UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

<Grado en Ingeniería Informática

en Ingeniería del Software>

Trabajo Fin de Grado

<Juego social para plataformas móviles con comunicaciones vía bluetooth>

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

<Grado en Ingeniería Informática

en Ingeniería del Software>

Trabajo Fin de Grado

<Juego social para plataformas móviles con comunicaciones vía bluetooth>

Autor: <Francisco Javier Reyes Mangas>

Tutor: <Juan Manuel Murillo Rodríguez>

Tabla de contenido

Índice de Ilustraciones

Índice de tablas

Resumen

La tecnología, en los últimos años, ha avanzado a pasos agigantados y lo ha hecho de tal manera, que incluso ha cambiado la concepción que tenemos del mundo de que nos rodea. Siempre ha sido así, pero gracias a una serie de acontecimientos o invenciones relativamente recientes, tales como la aparición de los smartphones o la idea de dotar a todo aparato de inteligencia, han hecho que todo este avance sea más cercano a las personas. Al fin y al cabo, el producto final de todo desarrollo está destinado a facilitar en mayor o menor medida la vida del ser humano de a pie. Siempre está presente la idea de aportar soluciones del día a día de las personas.

Además, ahora se concibe ese smartphone como un dispositivo que hace de todo: te da el tiempo, los resultados de tus deportes favoritos, información y contacto con tus amistades, posibilidades de entretenimiento con juegos… y se quiere también que todo sea sencillo de realizar (sin configuraciones complejas),

que la batería del dispositivo tienda al infinito y que los datos móviles contratados no se terminen rápidamente.

Este proyecto trata de entrar en el gap existente entre la tecnología y la persona, entendiendo por gap la poca transparencia que hay en ocasiones entre cierta tecnología y la forma en la que una persona debe usarla, y ofrecer una aplicación que lo resuelva en cierta manera, ayudándose de los últimos estándares relacionados con el Internet de las cosas, que están destinados fundamentalmente a las comunicaciones inalámbricas y abogan por el bajo consumo de energía.

Los objetivos de este proyecto son varios y están relacionados entre ellos. Principalmente se tiene como meta desarrollar una aplicación móvil para un sistema operativo ya extendido como lo es Android, que se comunique inalámbricamente exceptuando la conexión con redes externas y que consuma poca batería. Colateralmente, se pretende también investigar las distintas posibilidades de comunicación inalámbrica existentes actualmente para los dispositivos móviles, analizando y determinando cuál de ellas es la apropiada para este proyecto.

El desarrollo del proyecto se ha enfocado en diversas partes. La primera de ellas ha sido fundamentalmente de investigación sobre las posibilidades actuales existentes en los smartphones Android para comunicarse inalámbricamente y una vez se han comprendido las múltiples opciones se ha pasado a decidir el tipo de aplicación que se realizaría e implementarla.

Como resultado de todo esto, se ha creado la aplicación Shake It, un juego social multijugador que se centra en evitar configuraciones complejas características de los juegos que usan comunicación inalámbrica, minimizar la intervención del usuario salvo en las funciones que se requiera y evitar el gasto de batería excesivo, así como la conexión con redes externas. Todo ello, haciendo uso de una tecnología que facilita la consecución de estos requisitos.

A modo de conclusión y como experiencia realizando el proyecto, se ha conseguido que la aplicación citada sea funcional y estoy satisfecho con el trabajo realizado. Y aunque es mejorable y se pueden pulir varios detalles, ha servido para poder aplicar los conocimientos adquiridos en el grado a la hora de realizar un proyecto completo desde cero. Además, conocer la tecnología que se ha escogido para la realización de este proyecto abre puertas de cara al futuro, ya sea por el simple hecho de conocer una tecnología emergente o por la posibilidad de desarrollar otros proyectos que la usen.

1. Introducción

En este capítulo inicial del proyecto se tratará de contextualizar el proyecto aportando una visión sobre la tecnología actual, tanto en lo técnico como en lo que supone respecto de las personas que la usan.

En primer lugar, se hablará sobre la evolución de la tecnología sufrida en estos últimos años con respecto al mundo de la telefonía, relacionándolo acto seguido con el internet de las cosas y la forma en la que las personas perciben estos cambios, terminando con la indicación de algunos detalles sobre el sistema operativo con más presencia en España en lo que a los smartphones respecta: Android. En segundo lugar, se exponen las motivaciones que han llevado a la realización de este proyecto, así como los objetivos que se pretenden conseguir a la hora de realizarlo. Por último, se mostrará finalmente la aplicación a realizar para cubrir lo explicado en los apartados anteriores.

1.1 Evolución de la tecnología

A lo largo de los años, desde la primera aparición de un smartphone (allá por el año 2009) se ha vivido una revolución tecnológica cuanto menos; cambiando el concepto, lugar y forma en que realizamos diversas acciones o tareas gracias a estos dispositivos de diversos tamaños que nos facilitan el día a día. Tal vez al principio no fuera tan impactante, pero no tardó demasiado en implantarse la idea de tener uno de estos dispositivos, llegando hasta el punto en que hoy en día es más difícil encontrar a una persona que no posea uno de estos móviles a lo contrario.

Esto ha traído una serie de consecuencias directas a la evolución tan temprana que han sufrido estos aparatos, sobre todo en lo relativo a la tecnología que incorporan. Actualmente todo smartphone está dotado de módulos que permiten hacer fotos, conectarse a internet mediante Wifi o redes de datos móviles e incluso establecer comunicaciones con otros aparatos de forma inalámbrica, ya sea por Bluetooth o NFC, y así, otra serie de características más, que hacen que se conciba un móvil como una aparto inteligente. Y no solo respecto a la comunicación, sino también a su capacidad de cómputo o adquisición de información a través de los numerosos sensores que estos aparatos incorporan.

1.1.1 Internet de las cosas

Evolución similar se ha sufrido en otros ámbitos, no sólo en los smartphones, sino también en la idea o movimiento de querer hacer todo aparato inteligente, entendiendo por inteligente dotar a los aparatos que tenemos en nuestro entorno diario de los mecanismos necesarios para captar información y actuar por si mismos sin ayuda externa de una persona. Al final lo que se quiere conseguir es que los objetos nos sirvan a las personas, formando una amplia red de comunicaciones entre estos pequeños aparatos. Todo esto es lo que se conoce hoy en día como Internet de las cosas (IoT por sus siglas en inglés). Es ahora cuando más se hace eco de este concepto porque se ha llegado a un punto en el que somos capaces de desarrollar chips de procesamiento realmente

pequeños y estos chips, a pesar de su tamaño, son más potentes que sus predecesores y no solo eso, si no que consumen menos, permitiendo a un dispositivo aguantar una gran cantidad de tiempo sin necesidad de conectarse a una fuente de energía adicional.

Gracias a esta vertiente, protocolos de comunicaciones actuales se están adaptando a fin de que converjamos en un futuro a implementar toda esta tecnología en el mundo que nos rodea. Tanto el popular WiFi como el Bluetooth ya tienen estándares publicados y siendo utilizados para la consecución de todo esto. Estos estándares son WiFi Direct y Bluetooth LE o Smart. De igual manera, y posiblemente por todo lo relativo a esta idea, los smartphones desde no hace mucho tiempo ya incorporan chips que soportan ambos estándares.

1.1.2 Tecnología para inexpertos

Después de lo expresado en el apartado anterior, cabe mencionar otro aspecto que afecta directamente a las personas que usan la tecnología. Al fin y al cabo, el producto resultante de aplicar todas estas tecnologías, acaba siendo usado por personas no involucradas en ese proceso de creación. Nosotros entregamos ese producto, con su manual, esperando que efectivamente esa persona consiga usar el artefacto sin problema alguno.

A menudo nos encontramos con un gap entre a quién podemos denominar técnico y el usuario final, con efecto negativo directo con este último. Y es que al final las tecnologías tienen ciertos nombres o configuraciones complejas que no son admisibles por todo el mundo, llegando a frustrar al usuario o viéndose obligados a pedir ayuda a otra persona. Cierto es, que los llamados nativos digitales no tienen tanto problema con ello, pues su curva de aprendizaje es más bien leve con respecto al resto de las personas, pero no es lo ideal. Por poner un ejemplo, ¿cómo le explicas a alguien que es necesario que sincronice su dispositivo con otro para realizar tal cosa? O yéndonos incluso un paso más atrás, ¿qué es eso de la sincronía y por qué tengo que hacerlo?

Al final todos queremos que la tecnología sea accesible por cuantas más personas mejor sin excluir a nadie y se debe trabajar porque esto sea así. Es comprensiblemente complicado, hemos llegado a un punto el que existe demasiada distancia entre un extremo y otro, pero se debe luchar por ello ya sea haciendo que todo sea transparente hacia ese usuario final o facilitando el aprendizaje.

1.2 Android

Sin entrar demasiado en detalle, Android es un sistema operativo destinado a plataformas móviles. Desde que Google lo adquirió ha crecido como la espuma. Actualmente es el sistema operativo más distribuido por todo el mundo y especialmente en España es el más usado. Con toda esta aceptación es comprensible que sea un SO actualizado constantemente y acorde con respecto a las últimas tecnologías.

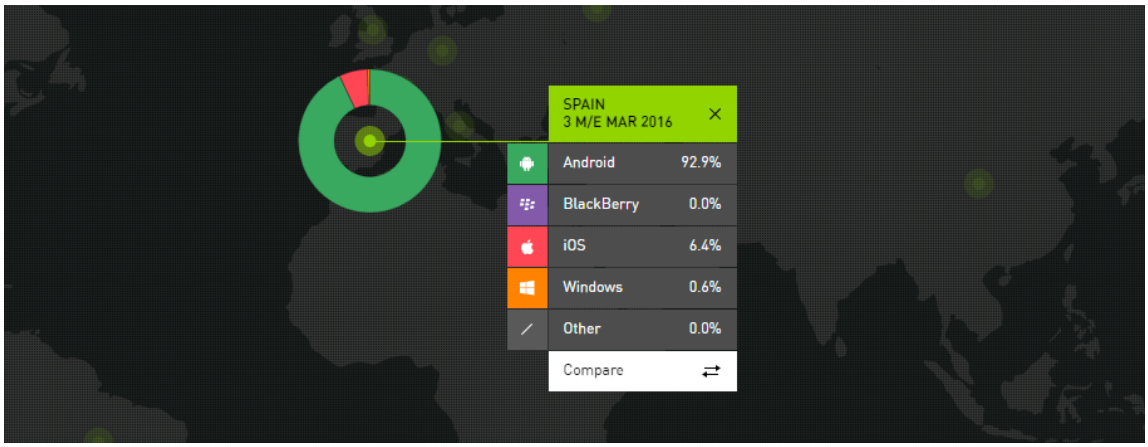

ILUSTRACIÓN 1: CUOTA DE MERCADO ANDROID. FUENTE: HTTP://WWW.KANTARWORLDPANEL.COM/GLOBAL/SMARTPHONE-OS-MARKET-SHARE/

Además, resaltar que Android soporta nativamente los nuevos estándares de la WiFi Alliance y Bluetooth SIG para el internet de las cosas.

1.3 Motivaciones y objetivos

Para entender el porqué del desarrollo de este proyecto se describen en los siguientes subapartados las motivaciones y los objetivos de dicho proyecto.

1.3.1 Motivaciones

Las motivaciones presentes para la realización de este proyecto son las siguientes:

1. Actualmente el mercado móvil está dominado, con una alta presencia, por el sistema operativo Android de Google y, desarrollar para esta plataforma te da, además de garantías de llegar a más público, formación que puede ser útil en un futuro.
2. De igual forma, con respecto a las tecnologías de comunicación inalámbrica, de las cuales algunas como el Bluetooth LE están muy presentes actualmente en todo lo que rodea al Internet de las cosas, ofrecen la apertura a un mercado emergente. Tanto lo están, que a partir de la penúltima versión de Android se da soporte nativo a estas tecnologías.
3. Entrando en la propia temática del juego que se propone, no existe ninguna igual y resulta interesante desarrollarlo por este motivo.
4. Otra de las motivaciones es descubrir si efectivamente se puede desarrollar un juego con una tecnología no diseñada específicamente para ello.
5. Hoy en día queremos hacer de todo con nuestro móvil, pero también queremos que la batería no sea un problema a pesar de hacer un uso intensivo del mismo.

Teniendo presentes dichas motivaciones se exponen a continuación los objetivos propuestos.

1.3.2 Objetivos

En este punto se tratará de detallar los objetivos que se pretenden alcanzar con el desarrollo de este trabajo fin de grado. A continuación, se listan dichos objetivos.

Para empezar, se define el objetivo general del proyecto. Dicho objetivo es tratar de desarrollar una aplicación móvil de un juego social que:

- Palie total o parcialmente los problemas expuestos en los apartados introductorios anteriores; por ejemplo, evitar la realización de pasos innecesarios por parte del usuario que éste no comprende por estar estrechamente relacionado con la tecnología.
- Use tecnologías de comunicación inalámbrica recientes.

Colateralmente se pretende también:

1. Explorar y valorar las posibilidades de comunicación inalámbrica que existen en el ámbito de los smartphones Android.
    a. Encontrar tecnologías aplicables al proyecto.
    b. Estudiar y comparar estas tecnologías.
    c. Determinar cuál es la tecnología óptima para el desarrollo del proyecto.
    d. Aplicar, en el desarrollo, nuevas tecnologías directamente relacionadas con el internet de las cosas.
2. Tratar de hacer una aplicación que intente minimizar al extremo la interacción por parte del usuario en las partes que no sea estrictamente necesario requerir de esa interacción, siendo ésta igualmente funcional.
    a. Bajo este precepto, comprender en mayor medida las necesidades de los usuarios cuando éstos se enfrentan a un sistema software.
3. Conseguir el desarrollo de un juego que no drene demasiada batería.

En lo relativo a lo académico:

1. Aplicar los conocimientos adquiridos en el grado a fin de hacerlos efectivos desarrollando este proyecto bajo procedimientos y fases distinguidas.

1.4 Aplicación propuesta

ILUSTRACIÓN 2: IMAGEN IDENTIFICATIVA DEL JUEGO

Shake It! (*Muévelo!* o *Agítalo!* en español) es un juego social cuyo objetivo principal está enfocado en realizar el mayor número de agitaciones (o *shakes*) haciendo de uso de un teléfono inteligente o smartphone capacitado para ello. Dicho juego posee la peculiaridad de ser completamente funcional sin necesidad de encontrarse conectado a una red que proporcione acceso a internet, siendo su modo de juego exclusivamente fuera de línea.

El tiempo que deberá estar agitando el móvil un jugador en la partida vendrá delimitado por un valor preestablecido en el momento de creación de la partida y que será transmitido entre dispositivos a lo largo del transcurso de la partida en cuestión.

2. Estado del arte

En este capítulo se va a tratar de mostrar, bajo un análisis de varias aplicaciones existentes en el mercado, cómo funcionan, qué bondades o carencias tienen dichas aplicaciones, similitudes con la aplicación realizada en este proyecto, con el fin de ayudar y mejorar la comprensión de la idea plasmada en el proyecto realizado.

Si bien han sido muchos los avances en diversos campos, se requiere de especial atención en los que han surgido en el área de la comunicación entre dispositivos, pues es la característica principal que abordamos con la aplicación que se va a realizar. Así pues, de este modo, y debido a la naturaleza del proyecto en cuestión, centro la investigación de mercado en el abanico de aplicaciones similares a la realizada, ya no por la temática del juego en cuestión, sino más bien por los modos de juego que estas ofrecen. Esto se traduce en la forma que estas aplicaciones o juegos se comunican para conseguir una experiencia

positiva en su cometido, siendo este distinto en cada una de las variopintas aplicaciones que existen en este ámbito.

Por lo tanto, es comprensible categorizar en cierta forma una serie de aplicaciones que compartan hábitos de uso y tecnologías que se aplican para establecer una comunicación efectiva entre dispositivos. El marco donde se engloban es el de los juegos, concretamente los que entran dentro del perfil de juegos multijugador, excluyendo los de pantalla dividida o de modo de juego por turnos en un mismo dispositivo, ya que para asemejarse a la aplicación que se ha realizado requiere que cada jugador participe desde su móvil y se comunique con otros de alguna manera. Aunque de igual forma, se tomarán como referencia dichas aplicaciones para resaltar sus bondades y sus defectos.

Teniendo en cuento lo anteriormente dicho, basándonos en el tipo de tecnología usada para la comunicación, algunas de estas categorías podrían ser las siguientes: juegos multijugador, uso de Bluetooth; juegos multijugador, uso de WiFi; juegos multijugador, uso de WiFi Direct o Wifi P2P.

2.1 Juego con múltiples modos de juego

El primer ejemplo que se muestra, un juego de nombre *Sea Battle 2*, se corresponde con el mítico juego de hundir la flota y como se puede apreciar en la figura que se muestra a continuación proporciona diversas formas de jugar una partida, desde un único jugador hasta juego online. Haremos hincapié en los módulos que están directamente relacionados e involucrados con la conectividad, estos son el modo de juego mediante bluetooth y el modo de juego online. Este último modo de juego hace uso de una conexión WiFi.

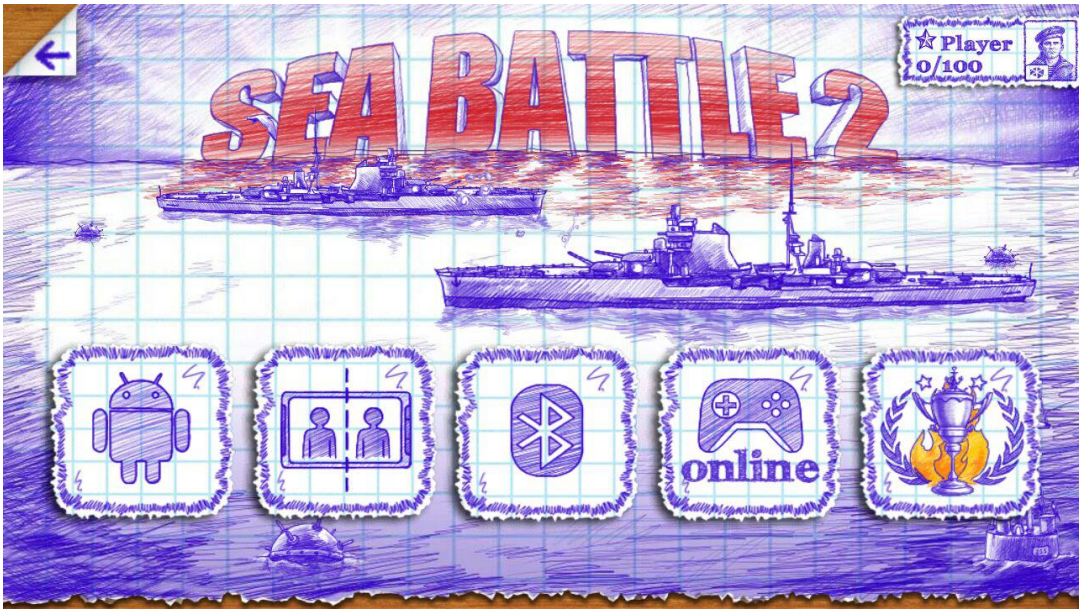

ILUSTRACIÓN 3: PANTALLA PRINCIPAL JUEGO SEA BATTLE 2

Empezando por el modo bluetooth, bien es sabido que debemos tener un conocimiento preconcebido del hecho de que, para poder realizar cualquier tipo de actividad, ya sea pasarse archivos entre dos teléfonos móviles o jugar a un juego como resulta en este caso, utilizando bluetooth se requiere de una sincronización previa de dispositivos si se quiere llevar acabo lo anteriormente citado.

Siguiendo con el ejemplo escogido, si no sincronizamos nuestros dispositivos desde el panel apropiado para ello, nos encontraremos ante la situación expuesta en la imagen que se muestra a continuación.

ILUSTRACIÓN 4: PANTALLA DE ESPERA JUEGO SEA BATTLE 2

Esta imagen se corresponde con el creador de la partida, que se encuentra esperando por un contrincante el cual nunca va a aparecer por la razón que ya hemos mencionado. De igual forma, al no estar sincronizados, el dispositivo que busca una partida tampoco la hallará, como se puede apreciar en la ilustración número tres.

