UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Diseño de un sistema universal de bajo coste para evitar olvidar bebés en vehículos usando tecnologías móviles y balizas Smart Bluetooth.

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Autora: **Lorena López Nortes**

Directora: Dra. Nieves Pavón Pulido

Codirector: Dr. Juan Antonio López Riquelme

Cartagena, 08 de Abril de 2019

Contenido

Tabla de Ilustraciones

Índice de tablas

Resumen.

En este trabajo de final de grado se tratará de encontrar una solución simple y económica al olvido de bebés por parte de padres o tutores, fundamentalmente en lugares que puedan suponer un peligro, tales como un vehículo. Para ello, se empleará una pequeña baliza que funciona usando el protocolo Bluetooth de Baja Energía (BLE), también denominada iBeacon, un dispositivo móvil inteligente o smartphone y una aplicación especializada.

El sistema desarrollado que se explica detalladamente en los capítulos siguientes permite que, a través de la colocación de un iBeacon en la sillita para coche del bebé, ésta se convierta en un elemento activo de seguridad del vehículo, de modo que se evite el olvido del bebé a través de la información que se envía al smartphone vinculado al iBeacon. Para ello, la baliza enviará a la aplicación del dispositivo móvil una señal de BLE con información relevante que incluye la identificación del iBeacon, para diferenciarlo de otros dentro del mismo rango, o la temperatura, entre otros aspectos importantes. Tal información sólo será recibida por el smartphone cuando entre dentro de su alcance. Además, en ausencia de información recibida (pérdida de la señal de la baliza), si el usuario no notifica a la aplicación que el bebé ha sido extraído de la silla, la aplicación detectará un posible olvido, avisando adecuadamente al usuario o a un conjunto de personas que podrían asistir al bebé, en caso de que el usuario principal no atienda la alarma lanzada.

El prototipo que se ha diseñado y verificado muestra unos resultados prometedores, ya que el sistema funciona adecuadamente, es adaptable a cualquier vehículo y a cualquier tipo de sillita, es fácil de utilizar y su coste es reducido.

Además de considerar, en este documento, los resultados obtenidos y las ventajas e inconvenientes del sistema desarrollado en comparación con otros existentes, se plantean una serie de trabajos futuros que permitirían mejorar el sistema propuesto.

Capítulo 1. Introducción, motivación y objetivos.

En este capítulo se describe cuál es la motivación que ha movido a la autora a diseñar el sistema propuesto, así como los objetivos que se persiguen.

La idea principal es desarrollar un sistema que evite a los usuarios de cualquier vehículo olvidar a un bebé que va correctamente en su sillita, usando una sencilla baliza que emite una señal de BLE, capaz de portar información relevante, que es leída por medio de un dispositivo móvil inteligente, y que permite emitir una alarma si el usuario no notifica que ha extraído al bebé del coche y ambos dispositivos (la baliza y el Smartphone), se desvinculan porque uno de ellos no se encuentra dentro del rango previsto.

1.1. Introducción.

En la actualidad, es innegable la sobreinformación a la que se somete cualquier persona cada día, los padres no son una excepción, y muchas veces eso provoca que desatiendan en algún momento a sus hijos, es el llamado con el término inglés "multitasking" o multitarea (que podríamos considerar como una traducción aproximada del concepto). El "multitasking" implica la realización de varias tareas a la vez. En el ejemplo que nos ocupa, los progenitores o tutores pueden estar pendientes de muchas tareas, tanto en el ámbito laboral como cualquier otro. En cualquier caso, muchas de las tareas que se realizan implican la consulta d un dispositivo móvil inteligente. En estas circunstancias, la realización de varias tareas simultáneas puede interferir en el correcto cuidado de los niños, en diferentes lugares.

El smartphone es el inseparable compañero de casi el total de la población y esto, aunque implica ventajas también trae consigo algunos inconvenientes. Entre ellos, pasar muchísimo tiempo frente a la pantalla, el uso de ciertos dispositivos (tabletas y móviles), como soborno a los más pequeños a la hora de la comida, la falta de concentración en la tarea que se está realizando o la imposibilidad de desconectar, aunque sea por poco tiempo. De hecho, el uso del smartphone puede llevar a los progenitores o tutores a descuidar determinados aspectos relacionados con el cuidado de los más pequeños.

Por otro lado, es cierto que el uso de dispositivos móviles también implica numerosas ventajas y detrás de gran parte de ellas están las numerosas aplicaciones que pueden utilizarse de forma sencilla para que ciertas tareas cotidianas sean más sencillas. Estas pueden ser de muy distinto tipo, desde una que envíe recordatorios de citas de cualquier tipo, aplicaciones para mejorar la comunicación y la conectividad entre familiares y amigos, las que fomentan

el buen uso de las redes sociales, cualquiera que permita llevar a cabo gestiones de forma rápida, tales como comprar los vuelos de la próxima escapada o muchas otras. Todas ellas ofrecen la posibilidad de realizar gran cantidad de acciones que hasta hace poco era impensable llevar a cabo sin un ordenador. Sin embargo, como ya se ha comentado, incluso las aplicaciones que pueden ayudar a resolver gran cantidad de tareas de manera eficaz, pueden distraer si se utilizan en momentos inadecuados (por ejemplo, durante la conducción de un vehículo), y otras muchas simplemente bombardean con información de poca o nula importancia o incluso alejan a las personas de su papel de buenos progenitores o tutores.

Conseguir aplicaciones que ayuden a ser mejores padres, tutores y/o cuidadores es esencial en una sociedad avanzada que tiene tanta dependencia de la tecnología. De modo que, si se evitan los inconvenientes mencionados y los desarrolladores se centran en diseñar aplicaciones que explotan las ventajas comentadas, es posible conseguir que el dispositivo móvil inteligente sea un aliado eficaz en la mejora del cuidado de los más pequeños o de los más débiles y dependientes.

La aplicación que se describe en este trabajo, como parte del sistema que se ha desarrollado, podría es un ejemplo de aplicación para dispositivos móviles inteligentes que ayuda a los usuarios a una mejor conciliación entre vida familiar, laboral y ocio.

1.2. Motivación y objetivos.

Son numerosos, por desgracia, los casos que se han dado, a lo largo del tiempo, de olvidos de bebés dentro del coche o en casa; de padres que, por despiste o prisas, han olvidado coger a sus hijos al salir de casa o se han olvidado de que los llevaban consigo al bajar del coche. En este último caso, puede resultar mortal, tanto en verano, cuando el interior de un coche puede alcanzar temperaturas incompatibles con la vida, como en invierno, cuando las temperaturas pueden ser extremas, pero en sentido contrario.

El caso más reciente en España ocurrió el pasado 5 de octubre, al olvidar un padre a su hija dentro del coche en lugar de dejarla en la guardería como de costumbre. Esto resultó mortal para la pequeña que pereció por deshidratación antes siquiera de que sus progenitores se percataran de la ausencia de la niña en la guardería.

Entristece pensar que algo tan fácil de evitar pueda tener consecuencias irreversibles. Ya en la escuela infantil de la pequeña sugerían la idea de un sistema para alertar a los padres de la ausencia de sus hijos en las instalaciones, pero esto podría no ser suficiente. Es precisamente

en proporcionar un modo de prevención contra estos desastres en lo que se centrará el desarrollo de este trabajo.

Nuestros objetivos serán:

- ~ Diseñar tanto la parte software como hardware de un sistema anti-olvido de bebés basado en la tecnología Bluetooth Low Energy o BLE.
- ~ Estudiar las técnicas utilizadas hasta ahora para evitar olvidar a los niños en el coche, así como sus características.
- ~ Estudiar todos los posibles componentes del sistema que se pretende diseñar y la selección de los mismos.
- ~ Desarrollar la aplicación que formará parte de dicho sistema.

1.3. Descripción resumida de capítulos.

En este apartado se resume brevemente el contenido de los capítulos que componen el trabajo.

1.3.1. Capítulo 1: Introducción, motivación y objetivos

Aquí se presenta la idea que se va a desarrollar, se resume el contenido de los siguientes capítulos y se habla sobre la causa que ha provocado la propuesta de la solución.

1.3.2. Capítulo 2: Estado del Arte

En el capítulo 2 se exponen todos los sistemas que podrían usarse para evitar el olvido de bebés en el coche, así como las distintas tecnologías empleadas para la localización, las comunicaciones inalámbricas o la transmisión de información de forma también inalámbrica. Se describen, de igual modo, los componentes de un smartphone y los sistemas operativos móviles que normalmente permiten el funcionamiento de estos dispositivos móviles inteligentes. Además, se explica el porqué de cada una de las elecciones realizadas durante el trabajo, considerando las ventajas sobre otras alternativas tecnológicas.

1.3.3. Capítulo 3: Diseño del sistema

En este capítulo se detalla el desarrollo del sistema integrado por los componentes anteriormente seleccionados y se profundiza en cada uno de ellos. Se muestra el "storyboard" y se explican todas las actividades que intervienen en la aplicación, así como la relación entre las mismas.

1.3.4. Capítulo 4: Implantación y prueba del sistema

En el capítulo 4 se diseña en primer lugar una batería de pruebas para relatar después lo acontecido en cada una de ellas y las correcciones realizadas en consecuencia.

1.3.5. Capítulo 5: Conclusiones y trabajo futuro

Finalmente, en este capítulo se enumeran las conclusiones a las que se ha llegado tras la realización del trabajo y se proponen posibles mejoras y acciones adicionales.

Capítulo 2. Estado del arte.

En este capítulo se expondrán las distintas tecnologías existentes capaces de resolver el problema planteado. Se comenzará por los sistemas de protección para la infancia que han sido comercializados hasta nuestros días en lo referente a vehículos, pues lo que se persigue en este trabajo es precisamente el desarrollo de un sistema de este tipo.

Seguidamente, se estudiarán las diversas tecnologías usadas en sistemas de comunicación, así como la aplicación de estas en las comunicaciones inalámbricas destinadas a localización. Este apartado tratará tecnologías como WPAN, las distintas versiones de Bluetooth, ZigBee o Ultra Wide Band. Se profundizará, además, en los sistemas de posicionamiento local o LPS dado que son los más convenientes para nuestro fin.

A continuación, se verán ejemplos de sistemas que previenen del olvido de objetos cotidianos haciendo uso de las tecnologías hasta ahora descritas. Se pasará entonces a evidenciar sus carencias reforzando, de este modo, la elección de las balizas iBeacon de Estimote como solución del presente caso.

2.1. Sistemas comerciales de protección para la infancia (anti-olvido).

Actualmente existen varios sistemas disponibles en el mercado que resultan útiles a la hora de no olvidar a niños en el coche. Los más simples son las aplicaciones para móviles. A continuación, se describen brevemente algunas de estas aplicaciones:

- **Waze,** no está especializada para este tipo de problema, pero permite añadir a sus funcionalidades el envío de un mensaje con cualquier recordatorio una vez lleguemos a nuestro destino, como un mensaje que nos recuerde que viajamos con niños, una mascota o un anciano, por ejemplo.
- **Kars4Kids** hará sonar una alarma cuando se abandone el coche, obligando así a mirar atrás por si se viaja con niños, en este caso, para detectar que se deja atrás el coche se usará Bluetooth. Se puede configurar además para que suene a una hora determinada o para que no suene si se viaja solo.
- **Precious Cargo for Parents & Kids**, en este caso la propia aplicación pregunta al arrancar el coche (tras haberse conectado a éste por Bluetooth), si se viaja solo o se lleva algún niño como acompañante. En el supuesto de que se viaje con un niño se preguntará el nombre de éste y al apagar el coche se enviará una alerta al móvil.

También existen algunos otros dispositivos más completos que intentar poner solución a este problema:

- **Sensorsafe** es un sistema de monitorización de la marca alemana Cybex que incorpora a sus sillitas de coche un sensor en el clip del cinturón que al estar conectado por Bluetooth podrá enviar información al teléfono de los padres gracias a la correspondiente aplicación. La información la envía en caso de una situación de emergencia, como puede ser, que el niño se quede solo en el coche, o que la temperatura del coche sea demasiado alta o baja e incluso alerta si se ha conducido más de 2 horas seguidas sin descansar.

  La filial de Cybex en Estados unidos comercializa otro sistema parecido al **Sensorsafe** pero que en lugar de por Bluetooth se conecta directamente al puerto OBD del coche y ambos se comunican por una señal de radio estándar. Dará la voz de alarma en caso de que cerremos el coche y la sillita del niño siga abrochada o si el niño se suelta mientras conducimos, ver Ilustración 1.

Ilustración 1. Sensorsafe de Cybex.

- **Remmy Car Baby Alert** es un dispositivo universal que puede utilizarse en cualquier sillita infantil o asiento de coche. Basta con enchufar el dispositivo a la toma de mechero del coche y colocar el sensor de presencia bajo la tela del forro del asiento. Cuenta con dos funciones principales, una que nos avisa si el niño se desabrocha el cinturón o se levanta de la silla en marcha y otra que nos avisa si al bajar del coche el niño permanece dentro. El sistema está preparado para que no pueda ser silenciado, y en el caso de que el sensor se salga del asiento o se caiga también avisará.

- **Sunshine Baby iRemind**, es una almohada muy sensible y suave que está diseñada para ser colocada bajo el asiento del coche, detecta peso a partir de media kilogramo y por ello al colocar encima al bebé emitirá un sonido para que sepamos que está activado y procederá a la sincronización con nuestro teléfono móvil o llavero especial. Cualquiera de los dos emitirá una alarma si nos alejamos más de 5 metros del coche.
- **Driver's Little Helper** es otro dispositivo similar que en el caso de emitir la alerta de que nos estamos alejando del coche y que no se responda ante ella comenzará a llamar a contactos de nuestra agenda.

