CU-03: Obtener aplicaciones instaladas en el terminal:

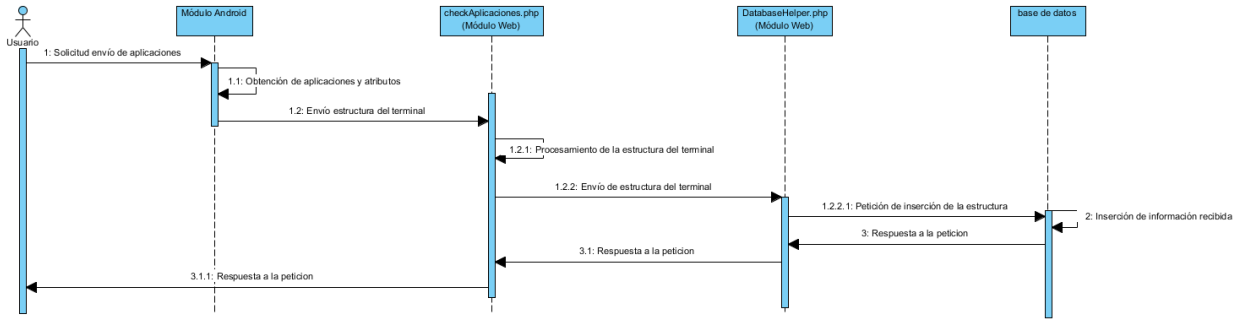

Ilustración 18 - Diagrama de secuencia 4; obtener aplicaciones instaladas

CU-04: Registrar nuevo dispositivo Wearable:

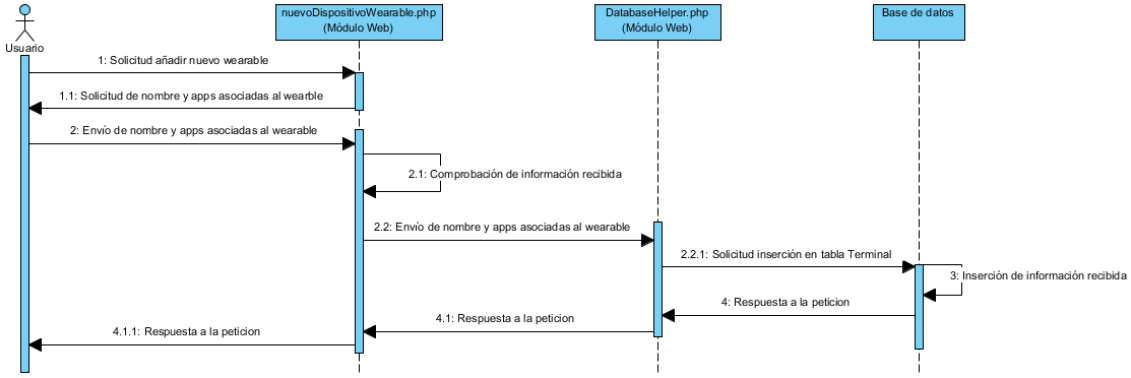

Ilustración 19 - Diagrama de secuencia 5; registro de nuevo Wearable

CU-05: Solicitar nuevo análisis de similitud entre aplicaciones:

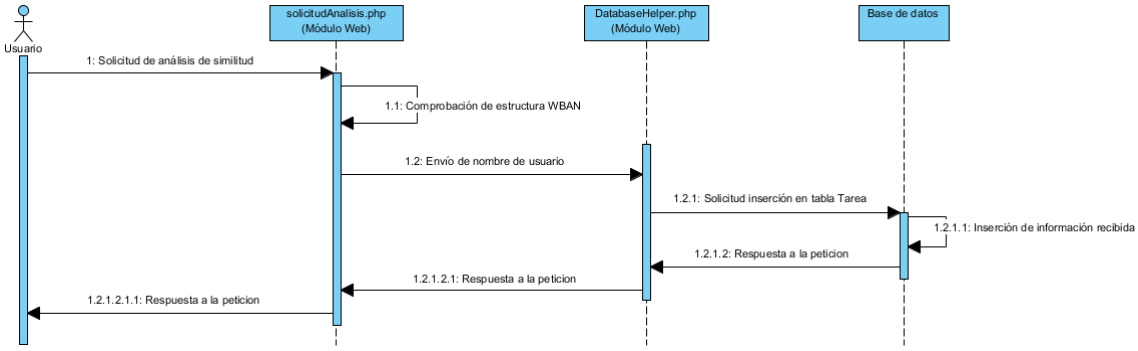

Ilustración 20 - Diagrama de secuencia 6; solicitud de nuevo análisis de similitud

CU-06: Realizar Evaluación de Similitud:

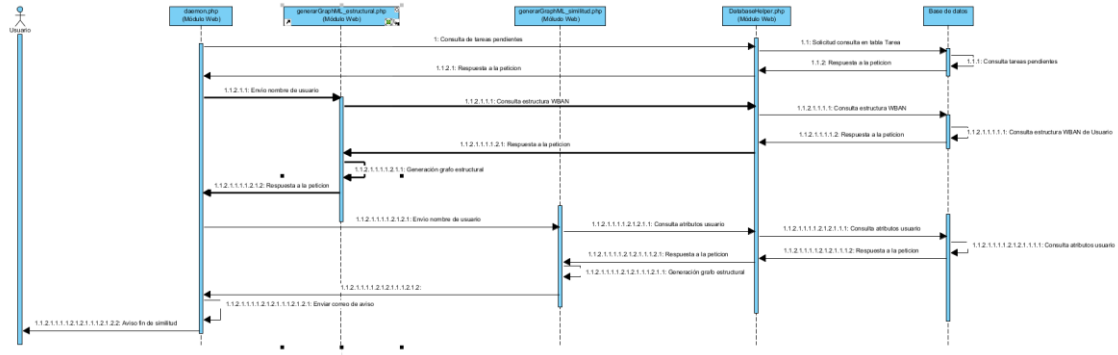

Ilustración 21 - Diagrama de secuencia 7; cálculo de similitud

CU-07: Añadir etiqueta a nodo (terminal o aplicación)

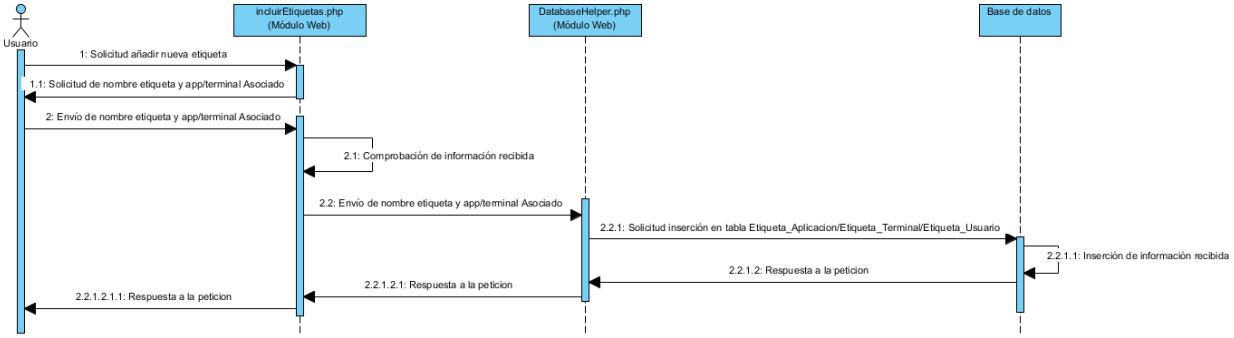

Ilustración 22 - Diagrama de secuencia 8; añádir nueva etiqueta a nodo

4.5 *Diseño de modelo físico de datos*

Durante este apartado se establecen las estructuras físicas de datos del sistema, teniendo en cuenta que, tal y como se dijo en el apartado X.X, se trata de una base de datos MySQL, es decir, que sigue un modelo relacional.

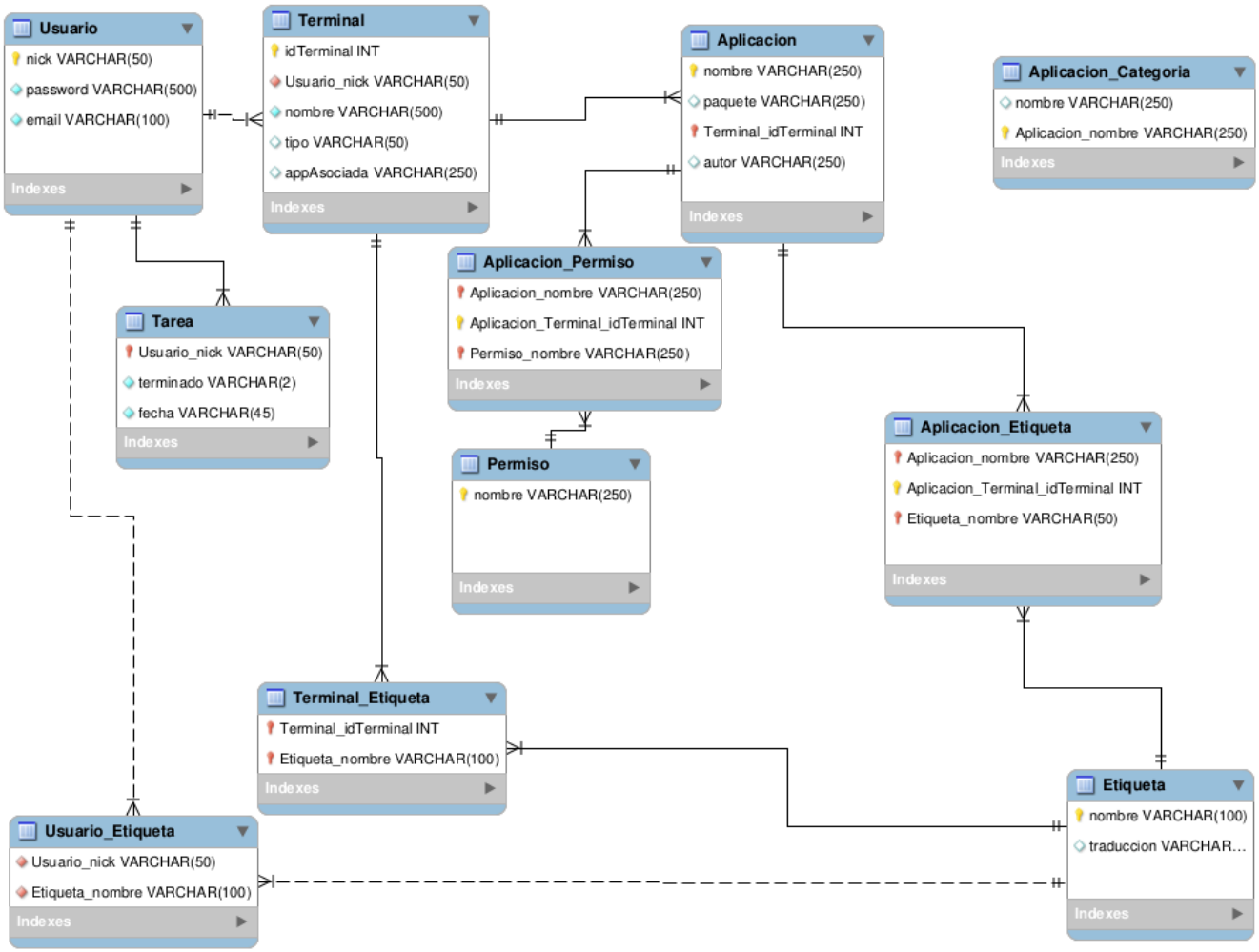

Ilustración 23 - Diseño físico de la base de datos de SimilDroid

Para una mejor comprensión de la base de datos del sistema, se va a detallar cada una de las tablas de la base de datos.

Tabla Usuario

La tabla Usuario se encarga de almacenar información relativa a cada uno de los usuarios del sistema. En primer lugar almacenará el Nick del usuario, que servirá

también como clave primaria de la tabla. Además, almacenará la contraseña del usuario (no en claro, sino el resumen de ella), y el correo electrónico, donde se enviará el correo de aviso de fin de cálculo de similitud.

Tabla Terminal

La tabla Terminal se encarga de almacenar información relativa a cada uno de los terminales del sistema. En primer lugar almacenará el id del terminal (que será la clave primaria de la tabla), junto al usuario al que pertenece el mismo. También se almacenará el nombre del terminal (p.e: Samsung Galaxy Note III), y el tipo de terminal (si se trata de un terminal Wearable, o no), y por último, el campo appsAsociadas, que almacena un array con los nombres de las apps asociadas a este terminal

NOTA: En la tabla terminal, el campo appsAsociadas, almacena un array en formato JSON con los nombres de las apps asociadas a este terminal. Esto solo tiene sentido para aquellos terminales de tipo Wearable, ya que, los Wearable, al no poder acceder al sistema SimilDroid, se debe seleccionar de forma manual las aplicaciones a las que está asociado. Por ejemplo, si un usuario tiene la aplicación de Twitter en un terminal, y su dispositivo Wearable le avisa, por ejemplo, mediante una vibración, que le ha llegado un twit nuevo, será necesario que el usuario añada el Wearable indicando que la aplicación Twitter es la asociada.

Tabla Aplicación

La tabla Aplicación, contiene todas y cada una de las aplicaciones de todos y cada uno de los terminales del sistema. Por ello, es necesario saber, de alguna forma, a que terminal pertenecen (Terminal_idTerminal). Se almacenará entonces, el nombre de la aplicación, el nombre del paquete Android al que pertenece y por último el autor, que servirá posteriormente para el cálculo de similitud.

Tabla Etiqueta

La tabla Etiqueta, únicamente contiene una lista de nombres de etiquetas, junto a la traducción de la misma (esto se hace para evitar hacer un uso excesivo de la *API de*

Google Translate, ya que muchas de las etiquetas usadas por el conjunto de usuarios son las mismas).

Tabla Tarea

La tabla Tarea, se encarga de almacenar cada una de las peticiones realizadas para el análisis de similitud. Tal y como se cuenta en el apartado *4.1.3 Controlador*, existe una cola de peticiones para realizar el análisis de similitud. Esta tabla contiene, el Nick del usuario que ha solicitado realizar el análisis, si el análisis se ha terminado o no se ha terminado y la fecha de la solicitud (que servirá como indicador de la posición de la cola), almacenada en formato Juliano.

Tabla Permiso

La tabla Permiso contiene una lista de todos los posibles permisos. Nótese que en Android, existe una lista cerrada de permisos, y aunque, existen otros tipos de permisos creados por diferentes organizaciones (como por ejemplo, grupos de permisos especiales de Facebook), los únicos que nos interesan son los de Android.

> **NOTA**: Para el correcto funcionamiento del sistema, será necesario precargar los permisos actuales de Android.

Tabla Aplicacion_Permiso

La tabla Aplicacion_Permiso, es la encargada de almacenar todos los permisos pertenecientes a una aplicación concreta. Es importante señalar, que dependiendo de la versión de la aplicación, los permisos pueden variar, por ese mismo motivo, hemos diferenciado las aplicaciones de un terminal de otro, aunque el nombre de la aplicación sea exactamente el mismo. Esta tabla almacena el nombre de la aplicación, el nombre del permiso, y el id del terminal al que pertenece la aplicación.

Tabla Aplicacion_Categoria

Esta tabla almacena la categoría de cada aplicación, es importante señalar, que la categoría de cada aplicación se obtiene llamando a la API 42matters *(véase 3.6.4 Librerías y API's)*. Se ha decidido crear una tabla aparte, ya que la API 42matters solo tiene en cuenta el nombre del paquete de la aplicación para obtener la categoría. Sería mucho más tedioso tener un atributo "categoría" en la tabla Aplicación, ya que, cada vez que se obtiene una nueva categoría, habría que realizar demasiadas modificaciones en la misma tabla (ya que inicialmente tendría valor NULL).

Tabla Aplicacion_Etiqueta

Al igual que ocurre con los permisos, esta tabla contiene las etiquetas de cada aplicación. Será importante almacenar el id del terminal, ya que cada aplicación es distinta en uno u otro terminal (aunque aparentemente sean iguales).

Tabla Usuario_Etiqueta

Almacena una lista de todas y cada una de las etiquetas que el usuario ha introducido en el sistema. Cada usuario introduce sus propias etiquetas (aunque en la tabla Etiqueta se almacena todo el conjunto de las etiquetas).

Tabla Terminal_Etiqueta

Tabla semejante a Aplicacion_Etiqueta, pero esta contiene las etiquetas pertenecientes a cada terminal, en vez de a cada aplicación.

4.6 Interfaces de Usuario

En esta sección del proyecto se presentará, de forma esbozada a lápiz, la interfaz gráfica diseñada para el usuario. Es importante entender que la interfaz gráfica de *SimilDroid* está, a su vez, subdividida en dos apartados (cada uno de los cuales se corresponde con un componente "cliente" del sistema).

NOTA: *Los botones de cada una de las interfaces, se han representado con sombra para intentar dar sensación de profundidad.*

4.6.1 Interfaz de usuario en Web

Una vez accedamos al sitio web donde está la página web de *SimilDroid*, aparecerá una pantalla como la siguiente:

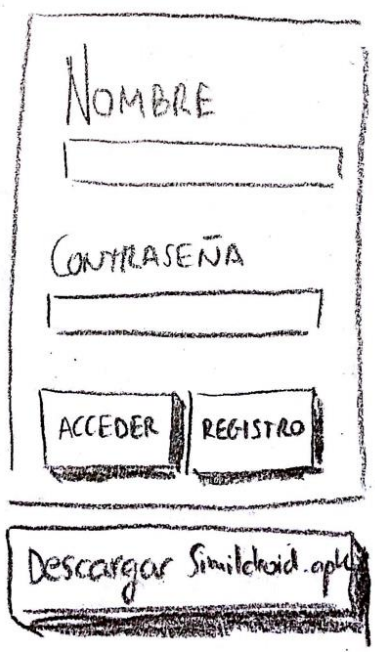

Ilustración 24 - Boceto página de inicio SimilDroid (Web)

Esta página se trata del login del sistema web. En esta página podemos realizar varias cosas:

- Acceder al sistema introduciendo nuestras credenciales de manera correcta.
- Acceder al registro del sistema.
- Descargar la aplicación SimilDroid.apk (que más adelante se explica).

Si decidimos acceder al registro del sistema veremos algo como lo siguiente:

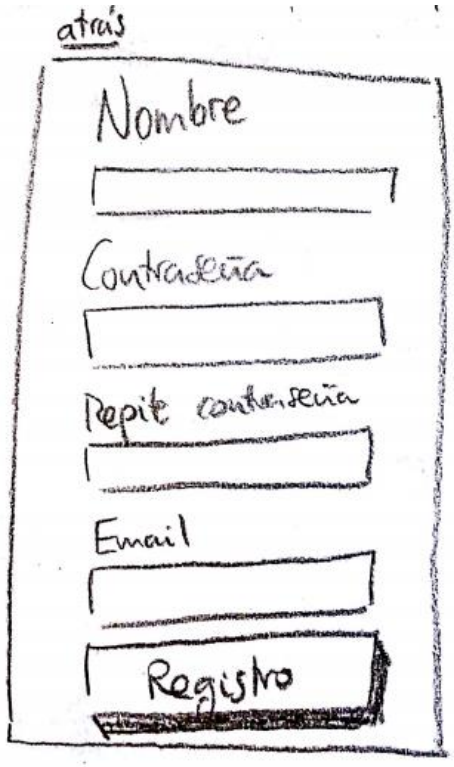

Ilustración 25 - Boceto página de registro en SimilDroid (Web)

En esta página podremos realizar nuestro registro introduciendo el Nick que queramos tener en el sistema (ha de ser único), la contraseña, la confirmación de la misma, y nuestro correo electrónico.