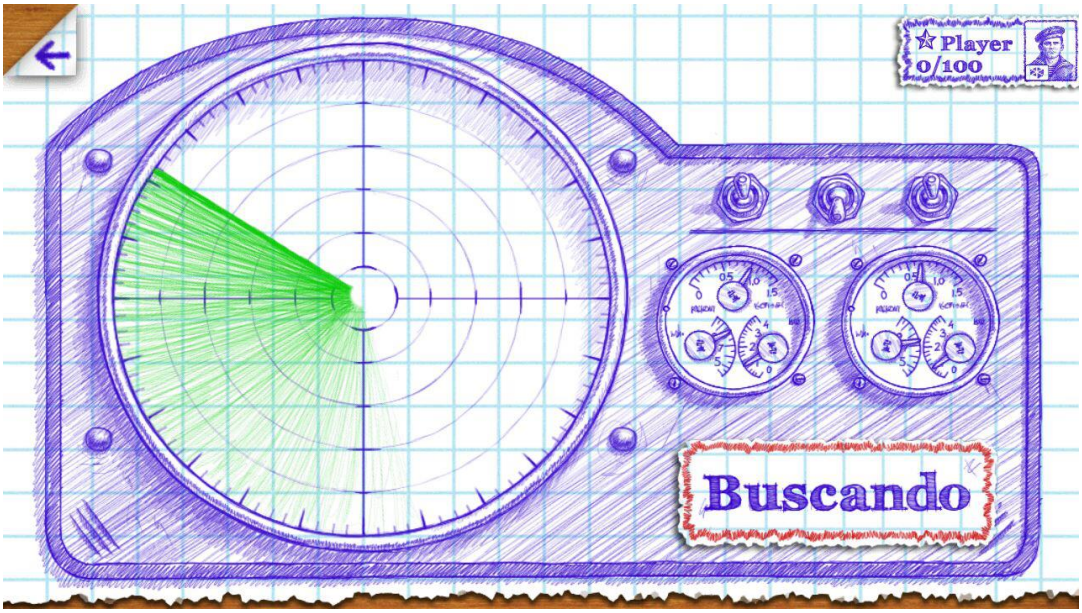

ILUSTRACIÓN 5: PANTALLA DE BÚSQUEDA DE JUGADORES

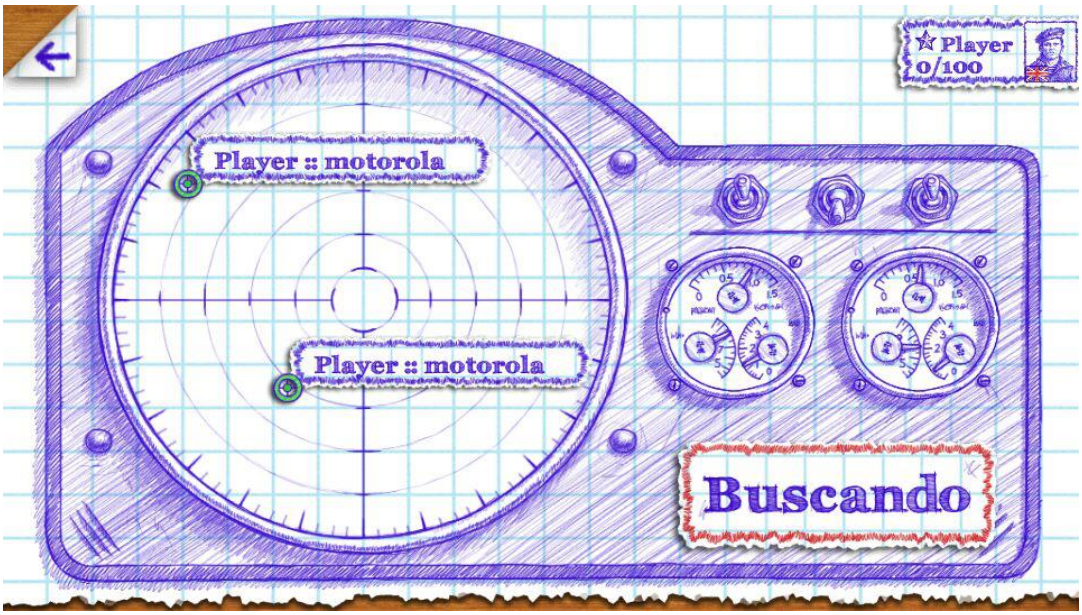

ILUSTRACIÓN 6: PANTALLA DE BÚSQUEDA CON JUGADORES ENCONTRADOS

Acto seguido, suponiendo que sabemos la razón de por qué no funciona sincronizaríamos nuestros dispositivos y volveríamos a intentar crear una partida para poder jugar finalmente, siempre que no surja ningún problema a la hora de

enlazar a los dispositivos. Y se está poniendo un ejemplo que el que solamente se involucran dos jugadores. ¿Qué pasaría si son tres o incluso más? ¿Tenemos que sincronizar todos los dispositivos? No se aplica para este juego, pero se podría requerir para otro perfectamente si se quiere conseguir un juego multijugador exento de la utilización de infraestructuras de redes externas al lugar donde se juega.

Si bien en esta aplicación se pueden apreciar las desventajas evidentes de usar bluetooth a la hora de desarrollar un juego, no se asemeja del todo con el tipo de juego que se realiza en este proyecto. La razón es que en este juego en concreto los jugadores pueden jugar simultáneamente y en el que aquí se desarrolla no; es un juego que funciona basándose en turnos.

2.1.1.1 Aplicación con un número amplio de jugadores

Otra de las características a tener en cuenta para facilitar la compresión del proyecto realizado tiene que ver con lo relativo al número de jugadores que un juego permite.

Después de una exhaustiva búsqueda uno de los juegos que cumple con este perfil se llama "*Chain Reaction*". Esta aplicación permite partidas de un amplio número de jugadores, ya sea desde un mismo dispositivo o multijugador online. Cabe destacar el número de jugadores implicados en la partida porque es raro el juego que admita tantas personas; lo habitual suele ser multijugador en los que intervienen solamente dos jugadores.

En la siguiente imagen se puede ver una captura del juego en cuestión en ejecución, donde se puede ver perfectamente una partida de ocho jugadores, lo cual es una cualidad destacable.

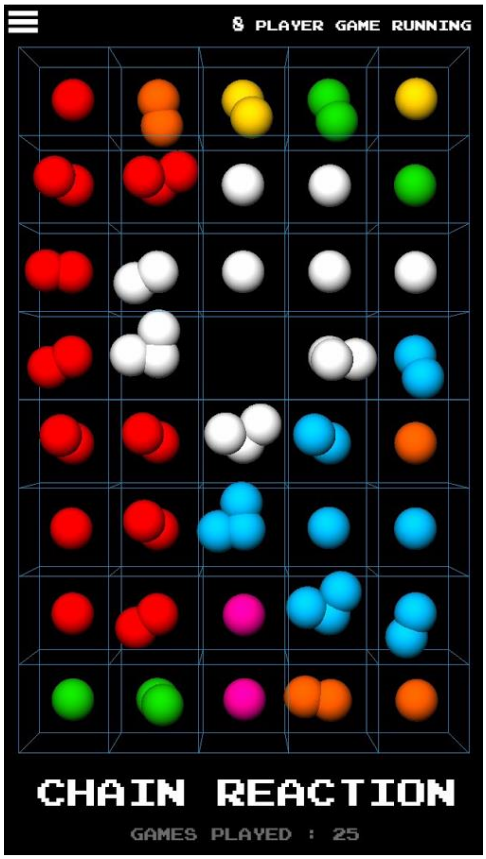

ILUSTRACIÓN 7: JUEGO CHAIN REACTION EN ACCIÓN

No lo es tanto que el multijugador que posee este juego sea online. En otras palabras, el juego requiere una conexión wifi o de datos efectiva para poder establecer contacto con otros jugadores a la hora de formar una partida. Como ya se ha mencionado, esto requiere de una infraestructura extra para poder funcionar, sin mencionar que no siempre podemos tener acceso; todo depende del contexto en el que nos encontremos.

Es una clara desventaja en el entorno en el que nos queremos situar. Si nos encontramos en una situación en la que un grupo de amigos están reunidos y quieren jugar, no deberíamos requerir de conectarnos a una red externa cuando nuestros dispositivos actualmente tienen las características necesarias para poder prescindir de esa infraestructura.

2.2 Indagando en la temática del juego que se desarrolla

Shake It!, tal como su nombre indica, tiene una temática definida fácil de intuir, que es la de agitar el terminal. Esto que nosotros consideramos como agitación es posible gracias a un sensor que se equipa actualmente en la mayoría de los smartphones existentes en el mercado. Este sensor lo conocemos como acelerómetro y puede ser objeto de múltiples aplicaciones. La nuestra en cuestión, es la de contar el número de veces que se agita un teléfono móvil.

Siendo tan específica esta funcionalidad que tiene la aplicación, se estudiarán aplicaciones que usen este sensor en mayor o menor medida, pues es muy complicado encontrar más de una aplicación con funcionalidad completa que tenga esta misma temática aquí tratada. Casualmente, aunque no está enfocada a ser una aplicación que ofrezca un juego multijugador, existe una aplicación en el Google Play Store[1] que posee exactamente esta funcionalidad, la de contar shakes o agitaciones.

El nombre de la aplicación citada no es otro que Phone Shake Counter y se muestra a continuación.

[1] Google Play Store: Tienda de aplicaciones desarrollado por Google Inc. que permite la descarga de múltiples aplicaciones.

ILUSTRACIÓN 8: PANTALLA PRINCIPAL JUEGO PHONE SHAKE COUNTER

ILUSTRACIÓN 9: PANTALLA DEL JUEGO EN ACCIÓN

Como se puede ver, es una aplicación simple, que hace lo que promete. Realmente no se puede saber más que el sensor que usa la aplicación para contar los shakes, ya que es el mecanismo que ofrece Android para detectar los movimientos del móvil.

De esta aplicación se puede resaltar sobre todo su interfaz simple. La manera en que muestra el número de agitaciones que se va consiguiendo durante el juego es clara y con la visibilidad perfecta. Además, de igual forma, nos muestra una cuenta atrás para saber el tiempo que le queda al usuario mientras agita el móvil.

Hay que destacar también los inconvenientes o carencias de esta aplicación. No se pueden apreciar si no la instalas en tu dispositivo, pero se tratará de explicarla sobre papel.

Esta aplicación cuenta una agitación cada vez que se hace un simple movimiento con el móvil, sea cual sea la dirección y sentido; es demasiado sensible al movimiento. Y esto, por definición, no es una agitación. Una agitación se entiende como una secuencia de movimientos repetidas de un lado a otro con relativas fuerza y rapidez. Es bueno saberlo de cara al desarrollo de nuestro juego, pues es un detalle a tener en cuenta.

2.3 Aplicaciones con la misma tecnología del proyecto

Actualmente las aplicaciones existentes que usen la misma tecnología empleada en este proyecto siguen una línea de utilidad similar y no es otra que actuar en consecuencia de la captación de información de balizas o beacons[2] propietarios. Estas balizas se sitúan en los locales de las empresas, ya sean restaurantes, tiendas u otros y emiten periódicamente cierta información que es captada por la aplicación propietaria que esté instalada en el smartphone de la persona que se

[2] Beacon: También conocido como baliza en español, es un aparato de un tamaño relativamente pequeño que emite señales periódicas con información preestablecida. Su característica principal es que puede durar mucho tiempo con una pila de botón convencional. Usan la tecnología Bluetooth LE para emitir.

acerque al establecimiento. El uso convencional es el de mostrar descuentos u ofertas especiales cuando estés cerca del sitio en cuestión.

Una aplicación de ejemplo para lo mencionado es la aplicación de McDonalds, al menos en algunas regiones, tienes balizas de este tipo que la aplicación lee y muestra ofertas o cupones a sus usuarios. Así lo confirmamos consultado una lista de aplicaciones que mantiene la librería usada en su página web.

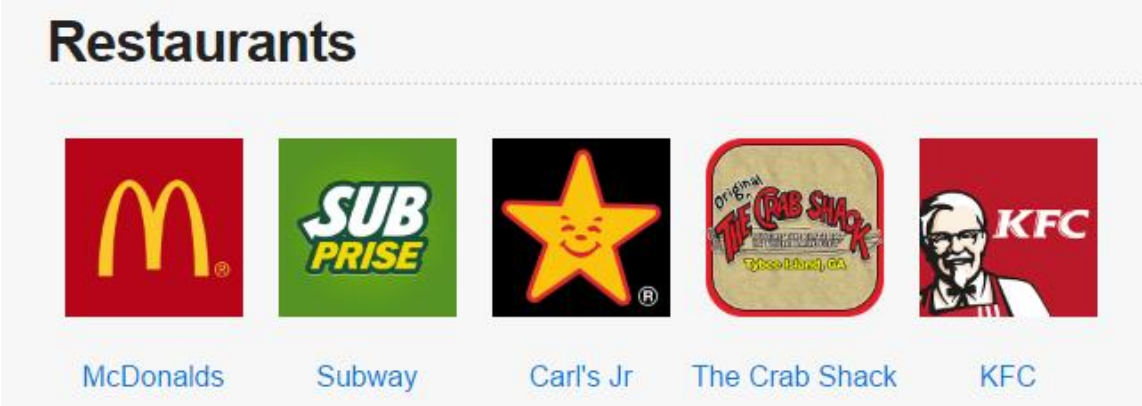

ILUSTRACIÓN 10: APPS USANDO ALTBEACON

Un ejemplo del funcionamiento está disponible a partir del minuto 1:16 en el siguiente video: https://www.youtube.com/watch?v=xab9YnhDnRE

3. Análisis y diseño

En este capítulo se encuentran las fases de análisis y diseño, donde se tratará de dar una visión inicial de este proyecto software, definiendo asimismo los requisitos, esquemas, flujos de interacción necesarios para especificar claramente lo que se quiere construir.

3.1 Requisitos

Primeramente, se definen los requisitos, pues es a partir de aquí donde se concibe el proyecto en sí y se dan a conocer las principales funcionalidades que tendrá la aplicación, así como una serie de restricciones que al final formal la aplicación mediante los requisitos funcionales y no funcionales.

3.1.1 Funcionales

En este apartado se enumeran y describen los requisitos funcionales del sistema desarrollado, que muestran la interacción entre el sistema y el entorno que le rodea además de servicios o funciones que debe proveer el sistema creado.

Los requisitos funcionales se encuentran descritos en la tabla que se expone a continuación.

TABLA REQUISITOS FUNCIONALES 1

Identificador	Descripción
RF1	Se debe ofrecer un acceso al sistema mediante la introducción de datos. Estos datos serán relevantes para la identificación de un usuario o de una partida.
RF2	Se debe introducir un nombre de usuario para poder unirse a una partida.
RF3	Se debe introducir un nombre de partida para poder crear dicha partida.
RF4	Los parámetros de configuración de la partida deberán ser aleatorios, así como el establecimiento de los turnos de juego.
RF5	Se debe ofrecer la posibilidad de crear o unirse a una partida ya creada.
RF6	La aplicación debe estar preparada para soportar un número indefinido de jugadores en la partida.
RF7	Si el usuario decide unirse a una partida, es de su elección unirse a la partida que guste, por lo que se deberá ofrecer esa posibilidad.
RF8	La aplicación será una aplicación destinada a dispositivos móviles.

RF9	La aplicación será dotada de un módulo capaz de contar el número de agitaciones que un usuario haga con el dispositivo móvil.
RF10	La aplicación debe informar al usuario que esté en una partida el momento en el que sea su turno.
RF11	De igual manera, se deberá informar al usuario que esté en una partida del momento el que ha terminado su turno.
RF12	La aplicación proveerá un mecanismo para aplicar una cuenta regresiva en el momento en que cada uno de los jugadores esté jugando su partida.
RF13	El usuario, una vez sea consciente de que es su turno, tendrá la potestad de decidir cuándo empezar su turno con algún mecanismo que permita hacerlo.
RF14	La aplicación, cuando el usuario se encuentre jugando su partida, proporcionará información visual de la cuenta regresiva asociada.
RF15	El usuario, dentro de su partida, debe saber en todo momento el número de agitaciones que lleva conseguidas.
RF16	La aplicación permitirá al usuario dejar la partida si ésta ya ha empezado.
RF17	El sistema debe mantener y registrar la información de la partida durante el transcurso de esta.
RF18	El sistema debe comunicar la información de la partida al resto de dispositivos involucrados en la misma.

3.1.2 No funcionales

Los siguientes requisitos no funcionales indican las restricciones de los servicios y funciones que ofrecerá el sistema. Se han dividido en distintas categorías según la naturaleza o ámbito de cada uno de ellos.

3.1.2.1 Interfaz

Todo lo relativo a la interfaz, lo relativo a las pantallas con que el usuario interactúa deberá cumplir con lo siguiente:

Identificador	Descripción
RNFI1	La interacción del usuario con la aplicación debe ser la esencial para el funcionamiento de la misma, es decir, debe intervenir solo en los casos necesarios.
RNFI2	A fin de mantener el foco de atención del usuario, la interfaz deberá ser inmersiva, para mantener en todo momento al usuario en el contexto de la aplicación.

3.1.2.2 Usabilidad

Bajo el nombre de usabilidad, se exponen los requisitos no funcionales relativos a la facilidad del usuario para navegar por la aplicación.

Identificador	Descripción
RNFU1	El sistema debe mostrar mensajes de error que permitan al usuario identificar el error que ha cometido o por qué no puede realizar una determinada tarea.
RNFU2	El sistema debe informar al usuario en el estado en el que se encuentra en cada momento que se realiza una acción.

3.1.2.3 Hardware

En lo que concierne al hardware, se han establecido las siguientes restricciones:

Identificador	Descripción
RNFH1	El dispositivo donde se ejecute la aplicación debe tener un chip Bluetooth que sea compatible con la especificación Bluetooth Low Energy o BLE.

RNFH2	El dispositivo debe estar dotado de un sensor, concretamente un acelerómetro.

3.1.2.4 Software

Respecto al software, se refleja lo siguiente:

Identificador	Descripción
RNFS1	La aplicación debe ser desarrollada para versiones superiores a Android 4.4 KitKat, es decir, desde Android 5.0 Lollipop.
RNFS2	Se utilizará el SDK de Android para desarrollar la aplicación.
RNFS3	El entorno de desarrollo (IDE) debe ser Android Studio.
RNFS4	El lenguaje de programación utilizado deberá ser Java.

3.1.2.5 Desempeño

En cuanto al desempeño de la aplicación:

Identificador	Descripción
RNFD1	El consumo de batería debe ser el mínimo posible.
RNFD2	El sistema debe tener un tiempo de respuesta aceptable.

3.1.2.6 Seguridad

Respecto de la seguridad del sistema frente a ataques o robo de información.

Identificador	Descripción
RNFS1	Los datos que la aplicación intercambie no deben contener información sensible.

3.2 Casos de uso

Bajo este apartado se recoge el diagrama de casos de uso con el que, de forma gráfica, se definen las acciones que desempeñan los usuarios o jugadores a la hora de interaccionar con el sistema en cuestión.

En primer lugar, se expone una breve descripción de los actores representados para esta ocasión y, posteriormente, se representa el diagrama de casos de uso en su totalidad, así como una descripción detallada del mismo.

3.2.1 Descripción de los actores

Si bien es relativamente sencillo comprender los actores con los interacciona nuestro sistema, se hace una breve descripción a continuación para ayudar a entenderlos un poco mejor.

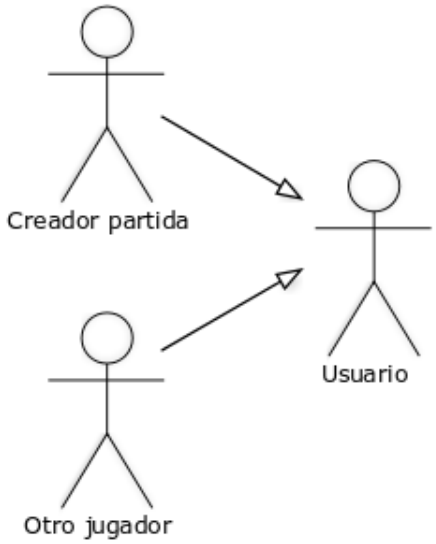

ILUSTRACIÓN 11: ACTORES CASOS DE USO 1

- **Usuario (creador de partida)**: Respecto del sistema actúa como una persona normal que usa la aplicación adquiriendo la pequeña peculiaridad que es la de tener el rol de creación de partida, lo cual hará que intervenga en algunas acciones diferentes.