La diferencia entre las tres aplicaciones para móvil descritas es, prácticamente, nula en cuanto a funcionamiento, quizá la mayor diferencia la presenta Waze, que por no ser una aplicación especializada para estos fines no cuenta con más que un recordatorio que no comprueba información con ningún tipo de sensor o dispositivo. En cuanto a las otras propuestas se aprecia en la

Tabla 1 que, en esencia, el funcionamiento es el mismo para todos los casos, diferenciándose entre sí por los apoyos físicos de los que obtienen la información necesaria para el envío de unas u otras notificaciones y por el precio.

Nombre	Tecnología	¿APP?	Hardware	Tipo de alerta	Precio
SensorSafe BT u OBD	Bluetooth	Si	Cinturón incorporable a la sillita del niño	Notificación si se desabrocha, si hay temperatura peligrosa o si se conducen más de 2 horas seguidas	50€
Remy Car Baby Alert	Sensor de presión bajo sillita	No	Sensor de presión, toma de mechero, buzzer	Alarma a través del buzzer si el niño se desabrocha o bajamos del coche sin él	79€
Sunshine Baby	Bluetooth	Si	Almohada con sensor	Notificación en móvil o en llavero especial	100€
Driver´s Little helper	Bluetooth	Si	Sensor de peso	Notificación en móvil	80€

Tabla 1. Comparación de los dispositivos anti-olvido de bebés.

A parte de las soluciones descritas, que pueden ser adquiridas independientemente de la sillita o del coche que tengamos, es interesante comentar que, actualmente, en algunos automóviles encontramos ya algunas funcionalidades que pueden resultar útiles para estos propósitos.

Un ejemplo es la protección de sobrecalentamiento en el habitáculo diseñada por Tesla para garantizar la seguridad de niños y mascotas en el habitáculo, y que consigue mantener una temperatura adecuada durante horas dentro del vehículo incluso estando apagado.

Por su parte, Hyundai incorpora el sistema de alerta de ocupantes de la parte trasera a partir de un sensor ultrasónico que es capaz de detectar el movimiento de un niño, alerta a través de la bocina o el smartphone conectado mediante el sistema Blue Link [1].

2.2. Dispositivos móviles inteligentes.

La traducción literal del término inglés "smartphone" es teléfono inteligente. La diferencia con los primeros teléfonos convencionales es la posibilidad de realizar acciones adicionales a enviar y recibir llamadas y mensajes en un ambiente completamente móvil. De hecho, posibilitan la realización de numerosas tareas para las que antes necesitábamos varios dispositivos diferenciados como, por ejemplo, el acceso a Internet, la reproducción de música y video, el envío de emails o la consulta de nuestra agenda entre muchísimas otras funciones.

Para ello, es necesario un sistema operativo (OS, del inglés Operating System) móvil, es decir, un programa que hace posible la realización de todas las tareas anteriormente descritas. Existen hoy día varios OS móviles de varias empresas, los principales son iOS de Apple, Windows Phone de Windows y Android de Google, que es el más popular y lo implementan la mayoría de las empresas, ver Ilustración 2.

Ilustración 2. Sistemas operativos móviles.

2.2.1. Sistemas operativos móviles

En este apartado se describen los principales sistemas operativos móviles mencionados.

2.2.1.1 Windows Phone.

Es el sistema operativo que Microsoft desarrolló para sustituir a Windows Mobile e intentar volver a ser competitivo en el mundo de los móviles. Apareció en 2010 y a pesar de ser el sucesor directo de Windows Mobile no es una actualización de la última versión de éste, sino que se diseñó prácticamente desde cero, presentando una interfaz totalmente nueva y un mejor comportamiento, así como un mayor control sobre las plataformas de hardware que lo ejecutan. Además, el sistema operativo ideado por Microsoft al tomar este nuevo nombre se hace incompatible con su predecesor Windows Mobile, haciendo así obligatorio el cambio a un nuevo terminar si se desea contar con la última versión [2].

La primera versión de este sistema operativo recibió el nombre de Windows Phone 7, ya que la versión de Windows Mobile anterior era la 6.5, después de esta versión vinieron algunas más, tales como Windows Phone 7.5 Mango, que incorporaba ya Internet Explorer 9 y la integración de Twitter con el acceso de Windows Live SkyDrive, Windows Phone 7.5 Tango que es una versión más asequible que Mango, Windows Phone 7.8 y Windows Phone 8 [3].

2.2.1.2 iOS

Es la propuesta de sistema operativo móvil por parte de la empresa Apple. Es una versión del sistema operativo que implementan los ordenadores de Apple pero que está diseñada para ser usada solamente en sus dispositivos móviles táctiles, es decir, iPhone, iPad y iPod touch. Se basa en el sistema operativo Unix y se presentó por primera vez en 2007 como sistema operativo diseñado en exclusiva para el primer iPhone. Permitiría a este último la capacidad de trabajar sin teclado físico, de acceder a navegar por Internet mediante Safari y de almacenar gran cantidad de canciones, tal y como ya permitía iPod. Cuenta además con actualizaciones periódicas que están disponibles para su descarga a través de iTunes, que es el software gratuito e indispensable para manipular y sincronizar cualquier archivo en estos dispositivos.

Una de las mayores diferencias que presenta frente a los demás sistemas operativos móviles es que tanto iOS como los demás sistemas operativos usados en los dispositivos de Apple son propietarios de la empresa y solo se usan en el mercado en los dispositivos para los que fueron creados.

Por otro lado, lo más valorado por los usuarios de este sistema operativo móvil es la capacidad de trabajar con múltiples programas a la vez y en segundo plano sin sacrificar con ello la eficiencia y la rápida respuesta de los dispositivos (multitarea).

Todas las aplicaciones para iOS se programan en Swift usando IDE Xcode [4].

2.2.1.3 Android

Es el sistema operativo móvil más utilizado en el mundo, con una cuota de mercado de más del 90%. Esto se debe a varios factores entre los que destaca que es totalmente libre, gratuito y multiplataforma por estar basado en el sistema operativo Linux, que también lo es. Además, como consecuencia de esto, puede ser usado modificado o mejorado por cualquiera que tenga la intención de hacerlo.

Las aplicaciones para este sistema operativo se programan en una variación de Java llamada Dalvik. Existen, asimismo, múltiples herramientas de programación gratuitas que, unidas a la sencillez de programación por ser Java un lenguaje de programación de uso ampliamente extendido, dan lugar a una cantidad de aplicaciones disponibles casi infinita. Estas pueden descargarse desde Play Store principalmente, aunque también cualquier desarrollador particular puede ofrecerlas por sus propios medios.

Este sistema operativo móvil, aunque existía desde bastante antes, había pasado desapercibido hasta que, en 2005, fue comprado por Google. Aun así, no fue hasta 2007 que fue presentado y apareció independientemente de cualquier teléfono móvil, lo que provocó que al principio la aceptación fuese lenta. Existen varias versiones las cuales tienen nombre de postres en inglés. Las últimas son Oreo 8.0 y la más reciente: Pie 9.0 [5], ver Ilustración 3.

Ilustración 3. Versiones de Android.

A la hora de optar por uno u otro sistema operativo móvil en el que desarrollar nuestra aplicación, lo que resulta más determinante es la cantidad de gente a la que pueda llegar nuestro trabajo, la existencia de plataformas libres en las que poder desarrollar las aplicaciones y la simpleza de uso de las mismas, así como del sistema operativo una vez se esté manejando el móvil.

Atendiendo a la mayor distribución posible de la aplicación podemos fijarnos en la cuota de mercado más reciente encontrada o en la predominancia por países de cada sistema operativo, ver Ilustración 4 e Ilustración 5.

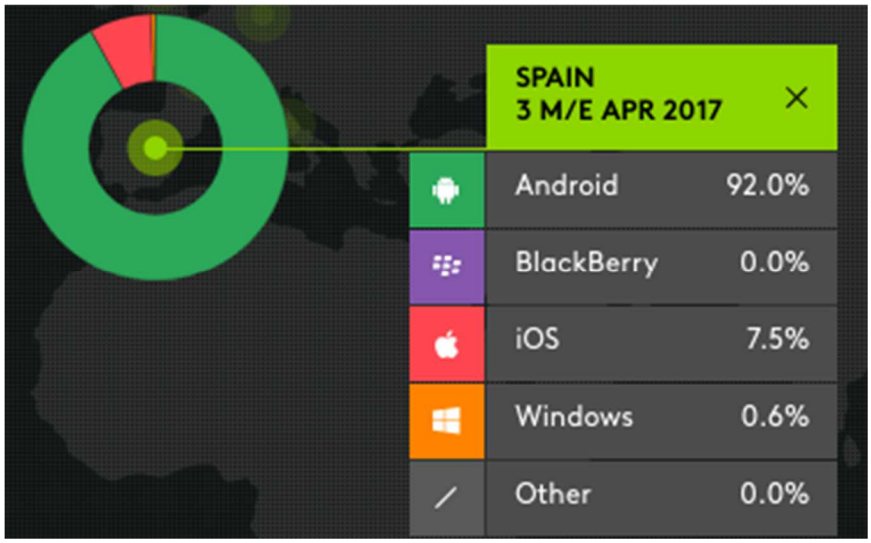

Ilustración 4. Cuota de mercado según OSM.

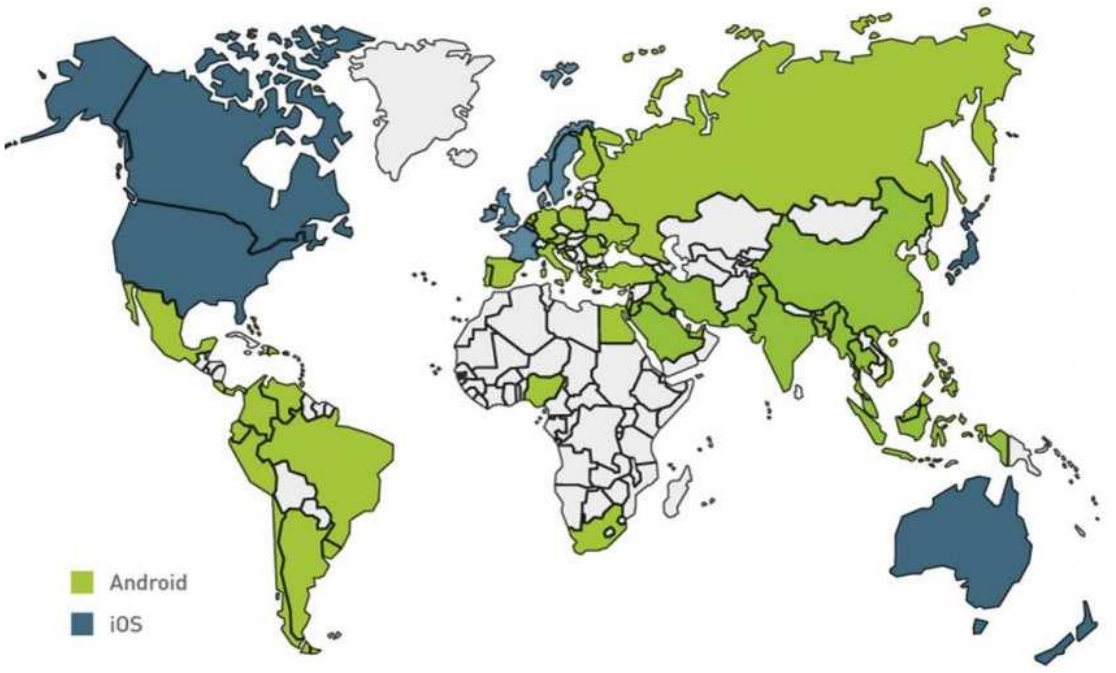

Ilustración 5. Sistema operativo móvil más usado por países.

Puede observarse, claramente, la predominancia de Android frente a cualquiera de los otros dos OS.

En cuanto al desarrollo de aplicaciones para uno u otro sistema operativo, en el caso de Windows Phone el lenguaje usado puede ser C++, C# NET, Visual Basic NET o JavaScript, todos ellos lenguajes bastante conocidos lo que no supone un inconveniente entonces para competir con Android o iOS. Estos últimos trabajan con Java y Swift respectivamente, lenguajes también ampliamente extendidos. Una de las diferencias fundamentales entre iOS y sus dos contrincantes es que, en su caso, las aplicaciones que se diseñen en este sistema operativo solamente podrán ejecutarse en dispositivos de esta marca que son parecidos entre sí y totalmente compatibles. No ocurre esto en el caso de Windows Phone, cuya propuesta puede ser compilada por distintos dispositivos de varias marcas, ni con Android, que es el más versátil de los tres.

Esta versatilidad puede resultar provechosa por el hecho de llegar a más dispositivos, pero es un inconveniente a la hora de diseñar aplicaciones, pues para que un diseño pueda adaptarse a la mayoría de dispositivos posibles tiene que estar muchas veces simplificada, lo que le puede restar atractivo visual entre otros aspectos.

A pesar de ello, en este trabajo se elegirá Android como opción definitiva por presentar más ventajas que inconvenientes.

2.2.2. Componentes de un dispositivo móvil inteligente.

Aunque los smartphones son entre sí muy distintos unos de otros dependiendo de la marca, del modelo, del precio o de la gama, en esencia cuentan siempre con los mismos componentes. Son estos los que se describen a continuación:

2.2.2.1 Procesadores ARM.

ARM es la empresa gracias a la cual ha sido posible el rápido desarrollo de los dispositivos móviles en estos últimos años. Los procesadores que fabrica se basan en el modelo RISC (Reduced Instruction Set Computer) y se introdujo en el mercado en el año 1985 con el primer modelo de procesador ARM. Este ha ido evolucionando año tras año hasta convertirse en nuestros días en la arquitectura de 32 bits más exitosa del mundo.