Una vez obtenido nuestro usuario, podremos hacer uso de la aplicación Android para introducir uno o más terminales en nuestra WBAN. Cada vez que instalemos la aplicación en un dispositivo, y la usemos correctamente (como se muestra en el punto 3.3.2), estaremos introduciendo un nuevo terminal en el sistema.

Cuando hayamos introducido algún terminal, podremos acceder al sistema mediante la página web. Introducidas las credenciales correctamente, veremos la siguiente pantalla:

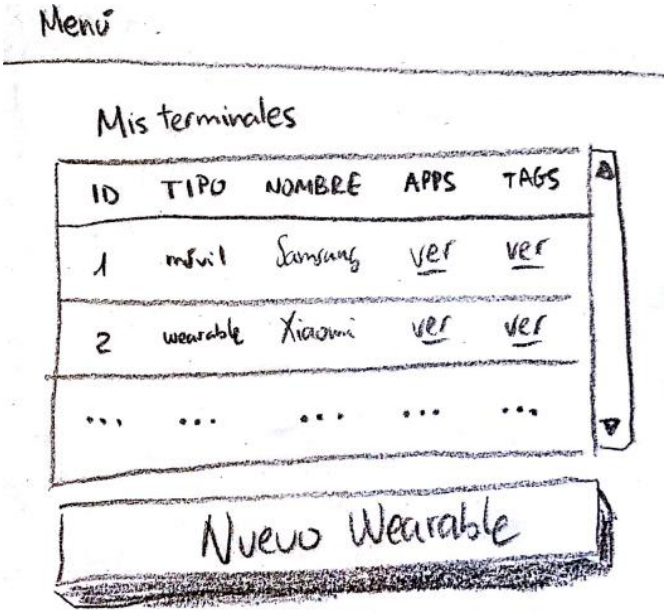

Ilustración 26 - Boceto página listado de terminales de la WBAN (Web)

Esta es la pantalla que muestra una lista de los terminales asociados a nuestra WBAN. Podremos entonces, añadir un nuevo dispositivo (Wearable) al sistema indicando qué aplicaciones, ya instaladas en alguno de nuestros terminales, están asociadas a este terminal. Para ello bastará con pulsar el botón *Añadir dispositivo Wearable* y rellenar el siguiente formulario:

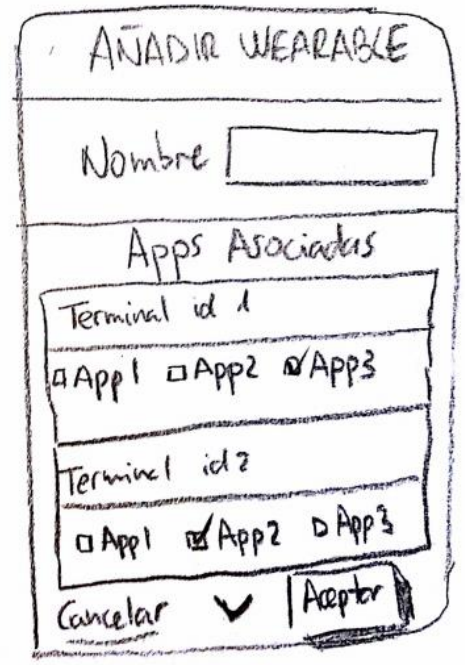

Ilustración 27 - Boceto pop-up para añadir Wearable (Web)

También podremos visualizar las aplicaciones asociadas a cada terminal, pulsando para ello el botón *Ver Aplicaciones* de la pantalla de terminales.

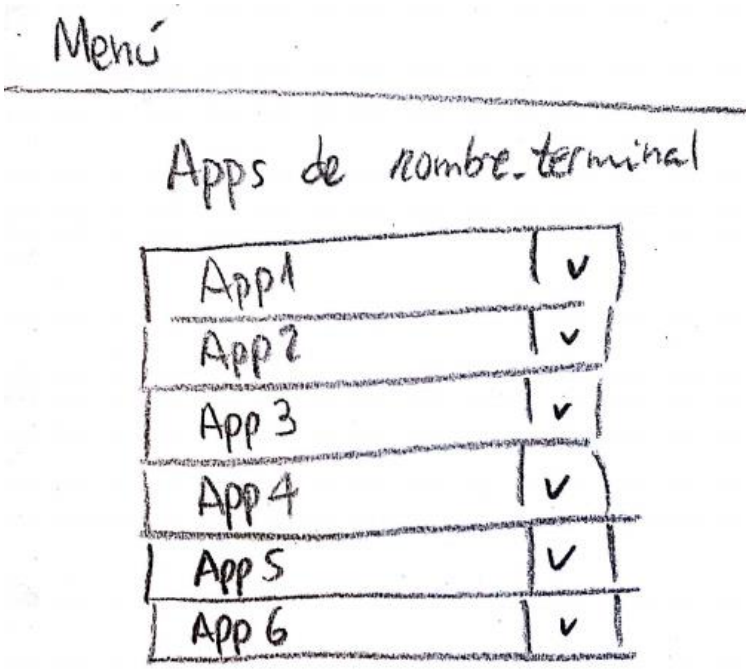

Ilustración 28 - Boceto listado de aplicaciones de un terminal concreto (Web)

Se mostrará entonces, el nombre del terminal, y una lista de desplegables que, al desplegarlo mostrará la siguiente información:

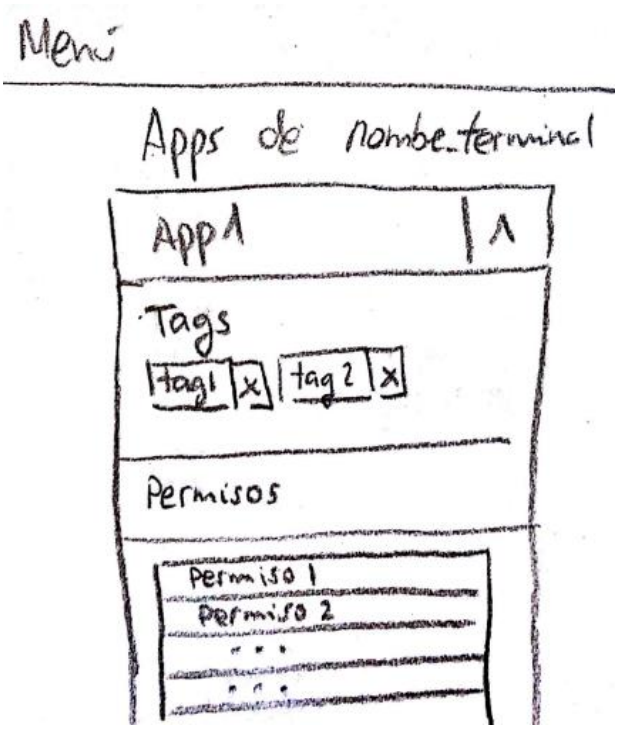

Ilustración 29 - Boceto de pliegue de aplicación en lista de aplicaciones (Web)

En esta sección se pueden ver cada una de las etiquetas añadidas a esta aplicación, además de cada uno de los permisos asociados a la misma.

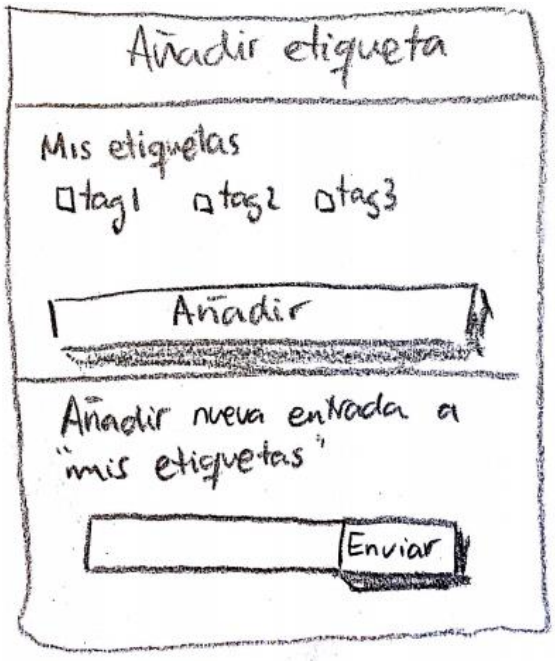

Ilustración 30 - Boceto de pop-up para añadir etiqueta (Web)

La lista de etiquetas que aparece, son las asociadas al usuario que está consultando ahora mismo sus aplicaciones, es decir, son personales de cada usuario. Para añadir nuevas entradas, tendrá que añadirse en el campo de la parte inferior y pulsar en Añadir. Una vez hecho esto, aparecerá una nueva entrada, de manera que podrá añadirse una nueva etiqueta.

Por otro lado, una vez cargada toda la WBAN del usuario en el sistema, podrá solicitar el análisis de similitud. Para ello tendrá que acceder a la sección *Realizar análisis de Similitud*:

Ilustración 31 - Boceto de página para solicitar análisis de similitud (Web)

Tras pulsar en el botón *Analizar*, se realizará la petición de cálculo de similitud. Es importante recordar que el cálculo de la similitud entre aplicaciones tarda unas cuantas horas.

Es por ello, que, hasta que no termine el análisis de similitud, el usuario no podrá realizar la evaluación de la similitud.

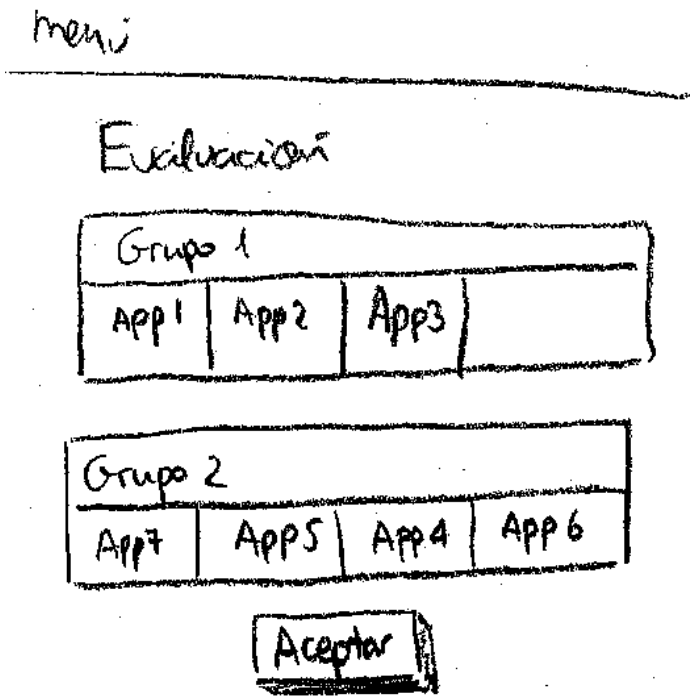

Ilustración 32 - Boceto de página para realizar la evaluación del sistema (Web)

El algoritmo devuelve un conjunto de clústeres o grupos, donde, vienen ordenados según similitud las aplicaciones.

El usuario, será el encargado de visualizar los grupos obtenidos, y modificarlos, en caso de ser necesario (de manera totalmente subjetiva, es decir, siguiendo los criterios que él vea necesarios).

Para poder realizar la modificación de grupos, el usuario únicamente tendrá que arrastrar aquellas aplicaciones que considere que están mal agrupadas, a otro grupo, donde, subjetivamente, quede mejor ordenador para el usuario.

Una vez terminada la evaluación, tendrá pulsar el botón de *Enviar*, para que quede almacenada su respuesta en el sistema.

4.6.2 Interfaz de usuario en Android

Una vez descargada la aplicación, veremos que aparece la siguiente pantalla, donde podremos introducir nuestras credenciales (usuario y contraseña) o registrarnos (esta funcionalidad nos llevará al registro web).

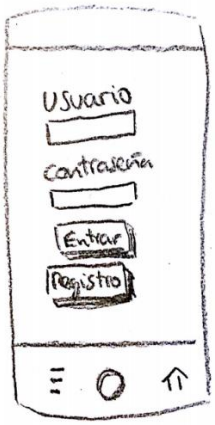

Ilustración 33 - Boceto actividad Login (Android)

Una vez autenticados en el servidor de manera correcta, aparecerá una pantalla similar a la siguiente.

La pantalla muestra la cantidad de aplicaciones encontradas en el terminal, junto a la lista completa de aplicaciones encontradas.

Si el usuario pulsa sobre el botón *Enviar,* enviará la información obtenida del terminal al servidor web, encargado de almacenar la información en la base de datos.

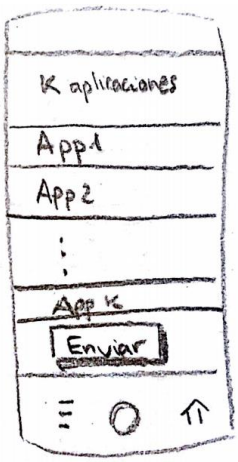

Ilustración 34 - Boceto actividad para listar y enviar apps al servidor (Android)

5. Implementación

Durante el apartado de implementación se detallará el proceso de instalación y adecuación del entorno de operación, además de definir la configuración del entorno de desarrollo.

5.1 Especificación del entorno tecnológico

Aquí se definen cuáles son las prestaciones tecnológicas necesarias para poder asegurar un correcto funcionamiento del sistema *SimilDroid*.

Tecnologías Hardware

- Memoria RAM de 8Gb de capacidad.
- Procesador Intel® Core™ i7-4700MQ CPU a 2.4 GHz
- Disco duro de 1 Tb de capacidad de almacenamiento.
- Tarjeta de red con conexión WiFi /Ethernet.
- Monitor de pantalla Samsung LS24F350FHUXEN 24" LED.
- Terminal móvil con SO Android 4.0+ (Xiaomi Redmi Note II).
- Terminal móvil con SO Android 4.0+ (Samsung Galaxy S3).
- Tablet con SO Android 4.0+ (Samsung Galaxy Tab S).

Tecnologías Software

- Sistema operativo Ubuntu 14.04 LTS.
- Servidor Apache httpd v2.2.31.
- Lenguajes de programación: PHP5, Java JDK 8, R v3.3.1.
- Android Studio v1.5.1.
- Gestor de base de datos MySQL v5.7.11.
- Librerías externas: Semilar, algoritmo R (desarrollado por tutora TFG), Semilar, WordNET y API Google Translate.
- VMWare Workstation 12 Player

5.2 Configuración del entorno de operación

Durante el presente apartado, se realizará una descripción de la configuración necesaria para hacer que el entorno de operación sea totalmente funcional. Así pues, se explicará la configuración necesaria en el servidor.

5.2.1 Configuración módulo Servidor Apache2

Tal y como se adelantaba en el apartado (), el servidor utilizado se trata de apache2. Es por ello que durante esta sección se describe la configuración que se ha llevado a cabo para el correcto funcionamiento del servidor [11].

La gran parte de la configuración del servidor de SimilDroid, sigue la configuración por defecto de Apache2. Únicamente se van a detallar las configuraciones modificadas.

- ***Módulo HTTPS***
  Para evitar que aparezcan problemas del estilo hombre en el medio e interceptación, se hace uso del protocolo HTTPS. Este protocolo es una modificación del conocido protocolo HTTP, añadiéndole seguridad gracias a la codificación de los mensajes mediante certificado digital, de manera que el usuario del sistema puede comprobar que realmente la información enviada no ha sido modificada.

Todo el contenido web que va a visualizarse de manera pública, estará contenido en la carpeta */var/www/html*. Dentro de esta carpeta tendremos el siguiente contenido:

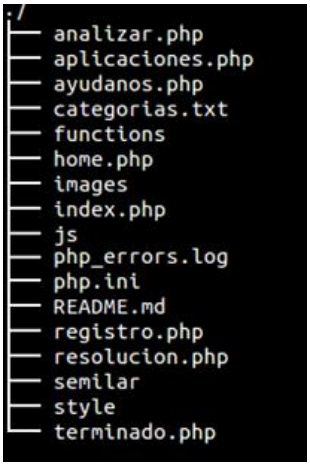

Ilustración 35 - Estructura interfaces web

Esta será lo que se conoce como carpeta raiz de la web del sistema web *SimilDroid*.

Tendremos que tener en cuenta, además, que la carpeta ***functions*** que aparece en la lista de la ilustración 35, contiene la capa ***Controlador*** de todo el sistema *SimilDroid*. Por lo tanto, la carpeta functions contiene la siguiente estructura:

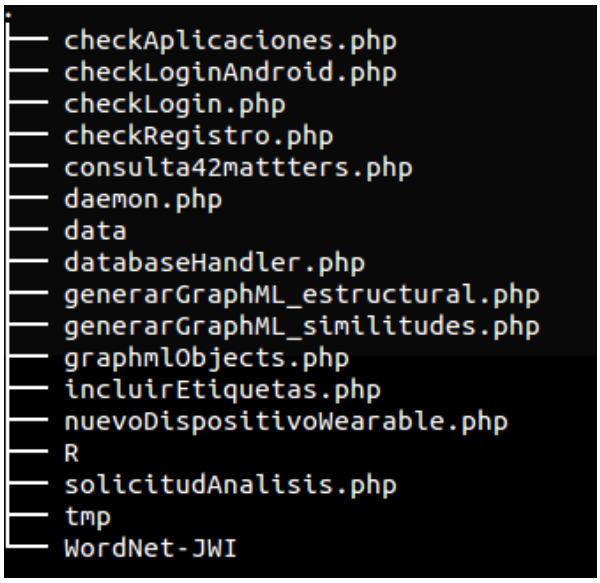

Ilustración 36 - Estructura controlador Web

5.2.2 Configuración módulo Android

Toda la configuración llevada a cabo sobre el cliente Android, se ha realizado haciendo uso de la herramienta *Android Studio*, herramienta proporcionada por los desarrolladores de *Android*.

La mayoría de las configuraciones que de la aplicación Android, son las definidas por defecto por la herramienta *Android Studio*. A continuación mostraremos algunas configuraciones extra realizadas en este proyecto.

- **Configuración del cliente HTTP (OkHttp):** para poder realizar peticiones de tipo POST al servidor web, era necesario configurar un cliente que pudiera realizar este tipo de acciones. El cliente seleccionado finalmente fue OkHttp. Para poder utilizarlo basta con descargar la librería OkHttp, y solicitar a Android que la compile al iniciar la aplicación. Para esto último se añade la línea:

  ```
  compile 'com.squareup.okhttp3:okhttp:3.0.1'
  ```

  en el fichero de configuración *build.grandle (Module:app)*, concretamente en la sección de dependencias (*dependencies*).

- **Firma de la aplicación Android:** se ha realizado para determinar que es un proceso necesario en este tipo de aplicaciones (que hacen uso de datos privados del usuario).

  Para realizar este proceso, en primer lugar se seleccionará la opción de *generate signed APK*.