- **Usuario (otro jugador)**: Del mismo modo, este actor adquiere una pequeña diferenciación respecto del otro, que no es otra que la de unirse a una partida en lugar de crearla, cambiando así el modo de interactuar en algunas partes del sistema.

3.2.2 Diagrama y descripción de los casos de uso

Con la intención de facilitar la comprensión de los casos de uso mostrados en Ilustración 12: casos de uso 1, se ofrece a continuación una descripción más detallada de los mismos, incluyendo los flujos que engloban dichos casos de uso. Para ello, se utilizan unas plantillas que permiten ver la descripción de forma más ordenada.

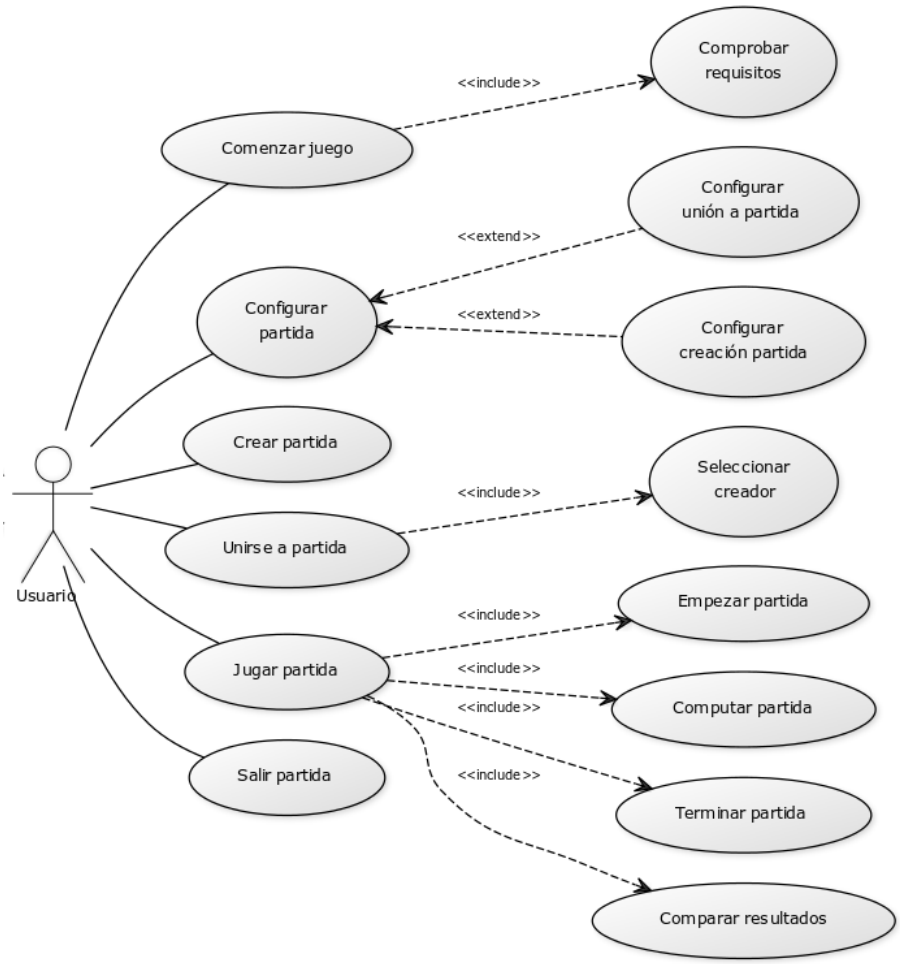

ILUSTRACIÓN 12: CASOS DE USO 1

Nombre	*Comenzar partida*
Descripción	Permite al usuario iniciar el juego.
Actores	Usuario (Creador de partida), Usuario (Otro jugador)
Precondición	-
Postcondición	El usuario ha iniciado el juego.
Flujo principal	1. El usuario solicita al sistema iniciar el juego. 2. Se realiza el caso de uso Comprobar requisitos. 3. El caso de uso Comprobar requisitos finaliza correctamente y el usuario inicia el juego sin problemas.
Flujo alternativo	3b. El caso de uso Comprobar requisitos finaliza sin éxito y el usuario no inicia el juego. Se cancela el caso de uso.

Nombre	*Comprobar requisitos*
Descripción	Permite comprobar los requisitos para la inicialización del juego.
Actores	Usuario (Creador de partida), Usuario (Otro jugador)
Precondición	El usuario ha efectuado el intento de inicio de juego.
Postcondición	El sistema ha comprobado los requisitos del juego y decide si se han cumplido o no.
Flujo principal	1. El sistema informa al usuario la confirmación de permisos que debe dar para continuar. 2. El usuario acepta los permisos y el sistema lo registra. 3. El sistema ha comprobado los requisitos.
Flujo alternativo	2b. El usuario deniega los permisos y el sistema lo registra.

TABLA 1: CASO DE USO CONFIGURAR PARTIDA

Nombre	*Configurar partida*
Descripción	Permite al usuario configurar una partida
Actores	Usuario (Creador de partida), Usuario (Otro jugador)
Precondición	El usuario ha solicitado configurar una partida
Postcondición	El usuario ha configurado una partida.
Flujo principal	1. El usuario solicita configurar una partida. 2. El sistema muestra el campo a rellenar correspondiente a la configuración de una partida. 3. El usuario rellena el campo en cuestión. 4. El sistema valida los campos.
Flujo alternativo	4a. La validación es correcta y, por tanto, se configura la partida correctamente. 4b. La validación es errónea y el sistema muestra por pantalla los errores cometidos.

TABLA 2: CASO DE USO CREAR PARTIDA

Nombre	*Crear partida*
Descripción	Permite al usuario jugar mediante la creación de una partida.
Actores	Usuario (Creador de partida)
Precondición	Se ha configurado la partida.
Postcondición	El usuario ha creado una partida.
Flujo principal	1. El usuario visualiza los jugadores que se han unido a su partida creada. 2. El usuario ejecuta la acción de comenzar la partida. 3. La partida se crea.

Flujo alternativo	-

TABLA 3: CASO DE USO UNIRSE A PARTIDA

Nombre	*Unirse a partida*
Descripción	Permite al usuario jugar uniéndose a una partida.
Actores	Usuario (Otro jugador)
Precondición	Se ha configurado una partida.
Postcondición	El usuario se ha unido a una partida.
Flujo principal	1. El usuario ha configurado una partida y quiere unirse a una partida. 2. Se realiza el caso de uso Seleccionar creador. 3. El caso de uso finaliza exitosamente y el usuario se une a una partida.
Flujo alternativo	3b. El caso de uso no termina correctamente y el usuario no se une a una partida.

TABLA 4: CASO DE USO SELECCIONAR CREADOR

Nombre	*Seleccionar creador*
Descripción	Permite al usuario jugar mediante la creación de una partida.
Actores	Usuario (Creador de partida)
Precondición	El usuario quiere unirse a una partida.
Postcondición	El usuario ha seleccionado un creador.
Flujo principal	1. El usuario quiere unirse a una partida. 2. El sistema informa de los creadores de partida que hay disponibles. 3. El usuario selecciona el creador de partida deseado. 4. El sistema registra la elección, lo comunica e informa al usuario de la selección.

	5. El usuario ha seleccionado un creador.
Flujo alternativo	3b. El usuario no selecciona un creador y el caso de uso se cancela.

TABLA 5: CASO DE USO JUGAR PARTIDA

Nombre	*Jugar partida*
Descripción	Permite al usuario jugar una partida
Actores	Usuario (creador de la partida), Usuario (otro jugador)
Precondición	Los jugadores han configurado y hay una partida en curso.
Postcondición	El usuario ha jugado una partida.
Flujo principal	1. Le llega el turno al jugador 2. Se realiza el caso de uso Empezar Partida. 3. Se realiza el caso de uso Computar Partida. 4. Se realiza el caso de uso Terminar Partida. 5. Se realiza el caso de uso Comparar Resultados. 6. El usuario ha jugado una partida.
Flujo alternativo	-

TABLA 6: CASO DE USO EMPEZAR PARTIDA

Nombre	*Empezar partida*
Descripción	Permite al usuario empezar una partida
Actores	Usuario (creador de la partida), Usuario (otro jugador)
Precondición	Se está en condiciones de empezar una partida.
Postcondición	El usuario empieza una partida
Flujo principal	1. El sistema informa al jugador de las características de la partida y del mecanismo oportuno para empezar la partida. 2. El usuario confirma el comienzo de partida.

	3. El usuario empieza una partida.
Flujo alternativo	2b. El usuario no confirma el comienzo de partida. 3b. Se cancela el caso de uso.

TABLA 7: CASO DE USO COMPUTAR PARTIDA

Nombre	*Computar partida*
Descripción	Permite al sistema computar movimientos del jugador.
Actores	Sistema.
Precondición	Se ha empezado una partida.
Postcondición	Se ha computado una partida.
Flujo principal	1. El usuario empieza la partida. 2. El sistema muestra un temporizador en pantalla. 3. El usuario agita el dispositivo. 4. El sistema registra los movimientos e informa del cómputo de los mismos. 5. El sistema indica la finalización del temporizador. 6. El usuario deja de agitar el dispositivo. 7. El sistema registra el número total de agitaciones. 8. Se ha computado una partida.
Flujo alternativo	-

TABLA 8: CASO DE USO TERMINAR PARTIDA

Nombre	*Terminar partida*
Descripción	Permite al usuario terminar la partida.
Actores	Usuario (Creador de partida)
Precondición	Se ha empezado y computado una partida
Postcondición	El usuario termina la partida.

Flujo principal	1. El usuario ha empezado y computado una partida. 2. El sistema muestra que ha terminado su turno. 3. El sistema muestra confirmación de terminación de partida. 4. El usuario ejecuta la acción de terminar la partida. 5. El usuario termina la partida.
Flujo alternativo	-

TABLA 9: CASO DE USO COMPARAR RESULTADOS

Nombre	*Comparar resultados*
Descripción	Permite al sistema comparar resultados
Actores	Sistema.
Precondición	Se ha jugado una partida.
Postcondición	Se han comparado resultados.
Flujo principal	1. Un jugador termina su partida. 2. El sistema muestra el resultado obtenido por pantalla. 3. El sistema compara resultados con jugador previo si lo hubiese y registra el ganador actual. 4. Se han comparado resultados.
Flujo alternativo	-

TABLA 10: CASO DE USO SALIR PARTIDA

Nombre	*Salir partida*
Descripción	Permite al usuario terminar la partida.
Actores	Usuario (Creador de partida)
Precondición	Usuario está en una partida

Postcondición	El usuario sale o no de la partida.
Flujo principal	1. El usuario está en una partida en curso. 2. El usuario quiere salir de la partida. 3. El usuario ejecuta la acción de salir de la partida. 4. El sistema muestra confirmación. 5. El usuario acepta y sale de la partida.
Flujo alternativo	5b. El usuario cancela y sigue en la partida. Se cancela el caso de uso.

3.3 Esquema preliminar

El siguiente esquema que se representa no pretende ser uno formal, sino más bien una serie de componentes entrelazados que definen una estructura que sigue el proyecto, declarando de forma abstracta la forma en la que funciona la aplicación. Es un concepto de partes diferenciadas para la consecución de la construcción del juego en su conjunto.

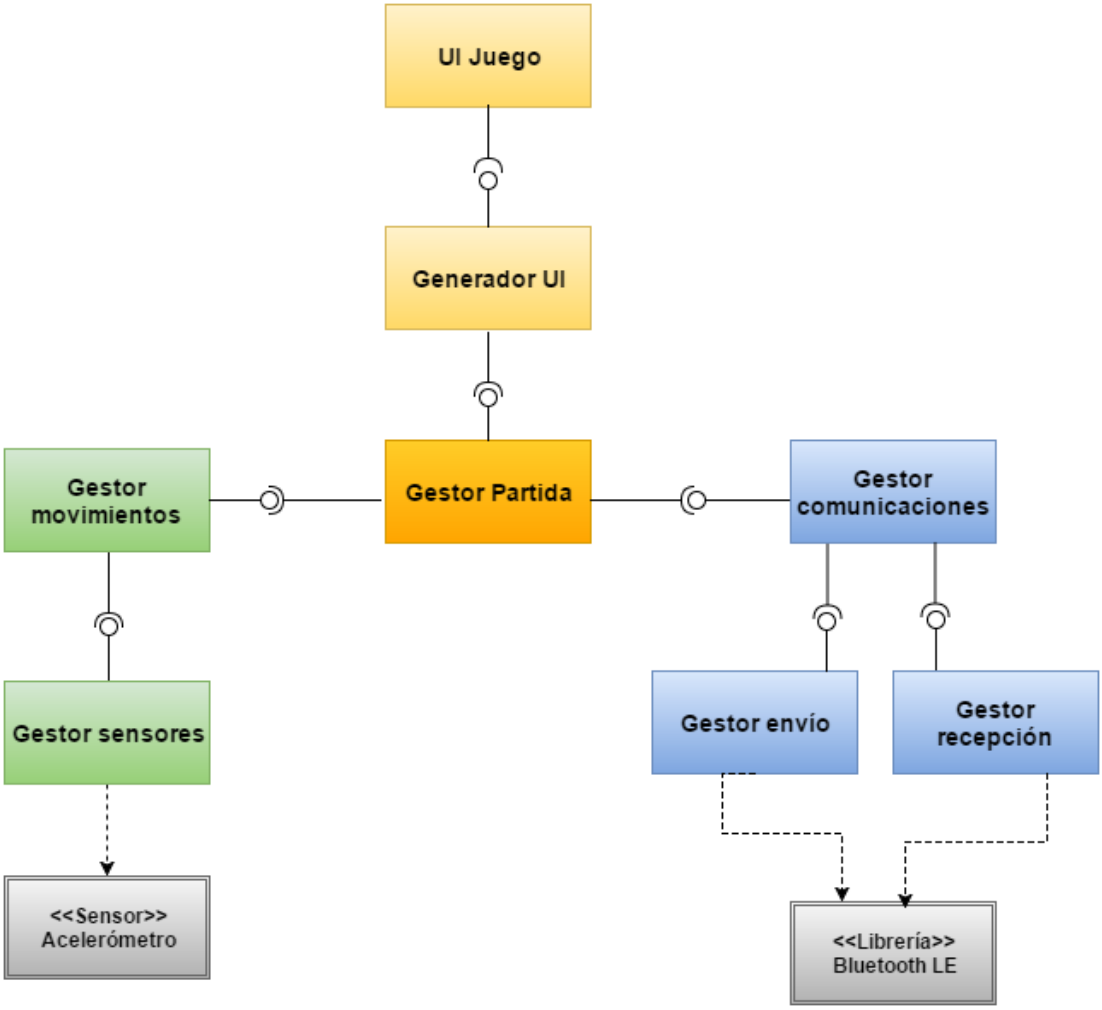

ILUSTRACIÓN 13: ESQUEMA COMPONENTES DE LA APLICACIÓN

En los siguientes subapartados se describe el esquema de forma más detallada y un posible flujo de ejecución entre los componentes.

3.3.1 Descripción del esquema y sus partes

En este apartado se tratará de explicar detalladamente cuál es la función de los componentes descritos en el esquema ilustrado anteriormente.

- Empezando por la parte superior, nos encontramos con el componente llamado ***UI Juego***. Se puede entender tal componente como la parte visible de la aplicación encargada en tu totalidad de mostrar todo referente visual. Es decir, es la parte formada por todas las pantallas que aparezcan en el juego.
- Pasamos entonces al componente denominado como ***Generador UI***, que se puede concebir como un paso intermedio entre la información que se trata en la aplicación y como se pasa esta información para que sea mostrada en la presentación visual. Se entiende pues, que de todo lo resultante de la partida se trata y se entrega para que sea visualizado.
- La parte encargada de todo lo referente a una partida que se juega estaría en el ***Gestor Partida***. Tanto los datos relativos a los jugadores que forman parte de la partida, como sus puntuaciones o el tiempo que dura cada ejecución de la partida de cada jugador estaría concebido dentro de este componente, así como la decisión de los turnos asignados para cada uno de esos jugadores.
- Dentro del ***Gestor de movimientos*** se encuentra toda la lógica relativa a tratar la información de los sensores a fin de que se decida cuando esos movimientos pueden ser computados como una agitación
- Por ***Gestor sensores*** se entiende toda la parte encargada de obtener la información que nos entrega el sensor, que será entregada al Gestor de movimientos para tratarla.
- Pasando a la parte de la comunicación tendríamos el ***Gestor de comunicaciones***, que mediante el ***Gestor de envío*** y el ***Gestor de recepción*** se harían cargo de todo el tema de transmisión de los datos de la partida.

- Por último, la comunicación haría uso de una librería para poder hacer efectiva esa comunicación por mediación del bluetooth. Y por la parte de sensores, se haría uso del acelerómetro, el cual entregaría las oscilaciones del smartphone a fin de saber su movimiento.

3.3.2 Descripción del flujo de ejecución de las aplicaciones respecto a acciones ejecutadas sobre la aplicación

Tomando como referencia el esquema realizado se muestra a continuación un posible flujo de ejecución a través de los componentes basándose en acciones hechas sobre la aplicación.

- **Comenzar juego:** Una vez seleccionado el juego, “UI Juego” se encarga de mostrar la interfaz correspondiente del mismo.
    - **Comenzar partida:** “UI Juego” muestra la pantalla de inicio de partida y da paso a comenzar la comunicación mediante el servicio ofrecido por el “Gestor de comunicación”, previa interacción con el “Gestor de partida”.
    - **Establecer partida:** En este punto los dispositivos involucrados deben comunicarse y establecer los parámetros en los que se basará la partida, por lo que entran en juego los componentes involucrados en esta comunicación, como pueden ser “Gestor de comunicaciones” y, por lo tanto, “Gestor de envío” y “Gestor recepción”, además del propio “Gestor de la partida”.
    - **Jugar partida:** Una vez se han establecido los turnos y los parámetros de la partida uno de los smartphones (gestionados por “Gestor de la partida”), al que le toque, se deberá empezar a tratar los movimientos que haga el jugador, para ello usamos “Gestor de movimientos” el cuál hará uso de “Cómputo de movimiento” y este, a su vez, se ayudará de la información entregada por “Gestor de sensores”.

- **Terminar partida:** Cuando se termine la partida, es decir, todos los jugadores involucrados hayan participado, se mostrará en las pantallas de los smartphones el jugador ganador y cuántos movimientos ha hecho éste, por lo que intervienen “Generador UI” y “Gestor comunicaciones”.

- **Terminar juego:** Se para toda comunicación, por lo que participa “Gestor de comunicaciones” y se cambia el contexto de la interfaz de la aplicación, por lo que intervienen “Generador UI”, “UI Juego” y “UI plataforma”.

3.4 Diseño de la interfaz

Investigando las distintas aplicaciones que son juegos, la mayoría coinciden en que siguen un diseño *Distraction Free*, que no es más que un diseño en el que la aplicación ocupa toda la pantalla, centrando la atención del usuario en su totalidad en la pantalla.

Se han hecho *mockups* previos para tener una concepción de cómo la aplicación ofrecerá las funciones al usuario mediante la interfaz. Además, estos mockups son interactivos, de modo que se puede simular el flujo de ejecución de la aplicación en cierta manera.

La herramienta utilizada para poder realizar estos mockups es gratuita, por lo que tiene un límite del número de pantallas que se pueden realizar. Por tanto, el flujo de la aplicación no se puede realizar de forma completa y ha sido dividido en varios más pequeños. La herramienta en cuestión es una herramienta online que se puede encontrar en la siguiente dirección url: https://www.fluidui.com/

3.4.1 Mockups

La pantalla principal de la aplicación se concibe como la presentación de la misma, en ella nos encontramos con el nombre de la aplicación, un logo y un botón para comenzar a jugar.

Si se diera el caso de hacer click en jugar y no tener el bluetooth activado, se ofrecería la posibilidad de activarlo desde la propia aplicación mostrando un diálogo emergente.