El microprocesador es el elemento más importante de un smartphone, así que si se aumenta su eficiencia también aumentará en consecuencia la del smartphone. Es, por ello, que últimamente la tendencia es aumentar el número de procesadores con los que cuenta cada

teléfono móvil. Esto se hace porque los procesadores son más eficientes cuanto menor es su tamaño debido a que, a menor tamaño, menor calor y menor consumo eléctrico, lo que, por extensión, indicará que dos procesadores de 45nm serán más eficientes que uno de 90nm.

ARM fabrica normalmente una estructura System-on-a-chip (SoC), que se refiere a la tendencia cada vez más frecuente de diseñar un único circuito integrado que contenga todos o casi todos los componentes de un ordenador o cualquier otro tipo de dispositivo informático. Esto tiene como finalidad que todos los componentes críticos de un dispositivo se encuentren en una misma zona relativamente pequeña.

2.2.2.2 Unidad de procesamiento gráfica.

Las siglas GPU se corresponden con Graphics Processing Unit, o lo que es lo mismo, coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante. Su función principal es trabajar en conjunto con la unidad central de procesamiento (CPU) aligerando así la carga de trabajo de ésta en videojuegos o aplicaciones 3D interactivas.

Esta unidad posee gran cantidad de unidades funcionales que se pueden dividir en dos grupos, las que procesan píxeles y las que procesan vértices, siendo estos dos las principales unidades con las que trabajará la GPU. Otra parte importante de una GPU es su memoria, que destaca por su rapidez.

El uso de la GPU depende tanto del sistema operativo como de la estructura del SoC. Por ejemplo, en el caso del SoC, la GPU entrará en juego cuando este no posea un chip destinado a la decodificación de vídeo. En el caso del sistema operativo será siempre la GPU la que se encargue de la renderización 3D en juegos y aplicaciones, ya que los núcleos de procesamiento de la CPU no sirven para tal fin.

Es interesante comentar que uno de los mayores problemas que Android encuentra es conseguir el uso de la GPU para la interfaz de usuario y la experiencia de navegación, problema que se solventó a medias a partir de la versión 4.0 gracias a su soporte 2D por hardware, que presentaba mayor potencia en la GPU por ser los SoCs más modernos. Sin embargo, no ocurre esto en el caso de iOS, que al tener que trabajar solamente con un hardware específico resuelve este asunto sin mayores dificultades.

2.2.2.3 Memoria RAM.

La memoria RAM o Random Acces Memory es uno de los componentes más importantes de un smartphone. En ella se almacenan todos los datos a los que se necesita acceder

rápidamente para un funcionamiento normal de nuestro teléfono y puede ser escrita o leída en cualquier momento. En realidad, en los smartphones se utiliza habitualmente una memoria RAM dinámica o DRAM, que posee una estructura en la que cada condensador de la placa de la RAM almacena un bit y estos condensadores son refrescados constantemente.

Lo más importante de este tipo de memoria es su capacidad, ya que su aporte al consumo de energía es mínimo en comparación con el de la pantalla o el procesador. Además, un smartphone común no suele utilizar simultáneamente grandes cantidades de memoria RAM por lo que es posible que estén funcionando a la vez varias aplicaciones distintas de manera fluida aunque la capacidad de la RAM no sea muy grande.[6]

2.2.2.4 Sensores y otro hardware

Además de todo lo descrito hasta ahora los smartphones, en su mayoría, cuentan también con una o varias cámaras, GPS, acelerómetro, giroscopio, brújula, sensor de luz y sensor de proximidad, así como con una pantalla táctil.

2.3. Tecnologías usadas para sistemas de comunicación en procesos de búsqueda y localización.

Cualquiera de las aplicaciones y métodos que se acaban de exponer parten de la base de alertar de que un objeto que tienen localizado sale del rango en el que debería mantenerse. La detección de que se está saliendo de este rango es posible gracias a alguno de los protocolos inalámbricos existentes. En concreto, todas las aplicaciones y dispositivos anteriormente descritos se valen de Bluetooth para alcanzar sus objetivos, pero esto no quiere decir que sea esta la única tecnología capaz de satisfacer dichas necesidades. Es por ello que, en las siguientes secciones, se describen algunas alternativas.

2.3.1. Redes inalámbricas de área personal WPAN.

Las redes inalámbricas de área personal se basan en el estándar IEEE 802.15 y permiten la comunicación en un rango de distancias bastante corto, en torno a 10 metros. Una conexión mediante este tipo de redes no requiere habitualmente ningún soporte físico y esto supone una importante ventaja que le permite implementar soluciones pequeñas, de bajo coste y eficientes en términos energéticos en múltiples dispositivos como pueden ser tabletas y smartphones.

Por otra parte, dos de sus principales puntos fuertes son el bajo consumo de energía y la baja velocidad de transmisión. Estas redes inalámbricas se basan en tecnologías tales como Bluetooth, IrDA, ZigBee o UWB. Estas, atendiendo a la aplicación podría decirse que Bluetooth está pensado para la conexión de dispositivos como un teclado inalámbrico o un altavoz portátil; IrDA se ideó para enlaces punto a punto entre dos dispositivos para que sea posible de este modo la transferencia de datos simples y sincronización de archivos; ZigBee está destinado a las redes inalámbricas fiables para el seguimiento y control de procesos, mientras que UWB se creó para enlaces multimedia de gran ancho de banda.

2.3.1.1 Bluetooth.

La tecnología Bluetooth es a día de hoy una de las más usadas por el tipo de dispositivos cuyo estudio nos ocupa. Esto es debido a su bajo coste, bajo consumo de energía y velocidades de transmisión de datos más que aceptables para los fines que se pretenden. Con el transcurso de los años se han ideado mejoras a este protocolo que apareció en 1998 las cuales buscaban potenciar unos u otros puntos según la aplicación para la que estaban concebidas.

- **Bluetooth Clásico**

Bluetooth forma parte del estándar IEEE 802.15.1. Se diseñó en origen para conexiones punto a multipunto, de corto alcance, con dispositivos baratos, bajo consumo energético y eliminando el uso de cables y sustituyéndolos por una conexión ad hoc por radio. Actualmente se desarrollan componentes y sistemas habilitados para Bluetooth para una gama de aplicaciones adicionales. Los dispositivos que incorporan esta tecnología pueden clasificarse en tres grupos distintos según su alcance máximo: Clase 1 (rango de 100 metros), Clase 2 (rango de 10 metros) y Clase 3 (rango de 1 metro). La clase más utilizada es la segunda y cualquiera de ellas usa la banda de 2,4 GHz que permite que dos dispositivos dentro del mismo rango compartan hasta 720 Kbps de velocidad de transferencia.

Varios dispositivos Bluetooth agrupados constituyen una red denominada picored (piconet), esta puede estar formada por hasta 8 dispositivos activos en una relación maestro-esclavo (master-slave). El primer dispositivo de la red Bluetooth será el maestro, y todos los que se unan después pasarán a ser los esclavos. En teoría, el rango de una picored podría alcanzar los 100 metros, pero esto solo en condiciones ideales, el rango típico es de 10 metros.

La seguridad de este tipo de redes se garantiza mediante la codificación de cada enlace y la protección frente a interferencias y escuchas. Además, dos picoredes pueden unirse para

formar una más grande llamada red dispersa o scatternet (ver Ilustración 6), llegando a compartir uno de los elementos que en una sería maestro y en todas las demás, esclavo.

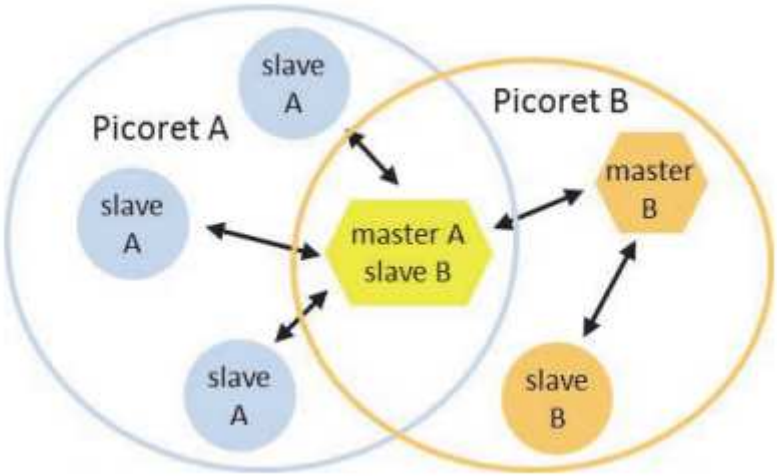

Ilustración 6. Red dispersa Bluetooth formada por dos picoredes.

- **<u>Bluetooth de baja energía BLE.</u>**

El Bluetooth de baja energía o BLE es un subconjunto del estándar Bluetooth v4.0 fue creado como resultado de una necesidad de reducir el consumo de su predecesor, el Bluetooth. Este protocolo ofrecía rangos de distancia de entre 1 y 30 metros, según la Clase a la que perteneciese, como ya hemos comentado, pero a mayor rango cubierto mayor era también la energía consumida y la potencia transmitida. Esta capacidad de cubrir un amplio rango resultaba por tanto en muchas ocasiones contraproducente pues por ejemplo, en la conexión manos libres por Bluetooth basta con unos 20 centímetros e incluso no es necesario que esta sea continua. Es por ello que ya en 2001 Nokia propuso una primera versión en lo que se conoció como Bluetooth Low End Extension que se convertiría tras el proyecto de investigación europeo en Wibree.

Al existir ya una alianza de fabricantes alrededor del estándar Bluetooth, el Bluetooth Special Interest Group (SIG) no tenía sentido la creación de una nueva especificación, así pues, los desarrolladores de Wibree se integraron dentro de las especificaciones de Bluetooth Low Energy en 2007 y BLE se integró dentro de la pila de protocolos de Bluetooth 4.0 en 2010.

Entre las grandes ventajas de esta tecnología podemos encontrar la aceptación obtenida por parte de las grandes plataformas a día de hoy, como pueden ser IOs, Android, Microsoft o Linux, además de la compatibilidad con ellas.

En cuanto a la pila de protocolos para Bluetooth Low Energy se tiene que se estructura como se observa en la Ilustración 7:

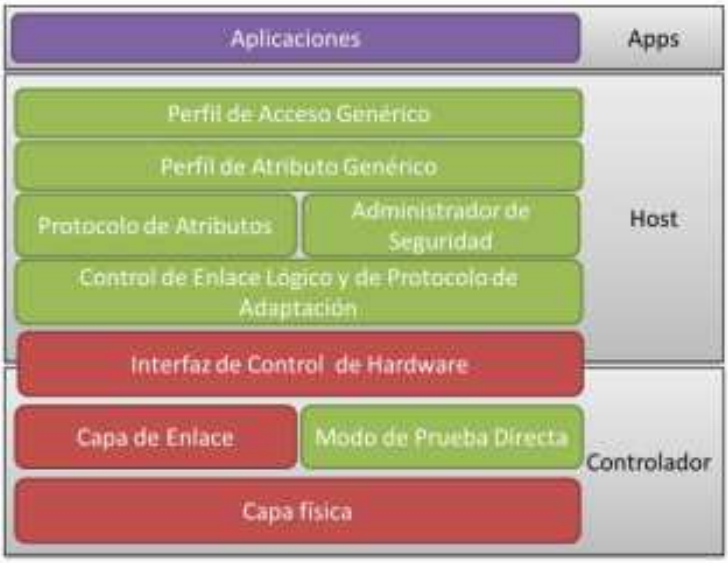

Ilustración 7. Pila de protocolos BLE.

- **Capa física**: es en la que se encuentra la circuitería de comunicaciones que posibilita los procesos de modulación y demodulación de señales analógicas que serán después transformadas en símbolos digitales.
- **Capa de enlace**: tiene dos funciones principales, gestionar las características tales como requerimientos temporales del estándar, chequeo de mensajes, gestión o filtrado de direcciones; y definir los roles (Advertiser, Scanner, Master and Slave)
- **Interfaz de Control de Hardware HCI:** es el protocolo estándar que posibilita la comunicación entre el Controlador y el Host
- **Control de enlace lógico y protocolo de adaptación**: es la que se encarga de dar formato a los mensajes que recibe de las capas OSI superiores y su posterior encapsulación en paquetes estándar BLE y el proceso inverso. También es la que se encarga de dar acceso y soporte a los dos protocolos fundamentales, que son el protocolo de Atributos (ATT) y el Administrador de Seguridad (SPM)
- **Perfil de Acceso genérico(GAP) y perfil de Atributo genérico(GATT)**: el GAP es el que se encarga de que el dispositivo sea visible para el resto y establece la forma de comunicación entre dos dispositivos. Por su parte, el GATT define como dos dispositivos BLE intercambiarán información cuando previamente han superado la fase de establecimiento de comunicación controlada por GAP.

Por otra parte, si se profundiza más en el funcionamiento del estándar se puede decir que es capaz de utilizar hasta 40 canales de 2MHz en la banda de 2.4 GHz. Además, emplea una técnica llamada "saltos en frecuencia" que consiste en interpretar una secuencia de saltos pseudo-aleatorios entre estos 40 canales que se han mencionado brindando así una gran robustez de cara a las posibles interferencias.

Todo esto sumado a unos costes de licencia nada intrusivos, la simplicidad de comprensión y la clara estructura de pila de protocolos muestran por qué es uno de los estándares que más auge experimental últimamente. [7]

- **Bluetooth 5.**

Es la quinta versión del protocolo de comunicación inalámbrica Bluetooth Classic creada con el fin de ampliar el rango alcanzado por las versiones anteriores además de mejorar la velocidad de transmisión de datos.