  Tendremos que definir, en primer lugar, un keystore:

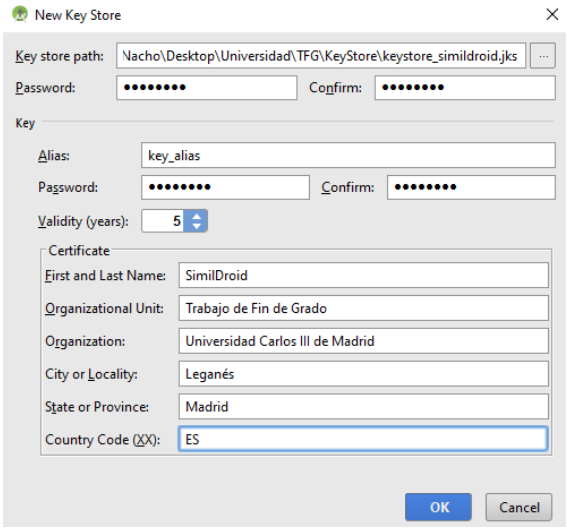

Ilustración 37 - Creación de keystore

Una vez creado el *keystore*, lo usaremos para realizar la firma de la aplicación.

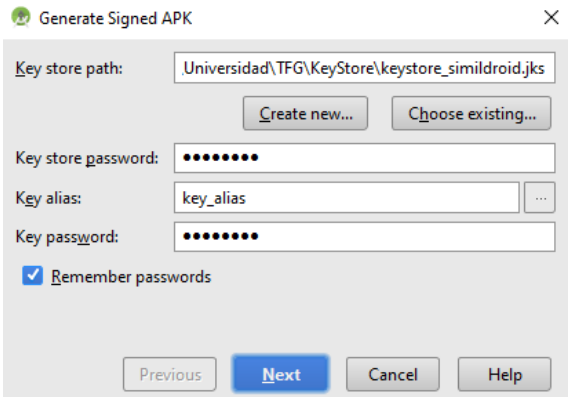

Ilustración 38 - Creación de apk firmada

Con respecto a la estructura que el proyecto que contiene la aplicación Android, es la siguiente:

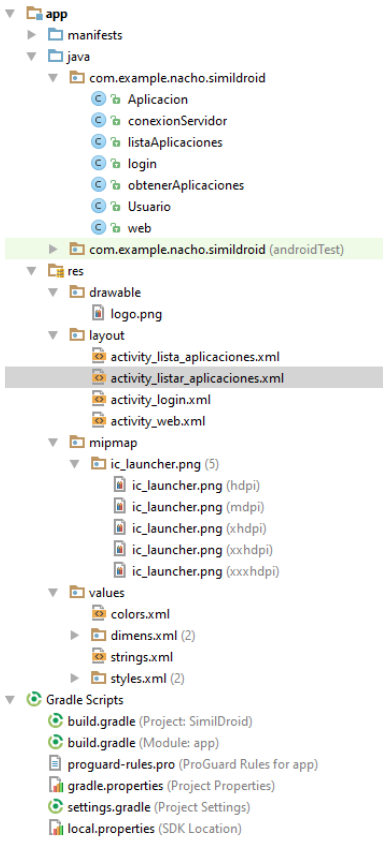

Ilustración 39 - Estructura Android

5.2.3 Configuración módulo R

Como se indica en el análisis de la arquitectura seleccionada (Apartado 3.3), el sistema cuenta con un módulo R para la ejecución del algoritmo desarrollado por el cliente. Como con los dos apartados anteriores, mostraremos el esquema de carpetas creado para el correcto funcionamiento del sistema. En la *Ilustración 36* - Estructura controlador Web puede observarse que existe una carpeta llamada R. Esta será la carpeta Raíz del módulo R.

La estructura del módulo R es un poco más compleja que la de los otros dos módulos:

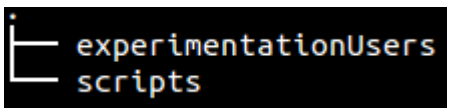

Ilustración 40 - Estructura de la carpeta raíz de módulo R

El contenido de estas dos carpetas es el siguiente:

- ***Carpeta scripts***: contiene todos los scripts necesarios para calcular la similitud entre las aplicaciones. Podemos decir, que esta carpeta contiene el algoritmo en sí. El cerebro del módulo R.
- ***Carpeta experimentationUsers***: esta carpeta contiene una carpeta por cada usuario al que se le ha realizado el análisis de similitud. Supongamos un total de cuatro usuarios, el contenido de esta carpeta será el siguiente:

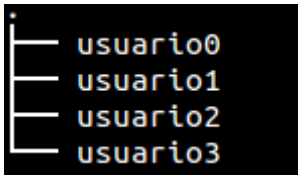

Ilustración 41 - Estructura de R/experimentationUsers

Cada una de las carpetas asociadas a un usuario, contiene, a su vez, una carpeta por cada ejecución que se ha llevado a cabo. Así por ejemplo, el usuario3 contiene estas ejecuciones:

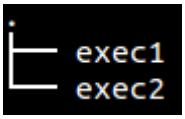

Ilustración 42 - Estructura de R/experimentationUsers/usuario3

Por cada ejecución, se crea el siguiente conjunto de elementos (esta vista en concreto muestra el contenido de exec1):

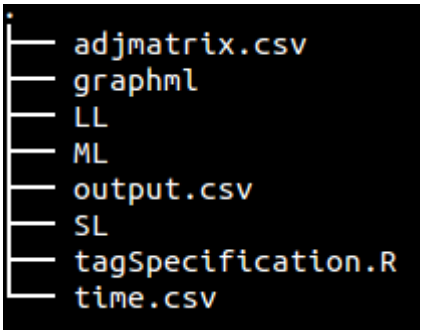

Ilustración 43 – Estructura de R/experimentationUsers/usuario3/exec1

Los elementos de la *Ilustración 43* – Estructura de R/experimentationUsers/usuario3/exec1 son los siguientes:

- ***adjmatrix.csv:*** matriz de adyacencia del grafo G_u.
- ***output.csv:*** esto son los resultados de la distancia intracluster para todas las variantes de esta ejecución.
- ***graphml:*** carpeta que contiene los grafos utilizados en esta ejecución. Si recordamos, existen dos grafos:
    - Grafo estructural con atributos.
    - Grafo con similitudes entre atributos.
- ***LL, ML, y SL:*** estas tres carpetas muestran las longitudes de path probadas. Así pues:
    - *LL:* longitud de path igual a al diámetro - 1.
    - *ML:* longitud de path igual ¾ del diámetro del grafo G_u.
    - *SL:* longitud de path igual a ½ del diámetro del grafo G_u.
- ***tagSpecification.R*:** este fichero contiene una lista con la definición de todos los tags del usuario. Es necesario para poder saber cuántos tags (o etiquetas) hay, y cuáles son.
- ***time.csv:*** almacena los tiempos de ejecución para todas las variantes.

Cada una de las carpetas LL, ML y SL, contienen un subconjunto de carpetas. Así por ejemplo, la carpeta SL, contiene lo siguiente:

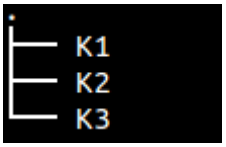

Ilustración 44 – Estructura de R/experimentationUsers/usuario3/exec1/SL

Cada una de las cuales se refiere a un tamaño de clustering distinto:

- ***K1:*** tamaño del cluster estimado como óptimo.
- ***K2:*** tamaño del cluster estimado como óptimo – 1.
- ***K3:*** tamaño del cluster estimado como óptimo – 2.

Dentro de cada carpeta ***KX***, existe lo siguiente:

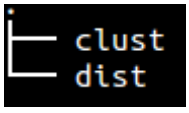

Ilustración 45 - Estructura de R/experimentationUsers/usuario3/exec1/SL/K1

Donde:

- ***Clust***: contiene los resultados del clustering para cada variante. Almacenando, además, los pesos de cada iteración.
- ***Dist***: contiene las matrices de cada iteración (distancia intracluster).

5.3 Interfaz gráfica final del sistema

En el apartado *4.6* se mostraban varios bocetos de cómo sería la interfaz de usuario. Con la intención de dar una mejor visión de cómo interactuaría un usuario con el sistema, se mostrará el resultado final de la interfaz de usuario.

Entonces, el funcionamiento quedaría de la siguiente manera:

- ***Login (módulo web):***

Ilustración 46 - Interfaz final login (web)

- ***Registro (módulo web):***

Ilustración 47 - Interfaz final Registro (web)

- ***Listado de terminales de la red WBAN (módulo web):***

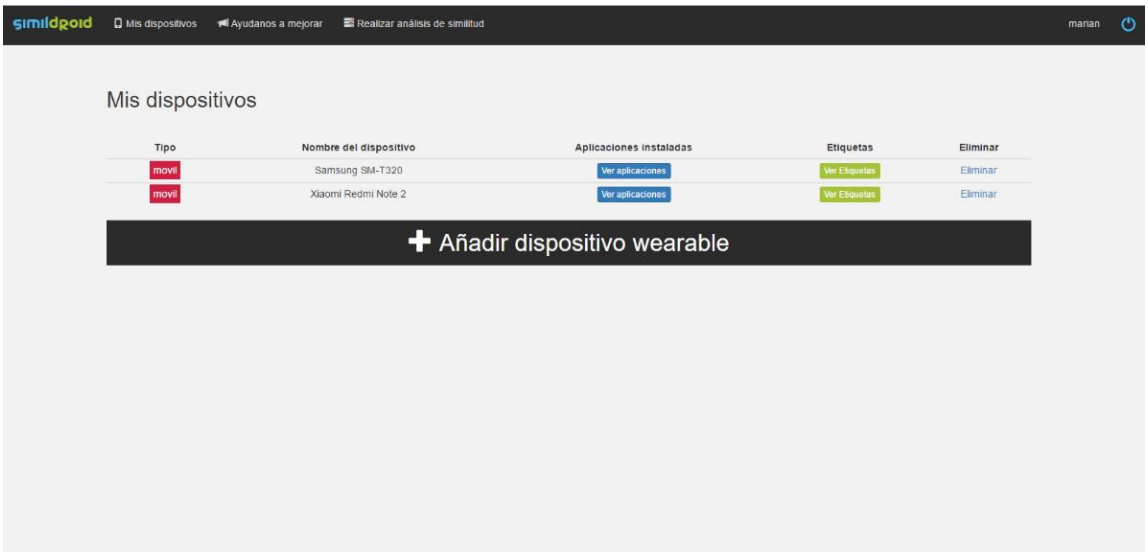

Ilustración 48 - Interfaz final listado terminales (web)

- ***Añadir nuevo dispositivo Wearable (módulo web):***

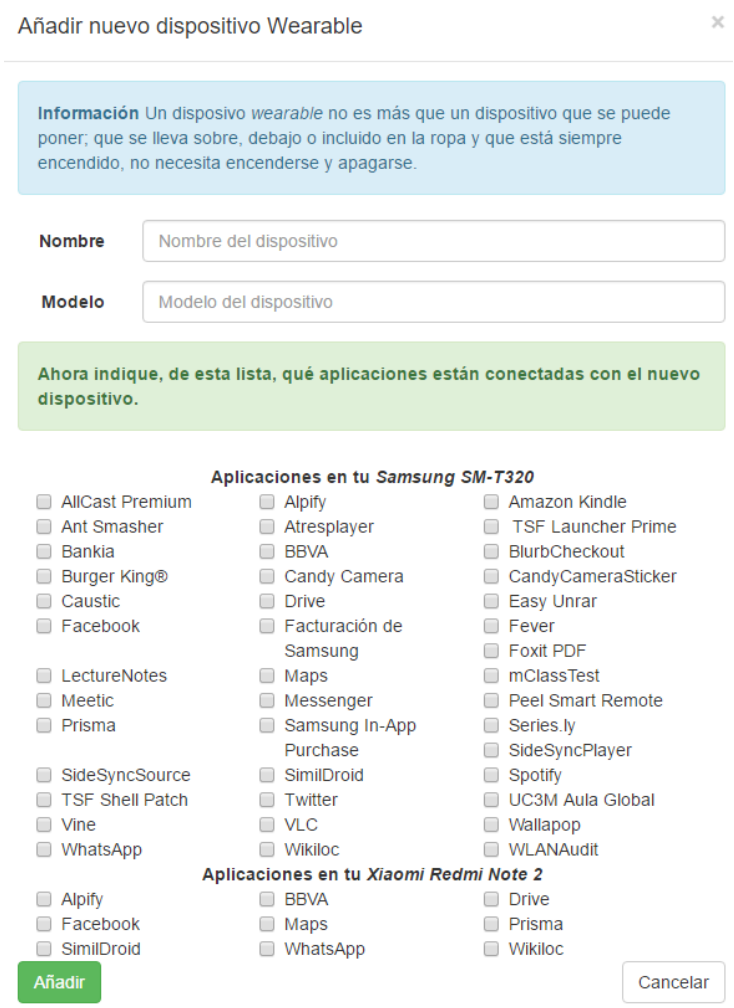

Ilustración 49 - Interfaz final Añadir Wearable (web)

- ***Listado de aplicaciones de un terminal concreto (módulo web):***

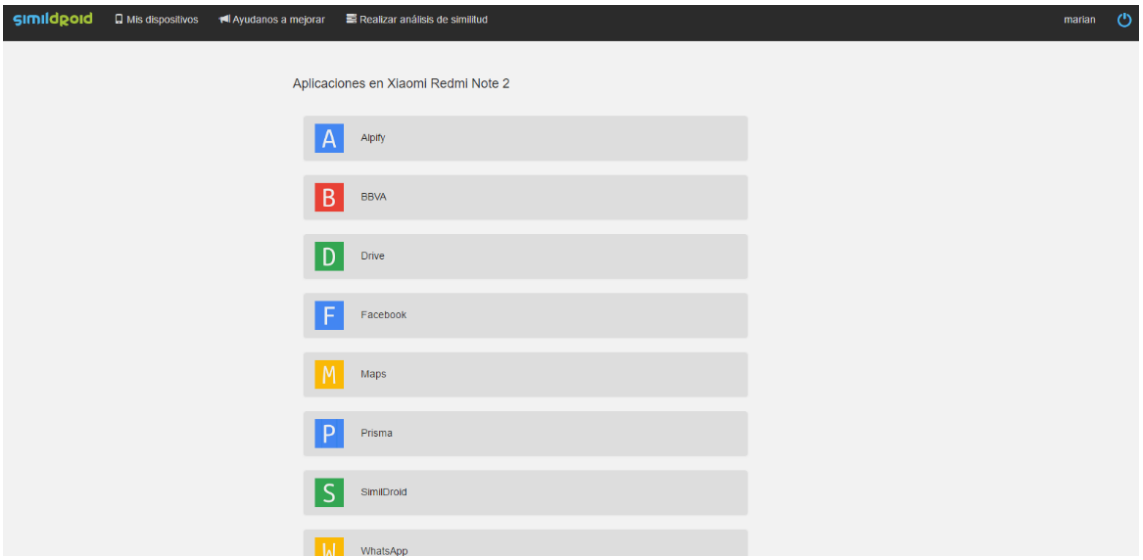

Ilustración 50 - Interfaz final listado aplicaciones (web)

- ***Elemento aplicación desplegado (módulo web):***

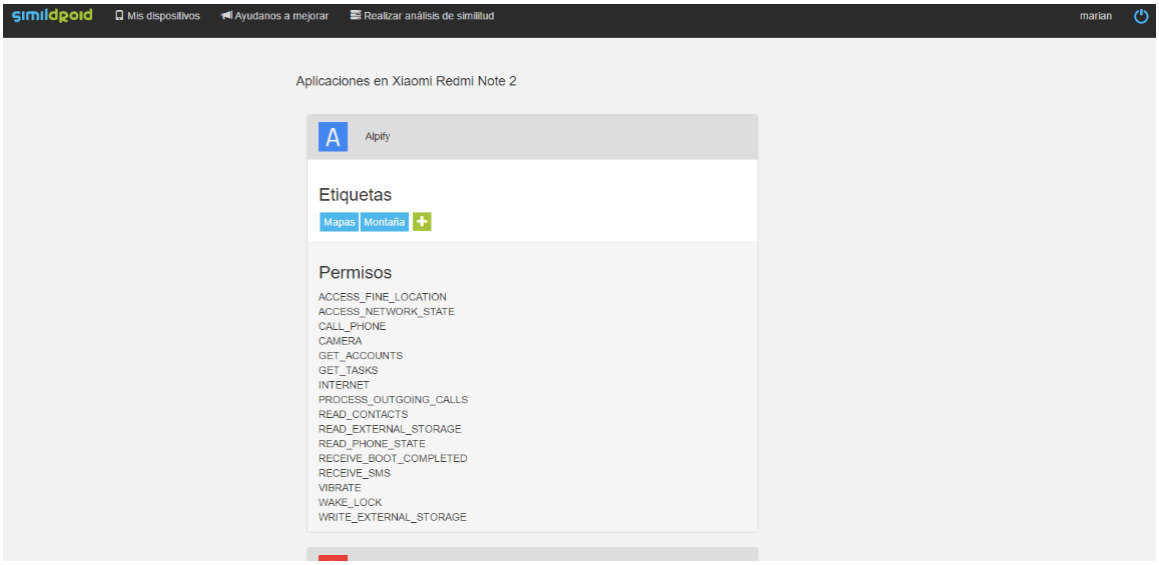

Ilustración 51 - Interfaz final información aplicación (web)

- ***Añadir nueva etiqueta (módulo web):***

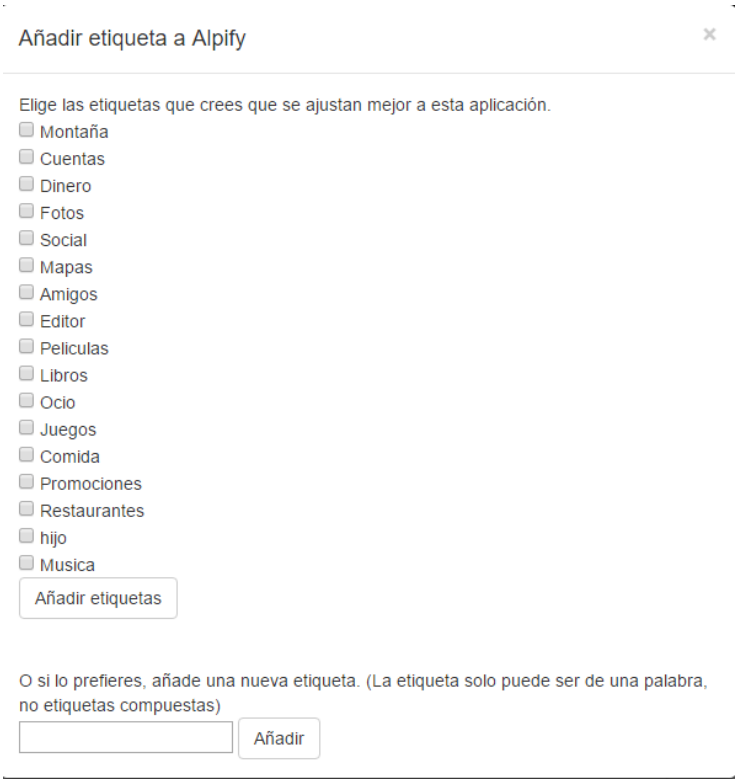

Ilustración 52 - Interfaz final añadir etiqueta (web)

- ***Realizar petición de cálculo de similitud (módulo web):***

Ilustración 53 - Interfaz final Solicitar similitud (web)

- ***Evaluación del cálculo de similitud (módulo web):***

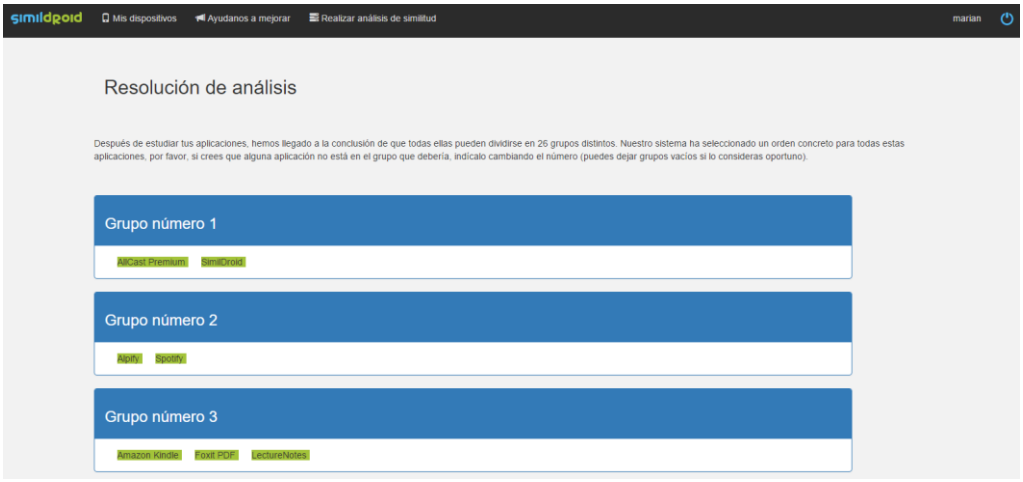

Ilustración 54 - Interfaz final evaluación de similitud (web)

- ***Login (módulo Android):***

Ilustración 55 - Interfaz final Login (Android)

- ***Lista de aplicaciones instaladas (módulo Android):***

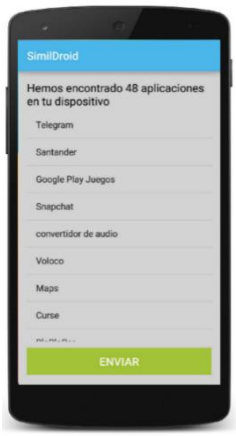

Ilustración 56 - Interfaz final aplicaciones instaladas (Android)

6. Evaluación del Sistema

En el presente apartado se mostrarán todas las pruebas necesarias para poder demostrar y cerciorarnos de que el sistema realiza todas las funcionalidades solicitadas por el usuario sin ningún problema. Servirá entonces para demostrar que el sistema cumple con los requisitos que se han definido con anterioridad.