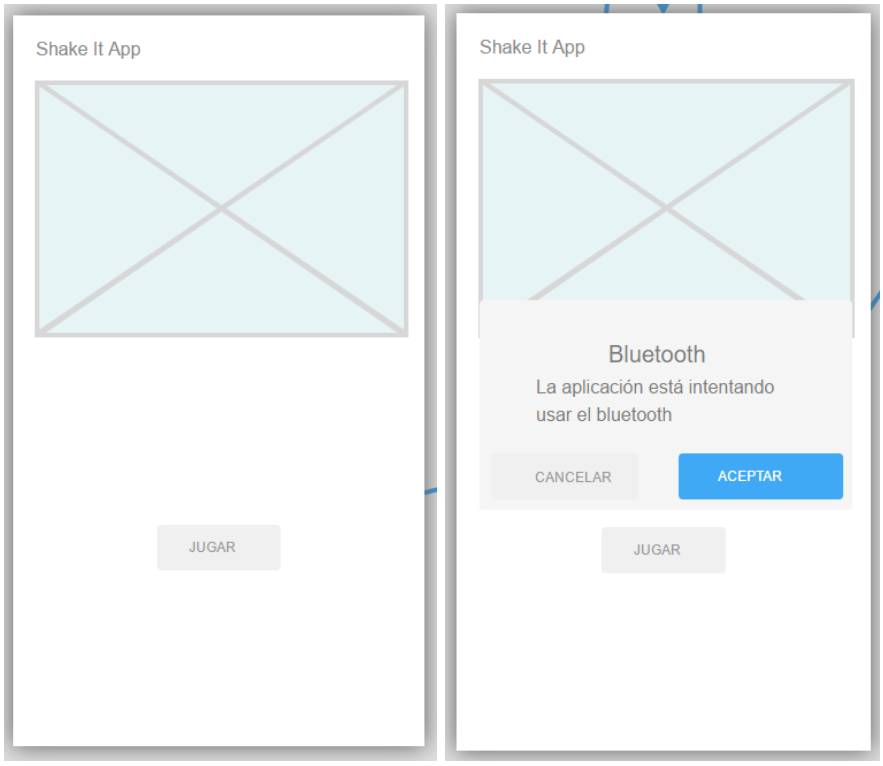

ILUSTRACIÓN 14: MOCKUPS PANTALLA PRINCIPAL

3.4.1.1 Configuración de la partida

A continuación, se muestran las pantallas donde se configurarían las partidas, tanto desde la parte del creador de la partida como de otro jugador que se una a una de las partidas creadas.

Previamente a la configuración de una de las opciones, se ofrece la posibilidad de elegir cada una de ellas.

- Existe un texto con la pregunta: ¿Qué quieres hacer?
- Se ofrece un botón para crear partida
- Se ofrece otro botón, pero para unirse a una partida

ILUSTRACIÓN 15: PANTALLA CONFIGURACIÓN PARTIDA

Según cual sea la opción elegida se mostrará un formulario u otro donde se podrá introducir:

- El nombre de la partida, si la opción escogida ha sido la de crear partida.
- El nombre de jugador, si la opción escogida ha sido la de unirse a una partida.

ILUSTRACIÓN 16: PANTALLAS INTRODUCCIÓN DATOS

A partir de aquí, habría que mostrar mensajes de error si el usuario intenta crear o unirse a una partida sin haber introducido ni un nombre de partida ni de jugador. Y si es posible indicar algún símbolo de color rojo, pues solemos asociarlo a que algo ha sucedido como no debería de haberlo hecho.

Las pantallas producto de esos errores podrían lucir de la siguiente manera, tal y como se muestra en la Ilustración 17: Pantalla configuración con errores

ILUSTRACIÓN 17: PANTALLA CONFIGURACIÓN CON ERRORES

Por último, si por el contrario se decide introducir un nombre en la caja de texto habría que mostrar un método de introducción de letras como puede ser el teclado.

ILUSTRACIÓN 18: PANTALLA CONFIGURACIÓN CON TECLADOS

La versión interactiva de estas pantallas se puede probar accediendo a través del siguiente enlace:

https://www.fluidui.com/editor/live/preview/p_lrP99hPeKBeTUBVTTQOvoOzJ5KRyW6hq.1465750727469

4. Implementación

En este capítulo se va a tratar toda la temática relativa a la implementación del proyecto, desde la tecnología aplicada para lograr la funcionalidad requerida hasta aspectos relevantes del código fuente escrito.

En primer lugar, se mostrarán las tecnologías candidatas a ser usadas para el desarrollo de este juego, indicando sus características principales y determinando por qué se ha escogido una o se han descartado otras.

En segundo lugar, se indican las principales librerías usadas en el proyecto, indicando cuál es la función para cada una de ellas.

Acto seguido, en tercer lugar, nos centraremos en la implementación del protocolo de comunicación de la aplicación, mostrando principalmente el formato de las balizas específicas para este proyecto.

Y, para terminar, nos centramos en la implementación específica de los casos de uso, así como en las clases relevantes del proyecto.

4.1 Tecnologías candidatas para usar en la comunicación

En lo que respecta a la comunicación entre dispositivos hay varias tecnologías aplicables al proyecto, pero sólo una cumple con los requisitos propuestos. En los próximos apartados se explican cada una de estas tecnologías, así como cada una de las razones que han llevado al descarte o aprobación de las mismas. Se comienza mostrando cada una ellas en orden de descarte para llegar finalmente a la tecnología escogida y facilitar el porqué de esta elección.

4.1.1 Wi-Fi

En primer lugar, se contempla el uso de Wi-Fi. Puede decirse que el Wi-Fi es una de las tecnologías más extendidas desde su aparición. Hoy en día hay adaptadores o módulos Wi-Fi en prácticamente cualquier aparato que requiera conectarse a una red bien sea para la obtención de datos o cualquier otro fin. Además, todos los smartphones existentes poseen este tipo de conectividad.

4.1.1.1 Estándares

Desde el primer momento que surgió el Wi-Fi, existe un estándar establecido por la Wi-Fi Alliance denominado 802.11 el cual define una serie de reglas y especificaciones en cuanto a algunas características de las redes Wi-Fi formadas. Algunos de estos parámetros tienen que ver con la velocidad de transmisión teóricas y prácticas, frecuencias de actuación, anchos de banda soportados, alcance de la señal, etc. Todas las diferentes especificaciones vienen diferenciadas por diferentes letras entre las que se encuentran la *b*, la *g*, la *n*, la *a* y una combinación de dos; la *ac, ad* y *ah*. De esta manera, los estándares actuales son los siguientes: *802.11b*, *802.11g*, *802.11n*, *802.11ª*, *802.11ac*, *802.11ad* y *802.11ah*.

Podemos visualizarlos en la tabla que se expone a continuación.

Estandar	Velocidad (teórica)	Velocidad (práctica)	Frecuencia	Ancho de banda	Alcance (metros)	Detalles	Año
802.11	2 Mbit/s	1 Mbit/s	2,4 Ghz	22 MHz	330		1997
802.11a	54 Mbit/s	22 Mbit/s	5,4 Ghz	20 MHz	390		1999
802.11b	11 Mbit/s	6 Mbit/s	2,4 Ghz	22 MHz	460		1999
802.11g	54 Mbit/s	22 Mbit/s	2,4 Ghz	20 MHz	460		2003
802.11n	600 Mbit/s	100 Mbit/s	2,4 Ghz y 5,4 Ghz	20/40 MHz	820	Disponible en la mayoria de los dispositivos modernos. Puede configurarse para usar solo 20 MHz de ancho y asi prevenir interferencias en una zona congestionada	2009
802.11ac	6.93 Gbps	100 Mbit/s	5,4 Ghz	80 o hasta 160 MHz		Nuevo estándar sin interferencia pero con menos alcance, aunque hay tecnologías que lo amplían. Más rendimiento y otras ventajas.	2013
802.11ad	7.13 Gbit/s	Hasta 6 Gbit/s	60 Ghz	2 MHz	300		2012
802.11ah			0.9 Ghz		1000	Wi-Fi HaLow	2016

ILUSTRACIÓN 19: ESTÁNDARES WI-FI

Los estándares más presentes actualmente en una amplia gama de dispositivos son el 802.11n e inferiores, llegando al 802.11ac en dispositivos de alta gama. Acotando y centrándonos en estos concluimos con que en la mayoría de los casos tendremos una velocidad práctica de 100 Mbits/s con un alcance de unos

820 metros y un ancho de banda comprensible, suficientes para la cantidad de datos que necesitamos en la aplicación.

4.1.2 Wi-Fi Direct

Wi-Fi Direct es una tecnología que soporta una conexión Wi-Fi entre dos dispositivos sin necesidad de un dispositivo intermedio. Es decir, se conectan directamente. La conexión efectuada puede ser uno a uno (dos dispositivos) o uno a muchos, formando un grupo. Al final lo que se forma es una red P2P sin requerir ninguna infraestructura por detrás que conecte a los dispositivos.

4.1.3 Bluetooth

Estamos ante otra tecnología de comunicación inalámbrica muy extendida y usada por múltiples dispositivos a lo largo del mundo. Su principal característica es que no requiere de dispositivos intermedios (como un modem o router) para establecer una conexión o crear una red local. Es el sucesor de la tecnología infrarrojos que antes portaban los móviles convencionales. Surge como alternativa a conexiones por cable haciendo uso de frecuencias de radio y es por eso que los aparatos que se conectan inalámbricamente usan esta tecnología.

Actualmente prolifera la versión 4.0, que es más un conglomerado de varias especificaciones, entre ellas el BLE.

4.1.4 Bluetooth LE

Bluetooth LE o BLE es una tecnología que nace para el internet de las cosas (Internet of Things o IoT). Tiene una premisa principal, que es la de necesitar el mínimo consumo de energía para operar. Esto da pie que un dispositivo que emite señales periódicas de pequeña información pueda funcionar durante meses con una pila de botón estándar, por ejemplo.

Está incluido en la especificación del Bluetooth 4.0 y es soportado nativamente por la mayoría de sistemas operativos, entre ellos Android.

4.1.4.1 BLE Advertising Mode

Junto con esta nueva especificación de bluetooth aparecen unos dispositivos llamados beacons que usan esta tecnología, BLE, introduciendo un nuevo concepto en el modo de comunicar datos; lo que se conoce como Advertising Mode (traducido al español podría ser algo así como modo de publicación o de emisión periódica de datos) y, por tanto, otra nueva forma de transmitir y recoger información del entorno que nos rodea.

Este mismo modo o la capacidad de realizar este tipo de transmisión está disponible en Android a partir de su versión 5.0 o Lollipop, en los dispositivos en los que el chip bluetooth lo permita o no esté limitado por una versión modificada del sistema operativo hecha por el fabricante de dicho dispositivo.

Usando este modo podemos transmitir pequeños paquetes de datos a un bajo coste energético dado que se ha diseñado para ello. Su funcionamiento es muy simple, uno de los dispositivos actúa como emisor de balizas o beacons que transportan cierta información y esa información es recibida por los dispositivos que se encuentran alrededor, de manera que pueden procesarla y actuar en consecuencia produciendo los efectos que nosotros deseemos.

4.1.5 Wi-Fi Direct vs. Bluetooth LE

Aunque ambas tecnologías nacen para trabajar en un mismo ámbito, el de posibilitar la conectividad fuera de línea, cada una tiene una serie de características que convienen más o menos según en qué situación nos encontremos.

Si lo que necesitamos es enviar o recibir una gran cantidad de datos entre dos dispositivos y, además, tener una comunicación bidireccional constante sin duda debemos inclinarnos hacia Wi-Fi Direct pues nos ofrece todas esas características. En cambio, si lo que más nos interesa es tener un consumo de energía realmente bajo y enviar o recibir pequeñas cantidades de datos debemos sin duda alguna decantarnos por la tecnología Bluetooth Low Energy porque es la más indicada para este tipo de comunicación.

En cuanto a rangos de distancia que permiten estas tecnologías ambas superan los 60 metros de distancia, lo cual es bastante extenso para el uso que se le puede dar en interiores o en comunicaciones de distancia relativamente corta.

4.1.6 Conclusión

Dado la naturaleza de este proyecto ambas tecnologías citadas anteriormente nos servirían para poder implementar la aplicación, pero dado que estamos buscando eficiencia, la mejor solución energética nos la proporciona Bluetooth LE y es precisamente la que se ha utilizado para ello. Además, requiere de especial atención que esta tecnología va a estar presente en los próximos años de una forma más habitual. Las balizas o beacons tomarán un papel más notable y así se puede confirmar con la próxima especificación que presenta el Bluetooth SIG, Bluetooth 5. En esta última versión se centran en impulsar la tecnología de los beacons especialmente, pero también se mejorará en cobertura y velocidad de transmisión.

4.2 Librerías de las que se hacen uso

En este apartado se citan y describen cada una de las librerías usadas para la implementación del proyecto con el fin de facilitar la comprensión de las mismas.

4.2.1 AltBeacon, Android Beacon Library

AltBeacon es una librería que nos permite detector beacons en Android. Y no solo detectarlos, también nos permite transmitir desde un dispositivo como si de un beacon se tratase.

La librería es completamente configurable para detectar múltiples variedades de beacons, incluso los que creamos nosotros de forma personalizada. A pesar de esto, Radius Networks, la empresa creadora de esta librearía, ha establecido por defecto el estándar AltBeacon y es el que se ha decidido usar en este proyecto.

4.2.1.1 Especificación

La especificación del protocolo que usa la librería AltBeacon define un formato concreto de mensajes que se emiten bajo "*beacon advertisements*". La información que viaja en ese mensaje emitido tiene que ver con la identificación del beacon emitido y con la distancia a la que se encuentra respecto del punto en el que se ha detectado dicho beacon. La responsabilidad de tratar la información o actuar en consecuencia de la lectura de la información pertinente recae directamente sobre el dispositivo que reciba esa información.

4.2.1.1.1 Objetivos del diseño

La especificación de AltBeacon ha sido diseñada bajo varios preceptos, expuestos a continuación:

1. Proporcionar la emisión de mensajes que puedan ser interceptados por dispositivos que actúen como escáneres.
2. Cumplir con la especificación Bluetooth 4.0 estableciendo un PDU [3] (Unidad de datos de protocolo) y estructuras de datos específicas.

[3] PDU. Protocol Data Unit. Unidad de datos de protocolo.

3. Conseguir que todas las partes interesadas no tengan restricciones con respecto de la implementación.
4. Permitir la implementación de características específicas que quiera proporcionar el proveedor, si es posible.

4.2.1.1.2 Formato del protocolo AltBeacon

El protocolo de AltBeacon hace uso de la estructura de datos para emisión tal y como se define en *Bluetooth Specification Version 4.0, Volume 6, Part B, Section 2.3 Advertising Channel PDU*.

La unidad de datos de protocolo de la especificación de AltBeacon está compuesta por varios campos de distintas longitudes y cada uno de estos campos tiene asignada una función en concreto.

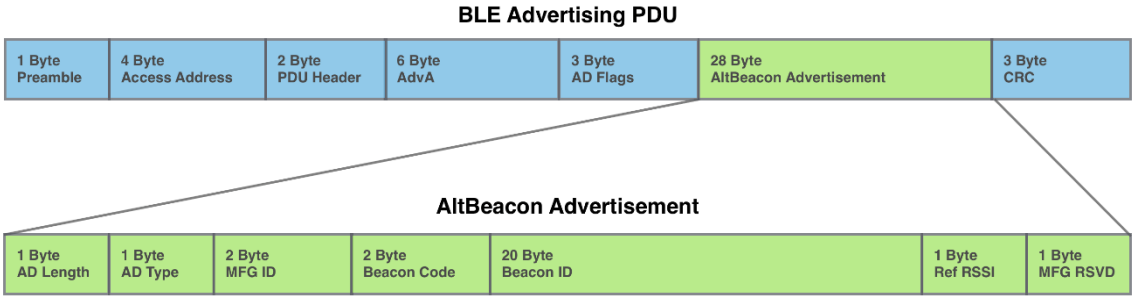

ILUSTRACIÓN 20. FUENTE: HTTPS://GITHUB.COM/ALTBEACON/SPEC

Dentro de la PDU definida, existen un total de 28 Bytes disponibles para poder transmitir balizas o beacons siguiendo la especificación propuesta y este campo se subdivide a su vez en más campos, cada uno con una función concreta, como sigue en la tabla Tabla 11: Campos especificación AltBeacon.

TABLA 11: CAMPOS ESPECIFICACIÓN ALTBEACON

Nombre del campo	Descripción	Valores aceptados
Longitud AD [Específico de MFG]	Longitud del tipo y de la parte de datos especifica de la estructura de datos del fabricante.	0x1B
Tipo de AD [Específico de MFG]	Tipo que representa la estructura de datos específica del fabricante.	0xFF
ID del fabricante	Código identificador del fabricante del dispositivo beacon.	La representación Little Endian del código del dispositivo del fabricante tal y como lo ha asignado el Bluetooth SIG.
Código del Beacon	Código de transmisión de AltBeacon.	La representación Big Endian del valor 0xBEAC
Beacon ID	Un valor de 20 Bytes para identificar al beacon emitido	La representación Big Endian del identificador del beacon. Como recomendación, los primeros 16 bytes deberán ser para identificar al emisor o transmisor y es resto de bytes puede subdividirse según convenga.
RSSI de referencia	Un valor de 1 Byte que representa la fuerza de la señal	Un valor de 1 Byte con signo desde 0 a -127.

	enviada por el transmisor.	
Reservado para el fabricante	Reservado para la inclusión de características especiales especificas del fabricante.	Un valor de 1 Byte desde 0x00 a 0xFF. La interpretación de este valor vendrá definida por el fabricante y debe ser evaluada en función del MFG ID o Id del fabricante.

4.2.1.2 Garantías

Al usar una librería de terceros podría surgir la idea de no tener un respaldo detrás en un futuro, pero desde su inicio es la librería más usada en Android para este tipo de comunicación, muchas empresas tales como McDonals, KFC, Coca-cola la usan por lo que no hay duda que está consolidada en el mercado. La lista con algunas de las más de 4000 aplicaciones que usan está librería se encuentran en el siguiente enlace: https://altbeacon.github.io/android-beacon-library/apps.html

4.2.1.3 Instalación

Si se usa Android Studio por defecto se usa el gestor de dependencias gradle y no tenemos más que añadir una simple línea para que compile:

```
compile 'org.altbeacon:android-beacon-library:2.7'
```

4.2.2 ButterKnife

ButterKnife es una librería que nos ayuda a inyectar vistas o recursos mediante anotaciones en el código Java que escribamos. Esto nos ayuda a agregar semántica y orden al código escrito, lo cual se ve reflejado positivamente en su legibilidad.

4.2.2.1 Ejemplos de uso

En Android cuando se accede a la vistas desde el código fuente tenemos que hacer un uso abusivo del método findViewById(). Con ButterKnife te evitas todo esto añadiendo una simple anotación al lado de la declaración del atributo.

```
@Bind(R.id.hosts_recycler_view)
RecyclerView mHostsRecyclerView;
@Bind(R.id.countdown_comunication_text)
TextView mCountdownText;
@Bind(R.id.status_comunication_text)
TextView mStatusText;
@Bind(R.id.winner_comunication_text)
TextView mWinnerText;
@Bind(R.id.start_game_button)
Button mStartGameButton;
@Bind(R.id.toolbar)
Toolbar mToolbar;
```

ILUSTRACIÓN 21: USO DE BUTTERKNIFE

4.2.2.2 Instalación

Si se usa Android Studio por defecto se usa el gestor de dependencias gradle y no tenemos más que añadir una simple línea para que compile:

compile 'com.jakewharton:butterknife:7.0.1'

4.2.3 Dexter

Dexter es una librería desarrollada por Karumi que tiene el objetivo principal de simplificar el proceso de petición de permisos para la versión 6.0 de Android (Marshmallow). Esta versión se caracteriza por pedir algunos permisos en tiempo real de ejecución de la aplicación, en lugar de declararlos explícitamente en el archivo manifest.xml del proyecto Android; esto implica que el usuario ahora puede permitir o denegar permisos una vez abra la aplicación y la comience a usarla.

4.2.3.1 Ejemplos de uso

La librería nos ofrece bastantes opciones, pero en lo que a pedir permisos se refiere tenemos dos formas muy sencillas de hacerlo.

Mediante el método *checkPermission()* podemos pedir uno o más permisos, todo depende del Listener que usemos; se ofrece un *PermissionListener()*, para pedir un único permiso o un *MultiplePermissionListener()*, para varios permisos a la vez.

A continuación, se muestra de forma gráfica como se expresaría en código la petición de varios permisos.

```
Dexter.checkPermissions(new MultiplePermissionsListener() {
    @Override public void onPermissionsChecked(MultiplePermissionsReport report) {/* ... */
    @Override public void onPermissionRationaleShouldBeShown(List<PermissionRequest> permis
}, Manifest.permission.ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION);
```

ILUSTRACIÓN 22: EJEMPLO DE USO DE DEXTER.