El aumento del rango de alcance en cuatro veces es especialmente útil en el caso de la conectividad con drones. Como novedad, en esta versión se posibilita la transferencia de audio a dos altavoces distintos al mismo tiempo, y es que esta revisión del protocolo inalámbrico se centra sobre todo en la mejora de la transmisión de archivos de audio.

Por otro lado, esta nueva versión hará posible que todos los dispositivos que cuenten con Bluetooth 5.0 puedan comunicarse mediante BLE.

A pesar de estas ventajas su adopción por parte de usuarios y empresas punteras en el sector es aún minoritaria [8].

2.3.1.2 IrDA

Estas siglas se refieren a la Asociación de Datos por Infrarrojos y conforman un conjunto de estándares para comunicaciones por infrarrojos utilizados para proporcionar conectividad inalámbrica a dispositivos que habitualmente utilizan cables para proporcionar conectividad.

IrDA es un estándar de transmisión de datos ad hoc diseñado para operar con distancias de hasta 1 metro y unidireccional. Es además de bajo consumo de energía y bajo coste, y trabaja con velocidades de 9600 bps a 4 Mbps aunque se está desarrollando los 16Mbps.

Esta tecnología se encontraba en muchos ordenadores portátiles y teléfonos de finales de los años 90 y principios del 2000, pero fue gradualmente desplazada por tecnologías como Wi-Fi y Bluetooth.

2.3.1.3 ZigBee

El término ZigBee describe un protocolo inalámbrico normalizado para la conexión de una Red de Área Personal Inalámbrico o WPAN, dicho protocolo está basado en el estándar 802.15.4 y fue desarrollado como un estándar global abierto que cumpliera con una alta fiabilidad, un bajo coste y consumo, una fácil implementación y bajas velocidades de transmisión de datos en redes de dispositivos inalámbricos.

Entre sus principales características podemos destacar la implementación global, pues ZigBee ha sido diseñado para la banda de 868 MHz en Europa, la banda de 915 MHz en Norte América o Australia y la banda de 2.4GHz que es una banda global aceptada en la mayoría de los países. Su velocidad de transmisión alcanza un máximo en 250Kbps, que es más que suficiente para suplir las necesidades de un sensor y de automatización cuando se usan redes inalámbricas.

ZigBee también está pensado para la creación de redes inalámbricas siempre que estas no exijan una gran cantidad de transmisión de datos. Una red ZigBee puede presentarse en 3 tipos diferentes de topologías (estrella, árbol o malla), ver Ilustración 8, y los dispositivos que la componen pueden ser de funcionalidad completa (FFD) o de funcionalidad reducida (RFD). Los de funcionalidad completa en concreto pueden operar en tres modos, que son: coordinador, dispositivo y coordinador de la WPAN. En contraposición, el RFD se diseñó solamente con fines muy simples, como puede ser un interruptor de luz.

Las diferentes topologías mencionadas se diferencian entre sí por las conexiones que permiten entre los dispositivos que las componen, por ejemplo, la topología en estrella la comunicación se establece entre los dispositivos y un controlador central único, mientras que en el caso de una topología en forma de malla cualquier dispositivo podrá comunicarse con cualquier otro, siempre que se encuentren uno dentro del rango del otro. La red en árbol es solamente un caso especial de la topología de malla en el que la mayoría de dispositivos son FFDs y cualquier RFD puede conectarse a la red como nodo "hoja" en el extremo de una rama. Todos los FFD pueden asumir la función de router y facilitar la sincronización a otros dispositivos y routers, siendo el coordinador de la WPAN uno de esos routers [9].

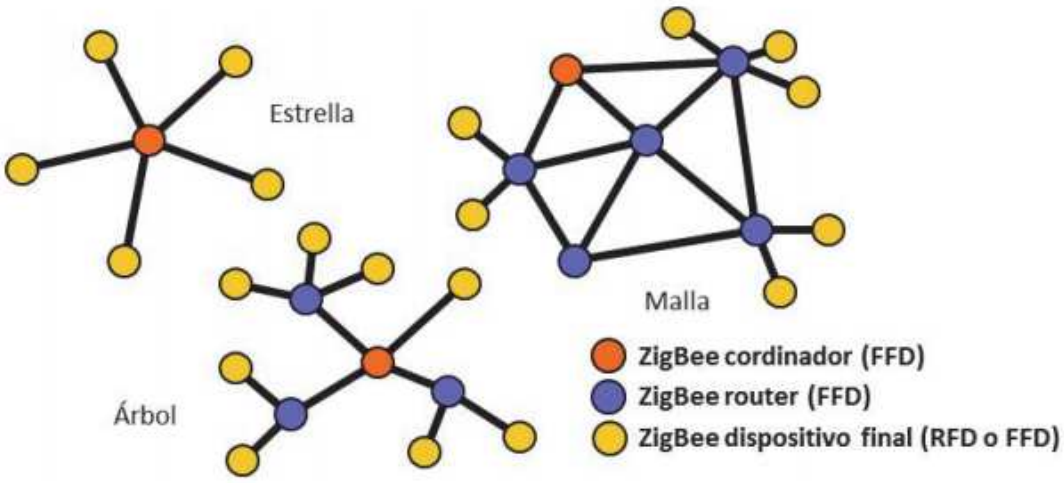

Ilustración 8. Diagrama de la estructura de una red ZigBee.

2.3.1.4 Ultra Wide Band

Este protocolo inalámbrico se basa en el estándar IEEE 802.15.3 y tiene un propósito distinto de los expuestos hasta el momento pues permite la transmisión de grandes archivos de datos a altas velocidades, pero en distancias cortas. La forma de transmitir información que usa esta tecnología es mediante la emisión de pulsos de muy corta duración y de gran ancho de banda.

UWB alcanza velocidades de transmisión de datos de más de 110 Mbps hasta 480 Mbps, pero no es capaz de transmitir estos datos entre dispositivos en el caso de que estos se encuentren a más de unos pocos metros. Esto, sin embargo, no supone un mayor impedimento para la mayoría de las aplicaciones multimedia de audio y video en el hogar.

Además, puede usarse como sustituto inalámbrico del cable de buses serie de alta velocidad como USB 2.0 y IEEE 1394 y las frecuencias en las que trabaja son de 3.1 GHz hasta 10.6 GHz, lo que supone una banda de más de 7 GHz de anchura. Es posible también compartir bandas de frecuencia entre dispositivos, lo que permite una alta productividad.

2.3.2. Redes Inalámbricas de Área Local WLAN

Este tipo de tecnología inalámbrica está diseñada para proporcionar acceso inalámbrico en zonas de hasta 100 metros. Las redes inalámbricas de Área Local se basan en el estándar 802.11 del IEEE y se comercializan bajo el nombre de WI-Fi, que es una abreviatura del término en inglés Wireless-Fidelity. Estas redes efectúan la transmisión de datos por medio

de radiofrecuencia, y es por ello que no es necesario el uso de cables. Una gran ventaja que presentan es la fácil, simple y rápida incorporación de nuevos equipos a la red.

En 1999 nació la Wireless Ethernet Compatibility Alliance o WECA, que es una asociación de varias empresas como 3Com, Nokia, Lucent Technologies o Symbol Technologies que tenían el propósito de suplir la falta de estandarización de normas u especificaciones referentes a las redes inalámbricas. Esta asociación pasaría en 2003 a llamarse Wi-Fi Alliance y desde entonces el número de empresas que se asocian a la misma ha ido aumentando.

Para la comunicación de redes sin cables basada en esta tecnología no fue necesaria la creación de ningún protocolo específico pues la WECA pasó a trabajar con las especificaciones del IEEE 802.11 que son básicamente los mismos que los del IEEE 802.3 salvo por el hecho de que este último, conocido por el nombre de Ethernet realiza las conexiones entre equipos mediante cables.

Otro de los aspectos a tener en cuenta es la posibilidad de coexistencia sin interferencia de más de una red Wi-Fi en el mismo espacio físico. Esto es posible gracias al SSID, que es el Service Set Identifier y que se compone de un conjunto de caracteres que se inserta en el encabezado de cada paquete de datos de la red identificándola así y permitiendo distinguirla de cualquier otra.

Los protocolos Wi-Fi que se han ido aprobando y actualizando a lo largo de los años son:

- **802.11:** La primera versión de este estándar fue lanzada en 1997 tras 7 años de estudios. El IEEE, al tratarse de una tecnología de transmisión por radiofrecuencia, determinó que el estándar pudiese operar entre 2.4 y 2.4835 GHz. Además, su tasa de transmisión de datos es de 1 o 2 Mbps y es posible utilizar las técnicas de transmisión Direct Sequence Spread Spectrum DSSS y Frequency Hopping Spread Spectrum FHSS
- **802.11b**: Esta es la primera actualización del estándar 802.11, fue publicada en 1999 y su principal característica era la posibilidad de establecer conexiones entre las velocidades de transmisión 1,2,5.5 y 11 Mbps. Se hizo necesario el uso de la técnica Complementary Code Keying CCK para trabajar de manera más efectiva con las velocidades de 5.5 y 11 Mbs. Además, el área de cobertura pasó a ser de 400 metros en exteriores y de 50 metros en entornos cerrados. Por otro lado, con el fin de mantener la transmisión lo más funcional posible, este estándar y los que vinieron tras él pueden hacer que la tasa de transmisión de datos disminuya hasta llegar a su

límite mínimo cuanto más lejos se esté del punto de acceso o que aumente hasta su límite máximo cuanto más cerca se esté del mismo.

- **802.11a**: Esta versión fue lanzada a finales de 1999, casi a la vez que 802.11b, y aunque ofrecía velocidades de transmisión mayores (6 Mbps, 9 Mbps, 12 Mbps, 18 Mbps, 24 Mbps, 36 Mbps, 48 Mbps y 54 Mbps) no se llegó a hacer tan popular como este. Su frecuencia de operación es distinta a la de los estándares predecesores, y es de 5 GHz, contando con canales de 20 MHz dentro de esa gama. Respecto a las técnicas de transmisión, en este estándar se hacía uso de la Orthogonal Frequency Division Multiplexing u OFDM en sustitución de las FHSS o DSSS.
- **802.11g**: Este estándar fue lanzado en 2003 y es totalmente compatible con el 802.11b, y considerado por tanto su sucesor natural. Su principal mejora es la posibilidad de trabajar con velocidades de transmisión de hasta 54Mbps
- **802.11n**: El desarrollo de esta especificación se extiende desde 2004 hasta septiembre de 2009, tiempo durante el cual se lanzaron dispositivos compatibles con la especificación inacabada. El atractivo principal de esta actualización reside en el uso de un esquema llamado Multiple-In Multiple-Out o MIMO y que es capaz de aumentar considerablemente las tasas de transferencia de datos por medio de la combinación de varias vías de transmisión. La frecuencia de transmisión puede ser de 2.4 o bien 5 GHz, lo que posibilita la compatibilidad con cualquiera de los anteriores estándares. Y respecto a la técnica de transmisión, la estándar es OFDM, pero con algunas modificaciones debido al uso del esquema MIMO.
- **802.11ac**: Este estándar, ver Ilustración 9, que es el sucesor del 802.11n ha ido desarrollando sus especificaciones entre 2011 y 2013, obteniendo la aprobación final de sus características en 2014. La ventaja más significativa es la velocidad de transmisión, que ronda los 403 Mbps en el modo más simple y que llega incluso a los 6Gbps en el más avanzado (que utiliza hasta 8 antenas). Al 802.11ac se le conoce también como Wi-Fi 5G, yes que trabaja a una frecuencia de 5 GHz y cada canal puede tener, por defecto, 80 MHz. El esquema que utiliza este estándar es MU-MIMO(Multi-User MIMO) que posibilita la recepción y transmisión de señales de varios terminales en la misma frecuencia. Por otra parte, se utiliza un método de transmisión llamado Beamfroming o TxBF.[10]

Ilustración 9. Estándar 802.11ac.

La Tabla 2 muestra una comparativa de las diferentes tecnologías de comunicación inalámbrica que se podrían usar en el sistema que se propone en este trabajo.

	Bluetooth	BLE	ZigBee	UWB	Wi-Fi
Bandas de frecuencias	2.4 GHz	2.4 GHz	2.4GHz 868/915 MHz	6-8.5 GHz en Europa	2.4 GHz
Tasa de transferencia	1Mbps	1Mbps	250kbps (2.4GHz) 40kbps (915MHz) 20kbps (868MHz)	500 Mbps(MB) Más de 1320Mbps(SD)	11 Mbps
Rango	100 m	1 m	10-100 m	1-10 m	100 m
Número de dispositivos	8	8	255/65535	8	32
Requisitos de alimentación	Media - Días de Batería	Muy Baja – De 2 a 5 Años de Batería	Muy Baja - Años de Batería	Muy Baja – Hasta 2 Años de Batería	Media - Horas de Batería
Precio	Accesible	Bajo	Bajo	Muy bajo	Alto

Tabla 2. Comparación de las tecnologías de comunicación inalámbrica.

Atendiendo a las diferentes características de las distintas tecnologías de comunicación, se puede comenzar eliminando Wi-Fi y Bluetooth como posibilidades debido a que la duración de su batería es insuficiente para nuestro fin y además su coste es mayor que el del resto. Por otra parte, IrDA ni siquiera se llega a incluir en la Tabla 2 por ser su velocidad de transmisión de datos extremadamente lenta, lo cual lo descarta como posibilidad desde el inicio. Por otro lado, UWB, aunque cuenta con hasta 2 años de batería y tiene el menor precio de todos, cuenta con una tasa de transferencia de datos que sobrepasa ampliamente lo que se busca. Esto nos deja con BLE y ZigBee, pero al fijarnos en el rango que cubren, ZigBee va desde los 10 hasta los 100 metros, lo cual nos hace descartarlo definitivamente pues en el presente caso se trabajará con rangos por debajo de los 10 metros. Finalmente se llega a la conclusión de que la mejor opción es BLE.