6.1 Definición del entorno del plan de pruebas

Durante el presente apartado se mostrará cual es el entorno utilizado para la realización de las pruebas.

Las pruebas serán realizadas sobre varios elementos. En primer lugar, el servidor web, mientras que en segundo lugar habrá un par de terminales para la obtención de las aplicaciones que tienen instaladas (y todos los atributos asociados). También será necesario el simulador propio de Android Studio para tener otro terminal que examinar.

El servidor web sobre el que se van a realizar las consultas tiene las siguientes características:

Disco duro	1 Tb
Memoria RAM	8 Gb
Sistema Operativo	Linux 14.04 LTS
Software del servidor	Apache2 HTTPD

Tabla 66 - Descripción servidor web

Por otra parte, las características de los otros dos elementos físicos de estas pruebas (los móviles), tendrán las siguientes características y estarán asociados a los siguientes nombres de usuario:

Marca	Samsung
Modelo	Galaxy S3
Sistema Operativo	Android
Versión del Sistema Operativo	4.3

Número de aplicaciones instaladas	15
Usuario propietario	Axel

Tabla 67 - Descripción dispositivo 1 de pruebas

Marca	Xiaomi
Modelo	Redmi Note 2
Sistema Operativo	Android
Versión del Sistema Operativo	5.0.2
Número de aplicaciones instaladas	9
Usuario propietario	Marian

Tabla 68 - Descripción dispositivo 2 de pruebas

Marca	Samsung
Modelo	Galaxy Tab Pro
Sistema Operativo	Android
Versión del Sistema Operativo	4.4.2
Número de aplicaciones instaladas	41
Usuario propietario	Marian

Tabla 69 - Descripción dispositivo 3 de pruebas

Marca	Samsung
Modelo	Galaxy Note 3
Sistema Operativo	Android
Versión del Sistema Operativo	5.0.2
Número de aplicaciones instaladas	41
Usuario propietario	Marian

Tabla 70 - Descripción dispositivo 4 de pruebas

6.2 Especificación del plan de pruebas

Con la intención de evaluar, por tanto, el cumplimiento de los requisitos del sistema, se definirán un grupo de pruebas sobre cada uno de los módulos del sistema. En este trabajo

se ha decidido realizar pruebas de caja negra ya que ayudarán a demostrar que las funcionalidades del sistema están bien realizadas, sin dar detalle de cómo está implementado.

El plan de pruebas, se ha decidido dividirlo en varios tipos, ya que cada tipo tiene una función distinta:

- **Pruebas unitarias:** pruebas creadas para evaluar un solo componente del sistema.

- **Pruebas de integración:** pruebas creadas para evaluar el correcto funcionamiento entre los distintos componentes del sistema (como un conjunto).

6.3 Especificación de pruebas unitarias

En este apartado se especifica qué pruebas unitarias se llevarán a cabo, y cuál es el formato de la tabla descriptiva de la prueba:

Cada prueba irá acompañada de una tabla que la describe:

PU-YY	
Nombre	
Módulo	
Descripción	
Entrada	
Salida	
Requisitos	

Tabla 71 - Formato prueba unitaria

Donde cada campo contiene la siguiente información:

- **PU-YY:** Identificador de la prueba compuesto por:
  - **PU:** código que identifica esta prueba como unitaria.
  - **YY:** código que identifica unívocamente la prueba unitaria dentro del conjunto de pruebas unitarias.
- **Nombre:** nombre descriptivo de la prueba.
- **Módulo:** módulo que se va a poner a prueba.
- **Descripción:** descripción concisa de la prueba.
- **Entrada:** Qué tiene que recibir la prueba.

- **Salida**: Qué debería de devolver la prueba.
- **Requisitos:** se refiere a qué necesita la prueba para poder llevarse a cabo.

PU-01	
Nombre	Obtener lista de aplicaciones
Módulo	Android
Descripción	Comprobar que la aplicación *SimilDroid* en Android obtiene la lista de aplicaciones instaladas en el dispositivo, junto a toda la información relativa a cada aplicación (permisos asociados, nombre de paquete y nombre del terminal).
Entrada	Nombre de usuario
Salida	Estructura JSON con el contenido relativo al terminal sobre el que se realiza el estudio (aplicaciones, permisos asociados, nombre de paquete asociado, nombre del usuario y nombre del terminal)
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema.

Tabla 72 - Prueba unitaria PU-01

PU-02	
Nombre	Añadir dispositivo Wearable
Módulo	Web
Descripción	Probar a introducir un nuevo dispositivo Wearable.
Entrada	Nombre del nuevo dispositivo y aplicación o aplicaciones asociadas
Salida	Se habrá creado un nuevo terminal con el nombre introducido en la entrada
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema. El usuario que realizará las pruebas habrá tenido que introducir al menos un terminal con al menos una aplicación en el sistema.

Tabla 73 - Prueba unitaria PU-02

PU-03	
Nombre	Añadir etiqueta de usuario
Módulo	Web
Descripción	Añadir una nueva etiqueta al conjunto de etiquetas del usuario. Estas etiquetas, servirán más adelante para poder etiquetar aplicaciones y/o terminales.
Entrada	El encargado de realizar la prueba decidirá el nombre de una nueva etiqueta y la insertará en el sistema.

Salida	La lista de etiquetas de usuario se verá incrementada en un elemento, el insertado en la entrada.
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema.

Tabla 74 - Prueba unitaria PU-03

PU-04	
Nombre	Añadir etiqueta a aplicación
Módulo	Web
Descripción	Añadir una nueva etiqueta a alguna de las aplicaciones instaladas en alguno de los terminales estudiados.
Entrada	Añadir una nueva etiqueta a alguna aplicación asociada a la WBAN de prueba.
Salida	La tabla *Aplicacion_Etiqueta* se verá modificada con las *k* etiquetas que haya seleccionado para dicha aplicación.
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema. El usuario que realizará las pruebas habrá tenido que introducir al menos un terminal con al menos una aplicación en el sistema.

Tabla 75 - Prueba unitaria PU-04

PU-YY	
Nombre	Añadir etiqueta a terminal
Módulo	Web
Descripción	Añadir una nueva etiqueta a algún terminal asociado a la WBAN de prueba.
Entrada	El usuario que realizará las pruebas elegirá una o varias etiquetas que describirán, de manera subjetiva, el tipo de terminal.
Salida	La tabla *Terminal_Etiqueta* se verá modificada con las *k* etiquetas que haya seleccionado para dicho terminal.
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema. El usuario que realizará las pruebas habrá tenido que introducir al menos un terminal.

Tabla 76 – Prueba unitaria PU-05

PU-06	
Nombre	Realizar petición de cálculo de similitud
Módulo	Web
Descripción	El usuario de prueba realizará una solicitud de cálculo de similitud para las aplicaciones de su WBAN.
Entrada	El usuario que realizará las pruebas pulsará el botón de realizar análisis de similitud.
Salida	La tabla *Tarea* se verá modificada con la nueva solicitud del usuario de prueba.
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema. El usuario que realizará las pruebas habrá tenido que introducir al menos un terminal, con al menos una aplicación.

Tabla 77 – Prueba unitaria PU-06

PU-07	
Nombre	Añadir etiqueta a terminal
Módulo	Web
Descripción	El usuario de prueba realizará la evaluación de la similitud entre las aplicaciones de la WBAN de prueba.
Entrada	El usuario de prueba recolocará las aplicaciones en los grupos en los que él crea que debería de estar.
Salida	Se crean dos ficheros, uno con la similitud calculada por el algoritmo del cliente, y otro con la similitud modificada por el usuario.
Requisitos	El usuario que realizará las pruebas ha tenido que registrarse y autenticarse en el sistema. El usuario que realizará las pruebas habrá tenido que introducir al menos un terminal, con al menos una aplicación. El algoritmo del cliente tendrá que haber realizado el cálculo de la similitud. El usuario tendrá que haber recibido un mensaje de aviso de fin de cálculo de similitud.

Tabla 78 – Prueba unitaria PU-07

6.4 Especificación de pruebas de integración

En este apartado se especifica qué pruebas de integración se llevarán a cabo, y cuál es el formato de la tabla descriptiva de la prueba:

Cada prueba irá acompañada de una tabla que la describe:

PI-YY	
Nombre	
Objetivo	
Precondiciones	
Acciones	
Postcondiciones	

Tabla 79 - Formato prueba de integraci

Donde cada campo contiene la siguiente información:

- **PI-YY:** Identificador de la prueba compuesto por:
  - **PI:** código que identifica esta prueba como prueba de integración.
  - **YY:** código que identifica unívocamente la prueba de integración.
- **Nombre:** nombre descriptivo de la prueba.
- **Objetivo:** módulo que se va a poner a prueba.
- **Precondiciones:** condiciones que se deben de dar antes de la prueba.
- **Acciones:** acciones necesarias para llevar a cabo la prueba por completo.
- **Postcondiciones:** condiciones tras la ejecución de la prueba.

PI-01	
Nombre	Comprobación Login desde Android
Objetivo	Comprobar si existe comunicación entre la aplicación cliente Android y el servidor Web
Precondiciones	▪ Servidor ha de estar en funcionamiento. ▪ Usuario de prueba ha de estar registrado en el sistema.
Acciones	▪ Descargar aplicación Android en el terminal de prueba. ▪ Abrir aplicación e introducir los credenciales de inicio de sesión. ▪ Pulsar botón “Acceder”

Postcondiciones	▪ Si existe comunicación, el usuario podrá visualizar las aplicaciones del terminal. ▪ Si no existe comunicación, aparecerá un mensaje de error.

Tabla 80 – Prueba de integración PI-01

PI-YY	
Nombre	Visualizar aplicaciones de un terminal añadido a la estructura WBAN de prueba
Objetivo	El objetivo es comprobar que se envía correctamente la estructura perteneciente a un terminal, y que el servidor web la almacena de forma correcta.
Precondiciones	▪ Servidor ha de estar en funcionamiento. ▪ Usuario de prueba ha de estar registrado en el sistema.
Acciones	▪ Usuario de prueba se loguea desde la aplicación Android, desde donde podrá ver todas las aplicaciones instaladas. ▪ Enviar estructura al servidor web.
Postcondiciones	▪ El usuario de prueba se loguea desde el navegador en el servidor web. ▪ El usuario de prueba podrá visualizar una nueva entrada en “Mis dispositivos”, correspondiente al nuevo terminal añadido. ▪ El usuario de prueba podrá visualizar la lista de aplicaciones instaladas en el terminal, junto a la lista de permisos asociados a cada una de las aplicaciones.

Tabla 81 – Prueba de integración PI-02

6.5 Resultado de pruebas unitarias y de integración

En este subapartado vamos a mostrar los resultados obtenidos para cada una de las pruebas descritas en los dos anteriores apartados.

Prueba	Módulo llevado a prueba		Resultado	
	Android	*Web*	*Superada*	*Fallada*
PU-01				
PU-02				
PU-03				
PU-04				
PU-05				
PU-06				
PU-07				
PI-01				
PI-02				

Tabla 82 - Resultados pruebas unitarias e integración

6.6 Pruebas con usuarios

Se han llevado a cabo varias pruebas con usuarios. Los usuarios son, concretamente, los indicados en el apartado *6.1 Definición del entorno del plan de pruebas*.

No se han hecho pruebas con más usuarios, ya que durante las pruebas, se han detectado problemas en el algoritmo, pues según los autores del método de ajuste de los pesos que se utiliza en el algoritmo del cliente, tras un número relativamente bajo de iteraciones, los resultados de los cálculos deben alcanzar cierta estabilidad. Sin embargo, esto no ocurre así de forma general en los casos de estudio analizados. Por tanto, era aconsejable ajustar primero el algoritmo antes de realizar pruebas más extensas.

Durante este apartado mostraremos, para cada uno de los usuarios, la salida que obtuvo el algoritmo, y la evaluación que realizó el usuario tras dicha salida. En el apartado *6.1 Definición del entorno del plan de pruebas* se define qué terminal pertenece a cada usuario.

Por otro lado, en el Anexo II, se adjuntan varios grafos creados por el sistema SimilDroid, que se introducirán en el algoritmo para ser evaluados.

6.6.1 Prueba con Usuario 1

Usuario 1	
Nombre	Marian
Número de dispositivos	2
Número total de aplicaciones	50
Número de clusters ofrecidos por el algoritmo	24

Tabla 83 - Datos de usuario de prueba 1

Resultados Usuario 1		
Grupo	**Salida**	**Evaluación**
1	AllCast Premium-1 Atresplayer-1 Peel Smart Remote-1 Series.ly-1 **SimilDroid-1** VLC-1	AllCast Premium-1 Atresplayer-1 Peel Smart Remote-1 Series.ly-1 VLC-1
2	**Alpify-1** Spotify-1	Spotify-1
3	Amazon Kindle-1 Foxit PDF-1 LectureNotes-1	Amazon Kindle-1 Foxit PDF-1 LectureNotes-1 **Drive-1** **Drive-2**
4	**Ant Smasher-1** CandyCameraSticker-1	CandyCameraSticker-1 **Prisma-1** **Prisma-2** **Candy Camera-1**
5	TSF Launcher Prime-1 SideSyncSource-1	TSF Launcher Prime-1 SideSyncSource-1 **TSF Shell Patch-1**
6	Bankia-1	Bankia-1 **BBVA-1** **BBVA-2**
7	**BBVA-1**	-
8	BlurbCheckout-1 **Candy Camera-1** Facturación de Samsung-1	BlurbCheckout-1 Facturación de Samsung-1
9	Burger KingÂ®-1 Fever-1 Meetic-1	Burger KingÂ®-1 Fever-1 Meetic-1
10	Caustic-1	Caustic-1
11	**Drive-1** Facebook-1 Vine-1 Facebook-2	Facebook-1 Vine-1 Facebook-2
12	Easy Unrar-1 **Prisma-1** Samsung In-App Purchase-1 SideSyncPlayer-1 **Prisma-2**	Easy Unrar-1 Samsung In-App Purchase-1 SideSyncPlayer-1
13	Maps-1	Maps-1 **Maps-2**
14	mClassTest-1	mClassTest-1

		UC3M Aula Global-1
15	Messenger-1 Twitter-1 WhatsApp-1 WhatsApp-2	Messenger-1 Twitter-1 WhatsApp-1 WhatsApp-2
16	**TSF Shell Patch-1**	-
17	**UC3M Aula Global-1**	-
18	Wallapop-1	Wallapop-1
19	Wikiloc-1	Wikiloc-1 **Wikiloc-2**
20	WLANAudit-1	WLANAudit-1
21	Alpify-2 **Drive-2** SimilDroid-2	Alpify-2 **Alpify-1** SimilDroid-2 **SimilDroid-1**
22	**BBVA-2**	
23	**Maps-2**	-
24	**Wikiloc-2**	

Tabla 84 - Resultados Usuario de prueba 1

NOTA: los valores de la columna *Salida* que están coloreados de **rojo**, indican que ese elemento ya no pertenece a ese grupo. Mientras que los valores de la columna de *Evaluación* coloreados en **verde**, indican que son valores nuevos de ese grupo.

Por otra parte, los nombres de las aplicaciones vienen acompañados de un número. Ese número es el identificador del terminal al que pertenecen. No debería de influir en nada dicho número para la evaluación del usuario.

Si estudiamos por encima la evaluación del usuario, podemos ver que hay bastantes grupos bien organizados (por ejemplo los grupos 9, 11, 12 y 15).

