ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA
INGENIERÍA DEL SOFTWARE

ITSME

Realizado por
Francisco Javier Lima Florido
Tutorizado por
Enrique Alba Torres
Francisco Javier Ferrer Urbano
Departamento
LENGUAJES Y CIENCIAS PARA LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Noviembre de 2016

Fecha defensa:
El Secretario del Tribunal

Resumen: Uno de los principales objetivos de Internet hoy día es el de establecer un contacto entre personas que se encuentran muy lejos unas de otras. En este Trabajo de Fin de Grado (TFG) contempla otro aspecto que también es importante, establecer conexiones con personas de nuestro entorno más cercano. Para ello vamos a aprovechar el potencial de los *smartphones* y las redes Bluetooth en el desarrollo de una aplicación con la que poder descubrir a personas cercanas que compartan nuestros gustos y aficiones, con el fin de ponernos en contacto con ellas. Dicho objetivo se ha cumplido dividiendo el TFG en dos partes: una aplicación Android y una API web a la que se accede a través de un servidor. Ambos módulos han sido desarrollados independientemente, utilizando tecnología, métodos y herramientas distintas para la implementación de cada uno, e integrándolos entre sí, conformando un sistema completo. Finalmente, dicho sistema consigue el objetivo de poner en contacto a las personas aprovechando los datos que proporcionan para realizar test de compatibilidad automáticos y permitiendo mantener conversaciones consentidas entre ellas de forma directa.

Palabras clave: Bluetooth, Android, conectividad, aplicación móvil, API web, compatibilidad, chat.

Abstract: One of the most important goals of Internet nowadays is to establish a connection between people who live far away from each other. The present degree thesis looks at other side which is also important, to establish connections with people from our enviroment. For that reason we take advantage of smartphones and Bluetooth networks in the development of an aplication with which you will be able to find close people who share our hobbies and preferences in order to contact them. This goal has been achived by dividing the degree thesis in two parts: An Android app and a web API which is acceded from a server. Both modules have been developed independently, using different technology, methods and tools to implement each part, and integrating them into a complete system. Finally, this system reaches the goal of putting people in contact by taking advantage of the information that they provide to perform authomatic compatibility tests and allowing them to talk to each other directly.

Keywords: Bluetooth, Android, conectivity, mobile app, web API, compatibility, chat.

Índice

Capítulo 1. Introducción

Como introducción se van a exponer las motivaciones que impulsan el desarrollo de este TFG, así como los objetivos que se esperan cumplir una vez finalizado el desarrollo completo del mismo.

1.1 Motivación

A menudo en situaciones cotidianas nos encontramos con que tenemos que pasar tiempo en salas de espera o en otros lugares en los que estamos rodeados de gente a la que no conocemos. Hoy en día una distracción recurrente en estos casos es la de utilizar aplicaciones instaladas en nuestro *smartphone* para leer el correo electrónico, jugar a juegos o enviar mensajes a nuestros amigos o conocidos, entre otras actividades.

Sin embargo, algunas personas prefieren entablar una conversación desenfada con alguien, en vez de estar mirando su móvil o esperar sin ningún tipo de distracción. Además de eso, este hecho hace que aquellas personas que tengan algún tipo de problema de sociabilidad se excluyan más, o en caso contrario, que se sientan aún más rechazados al ver que la mayoría de las personas a su alrededor está mirando su móvil (1).

Uno de los puntos a desarrollar de las *Smart Cities* es el de mejorar la comunicación entre los ciudadanos a distintos niveles (2), pero debido a problemas como los descritos anteriormente, parece que los *smartphones* también pueden influir negativamente en ello. Para mejorar esta situación es conveniente crear herramientas que mejoren la comunicación entre las personas que no conocemos en lugares como el transporte público, salas de espera o zonas de ocio. Para ello utilizaremos las redes inalámbricas que tanto potencial tienen en los *smartphones* y que cada vez tienen más influencia en las *Smart Cities* (3).

Como consecuencia de lo anteriormente expuesto, la propuesta principal de este TFG es la creación de una aplicación para *smartphones* que aproveche el potencial de las redes inalámbricas. En dicha aplicación los usuarios podrán buscar otros usuarios y decidir si inician una conversación con alguno de ellos basándose en un porcentaje de compatibilidad que es calculado para cada par de usuarios distintos. De esta forma, podrán establecer un primer contacto sin necesidad de hablar directamente con esa persona.

1.2 Objetivos

El objetivo principal de este TFG es el desarrollo de una aplicación para móviles Android que posibilite al usuario la comunicación con personas que están a su alrededor cuyos gustos o aficiones sean similares. Para esto el usuario tiene que crear una cuenta única en la que indicar sus gustos y otros datos que le da la posibilidad de iniciar sesión en la aplicación.

Como complemento importante, para saber cuáles de esas personas que se encuentran cercanas al usuario son las más indicadas, se pretende diseñar y aplicar un algoritmo de cálculo de compatibilidades que guíe al usuario a la hora de entablar una conversación y que aporte un mayor atractivo a la aplicación.

1.3 Estructura de la memoria

En esta sección se expone brevemente la estructura de la memoria y una síntesis del contenido de cada capítulo.

Capítulo 2: Tecnologías y recursos

Aquí se expondrán las tecnologías usadas en cada fase del trabajo y el porqué de su elección. También se incluye aquí los recursos utilizados como bibliotecas, plantillas y herramientas.

Capítulo 3: Especificación y análisis

En este capítulo serán incluidos los requisitos funcionales y no funcionales de las aplicaciones, así como los casos de uso y los actores de cada aplicación.

Capítulo 4: Diseño

Se presentarán los diagramas de distribución y clases, además de los diseños de las bases de datos remota y local.

Capítulo 5: Implementación y pruebas

Aquí se expondrá todo el desarrollo del trabajo. Como se ha implementado cada aplicación, que inconvenientes se han encontrado y como se han ido cumpliendo los objetivos.

Capítulo 6: Conclusiones y líneas futuras

En este capítulo se concluye exponiendo lo aprendido durante este TFG y los desarrollos adicionales que pueden mejorar la aplicación propuesta de cara al futuro.

Capítulo 2. Tecnologías y recursos

En este capítulo se explican las tecnologías usadas en cada proceso del desarrollo del trabajo, así como una justificación del uso de las mismas en lugar de otras alternativas, aportando referencias a su documentación.

2.1 Repositorio de código fuente

Como sistema de control de versiones, por su sencillez y potencia, se ha optado por el uso de un repositorio git alojado en GitHub. GitHub es uno de los servidores de alojamiento de repositorios más utilizado y conforma una gran comunidad de desarrolladores. De esta manera se da la oportunidad de que otros desarrolladores o personas interesadas en este trabajo puedan tener acceso completo a su código fuente e incluso colaborar en desarrollos posteriores de este mismo TFG.

Como herramienta de gestión de los repositorios se ha elegido *Source Tree* de la compañía *Atlassian* (4), que proporciona una completa y sencilla interfaz gráfica con la que gestionar tus repositorios alojados en distintos servidores y de distintos sistemas de control de versiones

Enlaces de los repositorios del proyecto Android y del proyecto de la API web:

Aplicación Android: https://github.com/jaliflo/TFGProject
API web: https://github.com/jaliflo/APITFG

2.2 Aplicación Android

En el desarrollo de la aplicación Android se ha optado por la vía nativa utilizando su *framework* propio en el *Integreated Development Enviroment* (IDE) *Android Studio*. Se ha considerado esta la mejor opción dado que el rendimiento de la aplicación desarrollada de forma nativa es más alto. Además, posee una completa documentación (5) y existen numerosos recursos en internet para el apoyo al desarrollo.

Para el envío de peticiones HTTP para comunicarse con la API web, se ha utilizado la librería *Retrofit* (6). Esta librería aporta métodos muy completos para el envío de las peticiones. Además, también aporta la capacidad de serializar y deserializar los objetos JSON intercambiados con la API web.

El chat vía Bluetooth es una parte fundamental de la aplicación, por lo que se ha aprovechado la plantilla de chat Bluetooth de los repositorios de Google. A esta plantilla se le han aplicado numerosas modificaciones para adaptarla al propósito de este trabajo.

Plantilla *BluetoothChat*:
https://github.com/googlesamples/android-BluetoothChat

2.3 API Web

Para el desarrollo de la API web se ha optado por usar el *framework* de C# .NET Core desarrollado por Microsoft (7). La gran ventaja de esta tecnología es que es multiplataforma y será de código abierto, aportando además sencillez a su instalación y utilización. El IDE utilizado para desarrollar la API web es *Visual Studio* 2015, también de Microsoft.

Para el propósito de este trabajo se ha optado por alojar la aplicación en el servidor web incluido en el paquete *Internet Information Services* (IIS) (8) que viene integrado en el sistema operativo Windows.

2.4 Bases de datos

Dado que se han desarrollado dos aplicaciones, en el presente TFG se han utilizado dos gestores de bases de datos. En cada caso se ha buscado escoger el más adecuado para el propósito de la aplicación.

2.4.1 Base de datos de la aplicación Android

Para la aplicación Android se ha optado por el uso de *SQLite* (9). Es un sistema de bases de datos muy sencillo y liviano que cumple perfectamente con los requisitos para este trabajo

2.4.2 Base de datos de la *API* web

Para la API web se ha optado por el uso de *Postgresql* (10). Es un sistema de bases de datos relacionales potente que aporta muy buen rendimiento y estabilidad para bases de datos grandes, es multiplataforma y posee una gran cantidad de tipos de datos.

Capítulo 3. Especificación y Análisis

En este capítulo se enumeran y describen los requisitos funcionales y no funcionales de la aplicación, se especifican los casos de uso incluyendo un diagrama, y se explican las entidades que conforman las bases de datos.

3.1 Requisitos funcionales

En la Tabla 1 y la Tabla 2 se enumeran los requisitos funcionales de la aplicación Android y la API dándoles una identificación y aportando una descripción de cada uno. Estos establecen la funcionalidad necesaria y las acciones que se deben cumplir en función de la interacción del usuario, o interacciones entre ambas.

ID	Requisito	Descripción
MA1	Iniciar sesión	Cuando se inicie la aplicación debe dársele la opción al usuario de introducir sus credenciales para iniciar sesión y acceder a toda la funcionalidad.
MA2	CRUD del usuario	El usuario podrá, antes de iniciar sesión, crearse una cuenta que posteriormente podrá consultar y modificar cuando desee
MA2.a	Crear perfil de usuario	El usuario podrá crear un perfil la primera vez que inicia la aplicación que incluya algunos datos obligatorios, como un nombre y una contraseña, y otros datos opcionales.
MA2.b	Consultar perfil de usuario	El usuario podrá consultar su perfil cuando desee una vez haya iniciado sesión.
MA2.c	Modificar perfil de usuario	El usuario podrá solicitar la modificación de los parámetros respecto a gustos y localización de su perfil.
MA3	Búsqueda de perfiles compatibles con el usuario	El usuario podrá solicitar a la aplicación que realice una búsqueda de usuarios cercanos a él y mostrarle sus nombres de usuario y un porcentaje de compatibilidad calculado durante el proceso.
MA4	Enviar petición de chat	El usuario podrá elegir a cualquiera de los otros usuarios mostrados en la lista para enviar una petición de chat.
MA5	Recibir petición de chat	En cualquier momento se podrá recibir una petición de chat que muestre datos del usuario que la envió.
MA6	Mantener conversación por chat	Cuando se inicia un chat los usuarios podrán intercambiar mensajes entre ellos.
MA7	Consultar historial de chats	En cualquier momento el usuario podrá ver un historial de chats en el que aparezcan todas las conversaciones mantenidas.

Tabla 1: Requisitos funcionales Android

ID	Requisito	Descripción
AW1	Crear usuario	La API debe recibir y comprobar los datos de creación de un perfil y en casos de estar todos correctos, insertar dicho perfil en la base de datos.
AW2	Comprobar credenciales de usuario	La API de poder recibir las credenciales de un usuario que intenta iniciar sesión en la aplicación, comprobarlas y de volver el resultado.
AW3	Calcular compatibilidad de usuarios	La API debe poder recibir una lista de dispositivos, buscar a los usuarios asociados a dichos dispositivos en la base de datos, calcular la compatibilidad con cada uno de ellos incluyéndola en una lista, y devolver dicha lista ordenada a la aplicación.