2.4. Uso de sistemas de comunicación inalámbricos para localización. Ejemplos de aplicación.

Las tecnologías de comunicación inalámbrica descritas hasta ahora pueden ser aplicadas a la localización dando lugar así a algunos de los sistemas de posicionamiento que se describen ahora.

Los sistemas de posicionamiento globales o GPS (ver Ilustración 10), se basan en el cálculo de la distancia entre una serie de satélites que están situados alrededor de la Tierra y esta, mediante las señales que envían al receptor. Esto permite calcular la posición de los mencionados satélites.

Ilustración 10. Posicionamiento global con satélites(fuente OpenStreetMap).

En cuanto a los sistemas de posicionamiento locales o LPS, se puede decir que realizan lo mismo, pero usando mecanismos locales en lugar de satélites y calculando la distancia a un punto más concreto en lugar de a la Tierra (ver Ilustración 11).

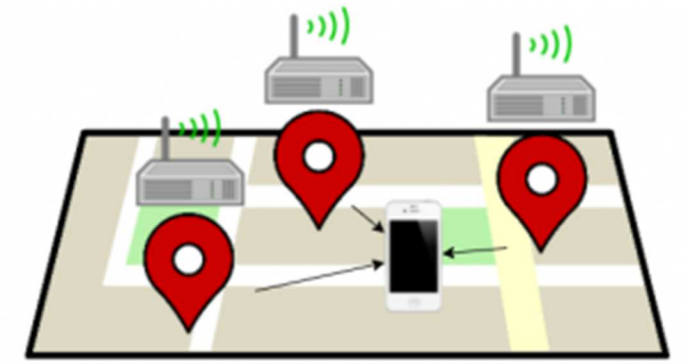

Ilustración 11. Posicionamiento local con Wi-Fi(fuente Wikimedia).

En el presente caso se profundizará en los sistemas de posicionamiento locales, en concreto en los sistemas de posicionamiento en interiores o IPS, ya que se considera son los más adecuados para afrontar el problema que nos ocupa.

Si se distingue entre posición, que son las coordenadas absolutas de un objeto en un sistema de referencia concreto, y localización, que se entiende por ubicación de ese mismo objeto o evento con relación a otro. Se puede definir un sistema de posicionamiento como un protocolo automatizado que permite determinar la posición de un objeto móvil espacial y temporalmente.

Actualmente para resolver el problema del posicionamiento en interiores no existe ningún estándar y hay múltiples tecnologías que son potencialmente válidas, pero las más usadas actualmente son las que usan radiofrecuencias. Se puede distinguir además entre las tecnologías de radio que usan ondas de banda estrecha (como Wi-Fi o Bluetooth) y las que usan ondas de banda ancha (Ultra Wide Band).

La combinación de tecnologías que se pueden usar para permitir la localización de objetos se muestra en la Ilustración 12.

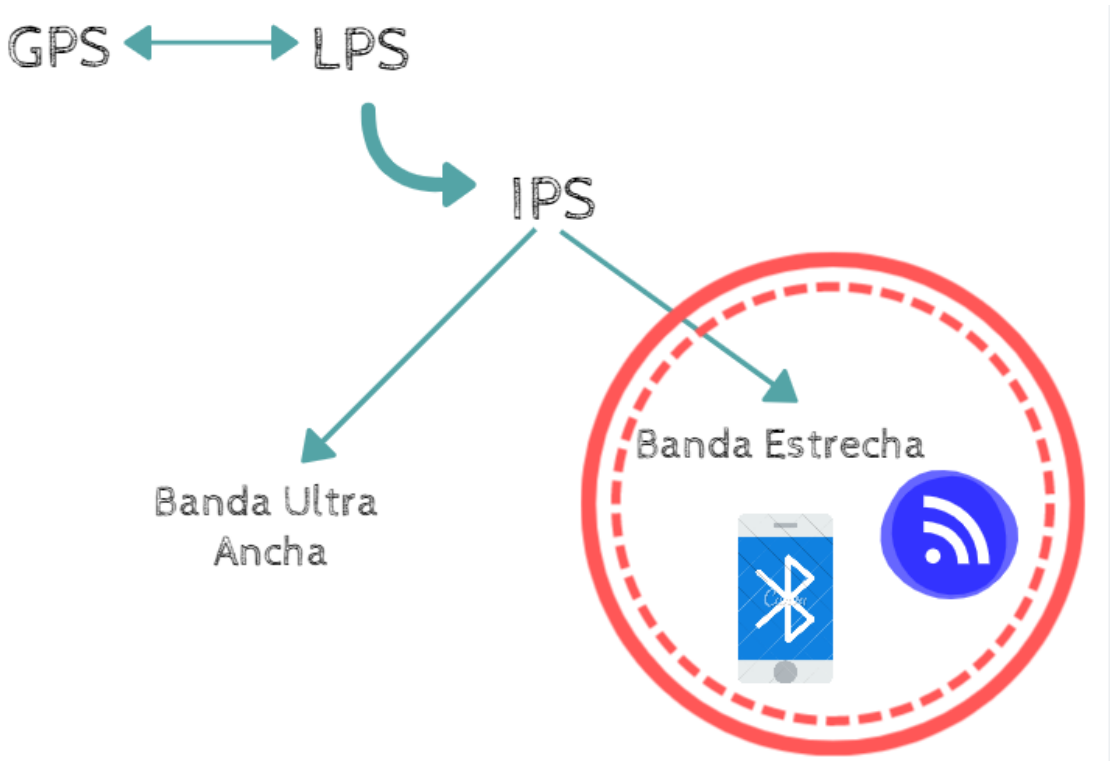

Ilustración 12. Combinación de los distintos sistemas de comunicación.

2.4.1. Sistemas de posicionamiento globales.

Los sistemas de posicionamiento globales son sistemas de radionavegación capaces de proporcionar en cualquier momento y condición atmosférica, información sobre la posición, navegación y cronometría de cualquiera que tenga un receptor GPS. Todo ello gratuita y simultáneamente a cuantos usuarios se requiera. Los dos que se exponen a continuación se diferencial principalmente en el número de satélites de los que reciben información para realizar su función.

2.4.1.1 Sistema de posicionamiento basado en satélites GPS.

El sistema de posicionamiento global GPS fue el primer sistema global de navegación por satélite del mundo y durante mucho tiempo el único existente. Consta de 24 satélites y su principal ventaja es su alcance ilimitado junto con su bajo coste y precisión.

En contraposición, al utilizar frecuencias de 1.575 MHz le es imposible atravesar objetos, por lo que rebota en ellos y esto implica que sea imposible su uso en interiores. Además, al estar los 24 satélites orbitando alrededor de la tierra son necesarios al menos 4 de ellos a la vez para poder fijar la señal, y un número menor hace imposible la obtención de coordenadas.

2.4.1.2 Sistema de posicionamiento global a la red de satélites GNSS-GPS.

El sistema global de posicionamiento por satélite GNSS engloba todos los sistemas de localización basados en satélites existentes. Inicialmente el GNSS está integrado por los sistemas GPS americano y el Glonass ruso, lo que suma un total de 56 satélites. Ya se comercializan sin embargo algunos sistemas que incluyen también el sistema Beidou-Compass chino o incluso el sistema Galileo de la Unión Europea.[11]

Ejemplos de aplicación

Los ejemplos de aplicación de los sistemas de posicionamiento globales son muchas, algunas de las que se pueden aplicar al campo civil son el **estudio de fenómenos meteorológicos**, que es posible porque las señales GPS al atravesar el vapor de agua en la troposfera varían su velocidad de propagación y esto, unido al hecho de que es el vapor de agua el causante de la gran mayoría de fenómenos meteorológicos permite predecir con cierta eficacia como evolucionará la meteorología. También se utilizan estos sistemas de posicionamiento para la **localización y navegación en regiones inhóspitas**, como por ejemplos en desiertos o regiones polares, donde la ausencia de obstáculos o marcas hacen bastante difícil su estudio en ausencia de estos sistemas. Resultan también convenientes estos sistemas en la predicción del movimiento de las placas tectónicas o en el inventario forestal y agrario, así como en **ingeniería civil** donde se aprovechan para realizar el seguimiento en tiempo real de las deformaciones que pueden producirse en grandes estructuras como pueden ser puentes o edificios situados en zonas de riesgo sísmico.

Otro campo en el que resulta interesante la aplicación de esta tecnología es la **sincronización de señales**, por ejemplo, para sincronizar relojes de las estaciones monitoras y luego poder conocer por triangulación si existe algún fallo en el sistema eléctrico y cuál es su situación. También se está estudiando el uso de las señales GPS para **guiar a disminuidos físicos**, como pueden ser invidentes, por las ciudades.

Algunos otros ejemplos de uso pueden ser la **navegación y el control de flotas de vehículos**, **los sistemas de aviación civil** o la **navegación desasistida de vehículos** [12].

2.4.2. Sistemas de posicionamiento local LPS.

Los sistemas de posicionamiento local pretenden suplir la incapacidad del GPS para trabajar en interiores. Se sirven de algunas de las distintas tecnologías de comunicaciones inalámbricas para determinar la posición en una estancia o edificio de algún objeto o individuo. Entre estas tecnologías destacan Wi-Fi, BLE y UWB.

2.4.2.1 Sistemas de posicionamiento basados en Bluetooth Low Energy(BLE).

Este tipo de sistemas son más bien de posicionamiento que de localización. Su característica principal es su bajo consumo y su corto alcance, además de conseguir un posicionamiento de precisión aceptable (entre 1 y 5 metros). El corto alcance de esta tecnología es uno de sus mayores inconvenientes pues, a más de 100 metros de distancia es totalmente ineficaz.

Su principio de funcionamiento consiste en el aviso de la pérdida de conexión entre un Tag o etiqueta(que suele ser un beacon) y un teléfono móvil, que han sido previamente emparejados.

Otra de las ventajas de esta tecnología es que las señales de BLE no se ven muy afectadas por la presencia de objetos sólidos, es decir, hechos con materiales conductores.

Por otro lado, es la intensidad de la señal recibida lo que proporciona la información sobre la posición, pero solamente permite conocer si nos encontramos cerca o muy cerca, no la posición exacta. Para ello se cuenta con técnicas de estimación de la misma mediante un muestreo previo que se describirán más tarde.

2.4.2.2 Sistemas de localización basados en Wi-Fi.

Es una tecnología de localización en interiores que al igual que el BLE se vale de radiofrecuencias de banda estrecha para determinar la posición de un objeto. Para ello se necesita un tag o etiqueta y un dispositivo que reciba las señales que este envía. En este caso el tag sería nuestro teléfono móvil y el receptor el router Wi-Fi.

Uno de los inconvenientes frente a la tecnología BLE es el precio del router, que es siempre mayor que el de un beacon. En contraposición, la señal Wi-Fi no requiere ningún permiso especial por parte del usuario para acceder a su posición mientras que en el caso de BLE sí que es necesario este permiso previamente a la localización.

Al igual que con BLE, este tipo de señal no se ve afectada por los objetos sólidos que puedan estar presentes y la determinación de la posición depende de la fuerza de la señal captada por el router Wi-Fi, siendo la precisión en este caso de entre 5 y 10 metros.

2.4.2.3 Sistemas de localización basados en Ultra Wide Band.

Este tipo de tecnología parte de la radiofrecuencia de banda ancha y al contrario que en el caso anterior, gracias a su naturaleza se podrá medir directamente la distancia de los objetos al receptor y no solamente la intensidad de la señal. Esto hace posible una precisión mucho mayor, incluso de hasta 10 centímetros.

Métodos de aproximación

Para el caso de los sistemas de posicionamiento local mediante BLE y Wi-Fi, tal y como se ha mencionado es necesario un método de aproximación que permita convertir las diferentes intensidades de las señales recibidas en distancias. Entre los métodos que se usan para esto distinguimos entre métodos deterministas, como son la triangulación y la trilateración, y los métodos basados en muestreos.

Tanto la triangulación como la trilateración consisten en calcular la posición de un objeto valiéndose para ello de una fórmula matemática y la dimensión de uno de los lados junto con dos de sus ángulos o bien los factores de cálculo de las distancias entre el objeto a posicionar y los puntos de referencia.

Por su parte, los métodos a los que se suele llamar "fingerprinting" consisten en el cotejo de unas medidas previamente almacenadas de casos anteriores de las que se conoce la distancia al receptor y las medidas que nos ocupan para poder determinar así por comparación la distancia deseada.

Ejemplos de aplicación

Los sistemas de posicionamiento locales pueden usarse eficazmente por ejemplo para indicar el camino que hay que recorrer para llegar desde el punto en el que nos encontramos a cualquier otro de un determinado edificio o lugar, como puede ser una universidad, un hospital, un centro comercial, una estación de tren…

También pueden resultar de utilidad a la hora de estudiar patrones de comportamiento en cualquiera de los lugares anteriormente mencionados además de en supermercados, museos o fábricas. Constituyen una forma muy sencilla y poco visible o aparatosa de ver en qué

lugares pasan más tiempo los visitantes, cuales son los lugares más visitados o cuáles nadie visita casi nunca.

Paralelamente pueden se utilizan también para el seguimiento a tiempo real de usuarios y dispositivos en hospitales, en fábricas, en museos y también en protocolos de evacuación. Siendo un poderoso complemento o incluso sustitutivo de las cámaras de seguridad.

Además, también posibilitan el envío de alertas por proximidad que puede resultar de ayuda en lo que a publicidad se refiere tanto sobre productos comerciales como de obras de arte o monumentos cercanos o en estaciones de trenes y paradas de bus para conocer cuanta gente hay en la próxima parada o cuanta se ha bajado en la anterior.