En esta captura se muestra como se piden permisos que tiene que ver con la localización del dispositivo, simplemente indicando su referencia de la clase Manifest.

Cabe mencionar que en toda activity u otro contexto hay que indicar una inicialización para poder usar Dexter; no es más que una línea que incluye Dexter.initialize(context);

4.2.3.2 Instalación

Para la instalación de esta librería, usando el gestor de dependencias gradle con Android Studio tan sólo hay que añadir la siguiente línea al archivo correspondiente de configuración:

```
compile 'com.karumi:dexter:2.2.2'
```

4.3 Comunicación entre dispositivos

Bajo este título se recoge toda la información referente a la comunicación entre dispositivos, incluyendo el protocolo creado específicamente para la aplicación de este proyecto.

4.3.1 Protocolo de comunicación

Para hacer efectiva la comunicación entre dispositivos se ha definido previamente un protocolo por el que se rige la aplicación, adaptándose a la especificación que ofrece AltBeacon. En función de este protocolo se ejecutarán las acciones pertinentes según en qué estado de ejecución se encuentre la aplicación en ese momento.

4.3.1.1 Mecanismo

Se usa el mecanismo típico de petición-respuesta que se usa principalmente en las arquitecturas clientes servidor.

A continuación, se representan las balizas o beacons que se crean para conseguir una comunicación efectiva entre los dispositivos Android que lleven instalada la aplicación.

Dentro de los campos que nos ofrece el formato AltBeacon tenemos disponibles tres IDs para enviar los datos que queramos. El ID1, el más grande los tres nos servirá para enviar la información requerida y los otros a modo de identificación de esa información.

Para el ID2, que identifica el estado de la partida o, dicho de otra manera, el momento en el que se encuentra la partida, se han definido los siguientes valores fijos:

- SEARCH_FOR_PLAYERS: Valor que indica que la información que viaja por el ID1 tiene que ver con la búsqueda de jugadores que quieran unirse a una partida.
- ITS_YOUR_TURN: Como el nombre indica, sirve para comunicar a un dispositivo que le ha llegado el turno de jugar.
- WAIT_FOR_YOUR_TURN: Si a la hora de decidir los turnos, juega el creador de la partida primero, es preciso indicarles a los demás que esperen por su turno.
- PREVIOUS_PLAYER: Sirve para indicar que la información enviada se corresponde con los datos del jugador que haya jugado previamente, si ha habido alguno.
- RESULTS_ARE_READY: Sirve para comunicar los resultados finales de la partida.

Además, en algunas situaciones, se precisa de una respuesta para la petición que se está realizando al enviar cierto mensaje, por lo que el ID3 se utiliza para distinguir entre una petición de información o una respuesta a esta petición. Los valores fijos establecidos para esto son:

- REQ: Se corresponde con una petición
- ACK: Se corresponde con una respuesta

Ahora, para cada par de ID3 e ID2 existentes, la información que viaja por ID1 es diferente y se detallan seguidamente, incluyendo la longitud de cada dato que se pone en ID1 (en bytes y campos que ocupa en formato hexadecimal).

- SEARCH_FOR_PLAYERS
  - REQ

- Nombre del host [10Bytes, 20hex]
- Dirección Bluetooth Host [6Bytes, 12hex]
- Formato en hexadecimal:
  0xNNNNNNNNNNNNNNNNNNDDDDDDDDDDDD

- ACK
  - Nombre del jugador [4Bytes, 8hex]
  - Dirección bluetooth jugador [6Bytes, 12hex]
  - Dirección bluetooth creador partida [6Bytes, 12hex]
  - Formato en hexadecimal:
    0xNNNNNNNNDDDDDDDDDDDDDDDDDDDDDD

- ITS_YOUR_TURN
  - REQ
    - Configuración de la partida [1Byte, 2hex]
    - Resultado ganador actual [1,5Bytes, 3hex]
    - Dirección Bluetooth ganador actual [6Bytes, 12hex]
    - Formato:
      0x000CCRRRAAAAAAAAAAAASSSSSSSSSSS
    - Caso especial cuando no hay jugador previo:
      0x000CC000000000000000SSSSSSSSSSS

- PREVIOUS_PLAYER
  - REQ
    - Dirección bluetooth jugador previo [6Bytes, 12hex]
    - Formato: 0x00000000000000000000DDDDDDDDDDDD

- WAIT_FOR_YOUR_TURN
  - REQ

- Dirección bluetooth del que va a jugar [6Bytes, 12hex]
- Formato: 0x00000000000000000000**DDDDDDDDDDDD**

- ACK
  - Dirección bluetooth del que espera [6Bytes, 12hex]
  - Dirección bluetooth del que va a jugar [6Bytes, 12hex]
  - Formato: 0x00000000EEEEEEEEEEEE**DDDDDDDDDDDD**

- RESULTS_ARE_READY
  - REQ
    - Resultado del ganador [1,5Bytes, 3hex]
    - Dirección bluetooth ganador [6Bytes, 12hex]
    - Formato: 0x00000000000000000**RRR**DDDDDDDDDDDD

4.4 Entorno de desarrollo

Bajo este título se recogen el hardware y las herramientas softwares utilizadas para la consecución del desarrollo de este proyecto.

4.4.1 Hardware utilizado para el desarrollo

Para la consecución del desarrollo de la aplicación se ha hecho uso de los siguientes ordenadores personales con una serie de características técnicas suficientes para correr las herramientas software utilizadas. Estas características son:

- PC Portátil:
  - CPU: Intel Core i5-2450m
  - RAM: 8GB
  - GPU: NVIDIA Geforce GT 630m
  - SO: Windows 10 Home 64bits
- PC Sobremesa:
  - CPU: Intel Core i7-6700K
  - RAM: 16GB
  - GPU: NVIDIA Geforce GTX 970
  - SO: Windows 10 Pro 64 Bits

De igual modo, para depuración de la aplicación se han hecho uso de varios dispositivos móviles con el hardware apropiado, que son:

- Motorola Moto E2 4G
  - Características principales
    - CPU: Qualcomm Snapdragon 410 Quad-core 1.2 GHz Cortex-A53
    - GPU: Adreno 306
    - Chip Bluetooth 4.0, compatible con la especificación Bluetooth Low Energy y su *Peripheral Mode*, necesario para utilizar la tecnología usada en el proyecto. Una lista completa

de dispositivos que son compatibles con esta tecnología puede ser encontrada en el siguiente enlace: https://altbeacon.github.io/android-beacon-library/beacon-transmitter-devices.html (Enlace válido a fecha de junio de 2016).

        - Sistema operativo: Android 5.1 Lollipop
- BQ Aquaris E5 HD:
    - Características principales:
        - CPU: MediaTek Quad Core Cortex-A7, @1.3 GHz
        - GPU: Mali 400 MP2
        - Chip Bluetooth 4.0 LE
        - Sistema operativo: Android 5.0 Lollipop

4.4.2 Software específico para el desarrollo

Se han utilizado una serie de herramientas para el desarrollo de la aplicación de este proyecto. A continuación, además de listar cada una de estas herramientas se ofrecerá un enlace a la ubicación de cada una de ellas para hacer más amena la instalación y configuración del proyecto.

Las herramientas citadas son:

- Como entorno de desarrollo o IDE, se ha hecho uso de la herramienta oficial que propone Google desde que la desarrollan, que es Android Studio. A fecha de redacción de este documento se encuentra la versión 2.1 y está disponible a partir del siguiente enlace: https://developer.android.com/studio/index.html
- De igual forma, haremos uso del SDK (Software Development Kit) que nos ofrece Google para desarrollar para la plataforma Android. No es necesario obtenerlo pues viene integrado con el IDE Android Studio.
- En Android, se puede desarrollar principalmente con dos lenguajes, soportados oficialmente. Estos lenguajes de programación son C++ y Java. Nosotros en esta ocasión hemos usado Java por facilidad de configuración

en el entorno y por familiarización con respecto al lenguaje. Para que funciona correctamente debemos obtener el JDK (Java Development Kit) que nos ofrece Oracle, empresa propietaria de este kit de desarrollo. En esta ocasión, se ha usado el JDK en su séptima versión y es accesible y descargable desde el siguiente enlace: http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html

4.4.3 Estructura de un proyecto Android

La estructura de los paquetes y carpetas que forman el proyecto están de acuerdo a la forma convencional en que se crean los proyectos en el IDE Android Studio y en general es de esta manera para todo proyecto realizado para la plataforma Android.

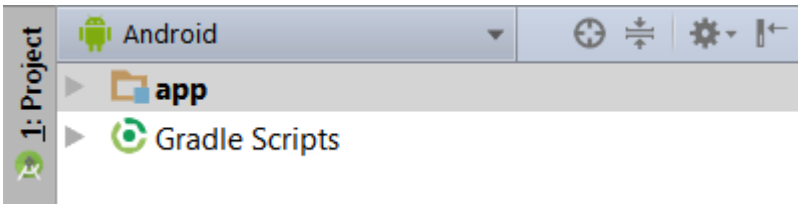

ILUSTRACIÓN 23: CARPETAS RAIZ PROYECTO ANDROID

Desde la raíz del proyecto nos encontraremos con *app* y *Gradle Scripts*. Dentro de la primera están todos los archivos referentes a la aplicación que se está desarrollando y dentro de la segunda se encuentran los archivos necesarios para realizar la compilación del proyecto, de la cual se encargará automáticamente el IDE según los parámetros y configuraciones que se encuentren en esos archivos.

Desplegando el primer paquete, ***app***, nos encontramos con lo mostrado en Ilustración 24: carpetas dentro de paquete app. Aparecen tres paquetes principales: ***manifests***, donde se encuentra el archivo xml donde se declaran los componentes principales de Android usados en el proyecto; ***java***, donde se sitúan todos los archivos relativos al código fuente de la aplicación (clases, interfaces, etc.); y ***res***, utilizado para guardar todos los recursos que son accesibles desde

nuestro código de la aplicación, es decir, imágenes, cadenas de texto, iconos, plantillas, etc.

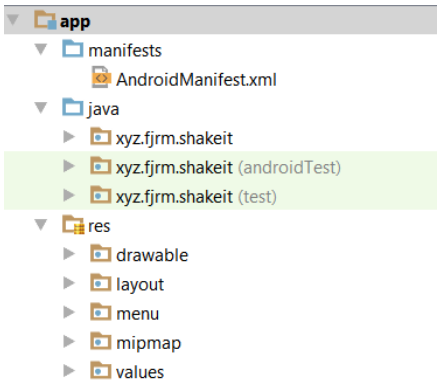

ILUSTRACIÓN 24: CARPETAS DENTRO DE PAQUETE APP

Entrando en detalle en la carpeta ***app/res/***, nos encontramos con lo siguiente:

TABLA 12: RECURSOS CARPETA RES

Carpeta	*Descripción*
/res/drawable/	Todas las imágenes y otros elementos gráficos utilizados son contenidos por esta carpeta. Además, Android, para facilitar el ajuste de estos recursos a la pantalla de los múltiples dispositivos existentes con diferentes densidades de pantalla, habilita unas subcarpetas para depositar en cada una de ellas nuestras imágenes para esas densidades, como sigue: • /drawable (recursos independientes de la densidad) • /drawable-ldpi (densidad baja) • /drawable-mdpi (densidad media) • /drawable-hdpi (densidad alta) • /drawable-xhdpi (densidad muy alta) • /drawable-xxhdpi (densidad más alta)
/res/mipmap/	En esta carpeta se sitúan los iconos que aparecen en el menú de aplicaciones de un smartphone Android. Al igual que los elementos gráficos de *drawable* se divide en subcarpetas

	relativas a las diferentes densidades de los dispositivos, siguiendo la misma nomenclatura, solo que con el prefijo mipmap.
/res/layout/	Esta carpeta contiene todos los ficheros xml que forman la interfaz de la aplicación. Cada uno estos ficheros de asociará posteriomente a activities o fragments desde el propio código fuente. Puedes definirte varios XML para diferentes configuraciones de pantalla, por ejemplo Smartphones pero también tablets.
/res/values	Contiene más ficheros XML, pero en esta ocasión tienen relación con cadenas de texto, estilos, definición de colores, tamaños, etc.
/res/menu/	Aquí se encuentran los ficheros XML relativos a los distintos menús que puedan tener las pantallas de la aplicación.

4.4.4 Estructura específica de nuestro proyecto

A pesar de que todo proyecto Android suele compartir la estructura de directorios y carpetas explicada en el punto anterior, es de libre elección la distribución de los ficheros fuentes para cada aplicación. Como se mencionó previamente, estos ficheros fuentes se encuentran dentro de la carpeta *java*. Dicho esto, se muestra acto seguido las carpetas creadas para ordenar el código en el proyecto desarrollado.

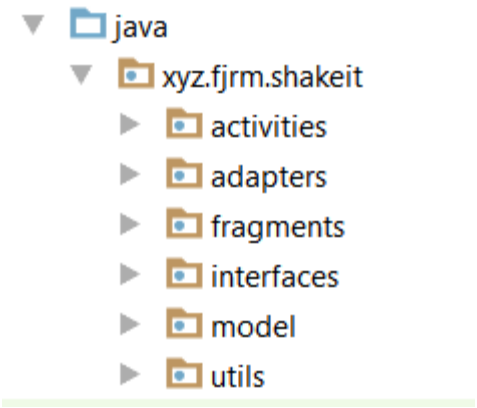

ILUSTRACIÓN 25: ESTRUCTURA PROYECTO SHAKE IT

En la siguiente tabla de describe lo que se encuentra en cada uno de los paquetes creados, aunque los nombres ya dan una idea bastante clara de lo que contienen. Los nombres de los paquetes en cuestión son: activities, adapters, fragments, interfaces, model, utis.

TABLA 13: PAQUETES ESPECÍFIFCOS PROYECTO SHAKE IT

Nombre paquete	Descripción
activities	Este paquete contiene todas las activities presentes en la aplicación.
adapters	Este paquete contiene los adapters. Un adapter se corresponde con una clase que sirve de ayuda a adaptar información o datos que quieres mostrar en una vista que sea una lista. Por ejemplo, para mostrar datos en una RecyclerView.
fragments	En este paquete se encuentran los fragments asociados con las activities.
interfaces	Las interfaces utilizadas para establecer contratos con otras clases. Para actuar en función de alguna acción que ocurra, a modo de listeners.

model	Las clases modelo para representar los objetos principales que conciben la aplicación se encuentran aquí.
utils	Todas las clases que sirven de ayuda para convertir formatos de datos se engloban aquí.

4.5 Ficheros fuente relevantes del proyecto

En esta sección se explicará de una forma más detallada alguna de las clases creadas en el proyecto que han resultado útiles a la hora de desarrollar el mismo.

Se adjuntarán capturas con el código incluyendo la documentación interna para facilitar la comprensión del mismo.

4.5.1 BluetoothChecker y AccelerometerChecker

Tal como su nombre indica, la única responsabilidad de esta clase es comprobar que el dispositivo que tenga instalada la aplicación y se disponga a jugar tiene bluetooth y, además, es compatible con la característica del Bluetooth LE Peripheral Mode para poder emitir como si de un beacon se tratase.

Asimismo, AccelerometerChecker se encargará de comprobar que el dispositivo tenga un acelerómetro.

Estas clases deben ser inminentemente usadas acto previo al empezar cualquier partida dentro del juego, pues es esencial para el correcto funcionamiento del resto de componentes. Si el dispositivo no cumple con estas condiciones no está capacitado para continuar con la partida.

```java
/**
 * Clase BluetoothChecker
 *
 * Encargada de realizar las comprobaciones necesarias para saber
 * si el dispositivo tiene bluetooth y es compatible con la característica
 * del Bluetooth LE Peripheral Mode.
 */
public class BluetoothChecker {
    public static final int REQUEST_ENABLE_BT = 0; /** Valor para pedir la activación del bluetooth */
    private static final String TAG = "BtChecker";
    private final BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    /**
     * Función para saber si el bluetooth está activado en el dispositivo.
     * @return true si el dispositivo tiene bluetooth activado, false en caso contrario
     */
    public boolean isEnabled(){
        if (mBluetoothAdapter == null) {
            throw new IllegalStateException("Debe usar la función hasBluetooth previamente.");
        }
        return mBluetoothAdapter.isEnabled();
    }

    /**
     * Función para saber si el dispositivo soporta el Bluetooth LE Peripheral Mode
     * para poder emitir balizas o beacons.
     * @param context contexto de la aplicación
     * @return true si el modo es soportado, false si lo contrario
     */
    public static boolean supportsAdvertising(Context context){
        return BeaconTransmitter.checkTransmissionSupported(context) == BeaconTransmitter.SUPPORTED;
    }

    /**
     * Función para saber si el dispositivo tiene bluetooth
     * @return true si el dispositivo tiene bluetooth
     */
    public boolean hasBluetooth(){
        return mBluetoothAdapter != null;
    }
}
```

ILUSTRACIÓN 26: CLASE BLUETOOTHCHECKER

```java
/**
 * Clase encargada de las comprobaciones necesarias
 * para comprobar si el dipositivo tiene acelerómetro
 */
public class AccelerometerChecker {
    private final Sensor accelerometer;

    public AccelerometerChecker(Context context){
        accelerometer =
                ((SensorManager) context.getSystemService(Activity.SENSOR_SERVICE))
                        .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    /**
     * Función encargada de decir si el dispositivo está dotado
     * de un acelerómetro.
     * @return true si el dispositivo tiene acelerómetro, false si no
     */
    public boolean hasAccelerometer() { return accelerometer != null; }
}
```

ILUSTRACIÓN 27: CLASE ACCELEROMETERCHECKER

4.5.2 GameNameParser y ScoreParser

Como sus nombres indican estas clases actúan como parsers. Esto quiere decir que su función principal es convertir datos de un formato a otro. Así, de esta manera GameNameParser se encargará de tratar los nombres de partida y de jugador y ScoreParser de los resultados que obtengan los jugadores en la partida.

Es necesario realizar estás conversiones pues a la hora de enviar esta información por bluetooth se requiere de una codificación a hexadecimal.

4.5.2.1 GameNameParser

La clase GameNameParser contiene las siguientes funciones:

```java
/**
 * Clase que siurve de ayuda para transformar las
 * cadenas de texto pasadas entre comunicaciones bluetooth.
 */
public class GameNameParser {

    /**
     * Función que convierte un texto en formato hexadecimal
     * a su homógeno en ASCII.
     * @param hex Cadena de texto en hexadecimal
     * @return cadena de texto en ASCII
     */
    public static String hexToAscii(String hex) {
        StringBuilder ascii = new StringBuilder();
        String str;
        for (int i = 0; i < hex.length(); i += 2) {
            str = hex.substring(i, i + 2);
            ascii.append((char) Integer.parseInt(str, 16));
        }
        return ascii.toString();
    }

    /**
     * Función que convierte una cadena de texto en formato hexadecimal.
     * @param asciiString Cadena de texto en ASCII
     * @return cadena de texto en hexadecimal
     */
    public static String asciiToHex(String asciiString) {
        char[] chars = asciiString.toCharArray();
        StringBuilder hex = new StringBuilder();
        for (char aChar : chars) {
            hex.append(Integer.toHexString((int) aChar));
        }
        return hex.toString();
    }
}
```

ILUSTRACIÓN 28: CLASE GAMENAMEPARSER

El nombre de las funciones es bastante semántico y la explicación de los métodos está en la documentación interna de la clase.

Es necesaria esta conversión para poder mostrar correctamente los nombres por pantalla, no sería de buen gusto mostrar dígitos hexadecimales.

4.5.2.2 ScoreParser

La razón de usar esta clase viene por la cantidad de agitaciones que podríamos enviar en las balizas que tenemos definidas. Como hemos visto antes, hemos asignado tres bytes para poder almacenar el resultado y enviarlo. Si el resultado lo enviamos en decimal tendríamos la posibilidad de enviar hasta un total de 999 shakes o agitaciones, pero si lo hacemos en hexadecimal obtenemos un total de FFF, que en su interpretación decimal llegamos hasta la friolera cantidad de 4095. Aunque tal y como se ha definido el juego por ahora sería imposible alcanzar tal cantidad, hemos dejado abierta esa posibilidad para futuras ampliaciones o adición de nuevos modos de juegos que podrían requerir de tales cantidades.