Tabla 2: Requisitos funcionales API

Estos requisitos permitirán definir los casos de uso del sistema, tanto de las interacciones del usuario con la aplicación y otros usuarios, como de las interacciones entre la aplicación Android y la API.

3.2 Requisitos no funcionales

En la Tabla 3 se enumeran todas las características no funcionales que se deberían cumplir en la aplicación, asociándolas a un código de identificación al igual que se han expuesto los requisitos funcionales.

ID	Requisito	Descripción
NFR1	Rendimiento	Debido a la espera que existe cuando el usuario solicita la lista de usuarios cercanos, la interacción entre el servidor y la aplicación Android debe ser lo más rápida posible.
NFR2	Disponibilidad	El servidor debe estar disponible el mayor tiempo posible.
NFR3	Usabilidad	La aplicación móvil debe ser lo más intuitiva y fácil de usar posible.
NFR4	Escalabilidad	Se debe poder aumentar la funcionalidad de la aplicación con facilidad, dada la naturaleza de está, será conveniente añadir nuevas funcionalidades.
NFR5	Seguridad	Las contraseñas deben ir almacenadas como un hash.

Tabla 3: Requisitos no funcionales

Estos requisitos se han descrito para ser cumplidos en la medida de lo posible con el objetivo de aportar una experiencia satisfactoria, cómoda y segura a cualquier usuario que utilice la aplicación.

3.3 Actores y casos de uso

En esta sección se introducirá una descripción de los actores que participan en el diagrama de casos de uso y se especificarán seguidamente dichos casos de uso.

Usuario:

Representa a cualquier persona que interactúa directamente con la aplicación móvil. Los usuarios también interactuarán entre sí utilizando el chat de la aplicación.

Sistema:

Representa al servidor web que se encarga de recibir las peticiones de creación de usuario, inicio de sesión y lista de usuarios cercanos.

La Figura 1 muestra un diagrama de los casos de uso y la relación de cada uno con los actores, cubriendo toda la funcionalidad descrita en el apartado de requisitos funcionales.

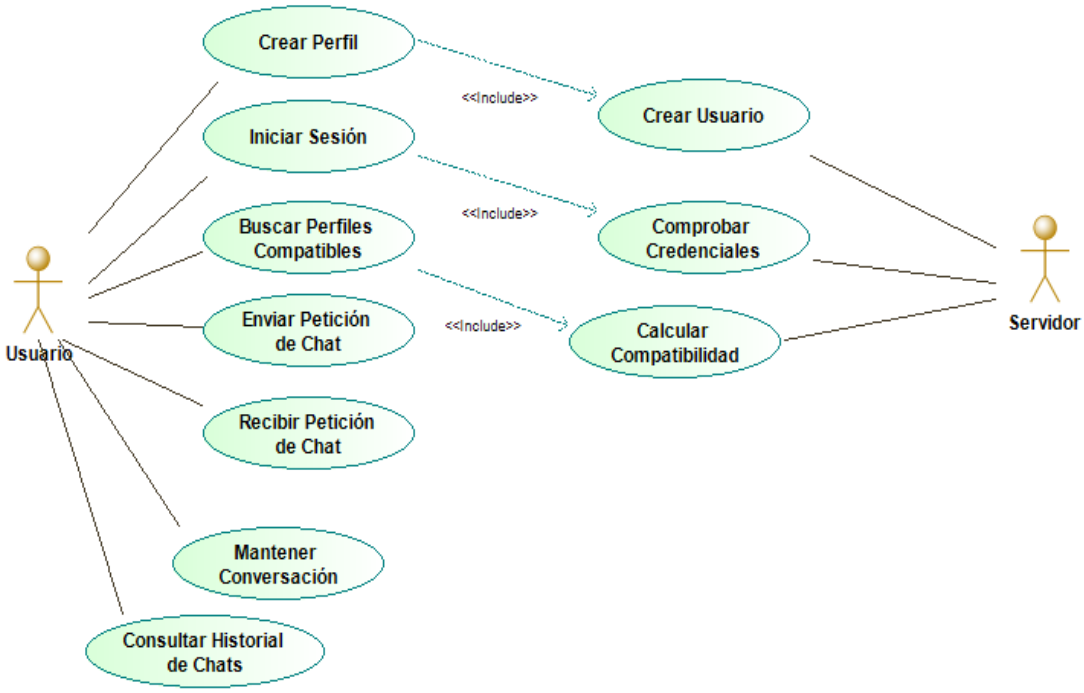

Figura 1: Diagrama de casos de uso

A continuación, en las Tablas 4, 5, 6, 7, 8, 9 y 10, se especifican los casos de uso más importantes de la aplicación Android, incluyendo su requisito asociado y los escenarios posibles de cada uno.

Iniciar sesión
Identificador: UC-01 **Requisito asociado:** MA1 **Descripción:** Cuando se inicia la aplicación, se mostrará un formulario de inicio de sesión. **Precondiciones:** 1. El usuario debe tener una cuenta creada. **Postcondición:** 1. El usuario habrá entrado a la aplicación. **Prioridad:** Alta
Escenario principal
1. La aplicación pide al usuario inicie sesión. 2. El usuario introduce usuario y contraseña y pulsará el botón de “Entrar”. 3. El usuario verá la pantalla principal de la aplicación.
Escenario alternativo-El usuario no tiene cuenta
1. La aplicación pide al usuario que inicie sesión. 2. El usuario pulsa en “Crear cuenta”. 3. La aplicación abre la ventana de “Crear perfil”.
Escenario alternativo-El usuario introduce datos incorrectos
1. La aplicación pide al usuario que inicie sesión. 2. El usuario introduce usuario y contraseña y pulsará el botón de “Aceptar” 3. La aplicación mostrará un mensaje emergente avisando de que los datos no son correctos y volverá a la pantalla de inicio de sesión.

Tabla 4: Caso de uso-Inicio de sesión

Crear perfil de usuario
Identificador: UC-02.a **Requisito asociado:** MA2 **Descripción:** Si el usuario no tiene cuenta, podrá crear una cuenta pulsando en “Crear cuenta”. Una vez creada irá almacenado de forma local, además de ser enviado al servidor. **Precondiciones:** 1. El usuario aún no tiene perfil o es la primera vez que se ejecuta la aplicación. **Postcondición:** 1. El perfil quedará completo y almacenado tanto de forma local como en el servidor. **Prioridad:** Alta
Escenario principal
1. Se le pide al usuario que rellene unos datos para crear un perfil que la aplicación usará. 2. Se abre una actividad con un formulario en el que se pedirán datos necesarios y datos opcionales. 3. Una vez todo este relleno, el usuario podrá darle a “Guardar”. 4. La aplicación almacenará el perfil en el dispositivo y lo enviará al servidor para que este también lo haga.

Escenario Alternativo-El usuario pulsa el botón de “Siguiente” cuando aún faltan datos necesarios para el perfil
1. Se le pide al usuario que rellene unos datos para crear un perfil que la aplicación usará. 2. Se abre una actividad con un formulario en el que se pedirán datos necesarios y datos opcionales. 3. El usuario pulsa “Siguiente” cuando aún faltan datos necesarios. 4. Aparecerá un mensaje dando un aviso de que aún faltan datos por rellenar. 5. Una vez el usuario completa los datos faltantes y pulse “Guardar”, la aplicación almacenará el perfil en el dispositivo y lo enviará al servidor para que este también lo almacene.

Tabla 5: Caso de uso-Crear perfil

Buscar perfiles compatibles
Identificador: UC-03 **Requisito asociado:** MA3 **Descripción:** El usuario podrá buscar otro usuario con el que entablar una conversación en el lugar en que se encuentre. La aplicación tratará de buscar un perfil que se adapte al del mismo. **Precondiciónn:** 1. El usuario debe tener un perfil creado y almacenado en el dispositivo y en el servidor **Postcondición:** 1. El usuario verá una lista de usuario cercanos a él ordenados en función de la compatibilidad. **Prioridad:** Alta
Escenario principal
1. El usuario pulsará un botón de “Buscar usuario”. 2. La aplicación abrirá una ventana mientras busca usuarios. 3. Una vez encontrados un usuario o varios se mostrarán en esa ventana ordenados en función de la compatibilidad.
Escenario alternativo-No hay usuarios cercanos
1. El usuario pulsará un botón de “Buscar usuario”. 2. La aplicación abrirá una ventana mientras busca usuarios. 3. Al no existir ningún otro usuario cercano, la ventana mostrará un mensaje informando de ello al usuario.

Tabla 6: Caso de uso-Buscar usuarios

Enviar petición de chat
Identificador: UC-04 **Requisito asociado:** MA4 **Descripción:** El usuario podrá enviar una petición de chat a cualquier usuario que haya encontrado. **Precondiciones:** 1. El usuario debe haberle dado a “Buscar usuario”. 2. La aplicación debe haber encontrado al menos un usuario cercano. **Postcondición:** 1. El usuario podrá entablar una conversación con el usuario que ha seleccionado. **Prioridad:** Alta
Escenario principal
1. El usuario pulsará sobre uno de los usuarios de la lista de encontrados. 2. La aplicación mostrará una ventana con los datos en que coinciden los dos usuarios y pedirá si quiere enviar una petición de chat. 3. El usuario pulsa sobre uno de los nombres que aparecen en la lista, la aplicación envía la petición al usuario marcado y espera una respuesta.

Tabla 7: Caso de uso-Enviar petición de chat

Recibir petición de chat
Identificador: UC-05 **Requisito asociado:** MA5 **Descripción:** El usuario podrá recibir una petición de chat enviada por otro usuario en cualquier momento. **Precondiciones:** 1. El usuario debe tener un perfil creado y almacenado en el dispositivo y en el servidor. 2. El usuario debe haber iniciado sesión. **Postcondición:** 1. La petición enviada será aceptada o rechazada. **Prioridad:** Alta
Escenario principal
1. El usuario recibirá una notificación de “Petición de chat recibida”. 2. La aplicación mostrará una ventana en la que se muestren los datos que coinciden entre ambos usuarios y se solicite aceptar o rechazar dicha petición de chat. 3. El usuario pulsará “Sí” o “No”.

Tabla 8: Caso de uso-Recibir petición de chat

Mantener una conversación por chat
Identificador: UC-07 **Requisito asociado:** MA6 **Descripción:** Cuando la petición de chat es aceptada, ambos usuarios podrán mantener una conversación de chat mediante la aplicación. **Precondiciones:** 1. El usuario que ha recibido la petición debe aceptarla. **Postcondición:** 1. Se mostrará una ventana de chat que podrán utilizar los usuarios para intercambiar mensajes. **Prioridad:** Alta
Escenario principal
1. La aplicación abrirá una nueva actividad en la que se verá un cuadro de mensajes enviados y recibidos, un cuadro para enviar mensajes y un botón de “Enviar”. 2. Los usuarios podrán intercambiar mensajes que se irán mostrando en el cuadro de mensajes.

Tabla 9: Caso de uso-Mantener conversación de chat

Ver historial de chats
Identificador: UC-08 **Requisito asociado:** MA7 **Descripción:** Existirá una opción en la aplicación mediante la cual el usuario podrá ver los chats que haya mantenido con otros usuarios. **Precondiciones:** 1. El usuario debe haber iniciado sesión en la aplicación. **Postcondición:** 1. Se mostrará una actividad en la que se verán los usuarios con los que se haya mantenido una conversación y podrán verse dichas conversaciones. **Prioridad:** Alta
Escenario principal
1. El usuario pulsará la opción de ver su historial de chats. 2. La aplicación mostrará una pantalla donde se muestre la lista de usuarios con los que ha chateado alguna vez. 3. El usuario podrá pulsar cualquier nombre de la lista para ver las conversaciones que ha mantenido con ese usuario.

Tabla 10: Caso de uso-Ver historial de chats

Por cuestiones de espacio y de importancia no se han incluido aquí los casos de uso de la API. Aun así, con lo expuesto en el apartado de requisitos y lo que se detallará en los siguientes capítulos se comprenderá de forma precisa su funcionamiento completo.