6.6.2 *Prueba con Usuario 2*

Usuario 2	
Nombre	Nacho
Número de dispositivos	4 (3 wearable)
Número total de aplicaciones	59
Número de clusters ofrecidos por el algoritmo	13

Tabla 85 - Usuario de prueba 2

Resultados Usuario 2		
Grupo	**Salida**	**Evaluación**
1	Adobe Acrobat-1 **Aptoide-1** CamScanner-1 Drive-1 **Google Play Juegos-1** Hojas de cÃilculo-1 MEGA-1 Root Explorer-1 **Santander-1** **com.android.gesture.builder-7** **Task Organizer-7**	Adobe Acrobat-1 CamScanner-1 Drive-1 Hojas de cÃilculo-1 MEGA-1 Root Explorer-1
2	å°□ç±³é£Žè¡Œæ'¬æ"¾æ□'ä»¶-1 ç^±å¥‡è‰°æ'¬æ"¾å™¨-1 è¯¬éŸ³è®¾ç½®-1 Spotify-1 **Tiempo BZ-1** **Sample Soft Keyboard-7**	å°□ç±³é£Žè¡Œæ'¬æ"¾æ□'ä»¶-1 ç^±å¥‡è‰°æ'¬æ"¾å™¨-1 è¯¬éŸ³è®¾ç½®-1 **Shazam-1** Spotify-1
3	Boomerang-1 Curse-1 Facebook-1 Fb Video Downloader-1 Imgur-1 Instagram-1 InstaSave-1 Messenger-1 Repost-1 Snapchat-1 Tonos para WhatsApp-1 WhatsApp-1	Boomerang-1 Curse-1 Facebook-1 Fb Video Downloader-1 Imgur-1 Instagram-1 InstaSave-1 Messenger-1 Repost-1 Snapchat-1 Tonos para WhatsApp-1 WhatsApp-1
4	Busco Un Chollo-1 Waze-1	**HERE Maps-1** Busco Un Chollo-1 Waze-1
5	**Chrome Beta-1** **Head Ball-1** **Magnetic Field Detector-1** Meme Generator Free-1 PPTV player-1 **Shazam-1** TimeLapse Video Recorder-1 Wallpaper Carousel-1 Example Wallpapers-7	Meme Generator Free-1 PPTV player-1 TimeLapse Video Recorder-1 Wallpaper Carousel-1 Example Wallpapers-7

6	FatSecret-1	FatSecret-1 **FatSecret-5** **FatSecret-6**
7	**HERE Maps-1**	**Aptoide-1** **com.android.gesture.builder-7** **Task Organizer-7** **Santander-1** **API Demos-7** **Head Ball-1** **Magnetic Field Detector-1** **Tiempo BZ-1** **Sample Soft Keyboard-7** **Wallpaper Carousel-5** **Widget Preview-7**
8	Infinite Slice-1 Minecraft - Pocket Edition-1 PokÃ©mon GO-1 **API Demos-7** **com.android.smoketest-7** **com.android.smoketest.tests-7**	Infinite Slice-1 Minecraft - Pocket Edition-1 PokÃ©mon GO-1 **Google Play Juegos-1**
9	Madrid Metro|Bus|Cercanias-1 **Widget Preview-7**	Madrid Metro|Bus|Cercanias-1 **Renfe CercanÃ-as-1**
10	**Renfe CercanÃ-as-1** SimilDroid-1	SimilDroid-1 **SimilDroid-7**
11	Wallapop-1 **SimilDroid-7**	Wallapop-1
12	Chrome Beta-5 **FatSecret-5** **Wallpaper Carousel-5**	**Chrome Beta-1** Chrome Beta-5 **Chrome Beta-6**
13	**Chrome Beta-6** **FatSecret-6**	

Tabla 86 - Resultados usuario de prueba 2

NOTA: los valores de la columna *Salida* que están coloreados de **rojo**, indican que ese elemento ya no pertenece a ese grupo. Mientras que los valores de la columna de *Evaluación* coloreados en **verde**, indican que son valores nuevos de ese grupo.

Por otra parte, los nombres de las aplicaciones vienen acompañados de un número. Ese número es el identificador del terminal al que pertenecen. No debería de influir en nada dicho número para la evaluación del usuario.

En esta prueba, existen una gran cantidad de grupos bien ordenados, aunque otros grupos (que contienen aplicaciones difíciles de categorizar), están más mezclados. El grupo que más destaca es el 3, ya que a este grupo ni le sobra ni le falta ninguna aplicación.

6.6.3 Prueba con Usuario 3

Usuario 3	
Nombre	Axel
Número de dispositivos	2 (1 wearable)
Número total de aplicaciones	16
Número de clusters ofrecidos por el algoritmo	7

Tabla 87 - Usuario de prueba 3

Resultados Usuario 3		
Grupo	**Salida**	**Evaluación**
1	**360 Security Lite-4** **My Boy! Free-4** **My OldBoy! Free-4** Telegram-4 WhatsApp-4	Telegram-4 WhatsApp-4 **Duo-4**
2	ACQUIRE-4 **Duo-4** SimilDroid-4	ACQUIRE-4 SimilDroid-4
3	Barcode Scanner-4 Wallapop-4	Barcode Scanner-4 Wallapop-4
4	Earth-4 HERE Maps-4 Waze-4	Earth-4 HERE Maps-4 Waze-4
5	mySamsung-4	mySamsung-4 **360 Security Lite-4**
6	PlayerPro-4	PlayerPro-4 **PlayerPro-5**
7	**PlayerPro-5**	**My Boy! Free-4** **My OldBoy! Free-4**

Tabla 88 - Resultados usuario de prueba 3

NOTA: los valores de la columna *Salida* que están coloreados de **rojo**, indican que ese elemento ya no pertenece a ese grupo. Mientras que los valores de la columna de *Evaluación* coloreados en **verde**, indican que son valores nuevos de ese grupo.

Por otra parte, los nombres de las aplicaciones vienen acompañados de un número. Ese número es el identificador del terminal al que pertenecen. No debería de influir en nada dicho número para la evaluación del usuario.

En este caso, quizá por tratarse de un terminal con muy pocas aplicaciones, la agrupación ha salido bastante bien. Realmente hay tres grupos muy bien organizados (2, 3 y 4).

6.6.4 Tiempos de ejecución de las pruebas con usuarios

En esta sección vamos a visualizar los tiempos que han tardado cada una de las ejecuciones de las pruebas anteriores.

En las tres gráficas se muestra el tiempo de ejecución acumulado según se van ejecutando las iteraciones. Cada línea hace referencia a una ejecución distinta con diferentes parámetros de configuración. Los valores de estos parámetros se muestran en la leyenda de cada serie: el primer número hace referencia a la longitud del camino aleatorio utilizado para el cálculo de la similitud, mientras que el segundo hace referencia al número de *clusters* K.

- ***Usuario 1:***

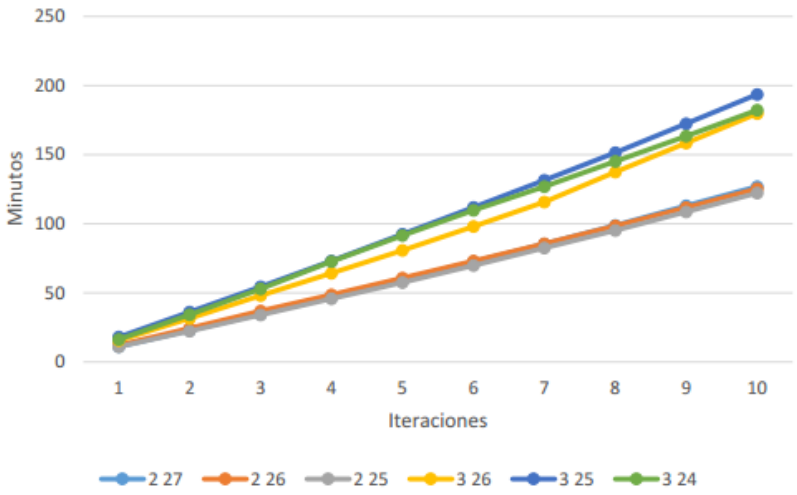

Ilustración 57- Tiempo de ejecución acumulado de las pruebas con el usuario 1

- ***Usuario 2:***

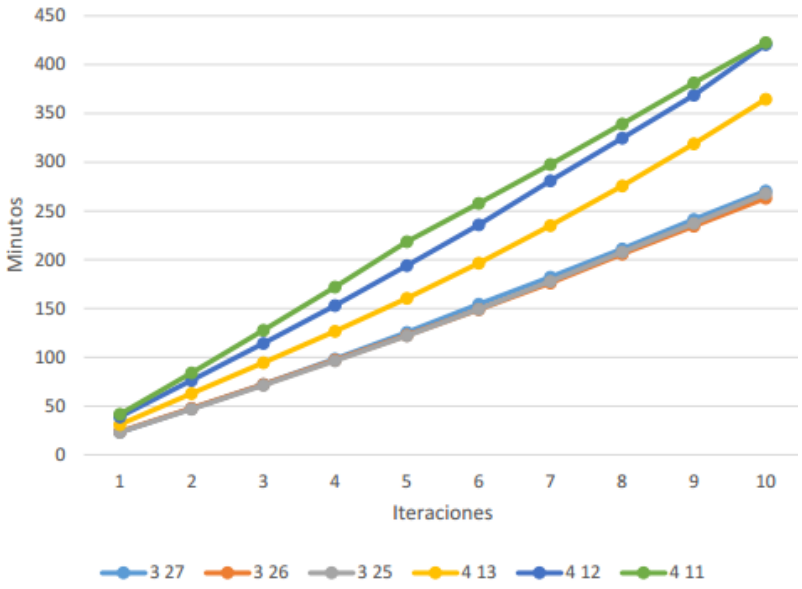

Ilustración 58- Tiempo de ejecución acumulado de las pruebas con el usuario 2

- ***Usuario 3:***

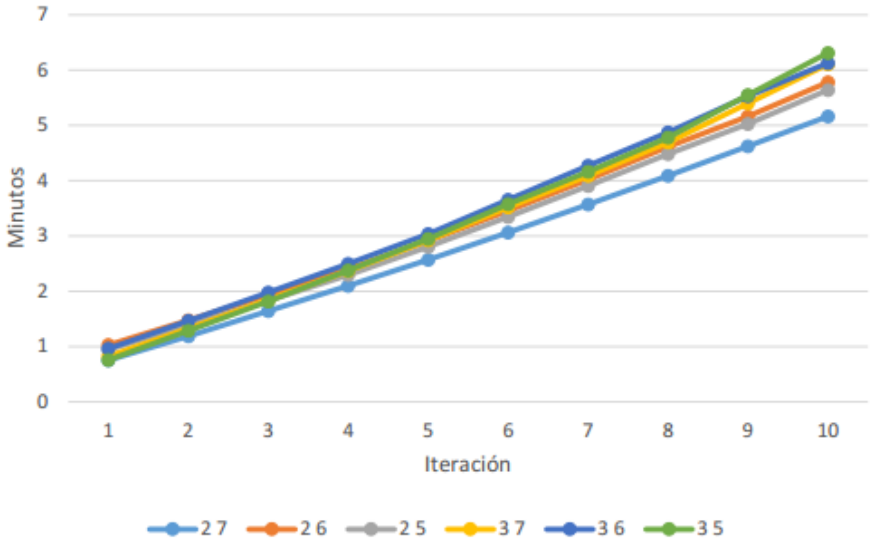

Ilustración 59 - Tiempo de ejecución acumulado de las pruebas con el usuario 3

Si prestamos atención, cuantas más aplicaciones haya en un dispositivo móvil, mucho más tarda en ejecutarse. Es más, podríamos decir que el al crecer el número de aplicaciones instaladas en un terminal, crece exponencialmente el tiempo que se tarda en realizar el análisis de similitud, y por tanto, el tiempo que se tarda en obtener el *clustering* de las aplicaciones.

7. Gestión del Proyecto

Durante el desarrollo del presente apartado se pretende exponer lo que se conoce como diagrama de *Gantt*, que tiene como finalidad, exponer el tiempo de dedicación previsto para las diferentes tareas realizadas en este proyecto. Además, se estudiarán los costes resultantes del trabajo realizado en el presente trabajo.

7.1 Planificación del proyecto

La duración del proyecto se establece en, aproximadamente, unos **seis meses** de trabajo. Así pues, el comienzo del mismo se establece el día **28 de Marzo del año 2016**, mientras que el final del trabajo se estima que será el mismo día de la entrega del TFG, este año establecido el día **26 de Septiembre del año 2016**.

Realmente, el periodo de trabajo efectivo son los seis meses anteriormente mencionados, aunque, durante los dos meses anteriores, hubo varias reuniones entre el alumno y la tutora para entender bien el objetivo del TFG y tomar contacto con los conceptos y las tecnologías que finalmente se han utilizado.

Una vez definido el rango del proyecto, vamos a determinar cuáles son las fases principales del mismo:

- Propuesta.
- Documentación.
- Análisis del sistema:
    - Análisis de requisitos de usuario.
    - Arquitectura del sistema.
    - Casos de uso.
    - Análisis de requisitos de software.
    - Análisis de las tecnologías.
- Diseño del sistema:
    - Diseño de la arquitectura del sistema.
    - Definición del entorno tecnológico.
    - Modelado físico de la base de datos.
    - Diseño de las interfaces de usuario.
- Instalación y configuración del entorno de trabajo.
- Implementación de la aplicación Android.

- Implementación del servidor Web.
    - Implementación interfaz.
    - Implementación de código generador de grafos.
- Implementación del módulo R.
- Unión de los dos subsistemas.
- Plan de pruebas.

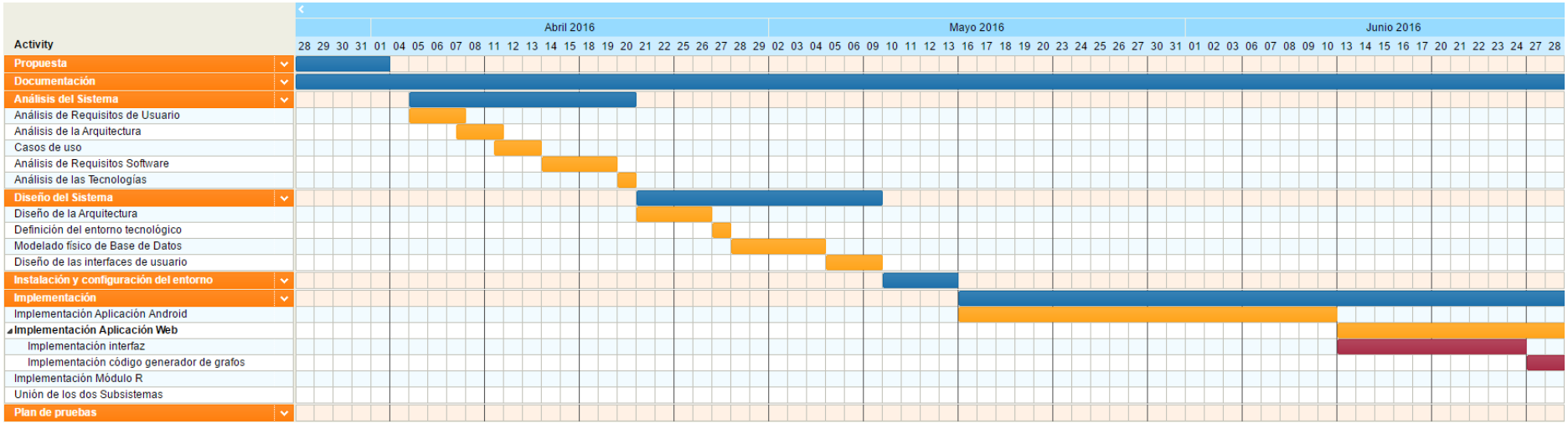

Ilustración 60 - Diagrama de Gantt (Parte A)

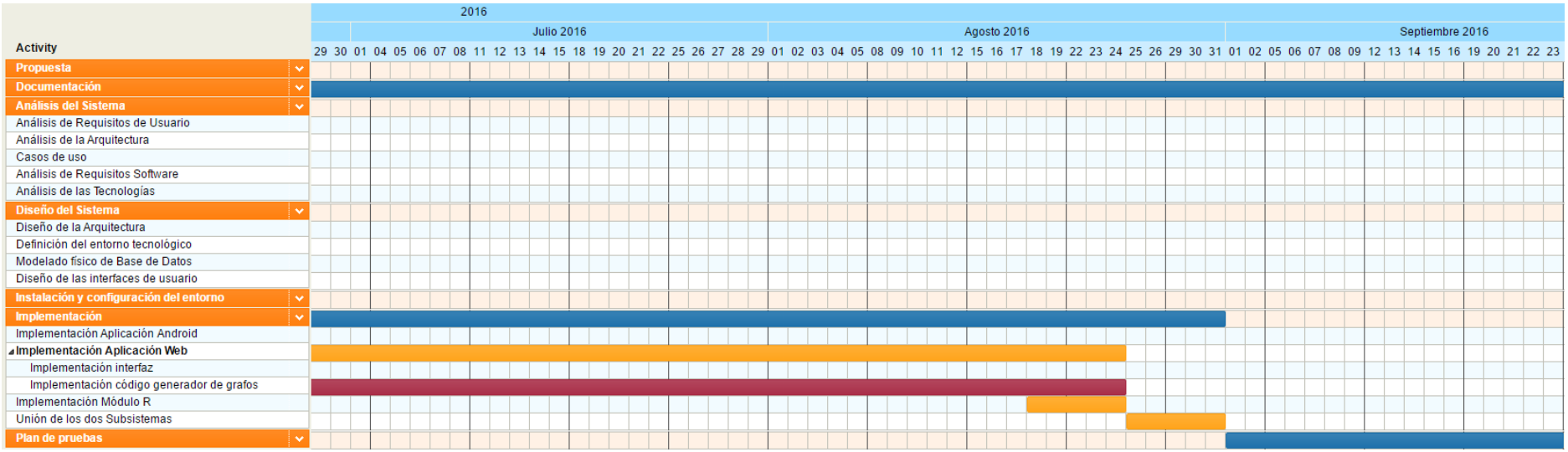

Ilustración 61 - Diagrama de Gantt (Parte B)

7.2 Presupuesto del proyecto

Tan importante es realizar una planificación del proyecto, como calcular cuales son los costes del mismo. Durante el desarrollo del presente apartado se detallarán los costes asociados al proyecto, ya se traten de costes materiales, como de costes personales, teniendo siempre en cuenta la planificación realizada anteriormente.

7.2.1 Costes materiales

Los costes materiales engloban todos aquellos costes relacionados con los recursos software, hardware y fungibles. Entre estos materiales se encuentran todos aquellos equipos necesarios para poder llevar a cabo el correcto despliegue del sistema *SimilDroid*.

Tipo de material	*Nombre del material*	*Coste sin IVA (euros)*	*Duración licencia (meses)*	*Vida útil estimada (meses)*	*Coste imputable al proyecto (euros)*
SOFTWARE	Ubuntu 14.04 LTS	0	Ilimitada	N/A	0
	Servidor Apache HTTPD	0	Ilimitada	N/A	0
	PHP5	0	Ilimitada	N/A	0
	R programming	0	Ilimitada	N/A	0
	VMware Workstation	0	Ilimitada	N/A	0
	JDK 8	0	Ilimitada	N/A	0
	Android Studio	0	Ilimitada	N/A	0
	MySQL	0	Ilimitada	N/A	0
HARDWARE	Intel Core i7-4700MQ, 8Gb de RAM, 1Tb Disco	800.12	N/A	48	100.02
	Monitor de pantalla Samsung	115.30	N/A	36	19.21
	Xiaomi Redmi Note II	140.68	N/A	24	35.17
	Samsung Galaxy S3	80.23	N/A	24	20.05
	Samsung Galaxy Tab S	251.21	N/A	24	62.80
FUNGIBLES	Materiales fungibles	50	N/A	6	50
COSTE TOTAL DE MATERIALES					287.24 €

Tabla 89 - Presupuesto en materiales

Es importante tener en cuenta que los costes materiales calculados tienen en cuenta la amortización, sabiendo que el proyecto dura aproximadamente 6 meses. Además, se ha

tenido en cuenta un coste de 50 € en concepto de materiales fungibles (bolígrafos, folios, tinta de impresora, *post-it*, etc.).

7.2.2 Costes personales

Los costes personales albergan aquellos gastos en personal. En este proyecto se han desarrollado varios roles, que, aunque hayan sido desempeñados por la misma persona, tienen un coste €/hora distinto. Es importante tener en cuenta que en el coste / hora está incluido el coste de seguridad social. Por otra parte, se tiene en cuenta también que cada día lectivo es igual a 8h de trabajo.