2.5. Sistemas "anti-olvido" de objetos existentes.

Partiendo en muchos casos de las tecnologías para la localización en interiores que se acaban de describir se desarrollaron otras parecidas como los sistemas de localización en tiempo real o RTLS. Estas y otras se usan para dar solución de diferentes formas al problema del olvido de objetos.

Estos sistemas de localización de objetos permiten identificar y rastrear automáticamente la ubicación de objetos o personas en tiempo real siempre que los mismos se encuentren dentro de un área delimitada, como puede ser un edificio. La localización será posible mediante las señales inalámbricas que reciban los puntos de referencia fijos desde los Tags RTLS. La forma de localización que este tipo de tecnología ofrece, es el posicionamiento de un objeto(Tag) frente a otro (punto de referencia fijo), por lo tanto, no brinda información de localización geográfica mediante GPS, tampoco indica información referente a velocidad, dirección u orientación espacial. Para suplir esta carencia, sin embargo, cuenta con baterías de larga duración, poco peso, la capacidad de trabajar tanto en interiores como exteriores y sin necesidad de un operador de telecomunicaciones móviles y el uso de protocolos abiertos.

Los RTLS se pueden basar en un amplio abanico de tecnologías, como los infrarrojos(IR), los ultrasonidos, Wi-Fi, RFID, UWB o BLE [13].

Normalmente la capa física de la tecnología de los RTLS es alguna forma de comunicación de radiofrecuencia(RF), como por ejemplo BLE (Bluetooth 4.0), UWB (Ultra Wide Band) o bien sistemas propietarios. Por otro lado, los Tags y puntos de referencia fijos pueden ser transmisores, receptores o ambos.

Hay múltiples empresas que comercializan distintos dispositivos que son fruto de la aplicación de esta tecnología, se ven a continuación algunos ejemplos:

- **Gigaset G-Tag**: este dispositivo, que tiene forma cuadrada y un tamaño no superior a 2 tarjetas SD se conecta a nuestro smartphone mediante Bluetooth y gracias a la aplicación asociada nos permite configurar los diferentes objetos que queremos tener controlados y las alarmas que queremos que nos alerten si nos alejamos del dispositivo marcado. Es posible además ver dónde está el objeto gracias al GPS o definir zonas como nuestra casa, donde no recibir alertas si nos separamos del objeto con seguimiento.
- **HipKey:** este dispositivo es similar al anterior, permitiendo también la configuración de alarmas luminosas o sonoras cuando nos alejamos del objeto al que hemos enganchado nuestro localizador. Incluye además algunas funcionalidades extra, como avisar a través del dispositivo localizador si nos estamos alejando de nuestro smartphone más de la distancia previamente fijada o detectar gracias a los sensores de movimiento incorporados que alguien se está llevando nuestras mochila o maleta.
- **Nokia Treasure TagMini:** la apuesta de Nokia es la más ligera si de peso se habla, pues solo pesa 6 gramos, permite básicamente lo mismo que los demás instrumentos, pero la conexión se realiza vía BT 4.0 lo que supone una reducción considerable del consumo.
- **Chipolo:** el funcionamiento de este es muy cercano al de los demás, cuenta con algunas características que lo diferencian del resto, como por ejemplo sus 6 meses de autonomía o la posibilidad de crear una lista de los Chipolo y los productos que hemos prestado asociados a ellos que hará que se envíe una notificación cuando la otra persona haya aceptado ese préstamo.
- **StickNfind:** este último ejemplo es el más ligero y discreto de todos y se centra sobre todo en localizar objetos mediante GPS y no en alertar de que nos estamos olvidando algo, aunque esto también es posible [14].

2.6. Carencias de los sistemas actuales y posibles propuestas de solución.

Una vez se han descrito todas las posibles alternativas para la implementación de la solución propuesta, así como las soluciones que se la han dado al mismo problema que se intenta

solucionar, se puede comenzar a descartar unas y a justificar el porqué de las elecciones realizadas.

Se han ido mencionando a la vez que se describían las distintas tecnologías, tanto los pros como los contras que tenían unas y otras, y que, en mayor o menor medida, cada tecnología suplía esas carencias con ventajas con las que las demás no contaban.

Lo que se busca en este proyecto es la localización de una sillita de bebé en un espacio reducido como es un coche. Por lo tanto, no es factible utilizar un sistema de posicionamiento global dado que éste falla en interiores. Se debe escoger entonces entre otra de las tecnologías disponibles, como es el sistema de posicionamiento local basado en BLE, Wi-Fi o bien UWB.

Con las tecnologías descritas hasta ahora, la localización quedaría resuelta de uno u otro modo. Lo que se busca también es la posibilidad de obtener información acerca de temperatura en el punto en el que se coloque el tag, lo cual puede ser satisfecho por cualquiera de las tres tecnologías de localización local descritas siempre que este cuente con sensor de temperatura.

Otra de las condiciones necesarias es la incapacidad de localización de un dispositivo a menos que esté configurado previamente junto con el receptor adecuado. Esto se hace necesario por tratarse de un dispositivo que irá unido a una persona, a un bebé en este caso, y ocurre que en múltiples ocasiones no se desea que este pueda ser localizado por cualquiera sino solamente por ciertas personas escogidas por el usuario.

Todo ello añadido a la comparación realizada anteriormente en la Tabla 2, hace evidente que la tecnología más conveniente para llevar a cabo la localización, así como la comunicación inalámbrica con el smartphone es BLE por su simpleza de uso, su bajo coste, su bajo consumo de energía y por tanto mayor duración de baterías , la adecuación de su rango de funcionamiento a nuestros propósitos y la velocidad conveniente de transmisión de datos entre otros.

Paralelamente se ha de decidir en cuál de los descritos sistemas operativos móviles se va a programar inicialmente la aplicación. La elección de Android frente a iOS se justifica, principalmente, porque el segmento de clientes de Android es mayor que el de iOS, al menos en Europa en general y en España, en particular.

2.7. Uso de balizas i-beacon para resolver problemas de localización. El caso de Estimote.

Los iBeacons (ver Ilustración 13), son la implementación por parte de la empresa Apple de un sistema de posicionamiento en interiores de bajo coste. Se basa para ello en BLE, Bluetooth de baja energía. Estos pequeños dispositivos brindan una nueva forma de proporcionar información basada en la posición del usuario o de cualquier objeto al que se adhieran. Funcionan en combinación con cualquier Smartphone que use IOS7 o superior o bien, Android 4.3 o superior. Se trata de pequeños transmisores que están continuamente enviando una señal Bluetooth la cual será recibida por nuestro teléfono móvil que rastreará continuamente en busca de una señal de ese tipo una vez instalada la aplicación pertinente. Una vez encontrada la señal adecuada se dispondrá a realizar una u otra acción, según se haya programado, normalmente algún tipo de notificación [15].

Ilustración 13. Explosionado de un iBeacon.

Generalmente los iBeacons se utilizan para localización en interiores y haciendo uso de esa información son útiles para una gran multitud de funciones. Por ejemplo, anuncios de ofertas al llegar a un punto en concreto de una tienda o centro comercial, servir de punto de referencia o localización, navegación en interiores, ya que ahí no podemos contar con la conexión GPS, integración con redes sociales y en definitiva mejorar la experiencia del cliente.

En la actualidad ya han empezado a implantarse en algunos comercios de Estados Unidos, por ejemplo. Algunas tiendas y cadenas integraron el uso de iBeacons para lanzar ofertas específicas a sus clientes, así, al ir pasando por distintos sitios de la tienda se le enviaban unas

u otras ofertas a cada uno. Una de las empresas que también apostó por el uso de los iBeacon fue Carrefour que los incorporó a sus cestas y carritos de compra para construir un mapa de calor y averiguar así qué zonas eran las más visitadas por sus clientes.

En los aeropuertos, estos dispositivos pueden ser también de gran utilidad, ya que se posibilita el envío de información a cada usuario acerca de su vuelo.

En cines se usa también para mostrar los próximos estrenos y fomentar así el interés del cliente para que vuelva. O en los hoteles, donde se utilizan para dar información sobre la ocupación de los restaurantes en tiempo real, de las actividades que se pueden realizar, y de la ocupación entre otras cosas.

En lo que al turismo respecta, pueden emplearse para enviar información de las obras y monumentos a los que nos aproximemos, tanto en un museo como al pasear por una ciudad.

Por otro lado, donde también podrían brindar múltiples posibilidades de uso es en el hogar, aunque por el momento es el ámbito menos desarrollado, servirían para programar mensajes de bienvenida, conexión automática del equipo de música o información sobre posibles incidencias [16].

Capítulo 3. Diseño del sistema.

En este capítulo se describe, en detalle, la solución que se propone para resolver el problema, tanto desde un punto de vista conceptual como de diseño, los materiales y métodos usados y cómo se ha llevado a cabo la implementación del sistema.

3.1. Introducción.

Una vez que se ha decidido la tecnología que se va a usar para el desarrollo de la aplicación, se puede concretar cómo será la misma.

En este proyecto se pretende crear un sistema que ayude a los padres a no olvidar a sus hijos o a cualquier persona u animal de compañía que lleven consigo en el coche. Para ello se ha pensado en una aplicación móvil en combinación con una baliza o tag del tipo iBeacon de Estimote. En este caso la aplicación desarrollada será compatible con cualquier dispositivo que tenga por sistema operativo a Android.

A grandes rasgos lo que hará es alertar mediante una notificación sonora en el teléfono en caso de que se salga de un rango previamente establecido o la temperatura esté fuera de los límites permitidos. Por otra parte, permitirá también conocer información acerca de la temperatura del lugar en el que se encuentre situado el iBeacon y enviar notificaciones o incluso llamar de manera automática a los contactos de una lista previamente confeccionada y almacenada en una base de datos.

La arquitectura del presente proyecto va desde que se inicia la aplicación y se configura el beacon hasta la recepción de la notificación pertinente pasando por varias fases. Los diferentes elementos que participarán en este proceso sin hacer distinción aún entre hardware y software (ver Ilustración 14), son:

- **iBeacon**: dispositivo que proporcionará la información acerca de si nos estamos olvidando al bebé en el coche.
- **Teléfono móvil**: dispositivo en el que se instalará la aplicación desarrollada y que recibirá información del iBeacon mediante BLE.
- **Aplicación Android:** hace posible la detección de si se ha salido del rango en el que se encontraba el bebé con el que se estaba viajando.

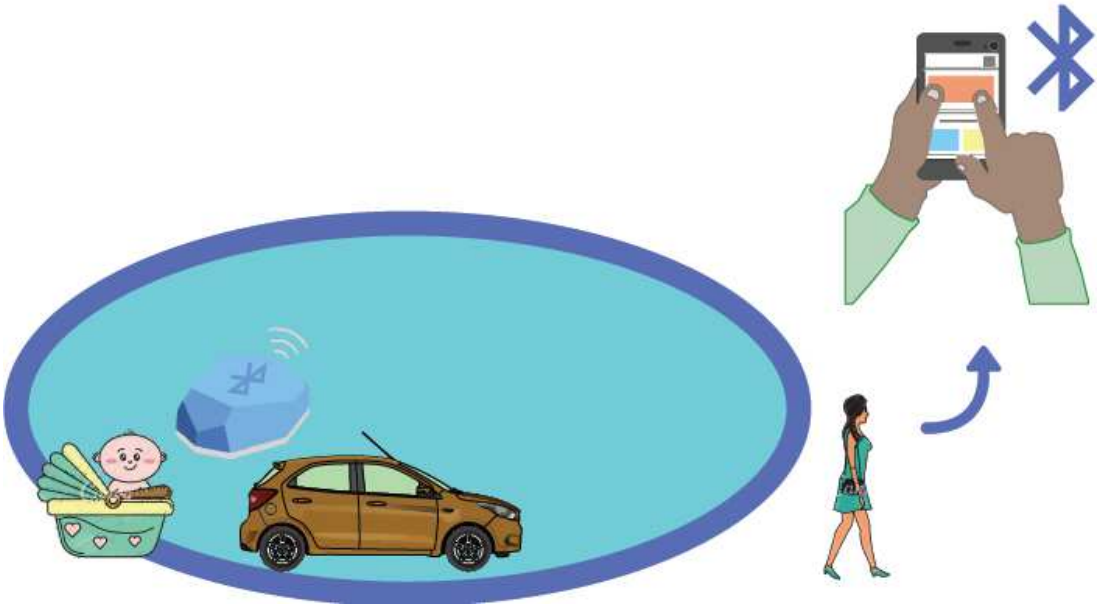

Ilustración 14. Esquema de funcionamiento.

Del esquema de funcionamiento mostrado casi todo es parte de la arquitectura hardware. Lo que se corresponde con el software es la parte implicada en la comunicación, del beacon con la aplicación y de ésta con un teléfono externo si fuese necesario (ver Ilustración 15).

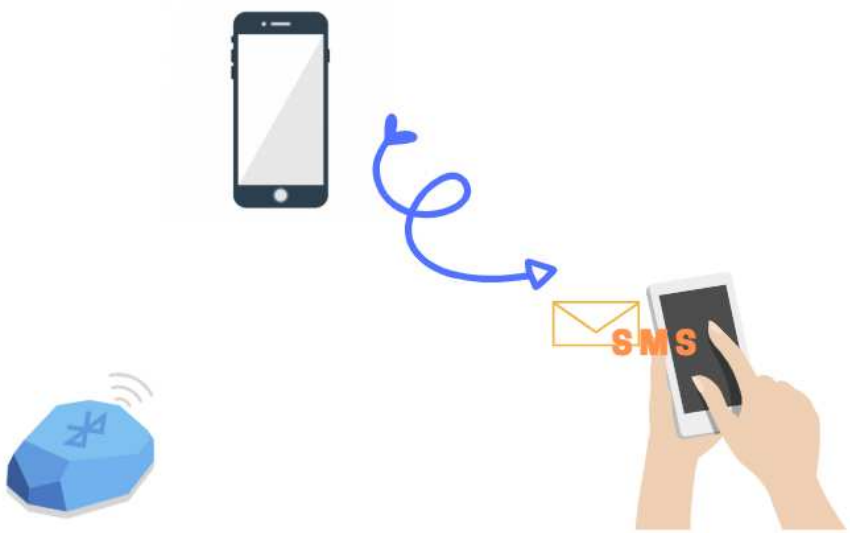

Ilustración 15. Esquema de comunicación.