3.4 Especificación de las bases de datos

Para el diseño de las bases de datos se ha tenido en cuenta el hecho de que la aplicación a desarrollar no sigue un modelo de datos usual de una aplicación de mensajería en la que el concepto de una conversación situada en un intervalo de tiempo no tiene por qué existir, ya que pueden alargarse indefinidamente en el tiempo.

3.6.1 Base de datos local

En la base de datos local serán almacenados los usuarios con los que se ha entablado una conversación en algún momento y los mensajes de esas conversaciones. Por ello, incluye las siguientes entidades:

-Usuario: se almacenarán el nombre de los usuarios y sus parámetros de compatibilidad.
-Chat: como forma intermedia entre el mensaje y el usuario, se incluye esta entidad como la representación de una conversación que ha tenido lugar en un intervalo de tiempo. Irá asociado a un usuario.
-Mensaje: incluirá el nombre del usuario con el que se ha intercambiado el mensaje, el contenido del mensaje y si ha sido enviado o recibido e irá asociado a un chat.

3.6.2 Base de datos remota

Su estructura es similar a la base de datos local, pero en ella irán almacenadas todas las cuentas de usuario. Está formada por las siguientes entidades:

-Usuario: contiene todos los datos del usuario, incluyendo sus contraseñas.
-Chat: estará formada por los dos instantes de tiempo en que se inicia y concluye una conversación y los dos usuarios que la han mantenido.
-Mensaje: incluirá al usuario que envía y al que recibe el mensaje, el contenido del mensaje y la conversación a la que está asociado.

Capítulo 4. Diseño

En este capítulo se incluirán diagramas que muestren el diseño de la aplicación en todos los niveles. Se apreciarán en detalle las interacciones entre los módulos del sistema, los modelos relacionales de las bases de datos y las relaciones entre las clases implementadas.

4.1 Diagrama de distribución

En la Figura 2 podemos ver el diagrama de distribución en el cuál se observa la comunicación entre los componentes que conforman todo el proyecto y la estructura implementada. La comunicación entre la API y la aplicación móvil se realiza mediante peticiones HTTP. Los datos enviados en estas peticiones se serializan en formato JSON.

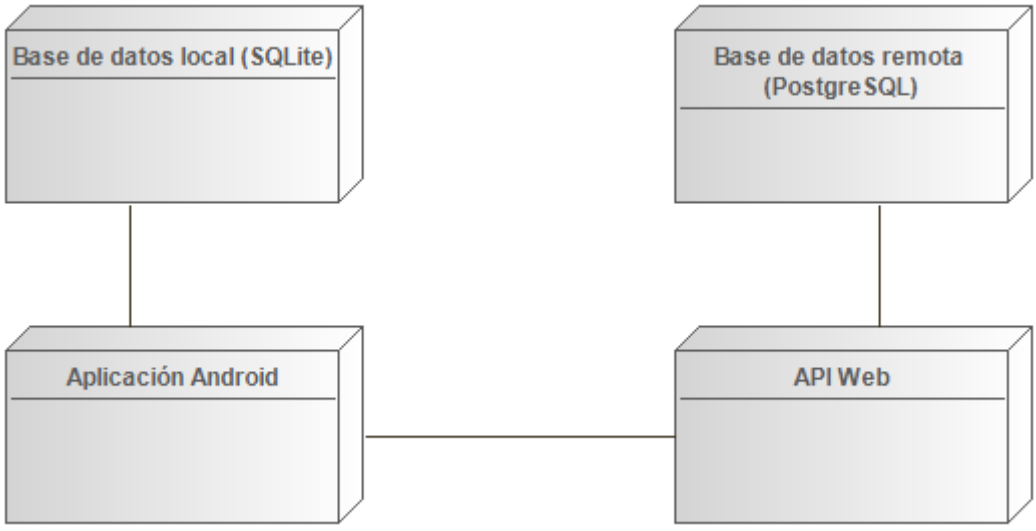

Figura 2: Diagrama de distribución

4.2 Modelos relacionales de las bases de datos

En las figuras Figura 3 y Figura 4 se muestran los modelos de ambas bases de datos. Los dos son bastante similares, ya que, en esencia, las bases de datos almacenan lo mismo. La principal diferencia radica en que en la base de datos local se almacena sólo lo asociado al usuario que utiliza la aplicación móvil y en la remota los datos de todos los usuarios de la aplicación.

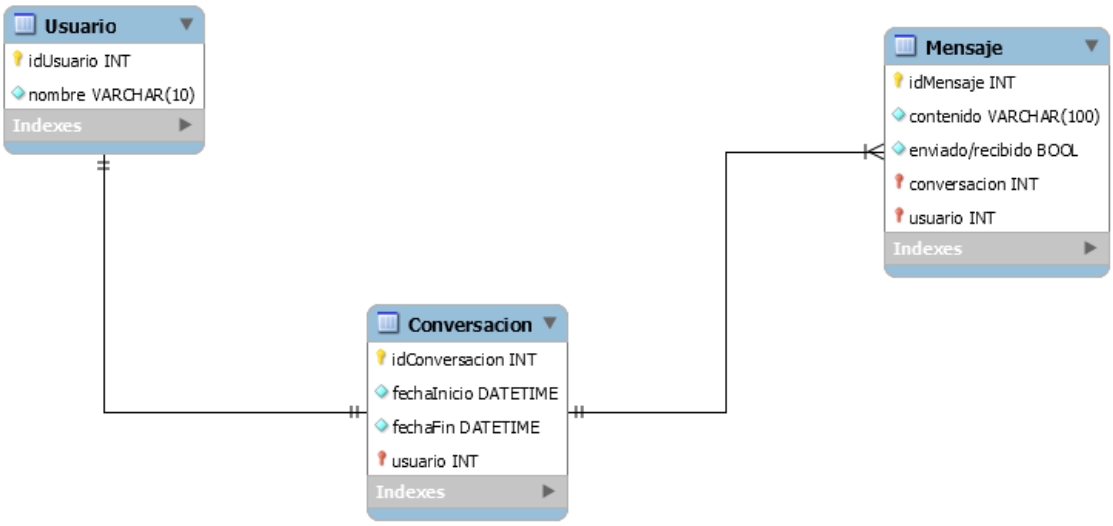

Figura 3: Base de datos local

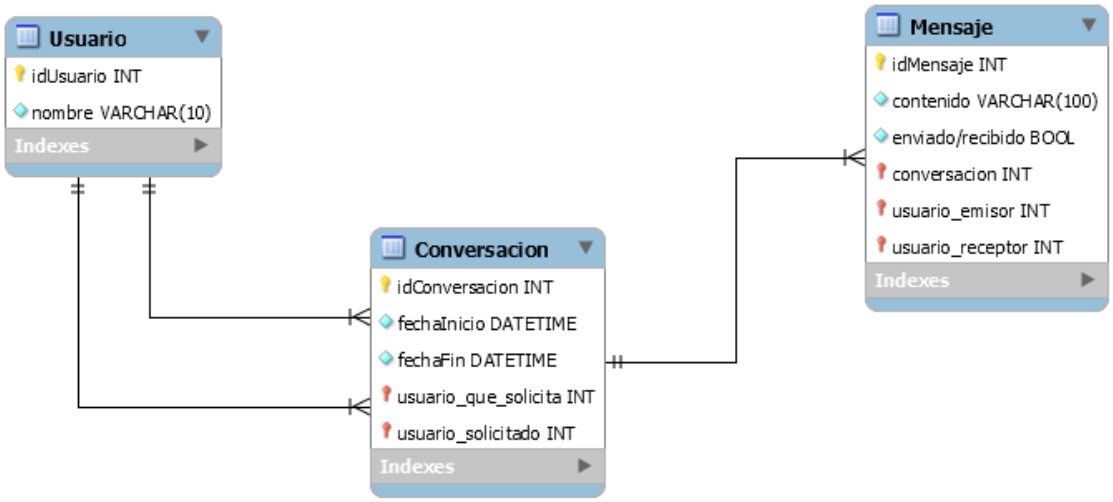

Figura 4: Base de datos remota

Como se puede observar en la Figura 3, en la base de datos local se relaciona a un solo usuario con una conversación, mientras que en la remota se relaciona a dos usuarios. Esto es debido a que, en el caso de la aplicación móvil, los datos del usuario se almacenan en forma de preferencias. Además, de forma local, se sabe que todas las conversaciones que se almacenan son mantenidas con el usuario que tiene instalada la aplicación.

Otra consideración a tener en cuenta es que en los diagramas relacionales no se ha visto necesario incluir todos los atributos de Usuario debido a que no son influyentes en el modelo. Aun así, hay que tener en cuenta que en las bases de datos se incluyen los siguientes atributos en la tabla Usuario además de los que aparecen en los modelos.

Atributos obligatorios:

-Contraseña: este atributo solo se almacenará en la base de datos remota. En el caso del almacenamiento local, se almacenará como preferencia.

-Dirección MAC Bluetooth: este atributo se recoge del dispositivo del usuario y si almacena en la base de datos remota. Este dato se utiliza cuando un usuario busca perfiles cercanos a él.

-Edad

-Ciudad y país: en el proceso de creación de perfil se le pide al usuario que ingrese su ciudad y su país de residencia por separado, pero en las bases de datos se almacena como una sola cadena de caracteres.

Atributos opcionales:

-Trabajo

-Hobbies

-Gustos musicales

-Gustos de cine

-Gustos de lectura

Los gustos y las aficiones se almacenan como cadenas de caracteres en las que cada elemento se encuentra seguido de una coma y un espacio.

4.3 Diagrama de clases

En la Figura 5 se muestra el diagrama de clases de la aplicación Android. En este diagrama se ven diferenciadas las clases en sus paquetes correspondientes y las relaciones entre ellas, que se explican a continuación:

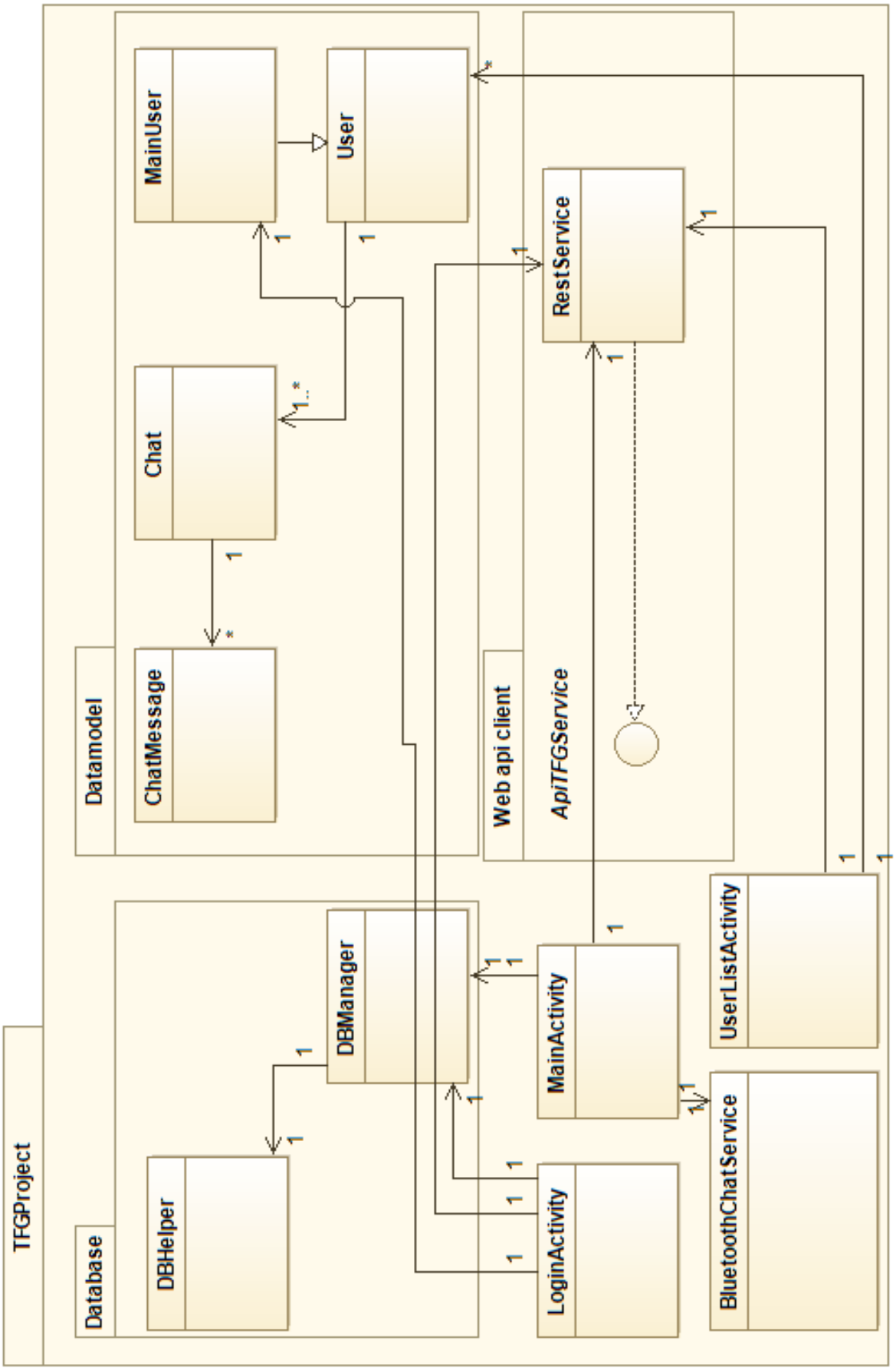

Figura 5: Diagrama de clases de la aplicación Android

- **Database:** La clase *DBManager* instancia a la clase *DBHelper* para poder crear la base de datos correspondiente que se va a gestionar. De esta forma, el resto de clases instancia a la clase *DBManager* abstrayéndose la gestión de la base de datos.
- **Datamodel:** Aquí se incluyen las clases correspondientes a las entidades de la base de datos, por lo que las relaciones entre dichas clases son la mismas que las de las entidades de la base de datos.
    - **MainUser:** Esta clase es la única que difiere, ya que representa al usuario que utiliza la aplicación. En *LoginActivity* se instancia esta clase para comunicarse con el servidor y comprobar las credenciales.
- **Web *API* client:** La clase *RestService* adapta la interfaz *APITFGService* para poder implementar los métodos que esta contiene desde el resto de clases.

En el caso de la API web, dada su simplicidad estructural, no es necesario incluir su diagrama de clases. Más adelante se explicará la estructura del proyecto y se comentará cada clase, aunque si es conveniente ahora exponer las relaciones más importantes:

- **Model:** Este es un paquete en el que se incluye la clase que gestiona la base de datos y todas las clases que se corresponden con el modelo de la base de datos remota. La relación entre estas clases es la misma que la de la base de datos.
- **Controllers:** En este paquete se incluyen los controladores que conformarán la lógica de la API. Estos controladores utilizan las clases del paquete *Model* para realizar las gestiones pertinentes sobre la base de datos (del mismo modo que con la clase *DBManager* en la aplicación móvil) y para intercambiar datos con la aplicación móvil cuando se recibe una petición.

Capítulo 5. Implementación y pruebas

En este capítulo se detallará todo lo referente a la estructura de los proyectos, toda la metodología seguida en la implementación y se incluirá una prueba demostrando todo lo desarrollado.

5.1 Estructura del proyecto Android

En la Figura 6 se muestra toda la estructura de carpetas y clases del proyecto Android. Más adelante se darán detalles sobre cada uno de los elementos y paquetes que contiene.

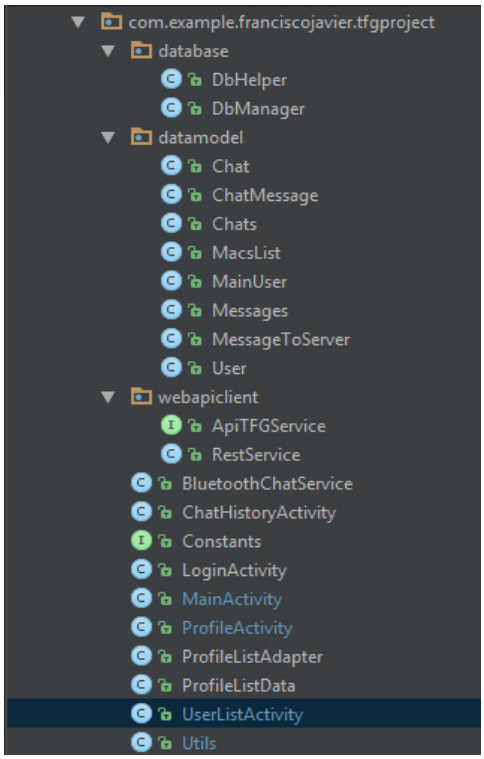

Figura 6: Estructura del proyecto Android

- **Database:** En esta carpeta se encuentran las clases correspondientes a la creación y gestión de la base de datos.
    - **DbHelper**: Esta clase contiene el nombre de la base de datos y los métodos para crearla y actualizarla.
    - **DbManager:** Esta clase contiene los métodos necesarios para interaccionar con la base de datos.

- **Datamodel:** En esta carpeta se encuentra todo el modelo de datos que necesita la aplicación para corresponderse con las bases de datos y la API.
    - **Chat:** Esta clase se corresponde con la tabla Chat de la base de datos local.
    - **ChatMessage:** Esta clase se corresponde con la tabla Mensaje de la base de datos local.
    - **Chats**: Esta clase sirve para encapsular una lista de chats en un solo objeto.
    - **MacsList:** Esta clase sirve para encapsular la lista de direcciones MAC que descubre la aplicación al buscar.
    - **MainUser:** Esta clase agrupa los datos almacenados en las preferencias del usuario de la aplicación.
    - **Messages:** Esta clase sirve para encapsular una lista de mensajes en un solo objeto.
    - **MessageToServer:** Esta clase sirve para serializar un mensaje al modelo de la base de datos remota.
    - **User:** Esta clase se corresponde con la tabla Usuario de la base de datos local.
- **WebAPIclient:** En esta carpeta se encuentran las clases necesarias para enviar las peticiones HTTP que se requieren.
    - **APITFGService:** Esta interfaz utiliza la librería *Retrofit* para crear los métodos necesarios para el envío de peticiones *HTTP*.
    - **RestService:** Esta clase implementa un servicio REST para la comunicación con el servidor.
- **Carpeta principal**: En esta carpeta se incluyen el resto de clases que componen el proyecto:
    - **BluetoothChatService:** Esta clase implementa todo el sistema de chat vía Bluetooth de la aplicación.
    - **ChatHistoryActivity:** Esta es la actividad que implementa la lógica necesaria para que el usuario pueda ver los chats almacenados en la base de datos local.
    - **Constants:** Esta interfaz incluye todas las constantes necesarias del proyecto.
    - **UserListActivity:** Esta es la actividad que implementa la lógica necesaria para que el usuario pueda ver los usuarios cercanos a él.
    - **LoginActivity:** Esta actividad implementa la lógica necesaria para que el usuario pueda iniciar sesión en la aplicación.
    - **MainActivity:** Esta es la actividad principal de la aplicación, que incluye la parte de iniciar búsqueda de usuarios y del chat.
    - **ProfileActivity:** Esta actividad implementa la lógica para la creación de perfiles.

- **ProfileListAdapter:** Esta clase es un adaptador para las listas hobbies y gustos de la creación de perfil.
- **ProfileListData:** Esta clase contiene el modelo de datos para la persistencia de los estados de las listas de aficiones y gustos.
- **Utils:** Esta clase engloba métodos estáticos útiles en distintos puntos de la aplicación.

5.2 Estructura del proyecto web

En la Figura 7 se observa la estructura del proyecto de la API. A continuación se darán detalles de cada elemento y paquete que conforma el proyecto explicando sus funciones en el mismo.

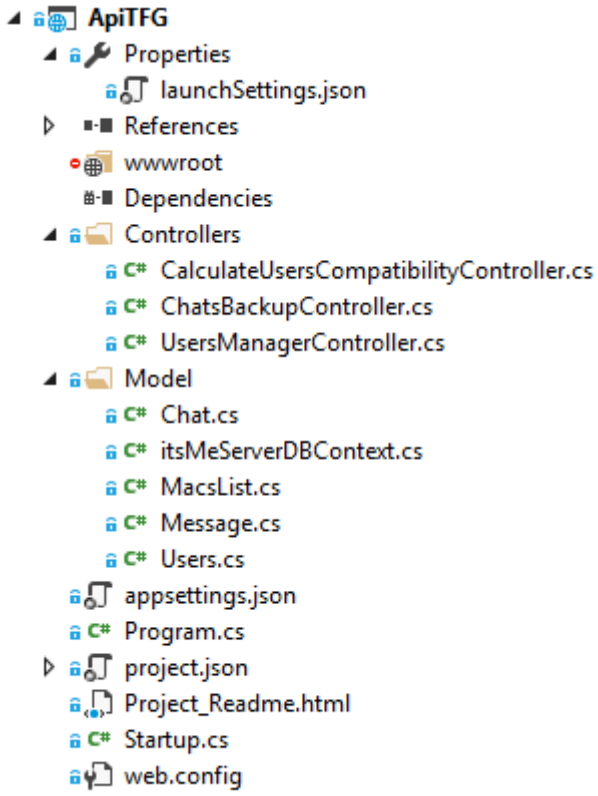

Figura 7: Estructura del proyecto web

- **Properties:** En este apartado se encuentran las propiedades del proyecto. Aquí podemos modificar las opciones de compilación y la configuración de ejecución.
- **References:** Esta carpeta incluye las referencias a todas las bibliotecas externas que se utilizan en el proyecto.

- **Wwwroot:** En esta carpeta se ubican todos los archivos referentes a la web. En este proyecto no se ha desarrollado una página web, solo una API, por lo que se encuentra vacía.
- **Controllers:** Esta carpeta contiene los controladores de la API que componen la lógica de esta.
    - **CalculateUsersCompatibilityController:** Esta clase contiene el método que recibe la lista de las direcciones MAC Bluetooth cercanas al usuario que las envía. Se encarga de buscar los usuarios correspondientes a esas MACs, calcular las compatibilidades y reenviar la lista con las compatibilidades.
    - **ChatsBackupController:** En esta clase se gestionan todas las peticiones referentes a las copias de seguridad de los chats de los usuarios.
    - **UsersManagerController:** En esta clase se encuentran los métodos para la creación y gestión de usuarios, y para la comprobación de las credenciales de inicio de sesión.
- **Model:**
    - **itsMeServerDBContext:** Esta clase es la gestiona todas las operaciones que la API requiera hacer en la base de datos.
    - **Chat:** Esta clase se corresponde con la entidad Chat de la base de datos.
    - **Message:** Esta clase se corresponde con la entidad Mensaje de la base de datos.
    - **User:** Esta clase se corresponde con la entidad *User* de la base de datos.
    - **MacsList:** Esta clase sirve para englobar una lista de direcciones MAC en un solo objeto para facilitar la serialización.
- **Resto de archivos:**
    - **Program:** Este archivo contiene la función principal del proyecto. En ella se realizan algunas configuraciones de construcción de lo que actuará como servidor web.
    - **Project:** Este archivo contiene todas las referencias a las herramientas y librerías utilizadas en la aplicación, así como distintos parámetros de configuración del proyecto.
    - **Startup:** En este archivo se configura la aplicación web al ser lanzada. Aquí se incluyen configuraciones varias de las bibliotecas, middlewares y otros servicios que se requieran.

5.3 Desarrollo del proyecto Android

En esta sección se van a enumerar y comentar de forma extensa las fases del desarrollo del proyecto Android, incluyendo los problemas encontrados, los cambios efectuados respecto a las primeras fases del TFG y la explicación de todas las decisiones tomadas.

5.3.1 Comunicación directa

La primera idea era incluir la opción de que el usuario eligiera si quería utilizar Bluetooth o *WiFi Direct* para la comunicación y la búsqueda de otros usuarios. Posteriormente se decidió avanzar en principio con la opción de Bluetooth, y por menor prioridad respecto a otras funcionalidades, se dejó la opción de *WiFi Direct* en segundo plano. Al final no se ha podido incluir esa opción en el resultado final, aunque en las primeras fases del trabajo se realizó un análisis del protocolo y se desarrolló una pequeña aplicación de prueba para investigar cómo podría implementarse en el resultado final.