Rol del personal	*Coste (euros/hora)*	*Número de horas estimadas*	*Coste total (euros)*
Analista	23	136	3128
Diseñador	19	104	1976
Programador	11.5	792	9108
TOTAL COSTES PERSONALES			14212

Tabla 90 - Presupuesto en personal

7.2.3 Presupuesto final

Teniendo en cuenta los costes de materiales, junto a los costes personales del proyecto, y, además, teniendo en cuenta que se va a tener en cuenta un beneficio económico de un 20% y un margen de riesgo de un 15% del coste total sin I.V.A.

Costes personales	14212 €
Costes materiales	287.24 €
Costes Directos	14499.24 €
Costes Indirectos (5% Costes Directos)	724.962 €
Precio total sin I.V.A	15224.20 €
Coste del I.V.A	3197.08 €
PRECIO FINAL DEL PROYECTO	**18421 €**

Tabla 91 - Presupuesto final del proyecto

El proyecto tiene un coste total de **DIECIOCHO MIL CUATROCIENTOS VENTIÚN** euros**, IVA incluido**.

8. Conclusiones y líneas futuras

Durante este último apartado, se pretende expresar las conclusiones obtenidas tras el diseño y desarrollo del sistema *SimilDroid*, además de mostrar las posibles líneas futuras para mejorar el sistema, ya sea en funcionalidad, seguridad o en cualquier otro aspecto.

8.1 Conclusiones personales y del sistema

Antes de entrar en detalle sobre las conclusiones finales con respecto a la funcionalidad y el alcance del sistema desarrollado, me he dado la libertad de realizar este apartado donde quiero dar a conocer mis conclusiones personales.

En primer lugar, me siento orgulloso de haber podido terminar un proyecto que hace unos cuantos meses veía muy complicado. Este trabajo me ha servido para aprender muchísimo sobre una gran cantidad de aspectos.

Hace más o menos seis meses no sabía ni cómo era posible desarrollar una aplicación Android. Tampoco cómo poder hacer que ésta aplicación llegase a comunicarse con un servidor. Cuando tuve que comenzar a desarrollar la arquitectura que se define en el apartado ***3.1***, pensé que no sería capaz, o al menos, que no llegaría hasta donde he llegado. Después de seis meses de trabajo, estoy más que satisfecho.

Después de varias reuniones con la tutora del Trabajo de Fin de Grado, puedo afirmar que realmente este proyecto cumple con las expectativas, y por tanto, con los requisitos acordados.

El sistema es capaz de captar la red WBAN de un usuario y analizarla, generando los grafos necesarios para poder poner en marcha el algoritmo. Por otro lado, el sistema permite a los usuarios evaluar la salida del algoritmo.

8.2 Conclusiones del cliente

Después de realizar diferentes pruebas con el sistema, y tras varias reuniones, tanto el autor del Trabajo de Fin de Grado, como el cliente, están de acuerdo en que el sistema desarrollado ha resultado muy útil, pues ha permitido desarrollar pruebas con usuarios reales, con diferentes *WBANs* y obtener la salida del algoritmo. La implementación del algoritmo en R no permitía realizar estas pruebas con facilidad, pues las entradas se construían manualmente (incluyendo los cálculos de las similitudes).

Los resultados de dichas pruebas han permitido detectar que el algoritmo no funciona como se esperaba. El cliente aplica un algoritmo de ajuste de pesos descrito en [8], donde se expone que tras un número de iteraciones pequeño, la distancia intracluster converge, provocando que los pesos queden estables y por tanto el clustering resultante también. Esto no ocurre así en todos los casos de estudio analizados…

Para demostrar que lo que se comenta en el párrafo anterior es cierto, se muestran las siguientes gráficas:

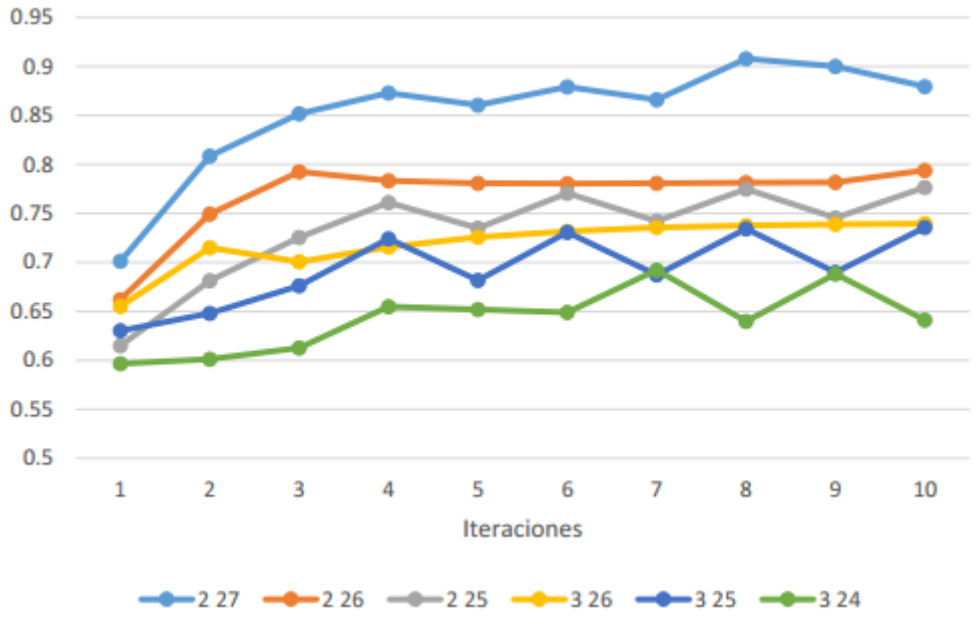

Ilustración 62 – Distancia intracluster del usuario 1.

Esta gráfica muestra la distancia intracluster obtenida de la prueba *6.6.1 Prueba con Usuario 1*. La distancia intracluster es realmente, la diferencia de similitud que existe entre los elementos pertenecientes a un mismo cluster. Para este usuario podemos ver que la distancia es, al menos, de un 0'6. Este valor depende de los datos que contienen los clusters. Lo verdaderamente importante de estas gráficas es ver que cada una de las

ejecuciones converge. En esta gráfica, la única línea que converge es la amarilla, correspondiente a la ejecución con Path = 3 y K = 26.

Durante la prueba llevada a cabo para el usuario 2, *6.6.2 Prueba con Usuario 2*, se obtiene la siguiente distancia intracluster:

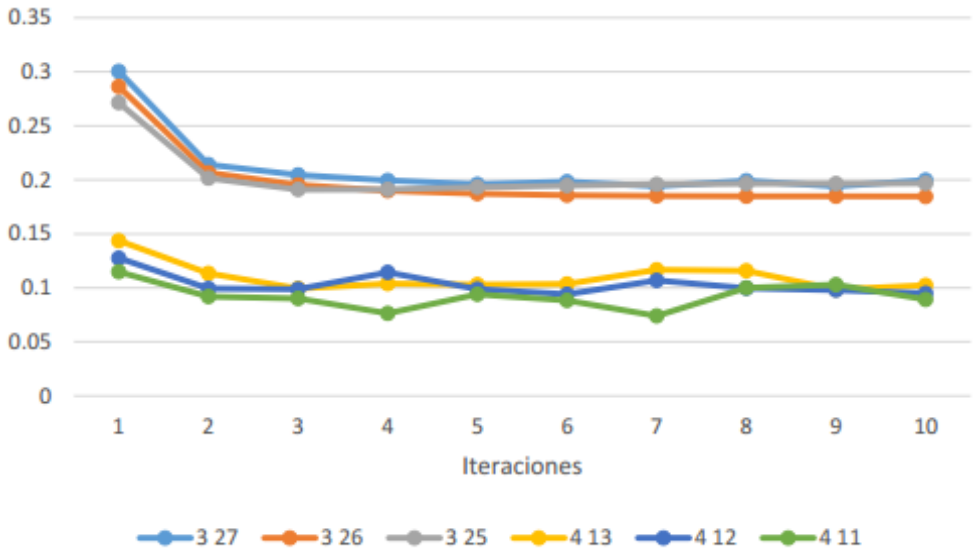

Ilustración 63 – Distancia intracluster del usuario 2.

En esta gráfica podemos ver que la distancia intraclusters varía con menos ondulación, incluso la ejecución representada con la línea naranja, parece que consigue converger.

Para el último usuario, *6.6.3* Prueba con Usuario 3*:*

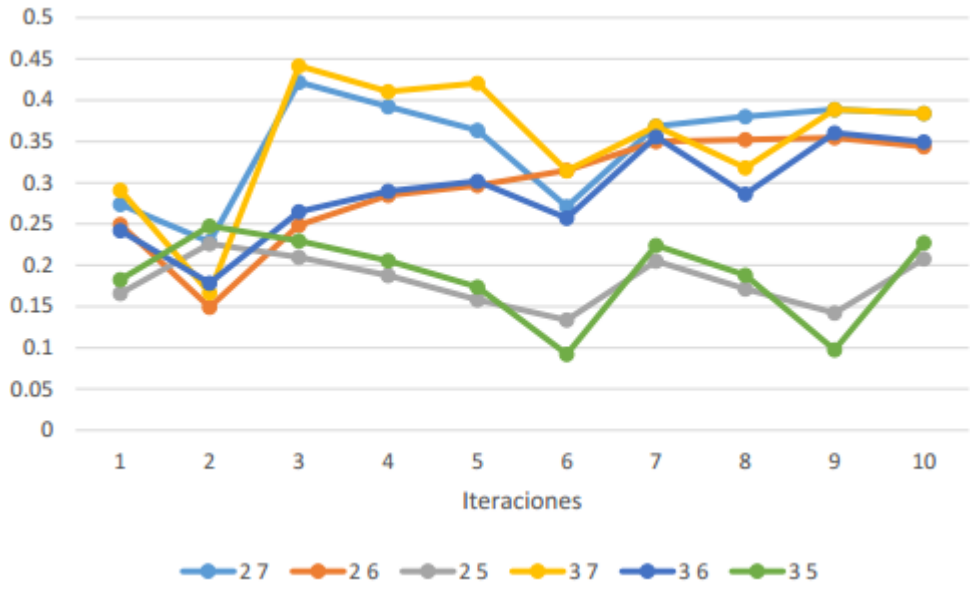

Ilustración 64 - Distancia intracluster del usuario 3.

En esta gráfica podemos ver que ninguna de las ejecuciones converge, de hecho es la gráfica más inestable de todas.

Además, para aportar una mayor información, se va a mostrar una gráfica que da información de qué clusters existen con qué ejecuciones. Este gráfico muestra justo la salida de la décima ejecución de las gráficas que aparecen en las ilustraciones 62, 63 y 64. Esto nos dará una idea de lo estable que es el algoritmo a la hora de decidir a qué cluster pertenece cada aplicación, ya que dependiendo del método elegido, existirán unos clusters u otros.

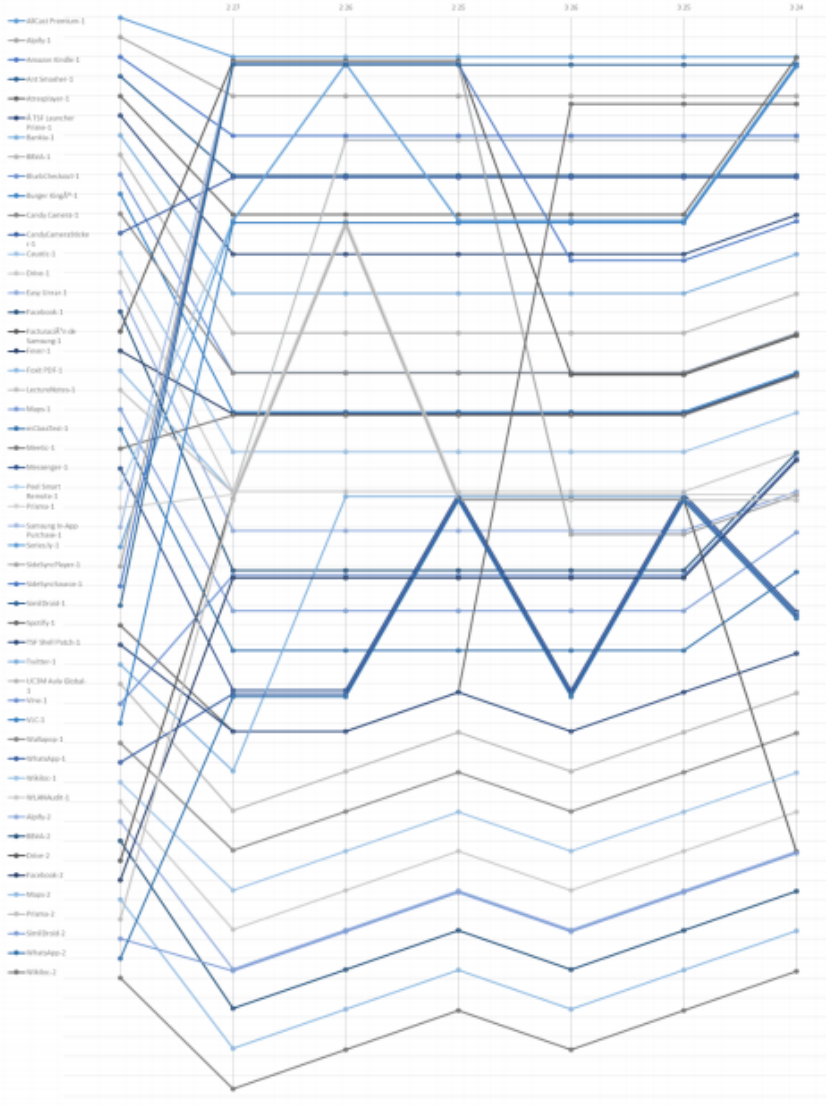

Ilustración 65 - Representación gráfica de las diferencias entre clustering del usuario 1

Podemos ver que para todas las ejecuciones, más o menos los resultados son similares.

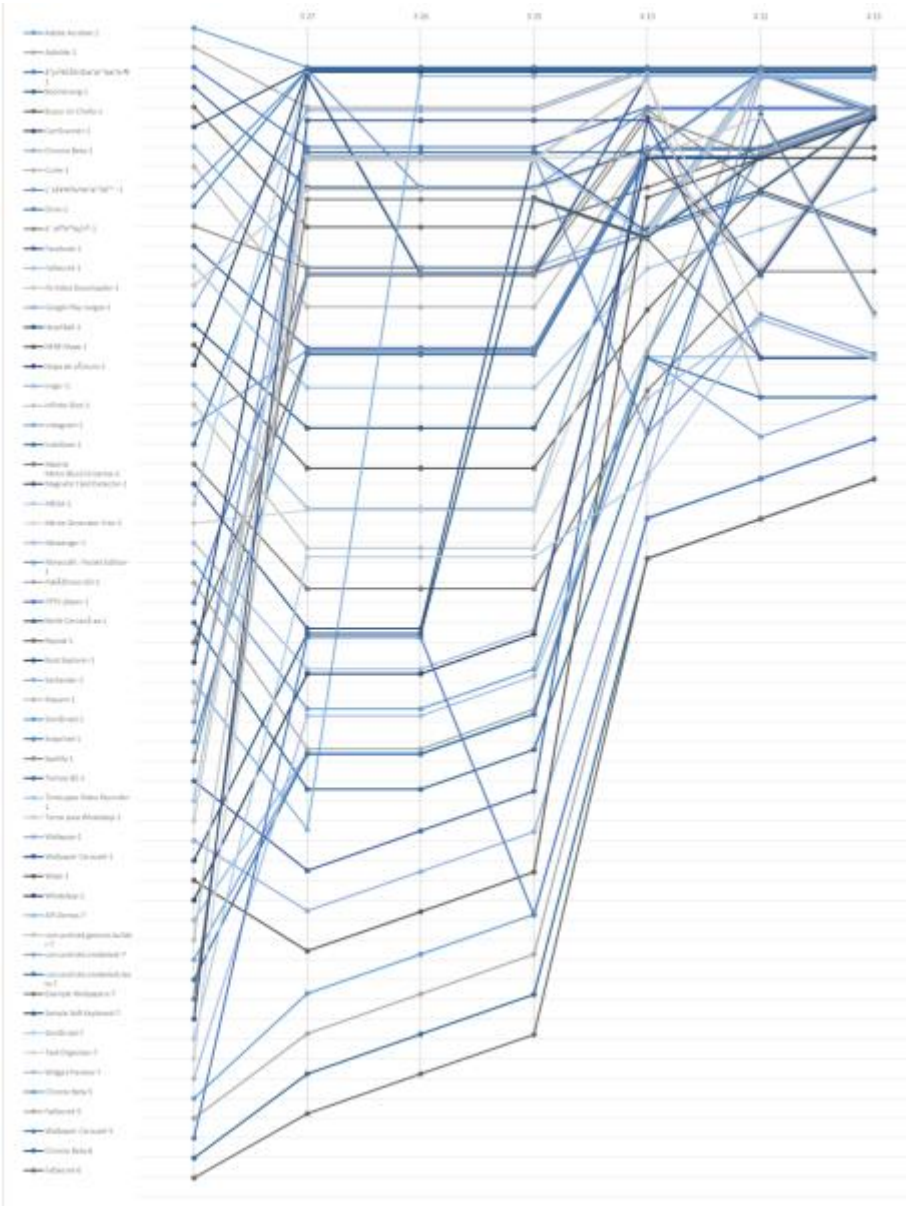

Ilustración 66 - Representación gráfica de las diferencias entre clustering del usuario 2

En la *Ilustración 66* - Representación gráfica de las diferencias entre clustering del usuario 2 podemos ver que dependiendo del tipo de ejecución, el clustering cambia radicalmente. Si nos fijamos en la últimas tres ejecuciones, los valores se ven alterados de manera drástica.

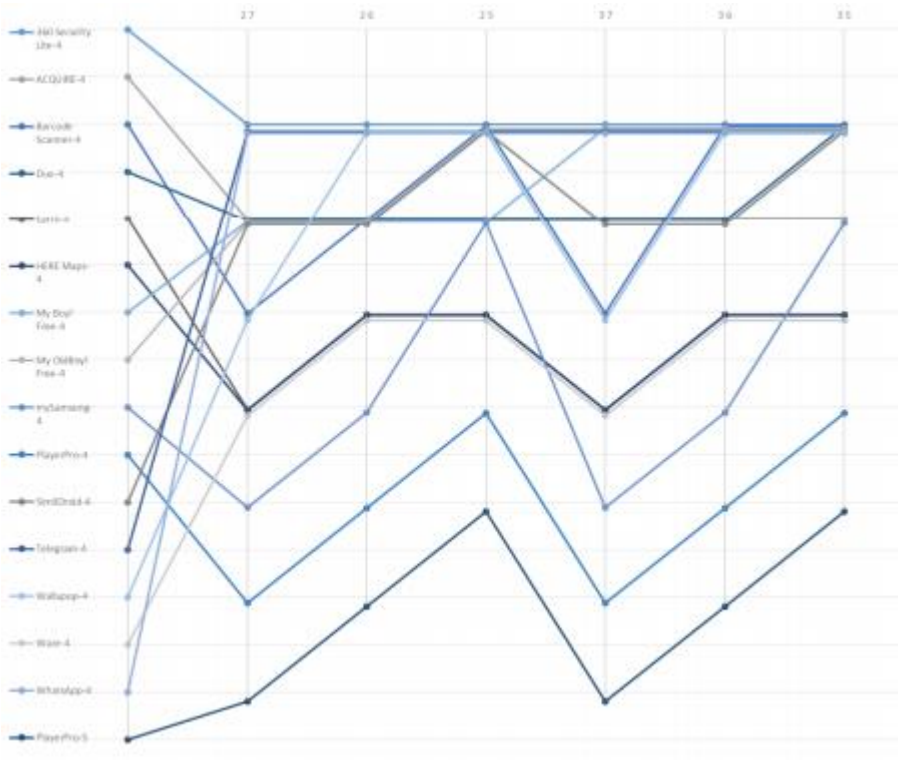

Ilustración 67 - Representación gráfica de las diferencias entre clustering del usuario 3

Por último, de la ejecución de las pruebas del tercer usuario, *6.6.3 Prueba con Usuario 3*, se ha extraído la *Ilustración 67* - Representación gráfica de las diferencias entre clustering del usuario 3. En esta ilustración podemos ver que el clustering durante todas las ejecuciones, es más o menos estable.