3.2. Arquitectura hardware.

Se entiende por arquitectura hardware la parte física del proyecto, es decir, todos los componentes mecánicos, eléctricos y electrónicos de este que hacen posible su correcto

funcionamiento. En el presente caso, como arquitectura hardware se tiene el smartphone, cuyos componentes internos han quedado ya descritos, y el iBeacon, aunque también podrían interpretarse como tal la sillita del bebé en la que se coloque el iBeacon e incluso el coche en el que se encuentre ésta.

3.2.1. Beacons.

Un beacon es un pequeño dispositivo capaz de enviar avisos y notificaciones de cualquier naturaleza, sirviéndose de Bluetooth de Baja Energía (BLE) a dispositivos cercanos que tengan el Bluetooth activado y sin la necesidad de permiso por parte de estos.

Cada beacon funciona como un mini faro, está continuamente enviando una señal BLE que no contiene más que su número de identificación. Este número de identificación es captado por un dispositivo que disponga de Bluetooth y enviado entonces a la nube a través de Internet donde se llevan a cabo distintos procesos que permite la determinación de la posición del beacon, entre otros aspectos.

3.2.1.1 Hardware del beacon.

Estos pequeños dispositivos son bastante simples en general y su parte física no es una excepción. Los beacons tienen forma de una especie de piedras pequeñas, contienen en su interior una pila de botón (por lo general), y un microcontrolador con un chip de radio BLE, recubiertos ambos por una protección de silicona resistente al agua.

El microcontrolador con chip de radio BLE es fabricado habitualmente por las empresas Texas Instruments o Nordic Semiconductor.

Las baterías que en general usan los beacons son como se ha adelantado del tipo botón pero en nuestro caso, los beacons de Estimote especiales para proximidad utilizan pilas alcalinas del tipo AA que suministran 2200mAh y que pueden llegar a durar hasta 3 años si se mantienen los ajustes de fábrica o incluso más si se modifican estos.

Otras empresas fabricantes de beacons u otras versiones optan por alimentación vía USB, lo cual elimina la necesidad de baterías, pero también la ausencia de cables y hace imprescindible la proximidad de una toma de corriente.

3.2.1.2 Firmware del beacon.

Todo beacon cuenta con un firmware único que lo distingue del resto y del que depende su correcto funcionamiento. Es aquí donde se puede hacer que la batería dure más o menos según las configuraciones que se decidan modificar.

Una de estas configuraciones es la **potencia de transmisión** o Tx Power, que consiste como su nombre indica en la potencia a la que el beacon, que está transmitiendo continuamente, envía la señal. La potencia es directamente proporcional al alcance del beacon y también, por extensión, al consumo de batería de este.

La frecuencia con la que un beacon envía la señal se denomina **Advertising Interval** y es otro de los responsables del consumo del beacon. Apple, por ejemplo, recomienda un Advertising Interval de 100ms, es decir, que se envíe señal 10 veces por segundo, pero lo cierto es que no existe un valor óptimo para este parámetro, sino que dependerá del tipo de aplicación y del uso que le demos al beacon. En lo que al consumo respecta se ha de aclarar que cuanto mayor sea el intervalo menor será el consumo, pero esto tiene como inconveniente el aumento de la capacidad de respuesta del dispositivo receptor.

La Tabla 3 muestra un resumen de las diferentes potencias, rangos y consumos según la distancia.

Intervalo	Potencia de Transmisión	Rango esperado	Duración batería
100 ms	3 (- 12 dBm)	35 m (115')	Más de 7 meses
300 ms(default)	3 (- 12 dBm)	35 m (115')	Más de 2 años
1000 ms o 1 s	3(- 12 dBm)	35 m (115')	Más de 4 años

Tabla 3. Comparación de características del firmware del beacon.

3.2.1.3 Protocolos de beacons.

Hasta ahora si algo está claro es que todos los beacons trabajan con BLE, este protocolo de transmisión de datos inalámbrico puede funcionar en dos modos, el modo Advertising y el modo Conectado. Este último hace uso del atributo genérico o GATT para transferir datos en una conexión. El modo Advertising por su parte, utiliza el perfil de acceso genérico para transmitir datos de uno a muchos, a cualquiera que esté escuchando de hecho, por esto mismo no garantiza la coherencia de los datos.

Estos pequeños faros BLE transmiten paquetes de Advertising periódicamente con un formato particular que dependerá del protocolo del que se haga uso, que depende a su vez de la empresa que lo haya desarrollado. Existen varios protocolos de comunicación por parte de tres empresas distintas.

La primera de ellas es **Apple**, que fue la primera en apostar por la implementación de BLE en este tipo de dispositivos. Este protocolo en concreto ofrece dos métodos distintos, Ranging y Monitoring que sirven para proporcionar estimaciones de proximidad cuando la aplicación está activada y para proporcionar información sobre si se está entrando o saliendo de un rango previamente establecido esté o no la aplicación activada, respectivamente. El paquete de Advertising de un iBeacon incluye un UIID, un Major y un Minor, que son tres conjuntos de caracteres cuya combinación compone la identificación de un beacon en concreto.

Google también quiso realizar su aportación a esta tecnología y lo hizo con un proyecto de código abierto al que se llamó Eddystone y cuyo fin era fomentar el Internet de las Cosas. Se diseño para que pudiese soportar múltiples tipos de paquetes de datos y actualmente se basa en un único método, Eddystone Discovery, que es el equivalente a iBeacon Ranging. Google pretende en un futuro proporcionar las API de Nearby y Proximity, que pondrán su protocolo al mismo nivel que el de Apple al aportar un método homólogo al iBeacon Monitoring.

Por último, la tercera empresa que propone un protocolo es **Radius Networks**. Su alternativa es gratuita y open-spec , cubre las mismas funcionalidades que Apple ofrece e incluso puede transmitir más cantidad de información por mensaje, pero no cuenta todavía con un amplio apoyo [17]. La Ilustración 16 resume los diferentes protocolos de comunicación según las diferentes compañías tecnológicas mencionadas.

Ilustración 16. Protocolos de comunicación de los Beacons.

3.2.1.4 Configuración de los beacons usados en la aplicación.

Como ya se ha comentado, el tipo de iBeacons usados en la presente aplicación son los especiales para proximidad. Antes de que puedan resultarnos útiles para este fin en concreto se deben ajustar unas configuraciones mínimas en la página que ofrece Estimote para tal fin: Estimote Cloud, ver Ilustración 17.

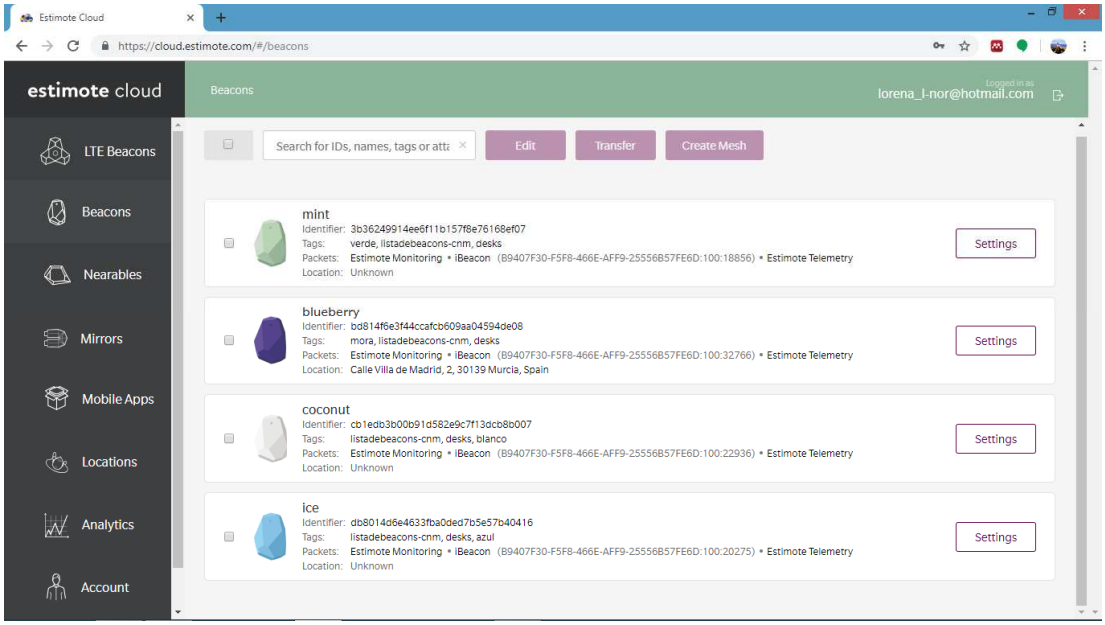

Ilustración 17. Captura de pantalla de Estimote Cloud.

En este sitio web aparecen no sólo los iBeacons, sino diferentes opciones en la barra de herramientas de la izquierda. Entre estas opciones se encuentra la de Mobile Apps, que son una serie de plantillas de aplicaciones básicas que proporciona Apple para simplificar la programación de sus beacons, ver Ilustración 18.

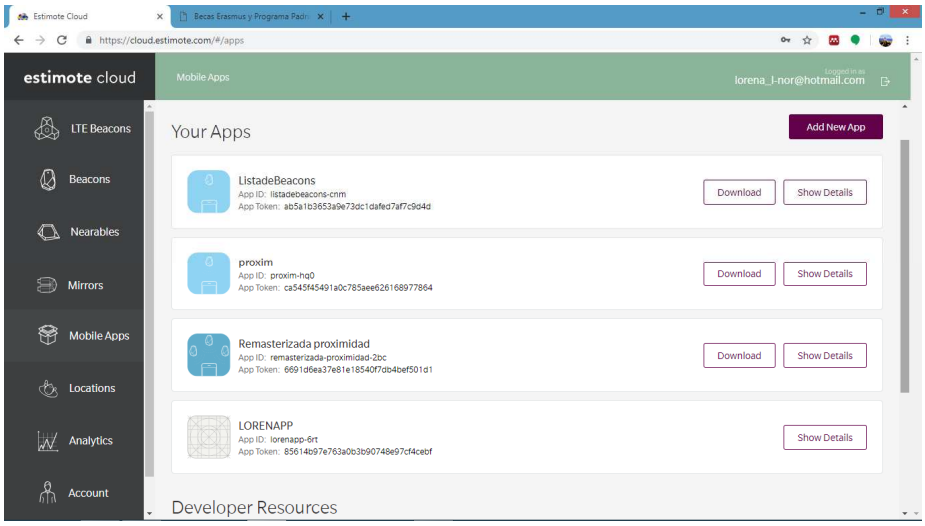

Ilustración 18. Plantillas para aplicaciones.

En este trabajo se tomó una plantilla para el caso de la proximidad y el rango y otra para el despliegue de la lista de beacons a la hora de asociarlos con un determinado perfil de usuario. Para que la aplicación que se está programando con Android Studio pueda comunicarse con la plantilla se necesita contar con algo que la identifique, su App ID y App Token, ambos deben aparecer en el código de Android Studio y serán los que permitirán la conexión de la App con Estimote Cloud y la obtención así de la información pertinente en cada caso.

Por otra parte, si en lugar de atender a las herramientas de la izquierda se selecciona uno de los dispositivos que aparecen en la pantalla, aparecen todos sus detalles y se pueden modificar sus ajustes a través del diálogo que se muestra en la Ilustración 19.

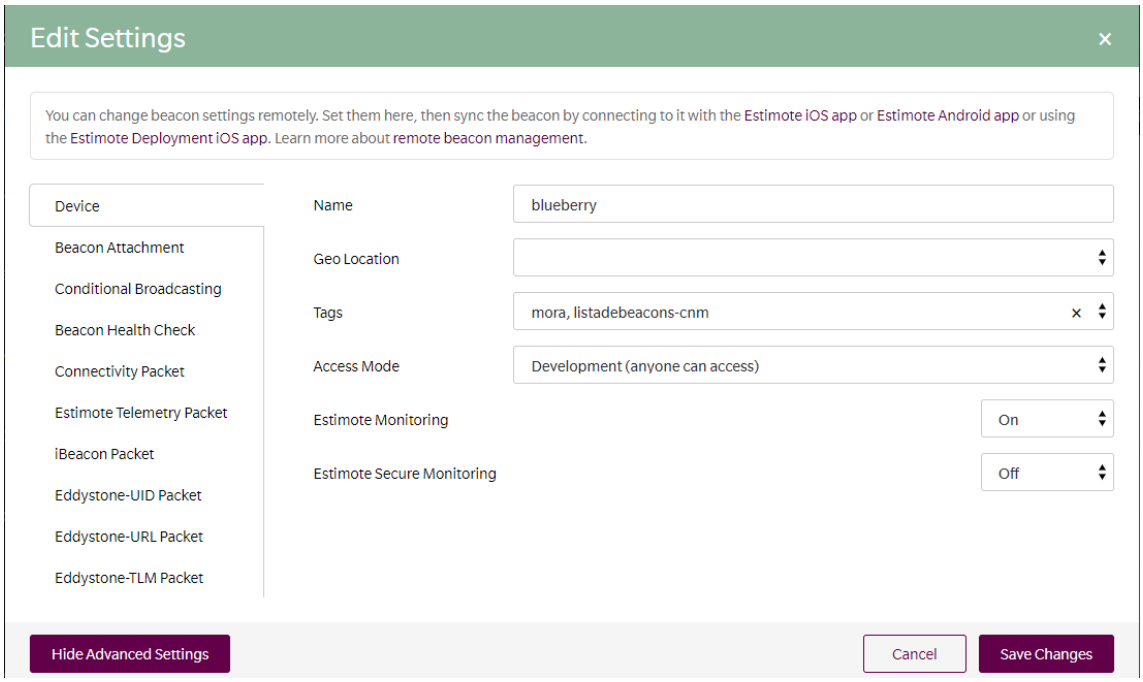

Ilustración 19. Pantalla de ajustes de un iBeacon.