En esta pequeña aplicación se implementó una lista simple en la que se mostraban los dispositivos encontrados por *WiFi Direct*. Se daba la opción de escoger uno de ellos para conectarse, y cuando el usuario del dispositivo seleccionado aceptaba, uno de ellos enviaba un mensaje que el segundo mostraba en la pantalla.

Esto es lo suficientemente útil como para ser usado de base cuando se quiera extender la aplicación para que también de la opción de mantener comunicaciones vía *WiFi Direct*.

En la Figura 8 se demuestra del funcionamiento descrito:

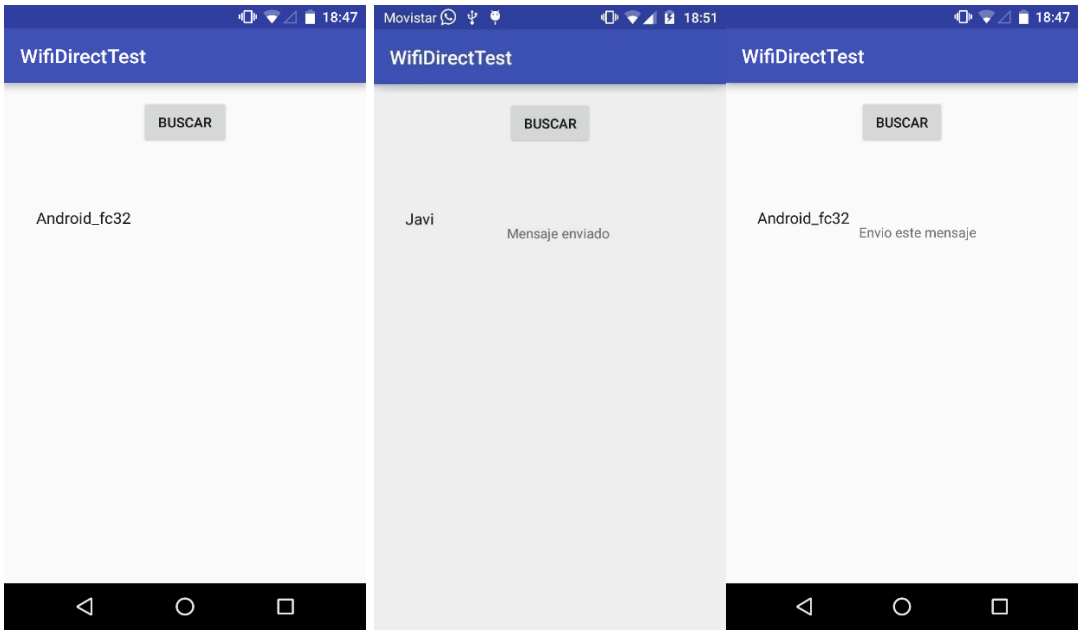

Figura 8: Funcionamiento de WiFi Direct

El motivo por el cual se decidió descartar la opción de *WiFi Direct* es que aún no está lo suficientemente extendido. Incluirlo hubiera supuesto disminuir considerablemente el público objetivo debido a que el número de dispositivos compatibles es reducido. Además, otro punto a favor de Bluetooth es su bajo consumo frente a *WiFi Direct*, lo cual es una ventaja teniendo en cuenta que la aplicación requiere su uso de forma continua.

5.3.2 Comunicación con la API web

La aplicación realiza una intensa labor de comunicación por medio de varias clases en las que se realizan peticiones HTTP para obtener y enviar distintos datos a la API. Para realizar estas peticiones se optó por incluir la librería *Retrofit*, que aporta mucha comodidad a la hora de realizar estas peticiones. Para usar esta librería se crea una interfaz en la que se especificarán los métodos que luego serán implementados donde se necesite. En el Código 1 se puede ver un ejemplo de cómo se especifican dichos métodos.

```
@POST("/calculateuserscompatibility/{id}")
public void getListOfNearbyUsers(@Path("id") Integer id, @Body
MacsList macsList, Callback<List<String>> callback);
```

Código 1: Ejemplo de método HTTP especificado

Una vez creada la interfaz se crea una clase que servirá de capa intermedia entre la interfaz y el resto de clases. Se puede ver en detalle la estructura de esta clase en el Código 2.

```
public class RestService {
    private static final String URL ="http://83.49.161.24:8080/API/";
    private retrofit.RestAdapter restAdapter;
    private APITFGService APITFGService;

    public RestService(){
        restAdapter = new retrofit.RestAdapter.Builder()
                .setEndpoint(URL)
                .setLogLevel(RestAdapter.LogLevel.FULL)
                .build();

        APITFGService = restAdapter.create(APITFGService.class);
    }

    public APITFGService getAPITFGService(){
        return APITFGService;
    }
}
```

Código 2: Clase RestService

Esta clase crea un adaptador que funciona como un servicio REST basado en la interfaz creada, que será utilizado como medio para conectar a la API e implementar los métodos HTTP que se necesiten.

Una vez creados estos dos archivos solo hay que implementar los *Callbacks* de las peticiones usando el *restService* para que la aplicación aporte el resultado deseado en caso de éxito o fallo en la petición. El Código 3 muestra un ejemplo de cómo se implementarían los métodos.

```
restService.getAPITFGService().login(user, new Callback<MainUser>() {
    @Override
    public void success(MainUser user, Response response) {
        //Exito
    }

    @Override
    public void failure(RetrofitError error) {
        //Fallo
    }
});
```

Código 3: Ejemplo de petición HTTP

5.3.3 Inicio de Sesión

La primera actividad de la aplicación móvil es la de *Login*. Esta es la parte más sencilla de la aplicación, sólo se encarga de aceptar las credenciales introducidas por el usuario y enviarlas al servidor para que compruebe con la base de datos si son correctas. Si el usuario ya tiene su cuenta almacenada en el dispositivo, el inicio de sesión se hace automáticamente utilizando los datos almacenados.

Desde esta actividad se puede acceder a la de creación de perfil mediante un botón incluido en la interfaz mostrada en la Figura 9.

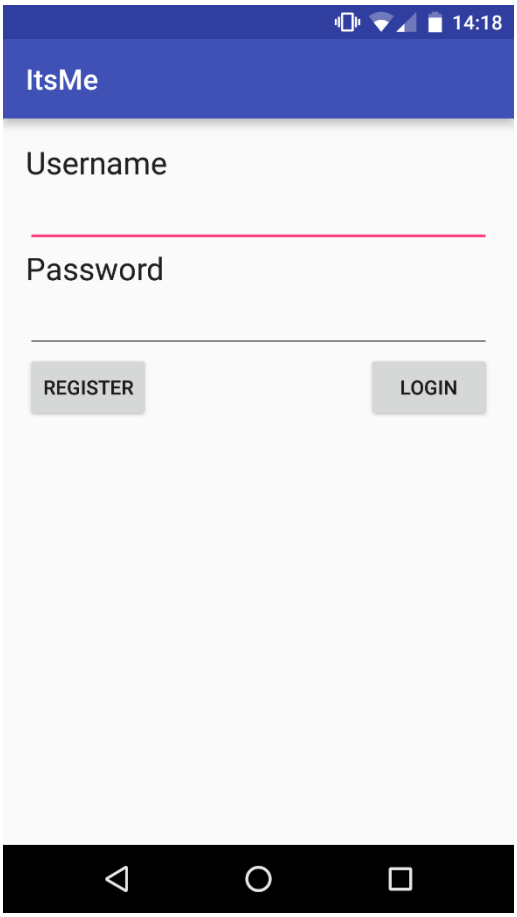

Figura 9: Interfaz de inicio de sesión

5.3.4 Creación de cuenta

En la Figura 10 podemos ver muestras de los pasos de creación de cuentas que se ha implementado optando por un proceso por pasos que resulte más ameno al usuario. Durante todo el proceso se le indica al usuario cuantos pasos hay en total y en cual se encuentra. En cualquier momento puede ir a un paso anterior utilizando el botón *back* de su *smartphone*. A continuación se detallan los datos solicitados en cada paso.

- **Paso 1:** Nombre de usuario, contraseña y edad. Estos datos son obligatorios para la creación de la cuenta.
- **Paso 2:** Ciudad y país de residencia, y trabajo. La ciudad y el país son obligatorios, desde el trabajo hacia adelante todos los datos son opcionales.
- **Paso 3:** Hobbies. En este paso y en los siguientes se muestra una lista con las posibles opciones que puede incluir el usuario en su perfil. Se pueden escoger desde ninguna hasta todas.
- **Paso 4:** Gustos musicales.
- **Paso 5:** Gustos cinematográficos.
- **Paso 6:** Gustos de lectura.

Una vez en el paso 6, si el usuario ha completado todo el proceso, tendrá que pulsar el botón guardar. Seguidamente la aplicación almacenará el perfil completo en el teléfono y a su vez lo enviará al servidor para ser almacenado en la base de datos remota.

Para las listas de gustos y la de hobbies se ha implementado un adaptador personalizado compuesto por una etiqueta y un *check box*. El principal problema de este tipo de listas es que al hacer *scroll* se pierden lo datos marcados y hay que volver a marcarlos para que aparezcan. Esto suponía un obstáculo importante dado que cuando se marca una opción se añade a una lista asociada. Cuando se hace *scroll*, parece que la opción se desmarca, pero en la lista se mantiene. Esto ocurre por las *ListView* destruyen los elementos que no están en la interfaz y los vuelven a crear cuando sea necesario. Para solucionar esto se ha asociado al adaptador *ProfileListAdapter* una clase *ProfileListData* con el objetivo de mantener la persistencia del *check box*. De esta manera, solo hay que añadir la lógica necesaria al adaptador para que cuando se cree una la vista de nuevo se compruebe si el *check box* debe ir marcado o no.

Una vez recogidos los datos del perfil, también se recoge la dirección MAC de Bluetooth del dispositivo. El propósito de esto se explica más adelante.

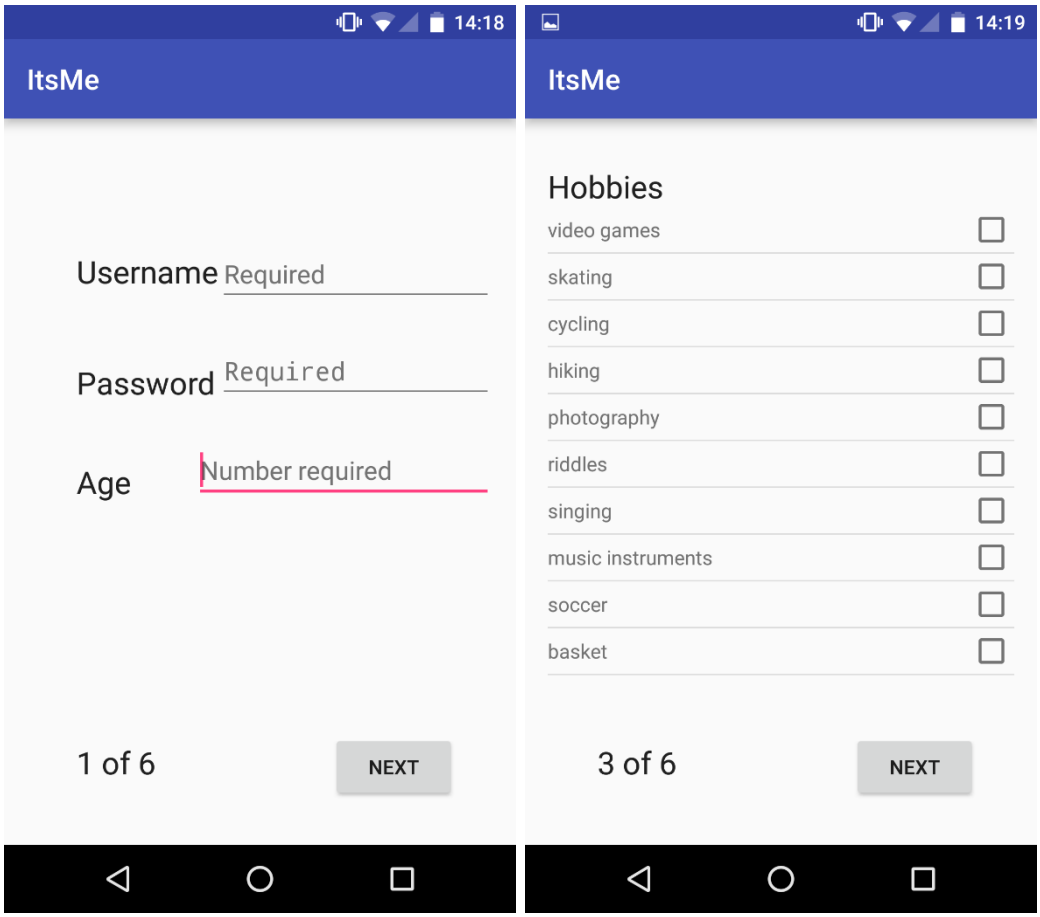

Figura 10: Pasos 1 y 3 del proceso de creación de perfil

5.3.5 Búsqueda de usuarios y chat Bluetooth

Introducir el uso de Bluetooth en la aplicación móvil fue relativamente sencillo dado que entre las plantillas de Android Studio se encuentra una aplicación básica de chat vía Bluetooth. De esa plantilla se han adaptado principalmente dos de sus clases: *DeviceListActivity* y *BluetoothChatService*. También se ha aprovechado parte de la clase *MainActivity* de la plantilla.