En conclusión, gracias a estas gráficas obtenidas de usuarios reales con dispositivos reales y atributos reales, el cliente puede determinar si el algoritmo está funcionando correctamente o no. En este caso, el algoritmo no funciona como se esperaba. Este tipo de información es de mucha utilidad para el cliente. Es lo que realmente necesitaba para poder seguir "puliendo" el algoritmo.

8.3 Líneas futuras

En los anteriores puntos de esta apartado, hemos demostrado que realmente el sistema *SimilDroid* cumple con las expectativas. Pero existen algunos elementos que harían que la herramienta fuera aún más completa:

8.3.1 Seguridad

Existen varias líneas en las que la seguridad de *SimilDroid* podría verse mejorada.

En primer lugar, la conexión que realiza el servidor Web a la base de datos, se realiza sin ningún tipo de mecanismo especial que oculte la contraseña de acceso. Por eso, una buena decisión sería encontrar alguna herramienta capaz de almacenar de alguna otra forma la contraseña, siempre y cuando no sea en claro.

Además, sería interesante realizar un estudio de los permisos Unix del servidor sobre el que corre *SimilDroid*. Durante el desarrollo de este Trabajo Final, los permisos de Unix no han sido un punto a tener en cuenta, cuando realmente es algo sobre lo que sí que hay que fijarse y tener controlado.

8.3.2 Análisis de los datos

Pensando en el análisis de los datos de salida del algoritmo, sería de muy buena ayuda para el cliente, crear una herramienta capaz de generar gráficas con las salidas del algoritmo. Estas gráficas recogerían la información necesaria, y podrían mostrarse a un usuario privilegiado. Este usuario privilegiado sería el único capaz de acceder a estas gráficas, por lo que realmente, sería necesario crear un portal privado de administración desde donde el usuario privilegiado pudiera visualizar estas gráficas.

Para la creación de gráficas podría hacerse uso de la librería *JpGraph* [14] para PHP5+, ya que proporciona una gran variedad de posibilidades de creación de gráficas.

8.3.3 Integración de nuevos módulos

Realmente este trabajo de fin de grado, podría haberse visto engrandecido de una manera abismal, si se hubiera conseguido realizar lo que el artículo [3] propone. Podría implementarse un nuevo módulo capaz de obtener la similitud que se extrae del algoritmo del cliente, y poder recomendar la política de privacidad más parecida a una nueva aplicación que entrase en el sistema. Lo único que habría que implementar, en este caso, sería la obtención de la política más adecuada, y conseguir que al instalar una nueva aplicación, el sistema *SimilDroid* se la pueda sugerir al usuario.

Bibliografía

[1] - Instituto Nacional de Estadística. (2014). *Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares*.

[2] - R. Thomas, D., R. Beresford, A., & Rice, A. (2015). *Security Metrics for the Android Ecosystem*. University of Cambridge, Computer Laboratory, Cambridge, United Kingdom.

[3] - González Tablas, A. I., & Estevez Tapiador, J. (2016). *Bootstrapping Security Policies for Wearable Apps using Attributed Structural Graphs*. Sensors (Basel, Switzerland).

[4] - Bettini, C., & Riboni, D. (8 de October de 2014). Privacy protection in pervasive systems: State of the art and technical challenges. *Elsevier*.

[5] - Chen, M., Gonzalez , S., Vasilakos, A., Cao, H., & Leung, V. (2010). Body Area Networks: A Survey. *Mobile Networks and Applications*.

[6] - Agencia Española de Protección de Datos. (1999). *Ley Orgánica 15/1999, de 13 de Diciembre, de Protección de Datos de Carácter Personal.*

[7] - Deza, M.M.; Deza, E. Encyclopedia of Distances; Springer: Berlin Heidelberg, Germany, 2009.

[8] - Cheng, H.; Zhou, Y.; Yu, J.X. Clustering large attributed graphs: A balance between structural and attribute similarities. ACM Transactions on Knowledge Discovery from Data (TKDD) 2011, 5, 12.

Fuentes de información (web)

[9] F. Lantigua, I. (04 de 04 de 2016). *Periódico El Mundo*. Obtenido de http://www.elmundo.es/sociedad/2016/04/04/5702621 9e2704e90048b465e.html el día 21 de 08 del 2016 a las 12:21

[10] Pautasio, L. (25 de 08 de 2015). TeleSemana.com. Obtenido de http://www.telesemana.com/blog/2015/08/25/estadisticas-mercado-de-smartphones-a-nivel-mundial/ el día 12 de 08 del 2016 a las 18:21

[11] Ubuntu Documentation. Help.ubuntu.com. Obtenido de https://help.ubuntu.com/lts/serverguide/httpd.html el día 4 de 09 del 2016 a las 08:55

[12] Semilar Toolkit 1.0.2 Obtenido de: http://www.semanticsimilarity.org/ el día 14 de 09 del 2016 a las 21:03

[13] WordNet 3.0 DataBase Obtenido de: http://wordnet.princeton.edu/ el día 14 de 09 del 2016 a las 21:15

[14] JpGraph 4.0.1 Obtenido de http://jpgraph.net/ el día 19 de 09 del 2016 a las 13:03

[15] 42matters Obtenido de https://42matters.com/ el día 1 de 08 del 2016 a las 12:02

Anexo I: Summary

I. Introduction

In the last years, mobile devices have turned into an indispensable element for the majority of the population. So much it is like that, that according to the National Statistics Institute, in the 96.7 % of the Spanish homes, at least a mobile phone exists [1], many of which are included in what we know as Smartphones. In 2016, 88.3 % of the Internet users in Spain, accedes to the net using an intelligent device, or a Smartphone [9].

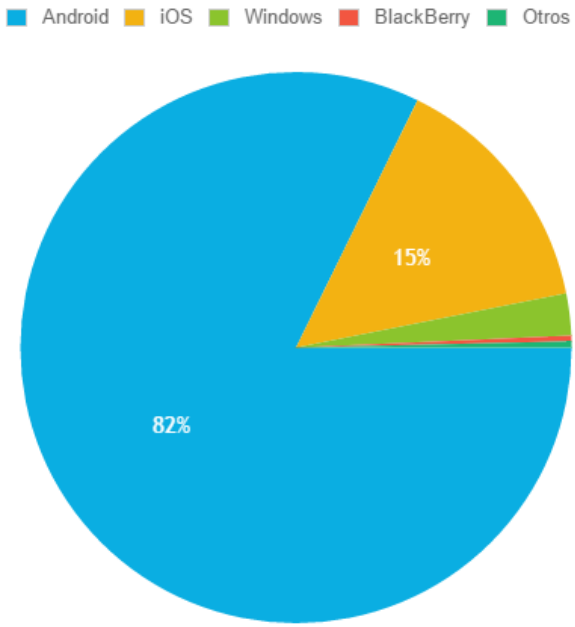

Ilustración 68 - Market Share by Operating System in 2015 [10]

Moreover, we consider that in the year 2015, more than 80 % of the commercialized Smartphones operate with an Android system [10], it is easy to realize that Android is one of the most used systems software day after day.

There is a serious problem with the Android operating system. According to a study done by the University of Cambridge, more than 80 % of the terminals with Android operating system, are considered " insecure terminals " [2].

With the qualifying “insecure", we refer that they are devices with a great probability of containing some type of malware (malignant software), like Trojans, spyware, or simply what we know as a common virus. The motive is very varied, from a lack of update of the software of the device, to a bad administration of the security policies of the terminal.

For all these reasons, the security and the privacy in Android is a problem that is necessary to tackle somehow. This field of research is, nowadays, very active.

There are multitude of studies that try to improve the security of the mobile terminals, besides assuring the privacy in them. It is important to bear in mind that, as the number of vulnerabilities (and therefore, of assaults) increases, it also exists an increase of the number of security solutions offered by researchers [2].

In addition, the applications tied to the mobile computation, are being increased in a drastic way in the last times, which carries, in addition, to an increase of the information given to service providers, many times without the knowledge of the user. [4]

Furthermore, the technological advances of last years in wireless technologies of communication, like the biosensors, together with the increasing development of technologies based on embedded systems " to wear " (what we know nowadays as Wearable device), is allowing the design, development and implementation of networks of Corporal area (Known as WBAN Wireless Body Area Network). [5]

A corporal area net (or like we will call it from now, WBAN), is not any more than a wireless communication net among electronic devices of low power.

These devices, as the electronic watches, the Smartphones, or similar devices, work together as to facilitate the life to the user, in occasions, offering a way to monitor important elements, as for example the health.

I.I Motivation

The article published by the tutor of the present end of degree Project belongs to the SPINY project, which belong to a group of researcher of the University Carlos III (Madrid), concretely to COSEC's laboratory, belonging to the department of Sciences of the Computation. This project uses the term “Internet of You " or IoY, to refer to those

nets that, monitored by intelligent telephones, are able to obtain information of the user. Some examples that the project provides are:

> *T-shirts that provide real time information for the user; intelligent pillboxes that they remember to a patient when it is the moment to take the medication and registering when THE PATIENT does it; and a new generation of medical intelligent devices with aptitude to be implanted, as pacemaker, insulin pumps and neurostimulators.*

The networks of this type have bigger problems of security and of privacy than in the traditional computation, due to this type of nets contains devices with multitude of sensors which can produce drains of sensitive information. One of the aims of the SPINY project is to approach, somehow, these problems related to the security and privacy of these networks.

Therefore, in the article of the tutor of this project, we pretend to undertake one of many problems of security that exist in type IoY nets, like WBAN nets; " We tackle the problem of the security policies and privacy for applications of new implantation introduction in wireless networks of corporal area (WBAN) " [3].

In this article, we want to tackle the problem previously commented through modelling grafos generated on the basis of the structure belonging to the WBAN net, besides certain attributes belonging to the elements that the same net contains. These attributes will be separated in two groups. On the one hand, we will have the subjective attributes, which will be defined by the user, and on the other hand, we will have objective attributes, which are inherent in its WBAN net.

The article proposes to demonstrate in a theoretical way, that, once studied all the components of the WBAN, in case of introducing a new application to the above mentioned net in any of its respective terminals / devices, those related applications can be found, so that the policies of these could be reused or adapted, for being suggested for the configuration of the new application.

In order to be able to demonstrate that really this can be carried out, it is necessary to realize a real study with applications, terminals, situations and especially, with real users.

For it, the author of the article (who we will be call a client from now on), has developed an algorithm in R programming language, manages to separate the applications of a WBAN net, in different groups, according to certain values of similarity.

I.III Aims

The basic aim of this project is to create a system capable of providing to external users a way of being able to represent their own WBAN, so that it will be taken to study to determine the similarity between the applications.

Ilustración 37 is a schematic overview to understand how SimilDroid.

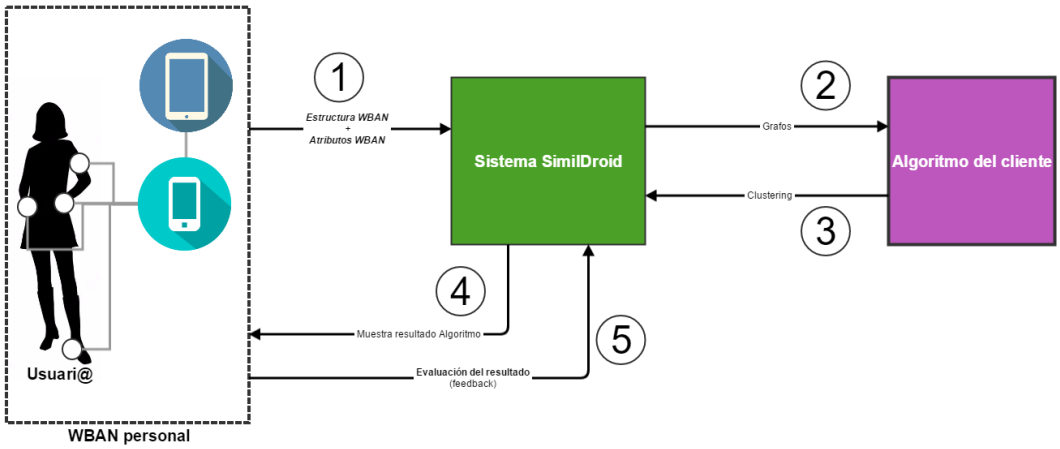

Ilustración 69 - Basic operating system Scheme

Firstly, the idea is to extract the structure and some attributes belonging to this. The SimilDroid system is the manager of gathering this information, and to transform them in order to be studied by the algorithm of the client. When the algorithm ends, it returns the results that will be showed to the user, who will be the person in charge of performing the evaluation of the algorithm (evaluating the exit of the same one).

Once obtained the result, in other words, the groups of applications, the user can evaluate, in a totally subjective way, if the result obtained by the system SimilDroid (which is responsable of executing the algorithm of the client), is correct or not.

The evaluation that the user does on the obtained result, will make the client to determine if its algorithm is effective, or if it needs some type of adjustment to be able to obtain a result that satisfies the user in a future.

II. Development

After studying the possible architectures of the system, it has been decided to use the following one:

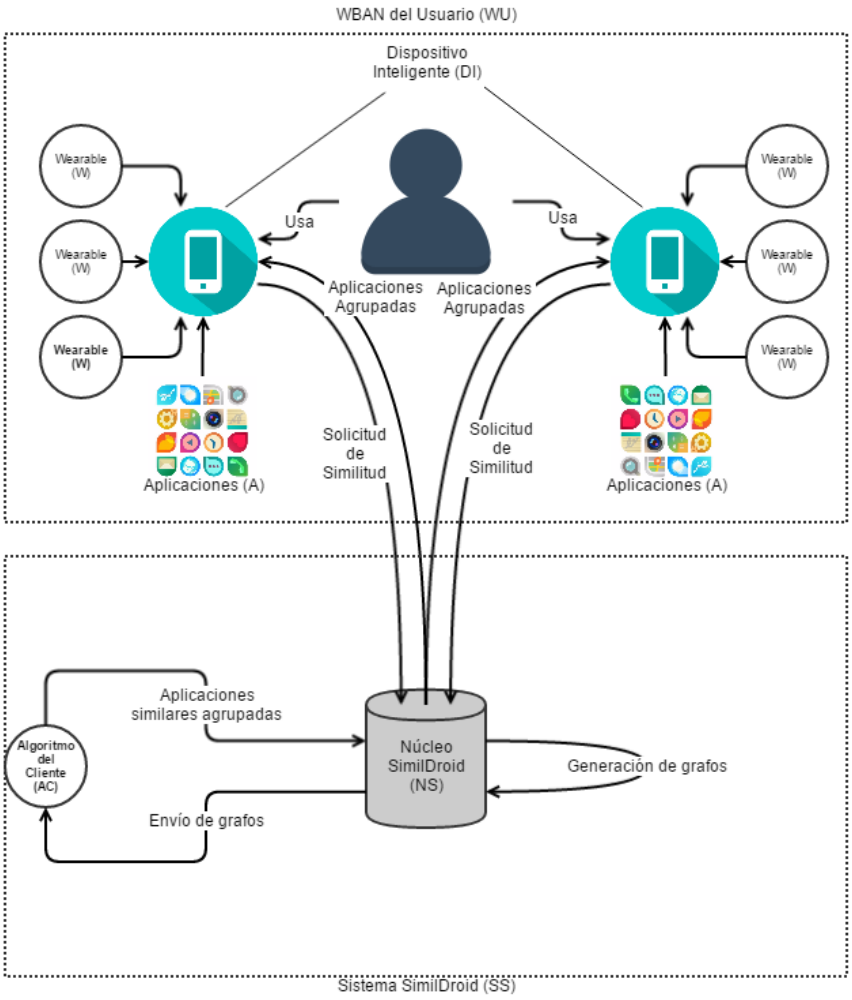

Ilustración 70 - Schema Architecture "Android + Web Server PHP"

The Android application generates a list of applications that will be sent to the Web Server. The mentioned server will be the manager of storing all the relevant information in its database.

The user will have to register all the devices of its WBAN network in the server one by one, using the Android application. Once registered all the terminals (Wearable included), the managements will be carried out across the Web Server.

Partly, this architecture makes the work more inconvinient, due to the user will have to use, on the one hand, the *SimilDroid.apk* application, and on the other one, the Web Server. However, this problem is compensated by the advantages that it provides. Since this is made in a comfortable environment for the programmer of the project, the development will be much quicker and satisfactory.

Therefore, the SimilDroid system will be composed by the following basic components:

- **WBAN of the User (WU):** it refers to the set of devices associated to the same user.
- **Smart Device (SD):** smart device skilled to support the Android operating system, besides being able to install applications on it. Normally, this device will match with a Smartphone, Tablet or similar.
- **Wearable (W):** an electronic device that "can be worn". This device will be connected to one or more applications of some of the Smart Devices of the WBAN of the user.
- **Applications (A):** the installed applications in each Smart Device.
- **SimilDroid's Kernel (SK):** it refers to the set of elements that will make the pertinent functions to obtain the graphs requested by the customer's algorithm.
- **Customer algorithm (CA):** it is the algorithm that the customer provides for its later evaluation.

The following technologies has been selected to carry out this project:

Web Server	Apache2 httpd
Manager Database System	MySQL
Clients	Android, Web Browser
Programm Lenguage	PHP, Android, Java, R
Libraries	Customer algorithm, Semilar, WordNet.