Además, aparte de la conversión de decimal a hexadecimal y viceversa, se ha añadido un formateador para que los números que se envíen sean siempre de tres cifras. De esta manera, mantenemos la longitud de ocupación en la baliza.

```
/**
 * Parser para devolver siempre un número de tres cifras.
 * Ej: Si un jugador consigue 15 shakes, se devolverá 015.
 */
public class ScoreParser {
    private static String pattern = "000";

    /**
     * Formatea el resultado recibido por parámetro.
     * @param score
     * @return cadena de texto de un número de tres cifras
     */
    public static String parse(int score) { return new DecimalFormat(pattern).format(score); }

    /**
     * Función de conversión de un número decimal en hexadecimal
     * @param score cadena de texto de un número en decimal
     * @return cadena de texto de un número en hexadecimal
     */
    public static String toHex(String score){
        return Integer.toHexString(Integer.parseInt(score));
    }

    /**
     * Función de conversión de un número hexadecimal en decimal
     * @param hexScore cadena de texto de un número en hexadecimal
     * @return cadena de texto de un número en decimal
     */
    public static String fromHex(String hexScore){
        return String.valueOf(Integer.parseInt(hexScore, 16));
    }
}
```

ILUSTRACIÓN 29: CLASE SCOREPARSER

4.5.3 TimedBeaconTransmitter

Dado el hecho de que todas las veces que se emite en la aplicación se hacía acompañado de un temporizador se ha decidido crear una clase para ello que facilite esta tarea.

Se han sobreescrito los métodos de la clase padre original de manera que cada vez que se inicia una emisión se inicia también el temporizador y una vez la cuenta atrás de este temporizador llega a su fin se detiene también la emisión. Evidentemente la emisión puede ser parada en cualquier momento como se podía hacer previamente.

En vez de crear dentro de la clase un temporizador con tiempos dijo se deja a libre elección el tiempo de duración de esa cuenta atrás. El constructor de esta clase recibe por parámetro una instancia de CountDownTimer[4], por lo que previamente se ha establecido el tiempo que durará la transmisión si ésta no es interrumpida antes.

En el mismo fichero se ha definido una interfaz también, por lo que en la parte del código que se use esta clase deberá implementarla a fin de que se reciban los callbacks[5] cada vez que la emisión se inicie correctamente o no, para poder actuar en consecuencia.

[4] CountdownTimer: Clase ofrecida por el SDK de Android para hacer cuentas regresivas.
[5] Callback: Se llama callback a la acción de “llamar de vuelta” que se produce cuando desde una parte del código se hace una llamada a una función y se espera una respuesta de ésta en un momento futuro.

```java
/**
 * Emisor de beacons con temporizador añadido
 */
public class TimedBeaconTransmitter extends BeaconTransmitter {
    private CountDownTimer countDownTimer;
    private OnTimedBeaconTransmitterListener listener;

    public TimedBeaconTransmitter(Context context, BeaconParser parser, CountDownTimer timer) {
        super(context, parser);
        this.countDownTimer = timer;

        if (context instanceof OnTimedBeaconTransmitterListener)
            listener = (OnTimedBeaconTransmitterListener) context;
        else
            throw new RuntimeException(context.toString()
                    + " debe implementar OnCountdownListener");
    }

    @Override
    public void startAdvertising(Beacon beacon, AdvertiseCallback callback) {
        super.startAdvertising(beacon, callback);
        countDownTimer.start();
    }

    @Override
    public void startAdvertising() {
        super.startAdvertising();
        countDownTimer.start();
    }

    @Override
    public void startAdvertising(Beacon beacon) {
        super.startAdvertising(beacon, new AdvertiseCallback() {
            @Override
            public void onStartSuccess(AdvertiseSettings settingsInEffect) {...}

            @Override
            public void onStartFailure(int errorCode) {...}
        });
        countDownTimer.start();
    }

    @Override
    public void stopAdvertising() {
        super.stopAdvertising();
        countDownTimer.cancel();
    }

    public interface OnTimedBeaconTransmitterListener {
        void onSuccessTransmission();

        void onErrorTransmission();
    }
}
```

ILUSTRACIÓN 30: CLASE TIMEDBEACONTRANSMITTER

4.5.4 ShakeDetector

Esta clase tiene la responsabilidad de interpretar los datos recibidos por el acelerómetro.

En esta clase implementamos el algoritmo de detección de agitaciones. Este algoritmo se encarga de decidir cuándo se dan las condiciones necesarias para concebir una serie de movimientos realizamos con el smartphone para considerar los mismos como una agitación.

En la imagen Ilustración 31: Implementación algoritmo de detección de shakes se puede visualizar el algoritmo implementado.

```
setCurrentAcceleration(event);
float maxLinearAcceleration = getMaxCurrentLinearAcceleration();
if (maxLinearAcceleration > MIN_SHAKE_ACCELERATION) {
    long now = System.currentTimeMillis();
    if (startTime == 0) {
        startTime = now;
    }
    long elapsedTime = now - startTime;
    if (elapsedTime > MAX_SHAKE_DURATION) {
        resetShakeDetection();
    } else {
        moveCount++;
        if (moveCount > MIN_MOVEMENTS) {
            mShakeListener.onShake();
            resetShakeDetection();
        }
    }
}
```

ILUSTRACIÓN 31: IMPLEMENTACIÓN ALGORITMO DE DETECCIÓN DE SHAKES

Este algoritmo es ejecutado cada vez que se recibe información del sensor, el cual tiene una frecuencia de refresco bastante alta, probablemente de milisegundos. No es algo que definamos nosotros, sino que viene establecido así por defecto.

En el siguiente listado de variables, se pretende explicar cada una de ellas para entender su función dentro del algoritmo.

- Locales del método:
  - MaxLinearAcceleration: Tiene en todo momento el mayor valor de aceleración lineal dado por uno de los ejes de coordenadas.
  - Now: Esta variable contiene el valor en milisegundos de la fecha actual en el momento que se llame a la función *System.currentTimeMillis().*
  - elapsedTime: Como su nombre indica, esta variable contendrá el tiempo transcurrido desde la última medición entregada por el acelerómetro y su valor viene dado por la diferencia entre *now* y *startTime*.
- Privadas de la clase:
  - startTime: Inicialmente esta variable tiene valor 0 y solo vuelve a tener este valor si se reinicia ese valor a 0, el resto de las veces recibirá la fecha actual en milisegundos, es decir, recibirá el valor de la variable *now*.
  - moveCount: Esta variable es un contador que acumula el número de movimientos que se van realizando dentro de la ventana de tiempo establecida para considerar este movimiento válido.
- Privadas estáticas finales de la clase: Son valores fijos usados como referencia de corte para comparaciones en el algoritmo
  - MIN_SHAKE_ACCELERATION: Establece la aceleración mínima necesaria para considerar un shake.
  - MAX_SHAKE_DURATION: Establece cuánto como máximo puede durar un movimiento.
  - MIN_MOVEMENTS: Por último, esta variable establece el número mínimo de movimientos necesarios para considerar una agitación.

Por lo que al final, como resultado de aplicar el algoritmo, notificamos gracias a la interfaz definida y por medio de un callback cuando se ha producido un shake, para poder sumarlos posteriormente.

4.5.4.1 Explicación de los métodos auxiliares

En la ilustración del algoritmo incrustada anteriormente aparecen varios métodos y funciones auxiliares que son utilizados para obtener ciertos valores que se utilizar en el propio algoritmo.

- setCurrentAcceleration (SensorEvent event): en este método se aplica un filtro de paso alto sobre los valores entregados por el acelerómetro para eliminar los efectos de la gravedad tal como se indica en la página de desarrolladores de Android facilitada por Google. Se puede consultar accediendo a la siguiente dirección: https://developer.android.com/guide/topics/sensors/sensors_motion.html
- resetShakeDetection(): Se encarga de reestablecer los valores de las variables *startTime* y *moveCount* a 0. Este método es llamado cada vez que se ha detectado un shake, para comenzar la detección de otro y cada vez que nos hemos pasado de la ventana de tiempo establecida para detectar un shake, por lo que se considera éste no válido y vuelta a empezar.
- getMaxCurrentLinearAcceleration(): Su función es la de comparar los valores de aceleración lineal obtenidos en cada eje de coordenadas para devolver el mayor de ellos, que es el que utilizará para decidir si se ha movido el teléfono móvil con la suficiente rapidez.

4.6 Implementación de los casos de uso

Como último tema a tratar en el capítulo de implementación se explican y localizan a continuación las partes del código fuente específicas que tiene relación directa con la implementación de los casos de uso en sí. Siguiendo el diagrama se verán en este orden:

- Comenzar juego
  - Comprobar requisitos
- Configurar partida
  - Configurar creación partida
  - Configurar unión partida
- Crear partida
- Unirse a partida
  - Seleccionar creador
- Jugar partida
  - Empezar partida
  - Computar partida
  - Terminar partida
  - Comparar resultados
- Salir partida

Para cada uno de ellos se indicará las activities o fragments involucrados (propios del framework de Android), así como los ficheros xml asociados a las vistas que contienen. De igual manera, se mostrará parte de la lógica que comprende cada una de las acciones.

4.6.1 Comenzar juego

El usuario abre la aplicación mediante el acceso directo del menú de aplicaciones y se muestra la pantalla principal de la aplicación, que se corresponde con la vista asociada a los ficheros XML activity_main.xml y content_main.xml. La activity

relacionada con estas vistas se denomina MainActivity y en ella se encuentra toda la lógica relativa a este caso de uso.

En la vista se ofrece un botón para comenzar el juego que desencadenará una serie de comprobaciones; requisitos para decidir si se puede comenzar la partida o no.

El click sobre este botón se captura y trata con el método onClick() y acto seguido se realizan las comprobaciones dentro del método.

Dichas comprobaciones se realizan con la ayuda de las clases AccelerometerChecker y BluetoothChecker, explicadas en una sección anterior.

Asimismo, si la versión del sistema operativo donde se ejecuta la aplicación es Android 6.0, se necesitan pedir permisión en tiempo de ejecución de la aplicación. Para ello, en el método promptPermissions() se encuentra el código para mostrar los mensajes emergentes que permiten al usuario aceptar o denegar los permisos, con ayuda de la librería Dexter.

4.6.2 Configurar partida

Todo lo relacionado con configurar partida tiene relación con las siguientes clases (activities y fragments):

- GameConfigActivity: Situada dentro del paquete llamado activities. Los archivos xml asociados son activity_game.xml y content_game.xml.
    - GameFragment: Directamente relacionado con la parte visual que ofrece elegir con dos botones crear o unirse a una partida. Vista asociada: fragment_game.xml.
    - GameJoinFragment: Fragment para mostrar el formulario de configuración de unión a una partida. Vista asociada: fragment_game_join.xml.

- GameNameCreationFragment: Fragment para mostrar el formulario de creación de una partida. Vista asociada: fragment_game_name_creation.xml.

Adicionalmente, para validar los campos se emplean las clases UsernameValidator y TextValidator que están dentro del paquete utils del proyecto.

4.6.3 Crear partida

Todo el proceso de creación de partida se realiza dentro de la activity *ComunicationActivity*, con ficheros XML asociados *activity_comunication.xml* y *content_comunication,xml*. Así pues, como va muy ligado a la comunicación se indican varios de los métodos donde se realiza el proceso.

En la clase BeaconHandler situada en el paquete utils/comunication está la lógica de recepción de balizas. Para el caso de creación de la partida nos situamos en el momento en el buscamos jugadores. Para poder buscarlos necesitamos crear la baliza necesaria con el método initReqBeacon() y comenzar a transmitirla haciendo uso del método startAdvertising() donde se hace uso de la clase TimedBeaconTransmitter explicada con anterioridad.

Una vez se reciba la confirmación por parte de un jugador que quiera unirse a la partida se mostrará por pantalla en una RecyclerView, la cual está definida en el paquete adapters, el usuario tendrá la posibilidad de hacer click en el botón de crear partida y será finalmente creada pasando a calcular la partida e indicar el turno a un jugador.

4.6.4 Unirse a partida

Al igual que en la creación de partida, la lógica relativa a la unión de una partida está también en la activity ComunicationActivity.

El proceso que se sigue es bastante parecido y se utilizan prácticamente las mismas clases y métodos que antes. En esta ocasión, de forma adicional, el

usuario precisa de visualizar los creadores de partida previamente para poder seleccionar con quién quiere jugar.

Entonces, una vez que la clase BeaconHandler recibe la baliza correspondiente y lo notifica mediante la interfaz (BeaconHandleListener, definida en el mismo fichero) que tiene como contrato con la activity, se precisa de la creación de una baliza respuesta para indicar al creador de que nos queremos unir a la partida. Para hacerlo, hacemos uso del método initACKBeacon() y transmitimos la baliza con startAdvertising(). Todo ello acto seguido de que el usuario haga click en el creador al que quiere unirse; el click se captura en la RecyclerView y se notifica con la interfaz definida para ello, OnClickHostListener, dentro del paquete interfaces.

De esto modo, finaliza la unión a una partida.

4.6.5 Jugar partida

En esta ocasión la activity relacionada con el jugar una partida se denomina ShakeActivity, situada, y como las otras ya mencionadas, está dentro del paquete activities.

Para jugar una partida previamente a un jugador le ha debido de llegar el turno y se le ha mostrado la vista asociada a la activity (formada por los archivos activity_shake.xml y context_shake.xml).

En dicha vista, además de la información de la partida, se muestra un botón que hace comenzar la partida cuando el usuario hace click en él. Este click se captura dentro de la propia activity con ayuda del método onClickStartButton().

Una vez esto se debe comenzar a computar los movimientos del jugador y el código fuente relativo a este cómputo se puede localizar en el paquete utils. El nombre de la clase que contiene este código es ShakeDetector y ya ha sido explicada en el apartado 4.5.4 ShakeDetector

Terminar la partida implica que el temporizador ha llegado a su fin (el temporizador se establece con el método startCountDownTimer()) y mostrar un mensaje emergente al usuario para que pueda confirmar que ha terminado la partida y así notificar al siguiente jugador o mostrar los resultados si él ha sido el último en jugar.

El mensaje emergente se muestra con ayuda del método showFinishDialog(), que usa la clase AlertDialog ofrecida por el framework de Android.

4.6.6 Salir partida

La acción de salir partida puede ser ejecutada en cualquier momento que nos encontremos en una partida por lo que esta vez están involucradas dos activities en el proceso, ComunicationActivity y ShakeActivity.

Para la consecución de esta característica sobreescribimos el método de la activity onBackPressed(). Este método se llama cada vez que el usuario hace click en el botón de ir hacia atrás en el smartphone. Capturamos ese intento y ofrecemos al usuario un mensaje emergente de confirmación. El mensaje lo creamos con AlertDialog.

Si el usuario hace confirma su salida se sale de la partida.

5. Manual de usuario

El contenido del siguiente capítulo ofrece una guía de usuario con el único objetivo de ayudar a toda persona que use la aplicación a fin de que esa persona pueda navegar correctamente a través de dicha aplicación.

Para empezar, se realiza una descripción completa de la aplicación, seguida de los requisitos necesarios para poder ejecutarla. Además, también se ofrece una guía de instalación y unas trazas completas de la ejecución incluyendo imágenes que facilitan la comprensión de cómo usar la aplicación.

5.1 Descripción de la aplicación

Shake It! es un juego multijugador, sin límite de jugadores, en el que tendrás que agitar el teléfono para ganar a tus amigos. El que más agitaciones consiga gana.

5.2 Requisitos de la aplicación

Para poder usar esta aplicación es necesario lo siguiente:

- Smartphone con sistema operativo Android en su versión 5.0 o superior.
- Smartphone con chip Bluetooth 4.0 Smart o Low Energy (LE).
- Smartphone con acelerómetro.

Puedes comprobar si tu dispositivo se encuentra en una lista donde se han comprado un gran número de ellos a fin de que puedas saber previamente si tu smartphones es compatible. La lista en cuestión es accesible a través del siguiente enlace: https://altbeacon.github.io/android-beacon-library/beacon-transmitter-devices.html.

Si tu móvil no está en esa lista o incluso si pone que no es soportado puedes instalar la aplicación y comprobar si ésta funciona.

5.3 Guía de instalación

Para instalar la aplicación deberás de obtener el fichero con extensión .apk e introducirlo en tu dispositivo, ya sea en la memoria interna o en un tarjeta SD, si la tienes.

Abre el archivo con tu explorador de archivos favorito y sigue las instrucciones.

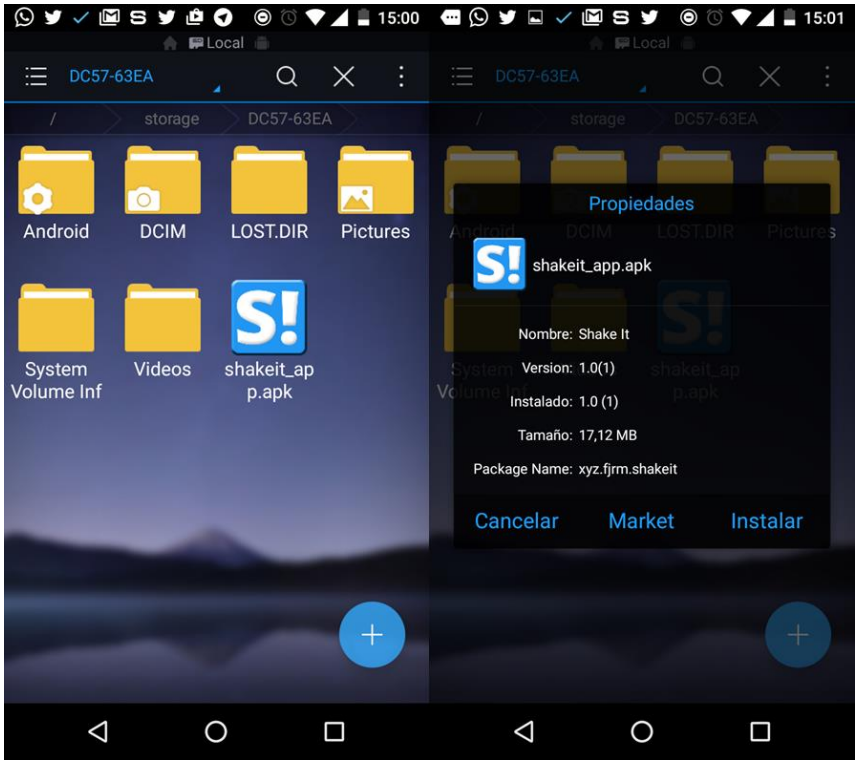

ILUSTRACIÓN 32: INSTALACIÓN SHAKEIT. ABRIR FICHERO APK

Al no estar disponible en el Google Play Store es posible que tu móvil te avise de ser una aplicación potencialmente maliciosa. Si esto ocurre deberás seguir los pasos que se describen a continuación.