La clase *MainActivity* mostrada en la Figura 11 se incluye todo el proceso de búsqueda de usuarios, un pequeño botón de menú en la parte izquierda y todo lo referente al chat. Lo primero que vemos es un botón en el centro de la pantalla abre un cuadro de diálogo y comienza a buscar. La clase de control de este diálogo es *UserListActivity*, que se ha desarrollado aprovechando algunas partes de la clase *DeviceListActivity* de la plantilla.

Previamente, al iniciarse la actividad principal, la aplicación comprueba que se dispone de Bluetooth en el dispositivo, solicita al usuario que lo active y establece el dispositivo como descubrible para poder ser buscado por otros usuarios.

Cuando la actividad *UserListActivity* se inicia, comienza a escanear por Bluetooth, realizando un escaneo de 8 segundos. Una vez terminado este escaneo envía al servidor todas las direcciones MAC descubiertas y espera una respuesta. En la respuesta se incluirán los nombres de usuario asociados a los dispositivos encontrados y la compatibilidad con dichos usuarios. Estos datos son mostrados en pantalla y se le da la opción de elegir uno de ellos. Una vez pulsado uno, se vuelve a la actividad principal y esta se encarga del resto.

Figura 11: Primera interfaz de la MainActivity

La clase *BluetoothChatService* es la encargada de todo lo referente a la comunicación por Bluetooth. En ella se incluyen métodos para conectar a un dispositivo, enviar una petición de chat, recibirla y todo el protocolo de intercambio de mensajes entre los usuarios. Esta clase no es un servicio de Android propiamente dicho, pero funciona de forma similar. Se inicia a la vez que la actividad principal y le proporciona todos los datos necesarios utilizando un manejador implementado en ella. La clase *BluetoothChatService* de la plantilla incluía hilos gestionados a través de distintos métodos durante las distintas fases del chat. Para esta aplicación la implementación de esos hilos ha sido modificada en gran medida al seguirse un proceso muy distinto al de la plantilla.

5.3.6 Historial de chats y base de datos local

La aplicación almacena los chats en una base de datos *SQLite* que se gestiona en la clase *DbManager*. En dicha clase se han implementado todos los métodos necesarios para insertar chats, usuarios y mensajes, además de para poder consultarlos en la actividad historial de chats.

Una conversación de chat termina cuando uno de los dos usuarios se desconecta o cuando se pierde la conexión por algún motivo. En ese momento se inserta el chat y los mensajes asociados en la base de datos local, relacionándolo con el usuario con que se ha mantenido la conversación. Posteriormente el usuario podrá ver la conversación pulsando sobre la opción *Chat History* del menú de la actividad principal.

La actividad *ChatHistory* presenta listados por el nombre todos los usuarios con los que se ha tenido alguna vez una conversación. Se puede pulsar sobre cada uno de ellos para ver la lista de chats mantenidos ordenados por fecha, y a su vez se puede pulsar sobre un chat para para ver los mensajes del mismo.

Se ha planteado incluir en esta actividad una opción de hacer una copia de seguridad de los chats almacenándolos en la base de datos remota. De hecho, se encuentra implementado casi todo lo necesario para ello y la base de datos remota también está preparada, sin embargo, por no ser muy necesario, se le dio una prioridad baja y se ha decidido no incluirlo.

5.4 Desarrollo del proyecto web

La API desarrollada es el segundo módulo clave para el funcionamiento del sistema. Es en este módulo en el que se ha implementado el algoritmo de cálculo de compatibilidad tan importante para el propósito de este TFG.

5.4.1 Base de datos y modelo

Para la base de datos remota se ha utilizado el gestor *PostgreSQL* a la cual se conecta utilizando *Npgsql*. Con la herramienta *EntityFramework* se ha realizado un *scaffolding* de la base de datos obteniendo las clases *itsMeServerDBContext* y todas las clases del modelo de datos. La clase *itsMeServerDBContext* se utiliza para realizar todas las gestiones a la base de datos que sean necesarias.

5.4.2 Creación de usuarios y comprobación de credenciales

En el controlador *UsersManagerController* se gestiona todo lo referente a los usuarios. Cuando un usuario crea su cuenta, los datos se envían al servidor y este llama al método *CreateUser* de este controlador. En dicho método se comprueba que no exista ya un usuario con ese nombre en la base de datos y que no haya ningún dato faltante. Si todo está correcto, se inserta al usuario en la base de datos y se devuelve un OK.

Cuando un usuario inicia sesión, la aplicación móvil envía las credenciales al servidor y este se encarga de comprobar que las contraseñas coinciden en base al nombre enviado. Si coincide, se devuelve un OK.

5.4.3 Cálculo de compatibilidades entre usuarios

Como se ha mencionado, cuando el usuario busca perfiles cercanos, la aplicación móvil envía la lista de dispositivos descubiertos al servidor. Este es el motivo por el cual se almacenan en la base de datos remota la dirección MAC de Bluetooth del dispositivo de cada usuario.

En el controlador *CalculateUsersCompatibility* se encuentra el método en el que se reciben la lista de MAC que envía la aplicación. Cuando se recibe una lista, se busca cada usuario asociado a cada MAC y se pasa a calcular la compatibilidad del usuario que ha enviado la lista con cada uno de los recibidos. El procedimiento para calcular la compatibilidad es el siguiente:

1. Se calcula la diferencia entre las edades de cada usuario, si esta es menor a 10, se suma a la compatibilidad un 25%.
2. Se comparan las cadenas de caracteres que contienen la ciudad y el país de cada uno. Si coinciden en la ciudad, se añade un 15%, si solo coinciden en el país, se suma tan solo un 5%.
3. Se comparan los trabajos y si coinciden, se suma otra cantidad.
4. Para cada una de las listas se sigue el siguiente procedimiento:
    a. Se comprueba que ninguna de las listas sea vacía.
    b. Si no es vacía, se mira cuál de las dos listas es mayor y se calcula el ratio de cada elemento en base a ese tamaño.
    c. Se miran las coincidencias y se suma a la compatibilidad total el ratio establecido anteriormente.

Cuando se ha terminado de calcular las compatibilidades, se ordena la lista de los usuarios en orden de dicha compatibilidad y se devuelve al cliente.

A continuación se detallan los ratios escogidos para cada parámetro:

- Edad: 25%
- Ciudad y país: 15%
- País: 5%
- Trabajo: 12%
- Elementos de las listas: (12/número de elementos de la lista mayor)%

Estos ratios han sido escogidos de forma arbitraria.

5.5 Pruebas

Para comprobar que todo funciona correctamente se ha realizado una prueba con dos *smartphones*. En cada uno de ellos se ha creado una cuenta y se ha realizado una simulación de todo el proceso.

1. En la Figura 12 vemos la primera fase que consistió en buscar a usuarios cercanos y ver que el usuario que desencadena la búsqueda es capaz de encontrar al segundo usuario.

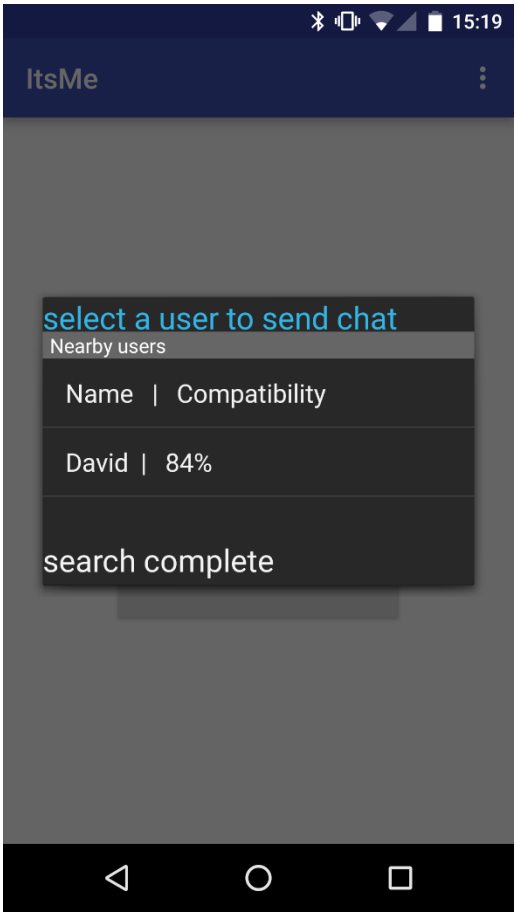

Figura 12: Resultado de la búsqueda

Los datos de los usuarios son los siguientes:

- Usuario 1:
  - Nombre: Javier.
  - Edad: 22.
  - Ciudad y país: Malaga-Spain.
  - Trabajo: Student.
  - Hobbies: video games, hiking, riddles, music instruments, soccer.
  - Gustos musicales: rock, metal, alternative, electro music, dubstep, pop, blues.

    - Gustos de cine: action, adventure, thriller, comedy, sciene fiction.
    - Gustos de lectura: fantasy, science fiction, adventure, thriller, graphic novel, comic.

- Usuario 2:
    - Nombre: David.
    - Edad: 16.
    - Ciudad y país: Malaga-Spain.
    - Trabajo: Student.
    - Hobbies: video games, soccer.
    - Gustos musicales: rock, metal, alternative, blues.
    - Gustos de cine: action, adventure, comedy, sciene fiction, history.
    - Gustos de lectura: fantasy, science fiction, adventure, thriller, graphic novel, comic, history.

A continuación se detallan los cálculos que dan lugar al porcentaje de compatibilidad que encontramos en la Figura 12:

- Edad: | 22-16 | = 6 –> ratio: 25%
- Ciudad y país: coinciden ambos –> ratio: 15%
- Trabajo: coinciden ambos –> ratio: 12%
- Hobbies: valor por coincidencia: 2% –> ratio: 4%
- Gustos musicales: valor por coincidencia: 1.71% –> ratio: 6.86%
- Gustos de cine: valor por coincidencia: 2.4% –> ratio: 9.6%
- Gustos de lectura: valor por coincidencia: 1,71% –> ratio: 10.29%

Existe un pequeño error dado que al realizar los cálculos la aplicación no toma todos los decimales si no que se redondea el resultado a la parte entera más próxima. A pesar del error de redondeo, se puede apreciar que el algoritmo cumple con las características descritas anteriormente.

2. En la segunda parte de la prueba se ha seleccionado al usuario de la lista para enviarle una petición de chat, se ha esperado a que este aceptara y se ha iniciado la fase de chat.
3. En la fase de la Figura 13 se ha probado a tener una conversación entre los dos usuarios, probando todos los aspectos de la parte de chat.