III. Results

Several tests have been performed to verify that the system works properly:

User 1	
Name	Marian
Number of device	2
Number of apps	50
Number of clusters	24

Tabla 92 - User Data 1 Test

Results User 1		
Cluster	**Output**	**Evaluation**
1	AllCast Premium-1 Atresplayer-1 Peel Smart Remote-1 Series.ly-1 **SimilDroid-1** VLC-1	AllCast Premium-1 Atresplayer-1 Peel Smart Remote-1 Series.ly-1 VLC-1
2	**Alpify-1** Spotify-1	Spotify-1
3	Amazon Kindle-1 Foxit PDF-1 LectureNotes-1	Amazon Kindle-1 Foxit PDF-1 LectureNotes-1 **Drive-1** **Drive-2**
4	**Ant Smasher-1** CandyCameraSticker-1	CandyCameraSticker-1 **Prisma-1** **Prisma-2** **Candy Camera-1**
5	TSF Launcher Prime-1 SideSyncSource-1	TSF Launcher Prime-1 SideSyncSource-1 **TSF Shell Patch-1**
6	Bankia-1	Bankia-1 **BBVA-1** **BBVA-2**
7	**BBVA-1**	-
8	BlurbCheckout-1 **Candy Camera-1** Facturación de Samsung-1	BlurbCheckout-1 Facturación de Samsung-1
9	Burger KingÂ®-1 Fever-1 Meetic-1	Burger KingÂ®-1 Fever-1 Meetic-1
10	Caustic-1	Caustic-1
11	**Drive-1** Facebook-1 Vine-1 Facebook-2	Facebook-1 Vine-1 Facebook-2
12	Easy Unrar-1 **Prisma-1** Samsung In-App Purchase-1 SideSyncPlayer-1	Easy Unrar-1 Samsung In-App Purchase-1 SideSyncPlayer-1

	Prisma-2	
13	Maps-1	Maps-1 **Maps-2**
14	mClassTest-1	mClassTest-1 **UC3M Aula Global-1**
15	Messenger-1 Twitter-1 WhatsApp-1 WhatsApp-2	Messenger-1 Twitter-1 WhatsApp-1 WhatsApp-2
16	**TSF Shell Patch-1**	-
17	**UC3M Aula Global-1**	-
18	Wallapop-1	Wallapop-1
19	Wikiloc-1	Wikiloc-1 **Wikiloc-2**
20	WLANAudit-1	WLANAudit-1
21	Alpify-2 **Drive-2** SimilDroid-2	Alpify-2 **Alpify-1** SimilDroid-2 **SimilDroid-1**
22	**BBVA-2**	
23	**Maps-2**	-
24	**Wikiloc-2**	

Tabla 93 - User 1 Test Results

NOTE: the values of the output column are colored in **red**. They indicate that this element doesn't belong to this group. Whereas the values of the column of Evaluation colored in **green**, indicate that they are new values of this group.

On the other hand, the names of the applications are accompanied from a number. This number is the identifier of the terminal to which they belong. The above mentioned number should not influence anything for the evaluation of the user.

If we slightly study the evaluation of the user, we can see that there are several groups well organized (for example the groups 9, 11, 12 and 15).

User 2	
Name	Nacho
Number of device	4 (3 wearable)
Number of apps	59
Number of clusters	13

Tabla 94 - User Data 2 Test

Results of user 2		
Cluster	**Output**	**Evaluation**
1	Adobe Acrobat-1 **Aptoide-1** CamScanner-1 Drive-1 **Google Play Juegos-1** Hojas de cÃ¡lculo-1 MEGA-1 Root Explorer-1 **Santander-1** **com.android.gesture.builder-7** **Task Organizer-7**	Adobe Acrobat-1 CamScanner-1 Drive-1 Hojas de cÃ¡lculo-1 MEGA-1 Root Explorer-1
2	å°□ç±³é£Žè¡Œæ'¬æ"¾æ□'ä»¶-1 ç^±å¥‡è‰°æ'¬æ"¾å™¨-1 è¯¬éŸ³è®¾ç½®-1 Spotify-1 **Tiempo BZ-1** **Sample Soft Keyboard-7**	å°□ç±³é£Žè¡Œæ'¬æ"¾æ□'ä»¶-1 ç^±å¥‡è‰°æ'¬æ"¾å™¨-1 è¯¬éŸ³è®¾ç½®-1 **Shazam-1** Spotify-1
3	Boomerang-1 Curse-1 Facebook-1 Fb Video Downloader-1 Imgur-1 Instagram-1 InstaSave-1 Messenger-1 Repost-1 Snapchat-1 Tonos para WhatsApp-1 WhatsApp-1	Boomerang-1 Curse-1 Facebook-1 Fb Video Downloader-1 Imgur-1 Instagram-1 InstaSave-1 Messenger-1 Repost-1 Snapchat-1 Tonos para WhatsApp-1 WhatsApp-1
4	Busco Un Chollo-1 Waze-1	**HERE Maps-1** Busco Un Chollo-1 Waze-1
5	**Chrome Beta-1** **Head Ball-1** **Magnetic Field Detector-1** Meme Generator Free-1 PPTV player-1 **Shazam-1** TimeLapse Video Recorder-1 Wallpaper Carousel-1 Example Wallpapers-7	Meme Generator Free-1 PPTV player-1 TimeLapse Video Recorder-1 Wallpaper Carousel-1 Example Wallpapers-7
6	FatSecret-1	FatSecret-1 **FatSecret-5** **FatSecret-6**
7	**HERE Maps-1**	**Aptoide-1**

		com.android.gesture.builder-7 **Task Organizer-7** **Santander-1** **API Demos-7** **Head Ball-1** **Magnetic Field Detector-1** **Tiempo BZ-1** **Sample Soft Keyboard-7** **Wallpaper Carousel-5** **Widget Preview-7**
8	Infinite Slice-1 Minecraft - Pocket Edition-1 PokÃ©mon GO-1 **API Demos-7** **com.android.smoketest-7** **com.android.smoketest.tests-7**	Infinite Slice-1 Minecraft - Pocket Edition-1 PokÃ©mon GO-1 **Google Play Juegos-1**
9	Madrid Metro|Bus|Cercanias-1 **Widget Preview-7**	Madrid Metro|Bus|Cercanias-1 **Renfe CercanÃ-as-1**
10	**Renfe CercanÃ-as-1** SimilDroid-1	SimilDroid-1 **SimilDroid-7**
11	Wallapop-1 **SimilDroid-7**	Wallapop-1
12	Chrome Beta-5 **FatSecret-5** **Wallpaper Carousel-5**	**Chrome Beta-1** Chrome Beta-5 **Chrome Beta-6**
13	**Chrome Beta-6** **FatSecret-6**	

Tabla 95 - User 2 Test Results

NOTE: the values of the output column are colored in **red**. They indicate that this element doesn't belong to this group. Whereas the values of the column of Evaluation colored in **green**, indicate that they are new values of this group.

On the other hand, the names of the applications are accompanied from a number. This number is the identifier of the terminal to which they belong. The above mentioned number should not influence anything for the evaluation of the user.

In this test, a great quantity of tidy well groups exist, though other groups (which contain applications that are difficult to categorize), are more mixed. The most remarkable group is number 3, due to it has the needed applications.

User 3	
Name	Axel
Number of device	2 (1 wearable)
Number of apps	16
Number of clusters	7

Tabla 96 - User Data 3 Test

Results of User 3		
Cluster	**Output**	**Evaluation**
1	**360 Security Lite-4** **My Boy! Free-4** **My OldBoy! Free-4** Telegram-4 WhatsApp-4	Telegram-4 WhatsApp-4 **Duo-4**
2	ACQUIRE-4 **Duo-4** SimilDroid-4	ACQUIRE-4 SimilDroid-4
3	Barcode Scanner-4 Wallapop-4	Barcode Scanner-4 Wallapop-4
4	Earth-4 HERE Maps-4 Waze-4	Earth-4 HERE Maps-4 Waze-4
5	mySamsung-4	mySamsung-4 **360 Security Lite-4**
6	PlayerPro-4	PlayerPro-4 **PlayerPro-5**
7	**PlayerPro-5**	**My Boy! Free-4** **My OldBoy! Free-4**

Tabla 97 - User 3 Test Results

NOTE: the values of the output column are colored in **red**. They indicate that this element doesn't belong to this group. Whereas the values of the column of Evaluation colored in **green**, indicate that they are new values of this group.

On the other hand, the names of the applications are accompanied from a number. This number is the identifier of the terminal to which they belong. The above mentioned number should not influence anything for the evaluation of the user.

Maybe because this a terminal with few applications, the group has turned out enough well. Actually, there are three groups very well organized (2, 3 and 4).

All these results have helped the customer to understand that there are problems related to the algorithm, since it is not able to to determine the correct groups of applications for the user.

Anexo II: Ejemplos de grafos generados

Con el objetivo de demostrar el funcionamiento del sistema, en este anexo se mostrarán algunos ejemplos de grafos generados mediante el sistema *SimilDroid*.

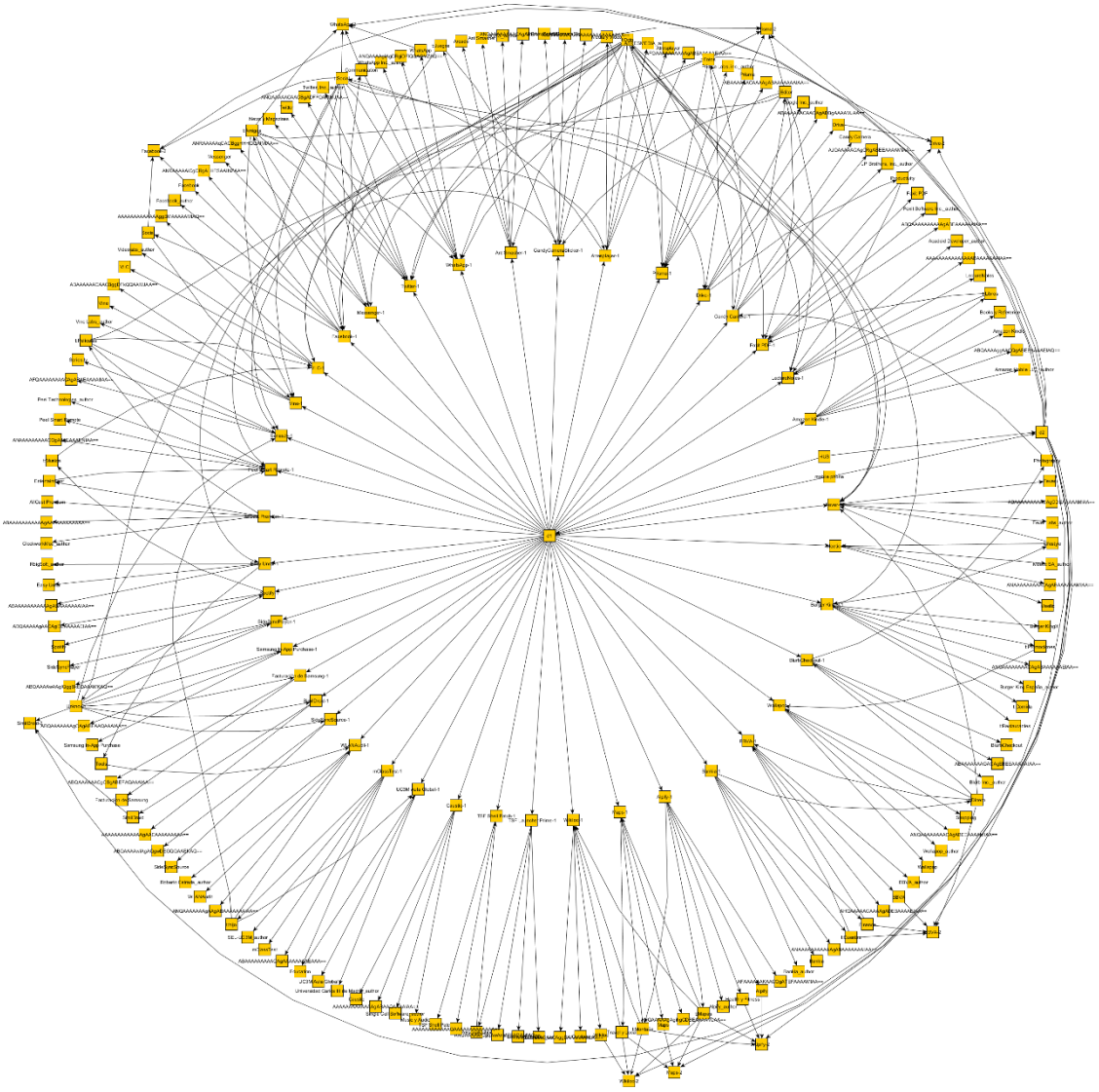

Ilustración 71 - Ejemplo de grafo Estructural con atributos

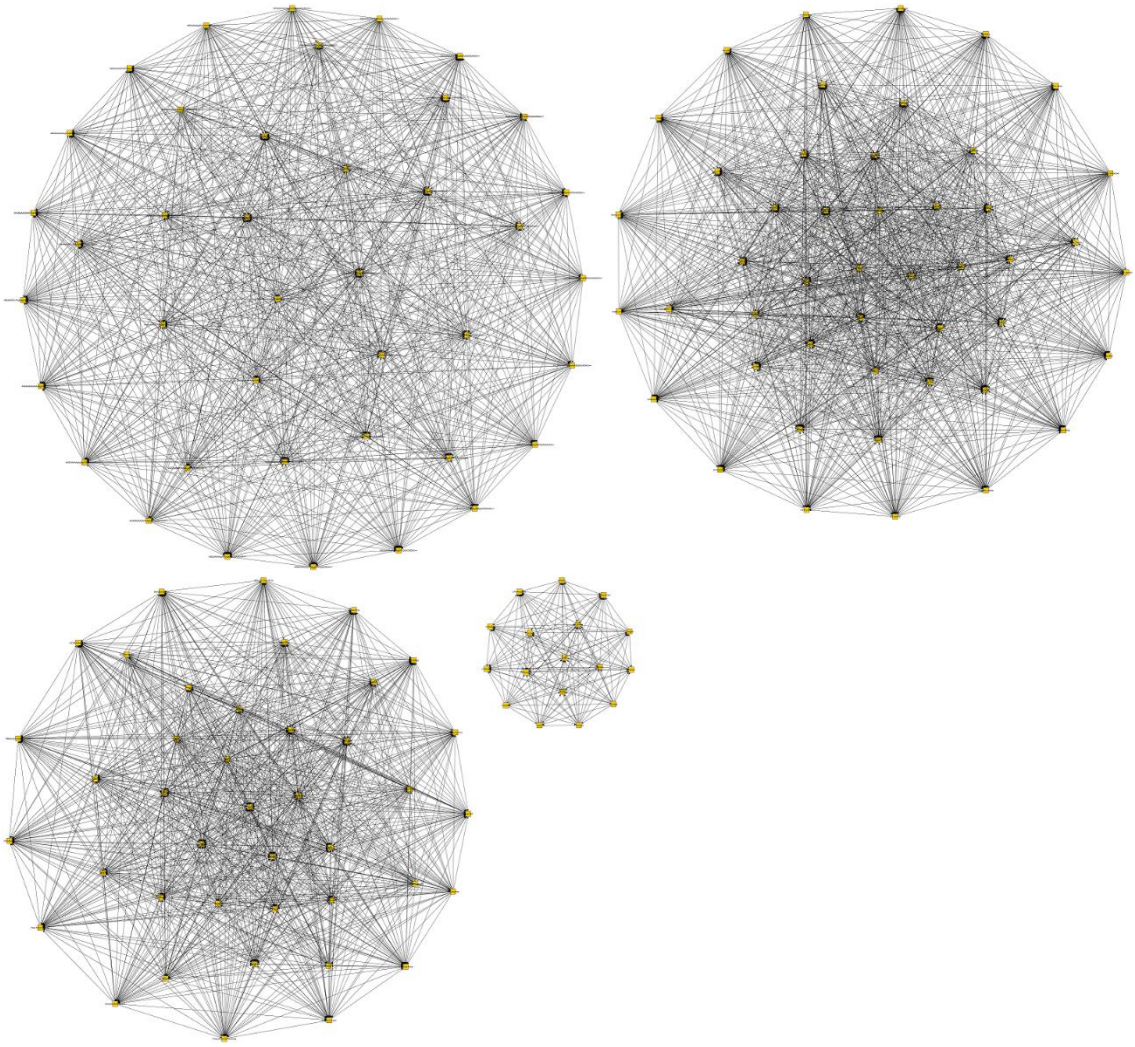

Ilustración 72 - Ejemplo de grafo de Similitudes

Se visualizan varios grafos distintos. Cada uno de ellos se corresponde con un tipo de atributo (tags, categoría, nombre y autor).

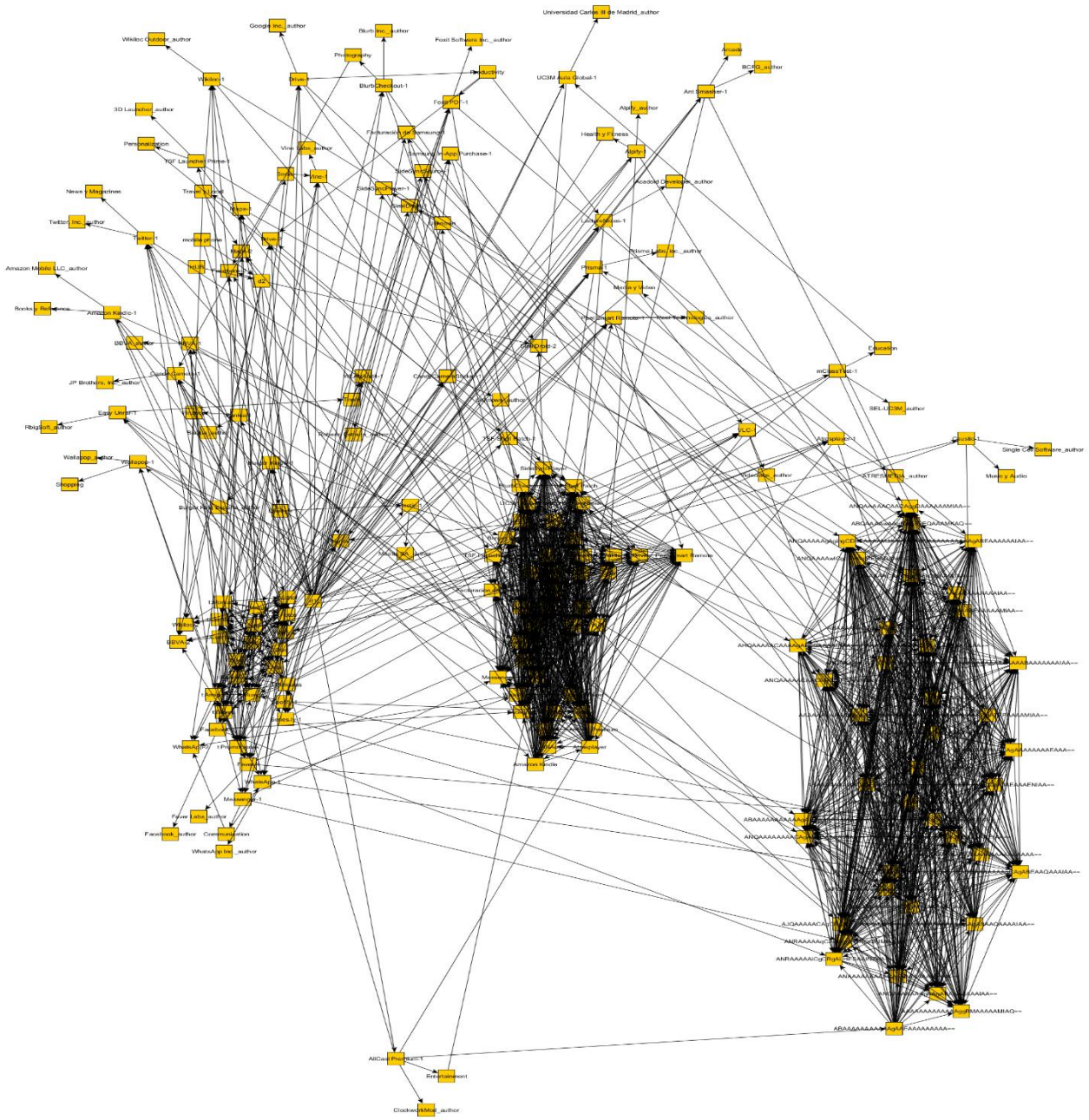

Ilustración 73 - Ejemplo de grafo de unión de los dos anteriores.