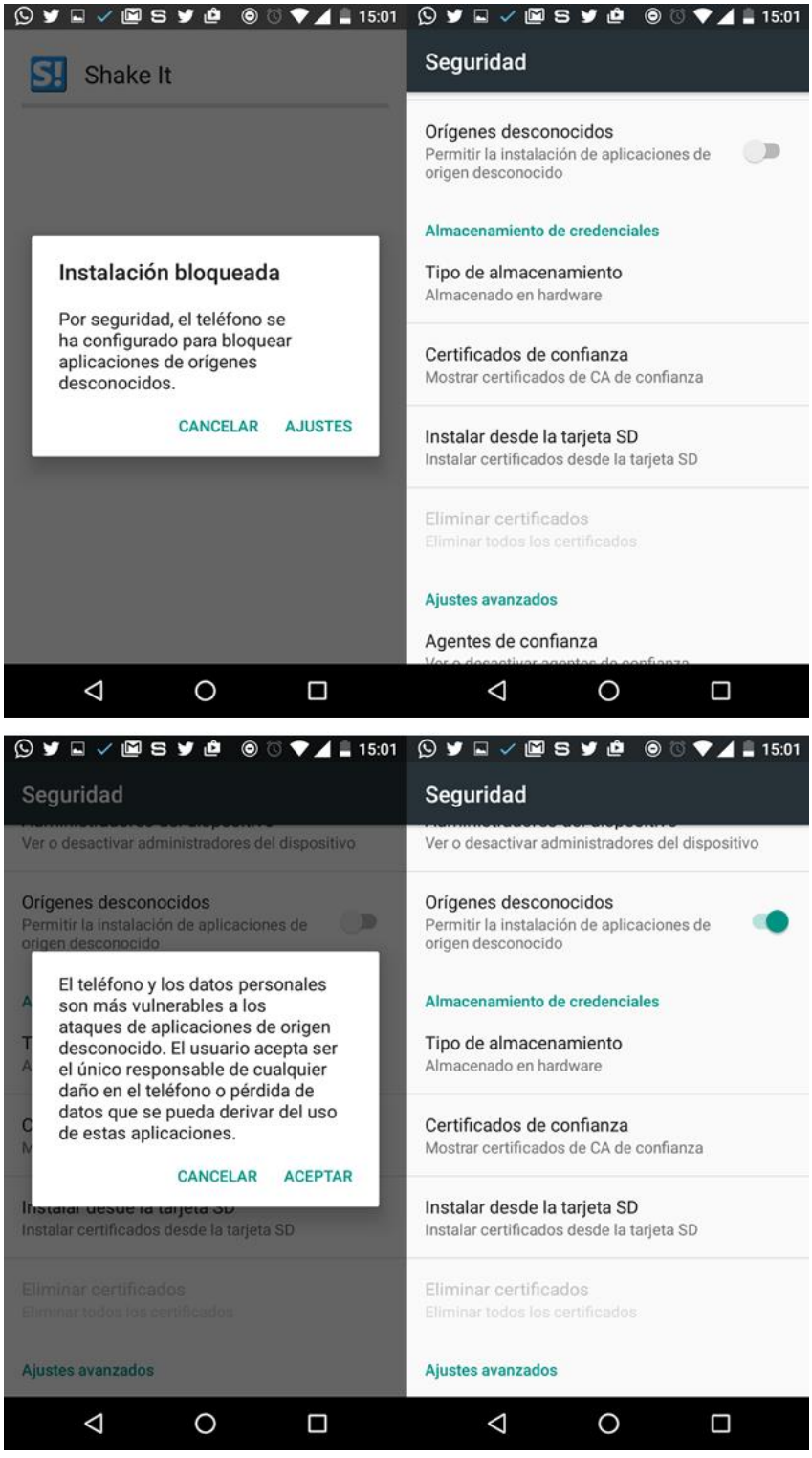

ILUSTRACIÓN 33: ACTIVACIÓN ORÍGENES DESCONOCIDOS

En la mayoría de los dispositivos se hace de la misma manera salvo modificación por parte del fabricante, que será similar. Debes dirigirte a Ajustes, tal y como se indica en el diálogo emergente y allí debes buscar la opción de Orígenes desconocidos, que se encontrará desactivada. Tan sólo tienes que activarla y volver a intentar instalar la aplicación de nuevo. Esta vez saltará el instalador y te permitirá instalarla sin ningún problema.

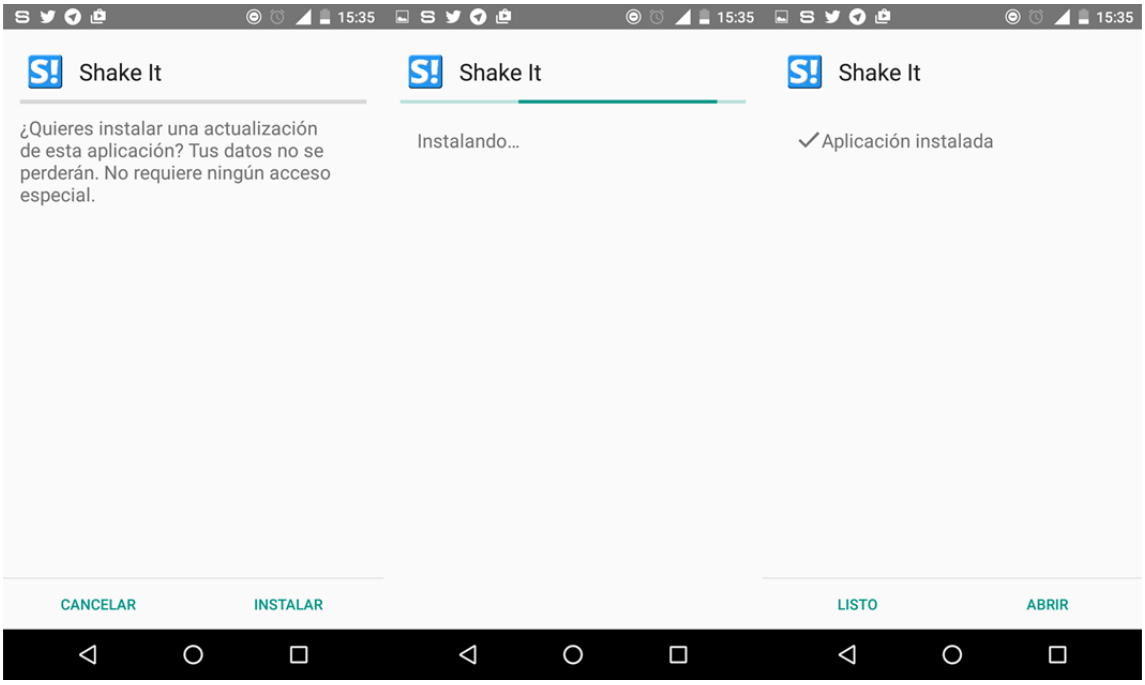

ILUSTRACIÓN 34: INSTALACIÓN SHAKE IT

Una vez finalice el proceso de instalación de la aplicación podrás abrila y empezar a utilizarla cuando gustes.

5.4 Ejecución y manejo de la aplicación

La aplicación es muy sencilla de manejar y no debería haber ningún problema a la hora de hacerlo. Igualmente, en el siguiente apartado se muestra la ejecución de una partida entre 2 jugadores. Se podrá visualizar cómo configurar una partida, como se muestran los jugadores involucrados, cómo se comienza la partida; todo por siguiendo esta guía.

El primer paso es abrir la aplicación. Para ello, te debes dirigir al icono de la misma, situado en el menú de aplicaciones de tu teléfono móvil o en el escritorio, si se ha añadido automáticamente como acceso directo.

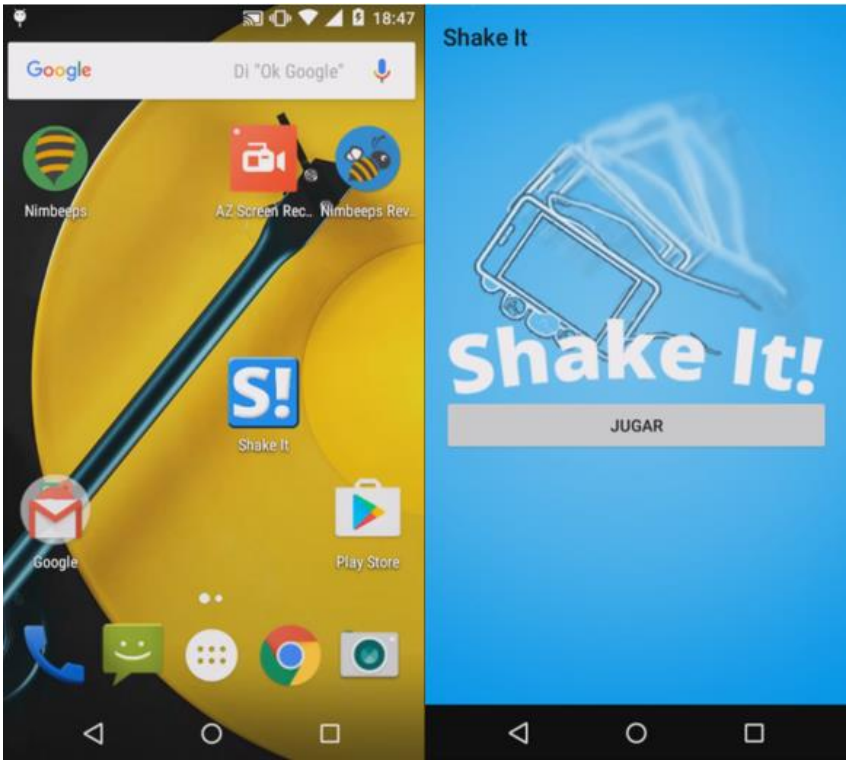

ILUSTRACIÓN 35: SHAKE IT. PANTALLA PRINCIPAL.

Una vez pulses en el icono, se abrirá la aplicación y aparecerá la pantalla principal. En ella hay disponible un botón que te llevará a la pantalla de configuración de la

partida, donde podrás decidir si quieres crear una partida o unirte a una creada por una persona cercana a ti.

Según la opción que elijas, aparecerá en la parte inferior de la pantalla un formulario con un campo a rellenar, que será del nombre de la partida si la opción escogida ha sido la de crear partida o del nombre del jugador si has escogido la opción contraria. En la ilustración que se expone acto seguido de puedes ver estos dos formularios. Cuando se introduzca la información requerida en ambos casos se pulsa el botón correspondiente de *Unirse* o *Crear*, y se abrirá una nueva pantalla.

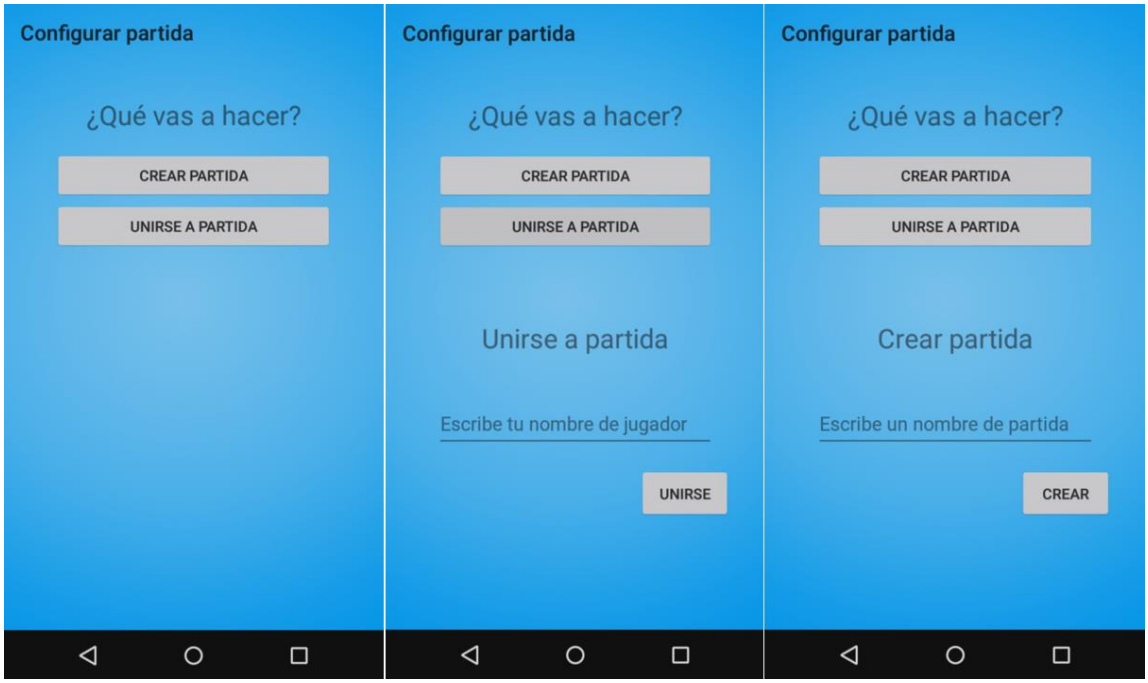

ILUSTRACIÓN 36: MANUAL DE USUARIO. CONFIGURACIÓN PARTIDA.

En las siguientes pantallas se mostrará un mensaje de espera, bien para creadores de partida, bien para jugadores. Una vez se encuentre a otros jugadores se mostrarán en una lista. En primer lugar, a la persona que se une a la partida le aparece el creador de la partida. Esta persona debe pulsar en el creador al que desea unirse. En este ejemplo sólo hay uno, pero podrían aparecer más. Cuando se realice esa pulsación, al creador de la partida le aparece el

jugador en cuestión que quiere unirse y podrá finalmente decidir si crea la partida, a no ser que esperase por más jugadores.

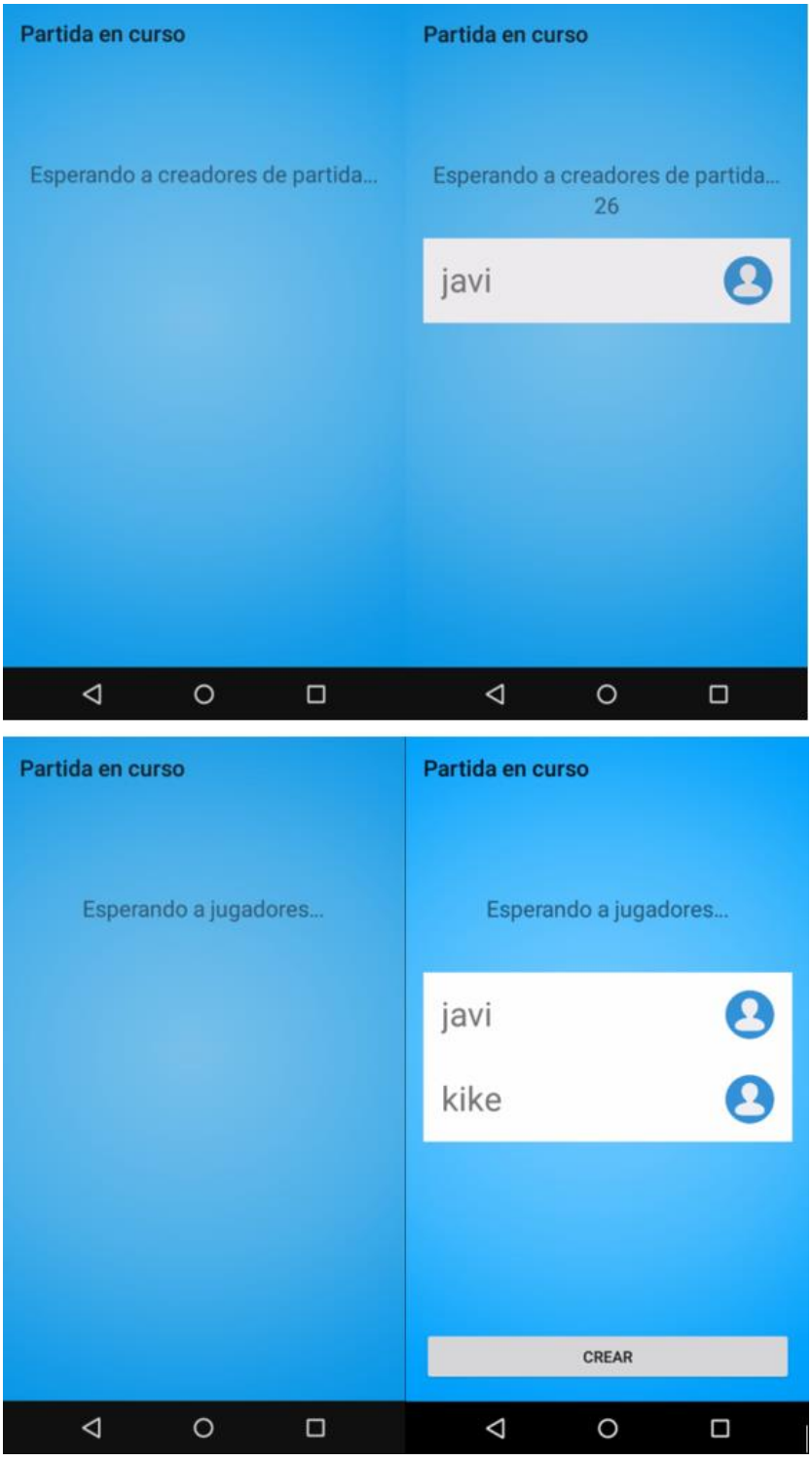

ILUSTRACIÓN 37: ESTABLECIMIENTO DE JUGADORES DE LA PARTIDA

Cuando el creador de la partida decida comenzar y pulse el botón *CREAR*, uno de los jugadores involucrados en la partida, incluido él mismo, podrá ser elegido como persona que comience la partida, ya que se decide aleatoriamente. Asimismo, dentro de un rango de valores, se decidirá el número de segundos que se debe agitar el smartphone para sumar shakes.

Así pues, a la persona que le llegue el turno podrá visualizar la pantalla donde puede comenzar a jugar y al resto de personas se les indicará que debes esperar por su turno, tal y como se muestra a continuación.

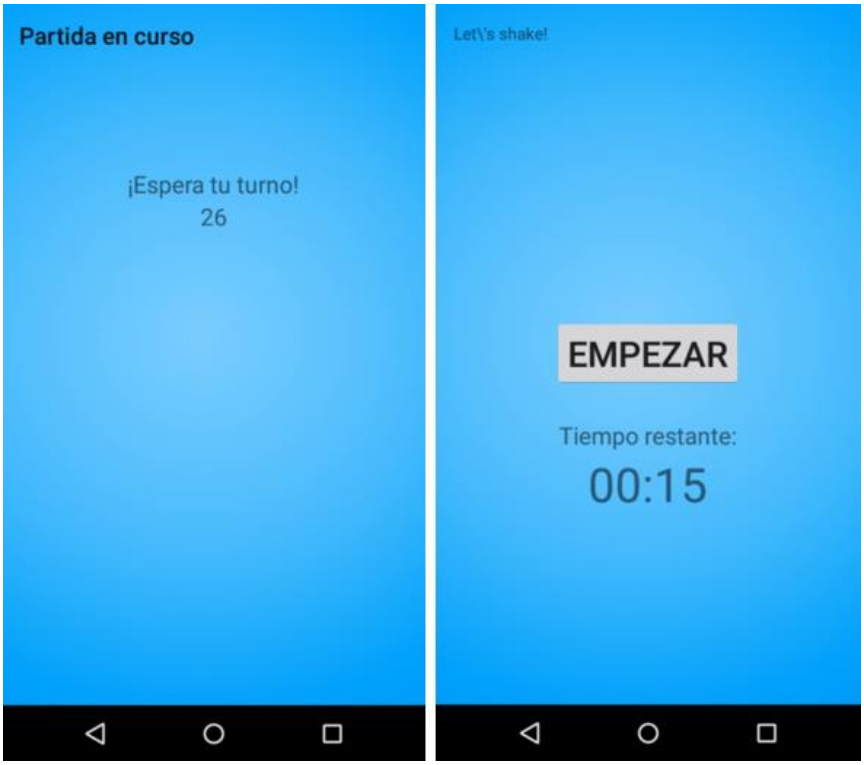

ILUSTRACIÓN 38: SHAKE IT. PARTIDA EN CURSO.

Siguiendo con la persona a la que le ha tocado jugar, ésta tiene la potestad de decidir cuándo va a empezar a agitar el móvil. En cuanto pulse el botón *EMPEZAR,* la cuenta atrás se iniciará y se dispondrá del tiempo que se muestre por pantalla para agitar el móvil. Una vez se termine de jugar, se mostrará una

pantalla con el resultado obtenido y con un mensaje que indica que debes esperar por los resultados.

ILUSTRACIÓN 39: SHAKE IT, PANTALLA RESULTADO

Esta pantalla permanecerá así hasta que todos los jugadores hayan participado en su turno.

Por último, la partida finalizará con un mensaje en el que se indica si has sido ganador o perdedor, en el que se dirá también el número de shakes que ha hecho el ganador.

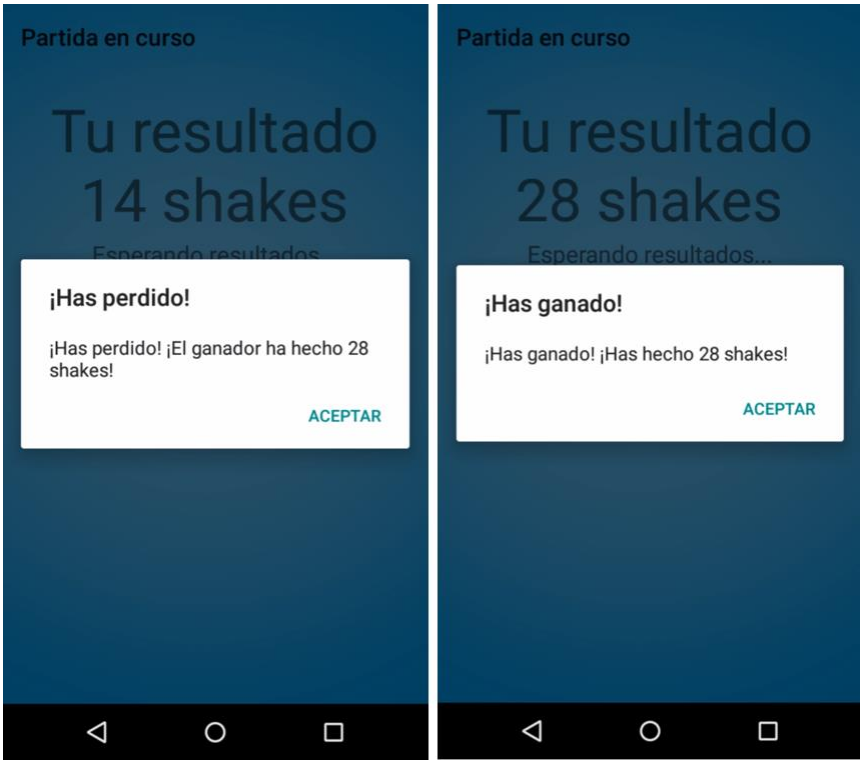

ILUSTRACIÓN 40: SHAKE IT. ANUNCIO DEL GANADOR.