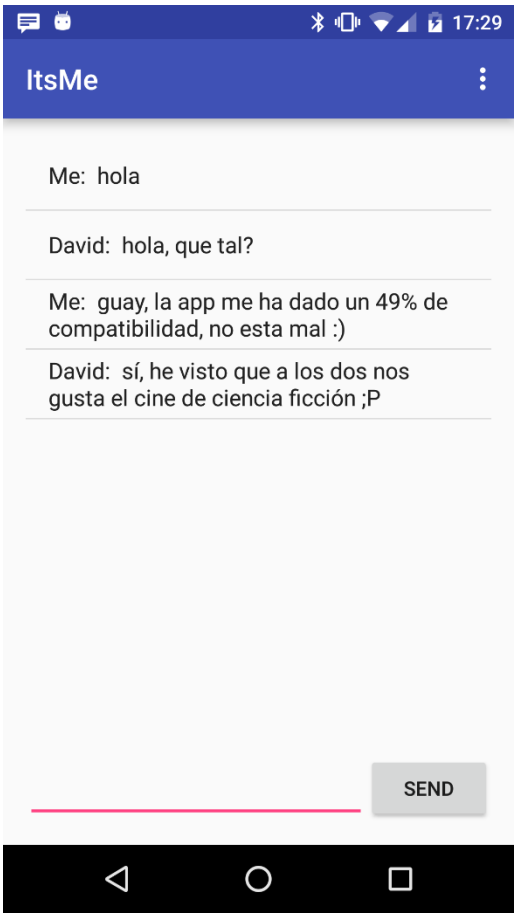

Figura 13: Prueba de chat

4. En la última fase se ha comprobado que, tras terminar el chat por la desconexión de uno de los usuarios, se ha almacenado bien y ha podido ser visualizado como se puede comprobar en la Figura 14.

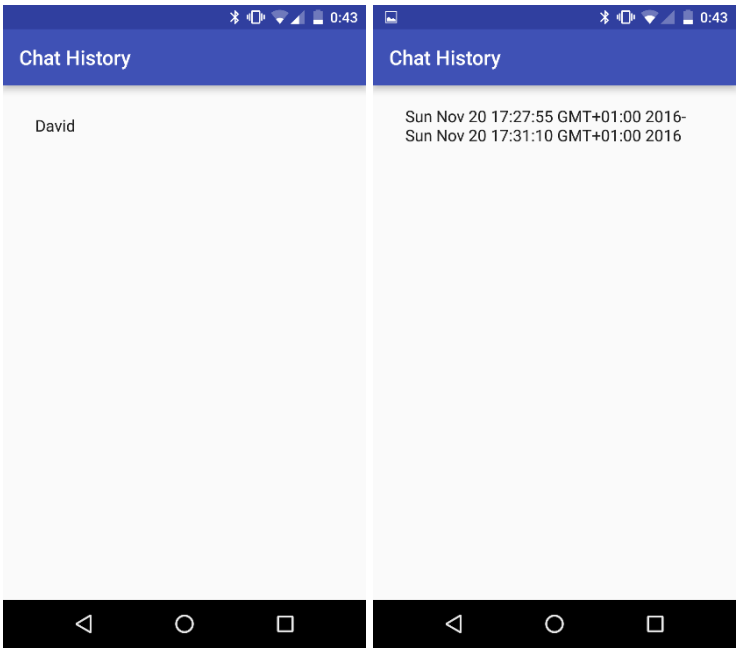

Figura 14: Historial de chats

Tras finalizar la prueba, cabe destacar que el intercambio de mensajes vía Bluetooth es muy rápido, tanto, que casi no hay tiempo de espera. Otro plus es que la respuesta del servidor también es bastante rápida, exceptuando la primera petición, que suele tardar bastante más, el resto de respuestas son casi instantáneas. Esto por supuesto teniendo en cuenta que sólo había dos usuarios haciendo peticiones y el cálculo de compatibilidad se ha hecho con un sólo usuario.

Capítulo 6. Conclusiones y líneas futuras

En la sección final se exponen las conclusiones obtenidas durante la realización de este TFG y en qué puntos podría extenderse o mejorarse, además de que opciones de mejora podrían aplicarse a esos puntos.

6.1 Conclusiones

Tras todo lo expuesto en los apartados anteriores queda claro que las redes inalámbricas en los *smartphones* poseen un potencial mucho mayor del que se aprovecha hoy en día. Esto es clave teniendo en cuenta que las redes inalámbricas son uno de los pilares de las *Smart Cities*, y es sumamente interesante aprovecharlas para una de las principales funciones por las que fueron creadas: mejorar la comunicación.

La aplicación móvil creada desarrolla esa idea proporcionando una herramienta de comunicación a corto alcance, con un rendimiento alto en el intercambio de mensajes. De esta forma iniciar una conversación con cualquier persona de tu alrededor utilizando la aplicación, lejos de parecer lento y aparatoso, resulta muy cómodo. Este hecho es apoyado considerablemente por el algoritmo de cálculo de compatibilidades. Gracias a este algoritmo la función de chat de la aplicación cobra más sentido y utilidad, ya que permite intuir hasta qué punto pueda resultar enriquecedor iniciar una conversación con cualquiera de los usuarios que se encuentren a tu alrededor.

Desarrollar esta aplicación ha supuesto una gran oportunidad para investigar en campos como el desarrollo de una api y el uso de redes inalámbricas. Además, empleando tecnologías novedosas como .NET Core y las nuevas versiones del SDK de Android se pueden desarrollar aplicaciones de calidad y que permiten ser ampliadas.

6.2 Líneas futuras

Tras el extenso trabajo llevado a cabo durante el TFG se han vislumbrado algunas posibles mejoras que podrían ser muy interesantes. A continuación se describen un subconjunto de aspectos que podrían ser mejorados:

- **Mejoras en la interfaz:** La interfaz ha sido diseñada de forma limitada a lo necesario para probar las funcionalidades. Podría mejorarse añadiendo elementos que den más *feedback* al usuario, cambiar los patrones de diseño de las UI o introducir elementos visuales como iconos o imágenes.
- **Añadir avatares a los perfiles:** Las aplicaciones de mensajería suelen incluir avatares opcionales en las cuentas de usuario y sería una interesante opción que añadir a esta aplicación. Esto permitiría tener una primera imagen del usuario con el que se va a entablar conversación, aportando riqueza al algoritmo, o poder encontrar fácilmente al usuario con el que se está manteniendo una conversación.
- **Mejorar el cálculo de compatibilidades:** El algoritmo implementado es quizás un tanto simple y podría mejorarse de diversas maneras. Añadir prioridades a la hora de escoger los gustos y hobbies, por ejemplo, o incluso añadirle inteligencia relacionando entre sí los gustos del usuario o teniendo más en cuenta las edades.
- **Añadir la opción de *WiFi Direct*:** *WiFi Direct* es una tecnología emergente que podría integrarse fácilmente teniendo en cuenta lo desarrollado. A pesar del motivo mencionado por el cuál no se ha incluido, sin duda es una tecnología emergente que dentro de poco sería conveniente incluir.

Referencias

1. Odriozola EE. Factores de riesgo y factores de protección en la adicción a las nuevas tecnologías y redes sociales en jóvenes y adolescentes. 2012: https://dialnet.unirioja.es/servlet/articulo?codigo=4113810.
2. Perlato A. Social network analysis: “Smart cities”: http://chorally.com/learn-impact-smartcity-using-social-networks-analysis/.
3. Williams B. Wireless communication networks advance smart city initiatives. 2014: http://americancityandcounty.com/blog/wireless-communication-networks-advance-smart-city-initiatives.
4. Documentación básica Source Tree: https://confluence.atlassian.com/sourcetreekb/sourcetree-basics-780870007.html.
5. Documentación Android: https://developer.android.com/index.html.
6. Página principal Retrofit: https://square.github.io/retrofit/.
7. Documentación.NET Core: https://docs.microsoft.com/en-us/dotnet/articles/core/index.
8. Sitio oficial de IIS: https://www.iis.net/.
9. Documentación SQLite: https://sqlite.org/docs.html.
10. Manuales PostgreSQL: https://www.postgresql.org/docs/manuals/.

Índice de Figuras, Tablas y Códigos

Apéndices

A. Manual de usuario de la aplicación Android

Para instalar la aplicación Android hay que pasar el archivo APK proporcionado al móvil. Cuando se instalan aplicaciones externas hay que modificar los ajustes para autorizarlas. La manera exacta de hacerlo difiere en cada versión, pero generalmente suele ser: Ajustes > Seguridad > Orígenes desconocidos.

Una vez permitida la instalación, para instalar el archivo APK solo hay que ir al lugar donde se haya colocado, pulsar sobre él y el asistente de instalación de Android se iniciará automáticamente.

Cuando se inicia la aplicación, lo primero que aparece es una la pantalla de inicio de sesión. Hay que pulsar en el botón "*Register*" (Figura 15) para iniciar el proceso de creación de cuenta.

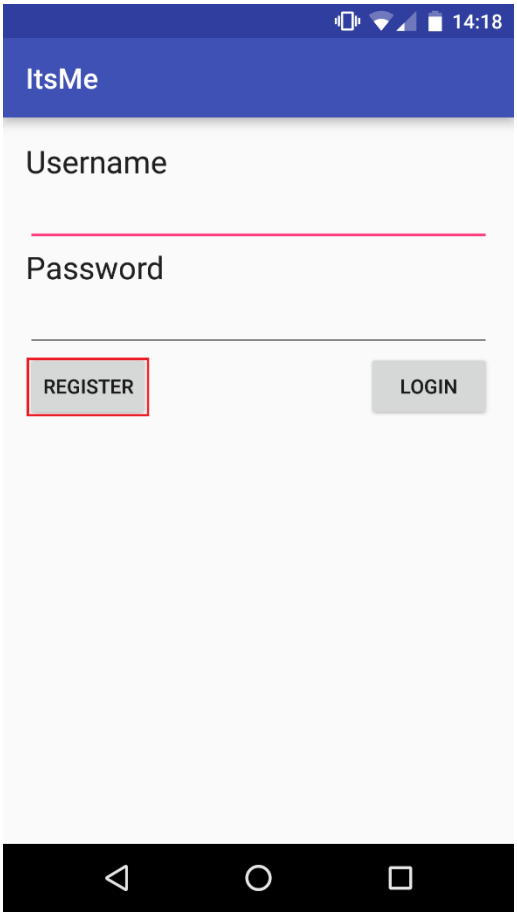

Figura 15: Inicio de sesión

Al pulsar sobre "*Register*" se inicia el proceso de creación de cuenta. Este proceso está dividido en varios pasos y en cada paso hay que introducir distintos datos. Durante el proceso se puede ir al paso anterior pulsando sobre el botón *back* del teléfono. Si se pulsa back en el paso 1, se volverá a la pantalla de inicio de sesión (Figura 16).

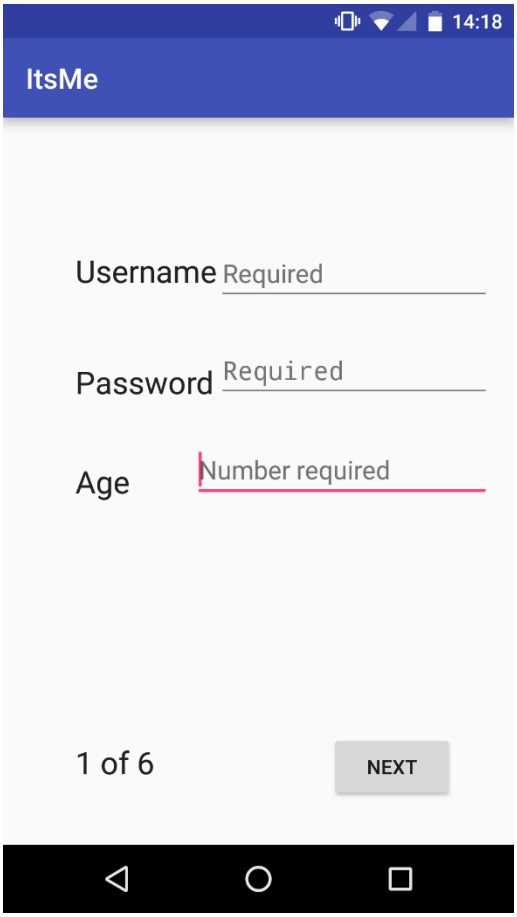

Figura 16: Paso 1 de creación de cuenta

Los datos que se requieran en cada paso están señalados como "*Required*". Si no se introducen esos datos la aplicación no dejará pasar al siguiente paso. Si se vuelve atrás en un paso, los datos introducidos se mantienen exceptuando en los pasos de 3, 4, 5 y 6.

En los pasos 3, 4, 5 y 6 (Figura 17) hay que escoger los gustos pulsando sobre el *check box* asociado a cada elemento de la lista. Puede volver a pulsarse sobre un *check box* marcado para desmarcar la opción. Hay que tener en cuenta que, como se ha especificado anteriormente, si se avanza o se retrocede en alguno de estos pasos la lista se reiniciará al volver.

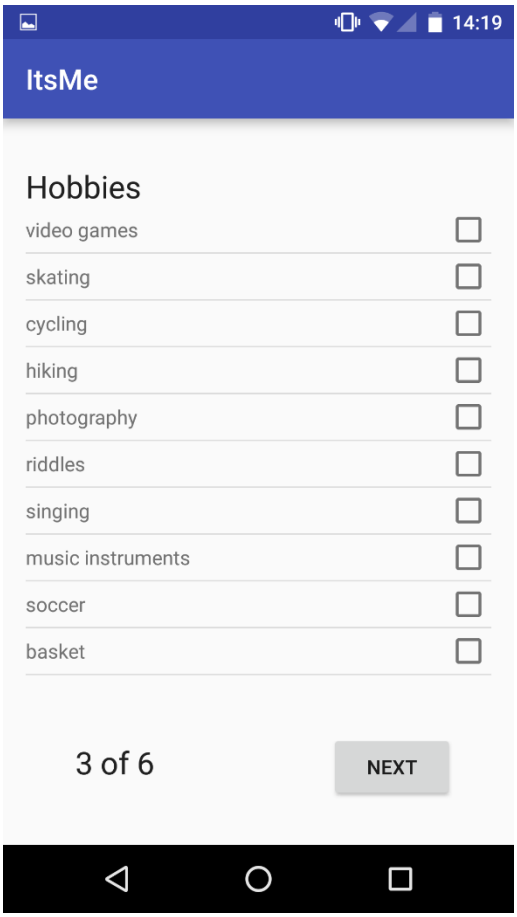

Figura 17: Paso 3 de creación de perfil

Cuando se completa la creación de perfil, si ya existe en el sistema un usuario con el nombre introducido, la aplicación avisará y el nombre introducido tendrá que ser modificado para poder crear el perfil.

Una vez creado el perfil la aplicación almacena en un archivo de preferencias todos los datos introducidos e inicia sesión automáticamente. A partir de ese momento, a no ser que se desinstale la aplicación o se eliminen los datos de la misma, se iniciará sesión automáticamente cada vez que se inicie. Si se borran los datos, basta con introducir tu nombre de usuario y tu contraseña para iniciar sesión y los datos se volverán a guardar.

Una vez en la pantalla principal lo primero que se solicita es que se active él Bluetooth indicando que una vez activado el dispositivo pasa a modo visible. La pantalla principal está formada simplemente por un botón grande en el centro (Figura 18) y al ser pulsado se abrirá un cuadro de dialogó y se iniciará la búsqueda de usuarios cercanos. Hay que tener en cuenta que la búsqueda de usuarios realiza varios pasos y cada uno posee una duración, por lo que la espera total puede ser de entre 9 y 12 segundos hasta obtener la lista de usuarios cercanos y sus compatibilidades.

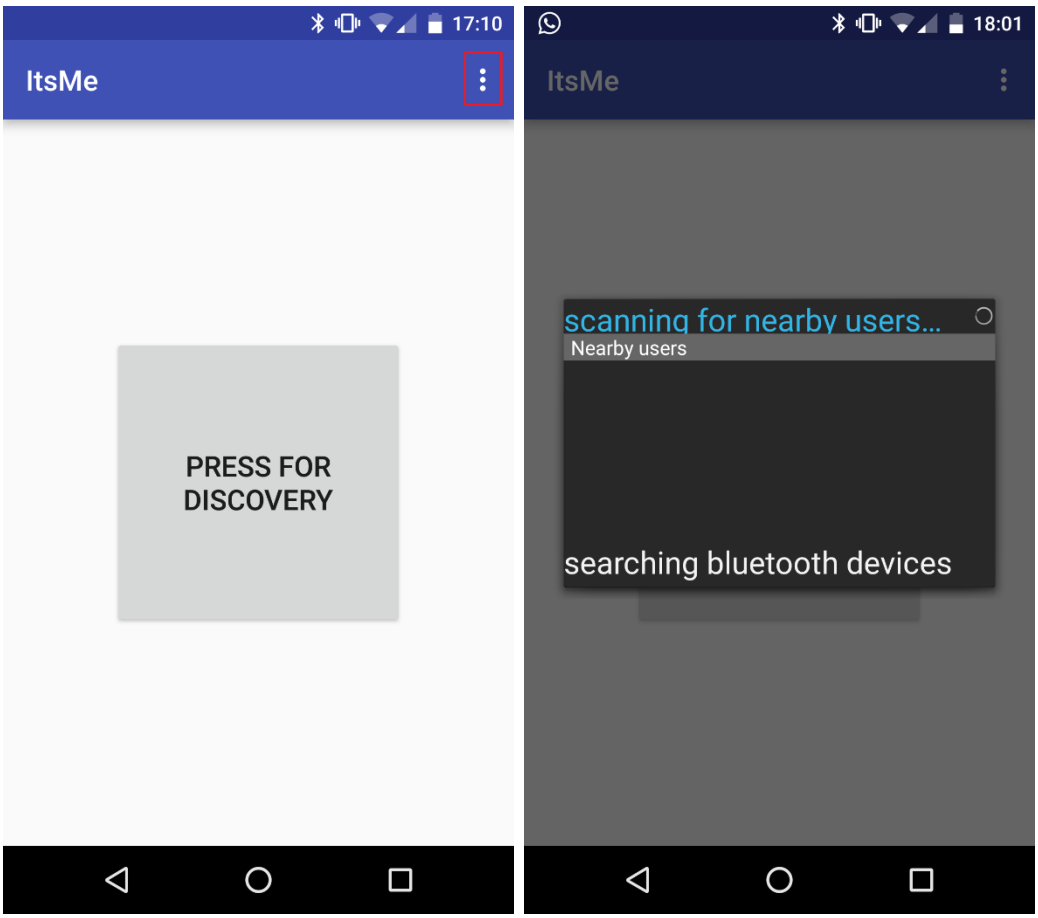

Figura 18: Pantalla principal de la aplicación y búsqueda de usuarios

Si no se encuentran usuarios cercanos, la aplicación avisará de ello. Si se encuentran, se podrá ver a todos ellos listados por orden de compatibilidad. Cuando se pulsa sobre uno de ellos, automáticamente se envía una petición de chat. Si el usuario acepta dicha petición, se pasa a la pantalla de chat (Figura 19).

Cuando se mantiene la aplicación abierta pueden llegar peticiones de chat de otros usuarios en cualquier momento (Figura 19). Si se acepta la petición, se pasa a la pantalla de chat.

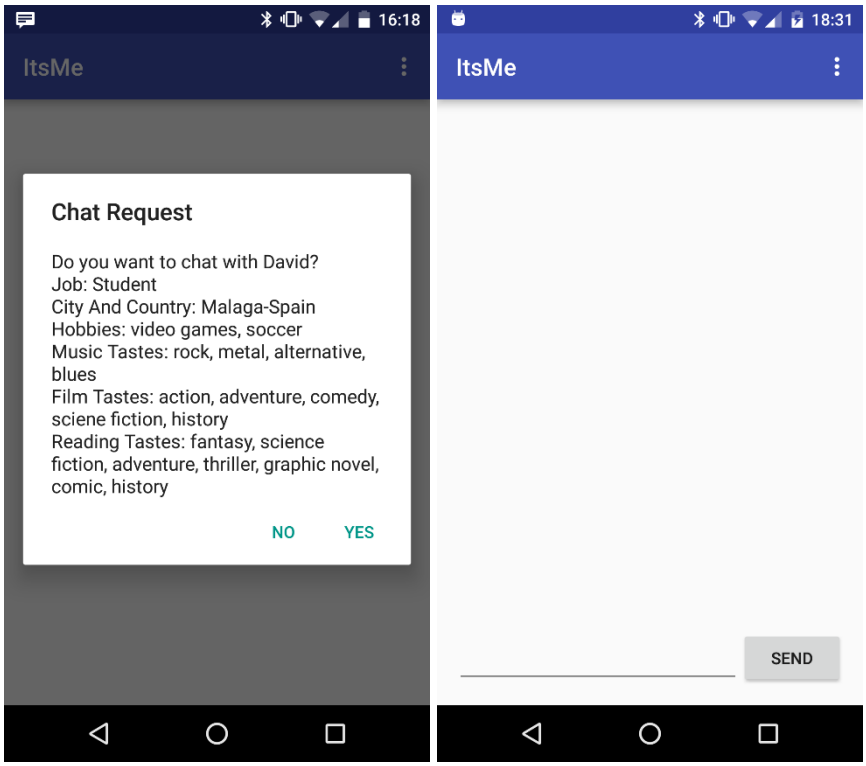

Figura 19: Petición de chat y pantalla de chat

Para terminar un chat, basta con pulsar el botón back del teléfono. Cuando se termina un chat, ya sea de esta manera, porque el otro usuario se desconecta o porque la conexión se pierde, la conversación se almacena y puede ser visualizada desde la opción de menú en la esquina de la pantalla principal (Figura 18).

B. Manual de instalación de la API web

Para poner en marcha la API en Windows no hace falta instalar ningún servidor ya que se aprovecha el paquete IIS integrado en el sistema. Es sencillo iniciar la API, aunque requiere instalar algunas herramientas.

- **Base de datos:** El gestor de base de datos *PostgreSQL* puede descargarse desde su página principal.

    - Enlace de descarga: http://www.enterprisedb.com/products/pgdownload.do#windows

    El archivo descargado es un instalador que incluye una guía durante todo el proceso. Una vez completada la instalación existen varias opciones para utilizar el gestor. En este caso procederemos a crear la base de datos a partir del archivo SQL facilitado. Para crearla puede hacerse desde la línea de comandos con el siguiente comando (Figura 20):

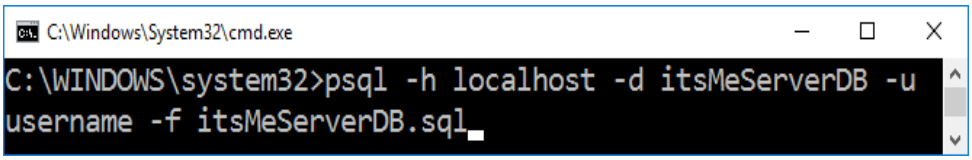

Figura 20: Comando de la base de datos

- .NET Core: Para instalar .NET Core, de la misma forma que la base de datos, puede descargarse desde su página principal. El archivo descargado también es un instalador que guía al usuario durante todo el proceso.

    - Enlace de descarga Windows x86: https://go.microsoft.com/fwlink/?LinkID=827525
    - Enlace de descarga Windows x64: https://go.microsoft.com/fwlink/?LinkID=827524

Una vez se ha instalado todo lo necesario se puede ejecutar la API desde la línea de comandos colocándose en la carpeta que contiene los ficheros de esta y ejecutando el siguiente comando:

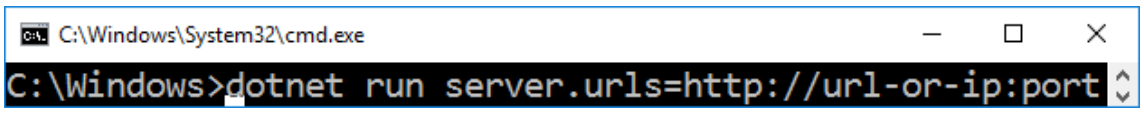

Figura 21: Comando para ejecutar la aplicación

Se ha configurado la aplicación con la opción de pasar como argumento la dirección y el puerto desde donde escuchará el servidor para facilitar la instalación de la API en cualquier máquina ejecutando la línea de la Figura 21.