Una vez se confirme la lectura del mensaje emergente se devolverá al jugador a la pantalla principal y estará en condición de volver a jugar otra partida.

5.4.1 Posibles mensajes de información o error

Hay situaciones en las que pueden aparecer mensajes emergentes informativos, por alguna acción que se vaya a realizar o de error, porque algo que se está realizando no esté admitido. Simplemente son mensajes de advertencia para saber que está ocurriendo.

A continuación, se expone una lista con los posibles mensajes que pueden aparecer:

- Si al pulsar el botón *JUGAR* de la pantalla principal no se tiene el bluetooth activado en el dispositivo, saldrá un mensaje emergente permitiéndote activarlo desde la aplicación. Una vez se permita, se puede comenzar a jugar.

ILUSTRACIÓN 41: MENSAJE EMERGENTE BLUETOOTH.

- Si se pulsa el botón de ir hacia atrás, el que viene por defecto, no en la aplicación, cuando se está en una partida en curso, aparecerá un mensaje

emergente para que sepas que, si lo haces, abandonarás la partida sin poder volver a unirte en ningún momento.

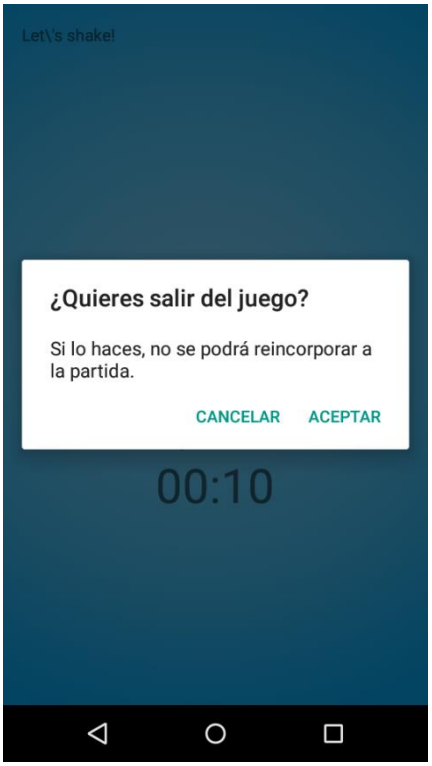

ILUSTRACIÓN 42: SHAKE IT. MENSAJE SALIR DEL JUEGO.

- A la hora de rellenar el formulario de configuración de partida es posible que te salte un mensaje de error. El nombre que se introduzca no debe ser mayor de 4 caracteres en el caso de unirte a una partida; el límite es de 10 para el creador de partida. Además, no se permiten caracteres especiales ni letras mayúsculas. De igual modo, si intentas empezar la partida sin rellenar los campos se mostrará otro mensaje.

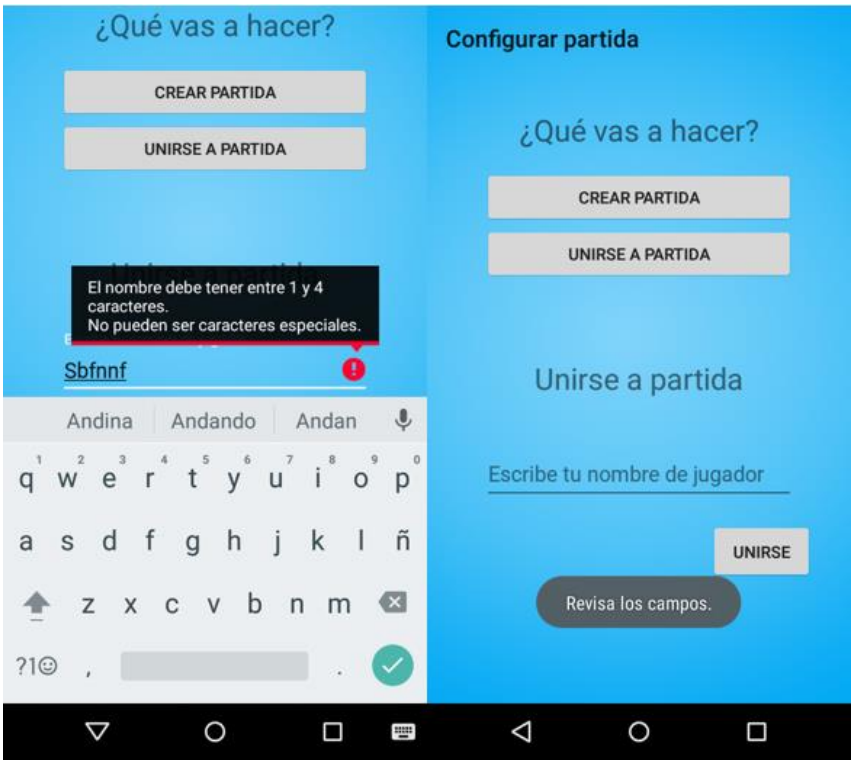

ILUSTRACIÓN 43: SHAKE IT. FORMULARIO. ERRORES.

6. Conclusiones

Este capítulo se expone a modo de conclusiones finales y personales sobre el desarrollo del proyecto, a medida que este se ha ido realizando.

El proyecto realizado ha partido desde una base prácticamente inexistente respecto a la información que hay en la actualidad sobre la tecnología que ha sido utilizada para la realización del mismo. Ha requerido de una investigación extra previa sobre todo lo relacionado con esta tecnología, así como de la API existente para poder realizar este tipo de comunicaciones. Se ha requerido del diseño de un protocolo de comunicación para poder hacer efectiva la comunicación entre los dispositivos que intervienen en una partida pues esta tecnología no está diseñada específicamente para ello. Y gracias a esta investigación previa ha sido posible el comienzo del desarrollo del proyecto.

Una vez hecho esto, se ha conseguido adaptar los conocimientos adquiridos a la idea inicial de la realización de un juego que cumpla las expectativas propuestas,

todo ello siguiendo un proceso de razonamiento, analizando los requisitos de la aplicación, casos de uso, esquemas preliminares, diseño de interfaz, y finalmente implementación del mismo.

6.1 Consecución de objetivos

Fueron múltiples los objetivos que se plantearon al principio del proyecto y parece conveniente verificar si se han llegado a cumplir.

Teniendo en cuenta de que ha sido posible realiza la aplicación planteada, comparando y eligiendo la tecnología propia para cumplir lo expuesto, como minimizar el gasto de batería o la interacción por parte del usuario para partes que no se requieran, puesto que van ligados a la tecnología aplicada, estamos en facultades de afirmar que se han cumplido todos en mayor o menor medida.

6.2 Posibles ampliaciones

Habiendo indagado en la materia, en este nuevo campo de comunicaciones, se ven múltiples posibilidades y/o aplicaciones de la idea plasmada en el proyecto.

Lo primero de todo mejorar la aplicación. Es funcional completamente pero evidentemente tiene más trabajo por delante. La intención es pulirla al máximo y publicarla en el Google Play Store. En general, mejorar la interfaz, la interacción con el usuario, la comunicación en sí misma, añadir más funcionalidades como por ejemplo la posibilidad de llevar un registro de las partidas realizadas, ofrecer otros modos de juego, etc.

Es interesante también contemplar la posibilidad de creación de una librería que facilite la creación de protocolos propietarios específicos de modo que sea más sencillo controlar la comunicación entre dispositivos usando esta tecnología. Es una idea que ha surgido durante el proyecto y pienso que es algo que promete.

En el último apartado de anexos se han añadido unas capturas en las que se mostraba una plataforma de juegos de estas características. En un futuro es

posible que la desarrolle a modo de englobar distintos juegos e incluso otras aplicaciones que hagan uso de esta tecnología de forma no convencional.

6.3 Aportaciones

La realización de este proyecto ha tenido como consecuencia la adquisición de competencias técnicas a la vez que personales.

Respecto a la técnica ya había realizado antes aplicaciones Android, pero nunca lo había hecho en el campo de comunicaciones bluetooth y mucho menos bajo estándares recientes. Esto me ha servido para formarme en un tema que si bien ya está presente entre nosotros lo estará aún más en los próximos años lo cual es estupendo pues me abre posibilidades en un mercado emergente. A su vez, no sólo en la comunicación ha servido para reforzar mis conocimientos en el framework de Android, como por ejemplo el acceso a sensores tales como el acelerómetro, y sin duda me será útil.

En lo personal y con respecto al grado, la realización de este proyecto me ha servido tal vez para tener cercanía con el proceso a seguir para la consecución de un proyecto desde cero, comprendiendo que es completamente necesario seguir unos pasos y metodologías estrictas si se quiere conseguir construir software de calidad. Una vez lo piensas, el grado me ha aportado la base necesaria para abordar este tipo de proyectos.

7. Bibliografía

En este apartado se muestran las referencias consultadas para la realización de la documentación y desarrollo del proyecto.

[1] Android Beacon Library. Consultado en febrero de 2016 de http://altbeacon.github.io/android-beacon-library/

[2] Android Beacon Library Apps. Consultado en febrero de 2016 de https://altbeacon.github.io/android-beacon-library/apps.html

[3] Dashboard. Platform version. Consultado en febrero de 2016 de https://developer.android.com/about/dashboards/index.html

[4] *Motion Sensors*, Consultado en febrero del 2016 de https://developer.android.com/guide/topics/sensors/sensors_motion.html

[5] Bluetooth. Consultado en mayo de 2016, de https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth

[6] Estándares y especificaciones de las redes Wi-Fi. Consultado en mayo de 2016 de https://norfipc.com/redes/tipos-redes-estandares-wi-fi-diferencias.php

[7] Bluetooth Technlogy Basics. Consultado en mayo de 2016, de https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics

[8] Android Local WiFi LAN Bluetooth WiFi Direct Multiplayer Games List. Consultado en mayo de 2016 de https://sites.google.com/site/androidmultiplayergames/home

[9] Wi-Fi Direct. Consultado en mayo de 2016 de http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[10] The differences between Bluetooth 4.0 and Wi-Fi Direct you need to know. Consultado en mayo de 2016 de http://www.makeuseof.com/tag/the-differences-between-bluetooth-4-0-and-wi-fi-direct-you-need-to-know/

[11] Bluetooth Core Specification. Consultado en mayo de 2016 de https://developer.bluetooth.org/TechnologyOverview/Pages/core-specification.aspx

[12] Blue News, June 2016. Consultado el 8 de Junio de http://link.bluetooth.com/m/1/58195719/b15916-c6bbe1ec-a292-43bc-b0ed-289bc62041f7/25/924/b9503bd4-66ea-4261-9e38-48620b7b3c14

8. Anexos

En este punto añadido, anexos, se adjuntan alguno de los bocetos realizados en el proceso de concepción de la interfaz de la aplicación móvil.

En las imágenes que se exponen a continuación se presenta un concepto inicial donde además de representar el propio juego desarrollado se plasma también la idea de tener una plataforma que englobe los juegos que se vayan realizando a modo de acceso de directo. Esto último es una de las ampliaciones que se proponen en las conclusiones de este proyecto.

Así pues, aquí están las capturas.

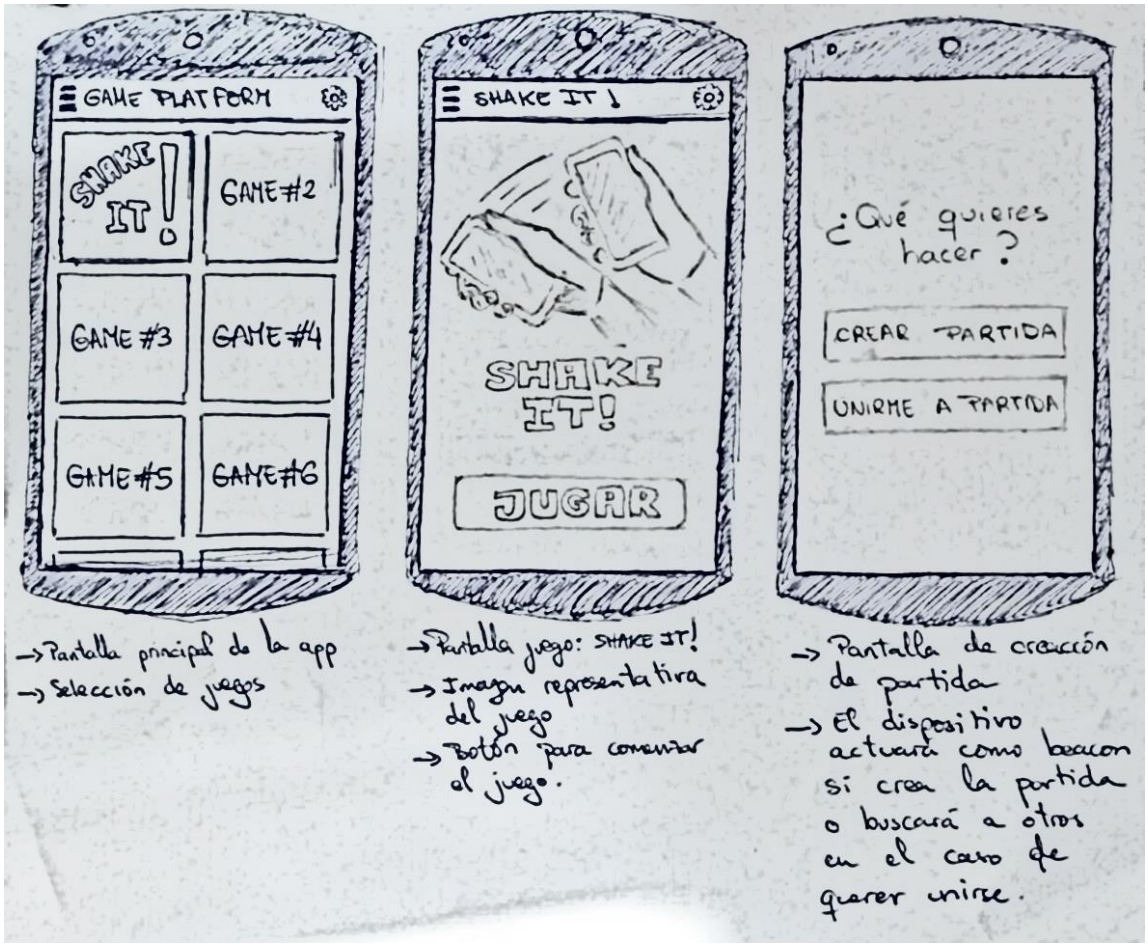

ILUSTRACIÓN 44: BOCETOS INICIALES 1

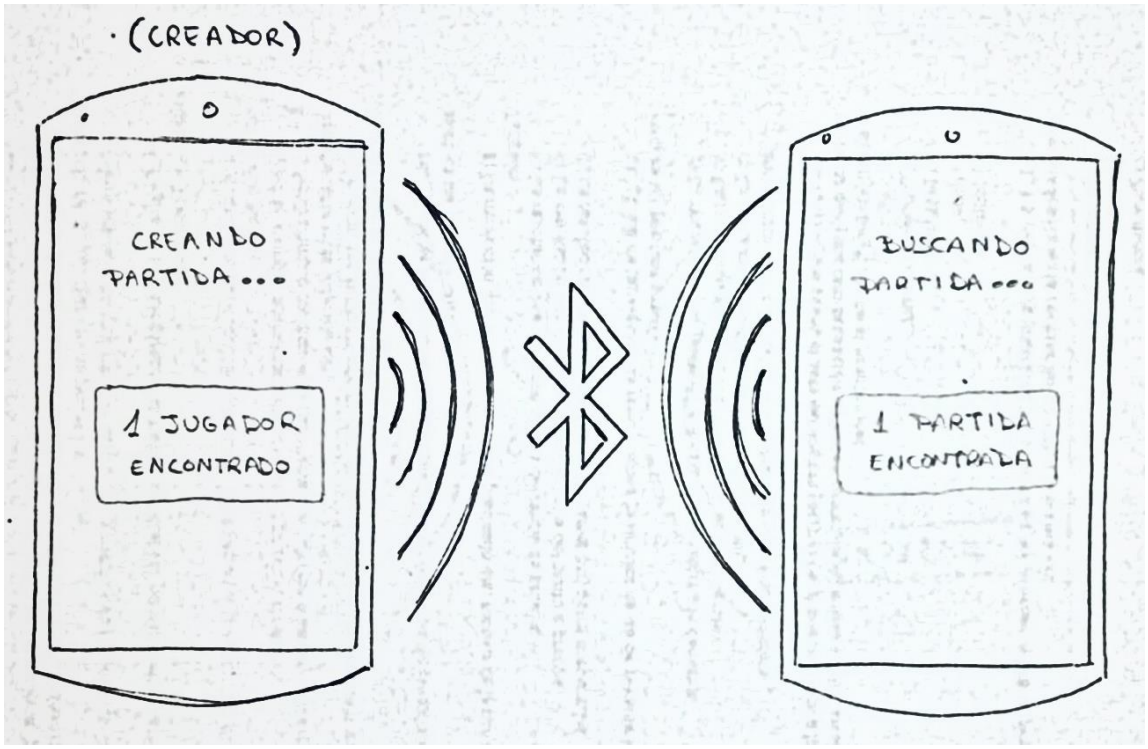

ILUSTRACIÓN 45: BOCETOS INICIALES 2.

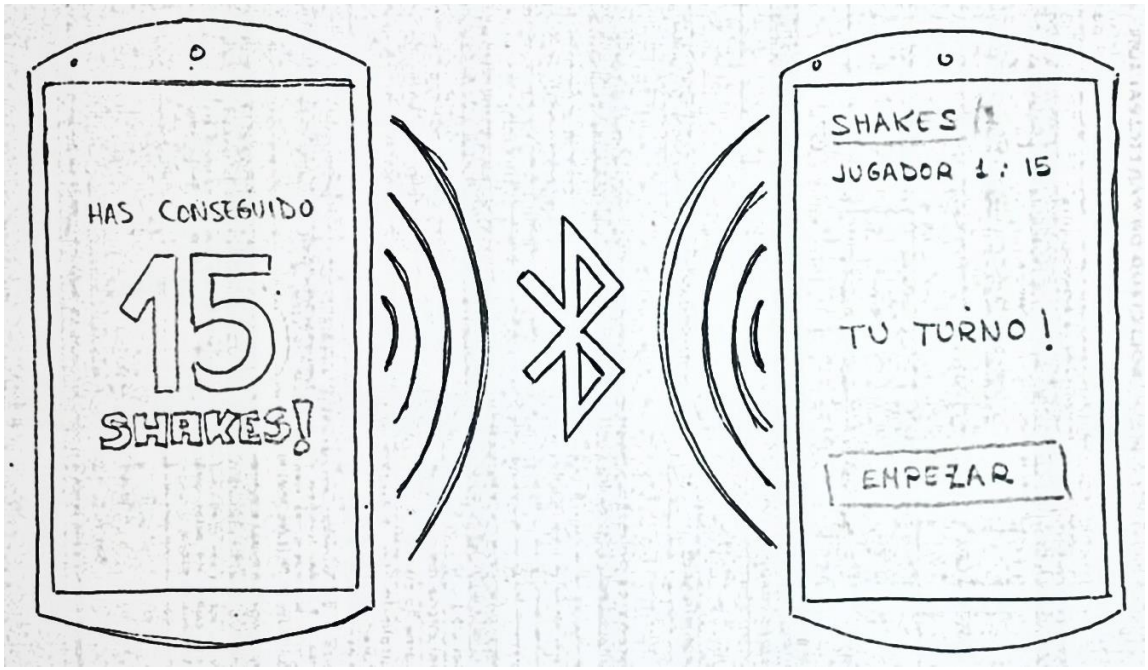

ILUSTRACIÓN 46: BOCETOS INICIALES 3.