Cada beacon que se desea usar en esta aplicación deberá tener 2 tags o etiquetas, uno que lo identifique, en este caso "mora" que se corresponde con los ajustes de proximidad y rango. Y otro que permita encontrarlo cuando se despliegue la lista de beacons que hace posible asociar cada iBeacon a un perfil distinto, "listadebeacons-cnm".

Estos que se acaban de describir son los únicos ajustes a realizar en la web de Estimote, el resto puede operarse desde la aplicación.

3.2.2. Descripción de los sistemas basados en estimotes. Aspectos de hardware y software.

Los sistemas de localización que se basan en iBeacons funcionan mediante la detección por parte del smartphone o cualquier otro receptor de las ondas emitidas constantemente por los iBeacons. Estas ondas son de BLE y por ello, aunque se esté emitiendo de manera continua la batería de los dispositivos no se consume rápidamente, sino que puede llegar a durar casi 2 años, dependiendo de la configuración particular de cada sensor.

Por otro lado, es necesario que el dispositivo que reciba las ondas esté programado para conectarse con Estimote Cloud, es decir, que disponga de una app. Esta le permitirá cotejar los datos que recibe con los que esperaba recibir, ya que cada iBeacon se identifica como ya se ha explicado más arriba mediante 3 parámetros: el UIID, el Major y el Minor.

Así pues, lo que se tiene en código de la aplicación son directamente estos parámetros o bien la identificación de la aplicación creada con las plantillas que se ofrecen en Estimote Cloud. Esta aplicación que se crea a partir de una plantilla puede asociarse a uno o varios iBeacons y tras esto la plataforma ofrece la posibilidad de descargarla y proporciona un par de parámetros de identificación, como son el nombre de la app y un conjunto de caracteres o "Token".

Volviendo a la localización, con los iBeacons es posible gracias a que, además de los datos para su identificación, envían también a cualquier dispositivo cercano y de manera periódica un indicador de fuerza de la señal recibida por el receptor (smartphone, por ejemplo), y la intensidad de señal medida preconfigurada a la distancia de un metro. Con estos datos, y el hecho de que la fuerza de la señal variará según estemos más cerca o más lejos del iBeacon se puede conocer la distancia entre el iBeacon y el smartphone realizando una comparación entre la intensidad de la señal recibida preconfigurada cuando el iBeacon está a 1 metro de distancia y la intensidad que se ha recibido del mismo. Y se deducirá así si el smartphone se encuentra dentro o fuera del rango de alcance del iBeacon.

En el caso de una red de varios iBeacons se puede conocer la posición exacta de cualquiera que se encuentre dentro de ella mediante un sistema parecido al usado por los sistemas de posicionamiento global y que ya se ha mencionado en el capítulo anterior, la trilateración.

La trilateración, que no debe confundirse con la triangulación, consiste en la determinación de ubicaciones aproximadas basadas en la geometría de los triángulos, y más concretamente

en las distancias de estos. Será necesario contar con al menos 3 iBeacons y sus correspondientes mediciones, ver Ilustración 20.

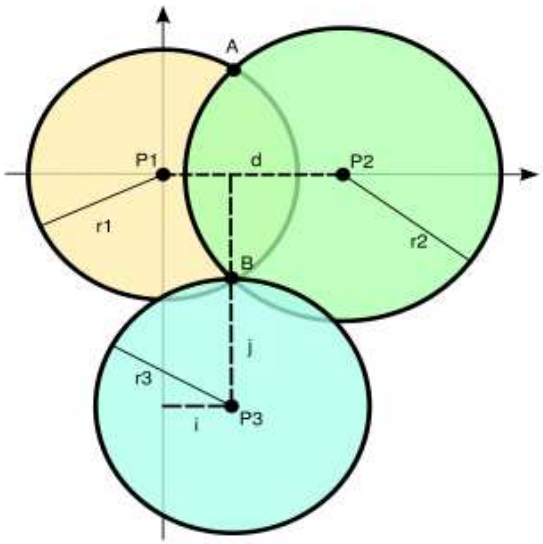

Ilustración 20. Esquema explicativo del método de trilateración.

En primer lugar, es necesario escoger las dos balizas más cercanas a nuestro dispositivo P1 y P2 y dibujar sus circunferencias con un radio equivalente a la distancia al smartphone r1 y r2. Después, se seleccionan los puntos pertenecientes a ambas circunferencias más próximos entre sí, es decir A y B, en los cuales podría estar ya la posición buscada. Seguidamente se escoge el tercer iBeacon más cercano, P3, y se dibuja su circunferencia correspondiente de radio igual a la distancia entre él y el smartphone. Finalmente, el punto más cercano a A y B de entre los puntos en los que esta tercera circunferencia corte a las otras dos se tendrá la posición inequívoca que se buscaba, que en este caso es B [19].

3.2.3. Arquitectura de los sistemas Android y su utilización con estimotes.

Como se acaba de decir, un iBeacon por sí solo no es de mucha utilidad, lo conveniente es que esté asociado a alguna aplicación móvil, en un smartphone dotado de capacidad de comunicación BLE, que posibilite su detección y que haga uso de los datos que es capaz de recolectar y enviar. Esta aplicación ha de ser instalada en el dispositivo, el cual puede implementar un sistema operativo móvil u otro. Como ya se expuso anteriormente, los más célebres son Windows Phone, iOS y Android. También se ha discutido ya, por qué se ha elegido Android como OS móvil. A continuación, se describirá la arquitectura del sistema operativo Android y su relación con los iBeacons.

La arquitectura de Android, como puede apreciarse en la Ilustración 21, consta de varios niveles o capas, las cuales utilizan elementos de la capa inferior para realizar sus funciones y es por esto que a este tipo de distribución también se la llama pila.

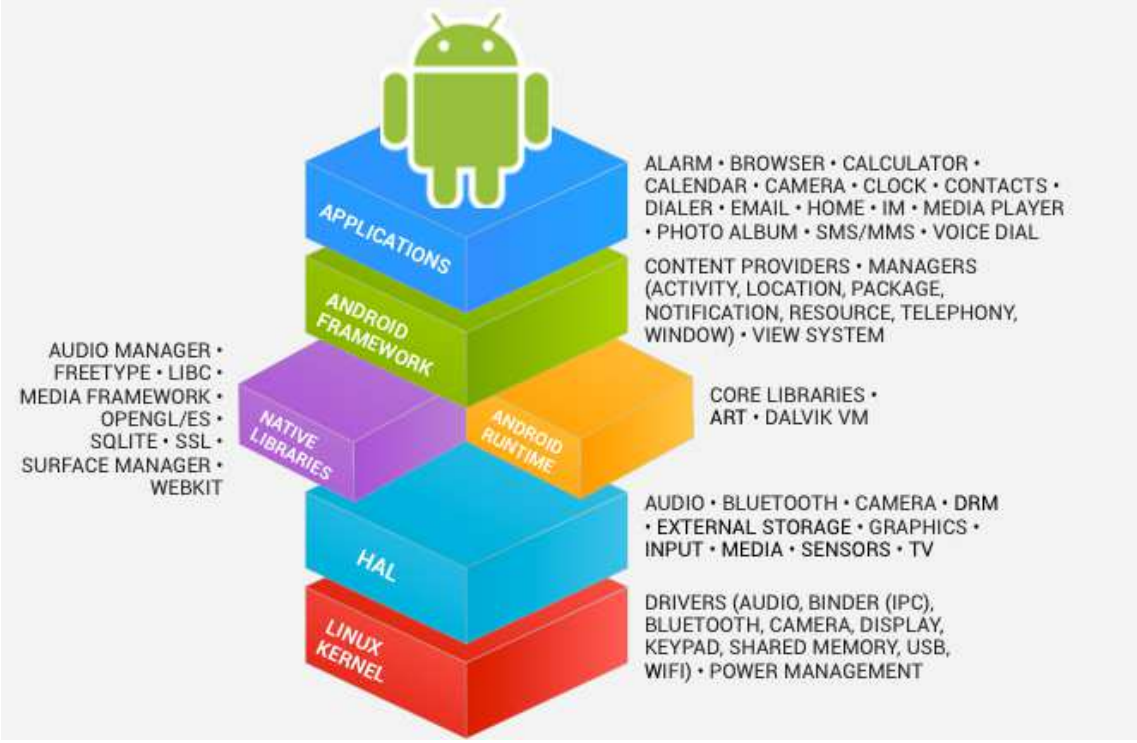

Ilustración 21. Estructura de Android.

~ La capa inferior se llama **Kernel**, que viene del alemán y significa núcleo. El núcleo del SO Android es Linux versión 2.6 y es similar al que incluye, por ejemplo, Ubuntu, pero con la particularidad de que está adaptado a las características de Android para ser ejecutado normalmente en un smartphone. Al igual que Ubuntu es de software libre y código abierto.

Esta capa es la única que se relaciona con el hardware y permite acceder a los componentes del mismo independientemente del modelo o características del dispositivo. Permite también la gestión de los diferentes recursos del teléfono y del SO como la batería, la memoria o los elementos de comunicación entre otros.

~ Le sigue la **capa de librerías o capa nativa**, que contiene partes como el HAL, las librerías nativas o los Daemons.

- **HAL o Hardware Abstraction Layer,** es decir, la capa de abstracción del hardware. Es la que hace posible la independencia del hardware tan característica de Android. El HAL actúa como una arquitectura genérica que representa a todos

los posibles tipos de hardware y que posibilita que el sistema trabaje siempre con las mismas instrucciones en todos ellos.

- **Librerías nativas**: escritas en C o C++ recogen una serie de funciones habitualmente usadas por las aplicaciones y las empaquetan en trozos de código genéricos que pueden ser usados por cualquier aplicación. Un ejemplo podría ser las librerías que permiten el soporte de Wi-Fi o SQLite que sirve para la gestión de bases de datos.
- **Daemons**: son también pequeños fragmentos de código destinados a ayudar a algún servicio del sistema.
- **Runtime de Android**: las aplicaciones están escritas en Java y son traducidas luego a bytecodes, pero normalmente, en un código Java de cualquier otra cosa distinta de una app el siguiente paso sería obtener un fichero JVM. Sin embargo, para las apps en particular se busca que ocupen poco espacio y por ello se creó una herramienta específica para traducir los bytecodes a un formato más ligero que JVM, ese es DVM y la herramienta asociada es Dalvik VM que crea un ejecutable que ya no tendrá que volver a traducirse, aunque se instale en un dispositivo totalmente distinto. A partir de Android 5.0 además, se reemplazó Dalvik por ART que consigue todo lo que su predecesora, pero en un 33% menos de tiempo de ejecución.

~ La siguiente capa es el **Entorno de aplicación**, que contiene todas las clases y servicios que utilizan directamente las aplicaciones para llevar a cabo las tareas que realizan. Se sirven para ello como es evidente de las librerías que antes han quedado descritas, así como del Runtime de Android. Algunas de los servicios que incluye son:

- **Views**: son un conjunto de vistas, muy útiles para la parte visual de la aplicación, como por ejemplo botones, cuadros de texto o listas.
- **Resource Manager**: da acceso a cualquiera de los recursos que queramos incluir en nuestra app, como imágenes, cadenas de texto traducidas o archivos de audio entre otros.
- **Activity Manager**: es el encargado de gestionar la pila de actividades y el ciclo de vida de estas, así como de proporcionar un sistema de navegación ente ellas
- **Notification Manager**: proporciona servicios que hacen posible notificar al usuario cuando se requiere mediante alertas personalizadas en la barra de estado.
- **Content Provider**: gracias a este es posible el paso de información entre aplicaciones mediante la creación de una capa de abstracción que permite

compartir paquetes de datos encapsulados. Esta capa de abstracción sirve para que se pueda acceder a esos datos del paquete sin perder el control sobre como se accede a la información. Un buen ejemplo es el de los contactos que se almacenan en un móvil, pues múltiples aplicaciones solicitarán el acceso a esos datos y podrán acceder a ellos haciendo uso del sistema descrito.

- La capa superior de esta pila es **la capa de Aplicaciones**, en la que se encuentran todas las aplicaciones del dispositivo, tanto las propias del sistema como las que se han instalado después por el usuario. Todas ellas comparten un mismo entorno de aplicación, librerías y demás capas inferiores y esto implica que se pueden crear nuevas aplicaciones que usen los recursos de las aplicaciones nativas y que se pueden crear aplicaciones que reemplacen a las nativas. Todo ello supone un control total del software que se ejecuta en el teléfono por parte del usuario y eso es lo que hace tan especial a Android [20][21][22].

3.3. Arquitectura software.

La parte software es, como ya se adelantó, la que se corresponde con la aplicación Android desarrollada. Antes de poder pasar a describir en profundidad el funcionamiento de la misma se deberán aclarar algunos puntos. Entre ellos el entorno de programación en el que se trabaja, que es Android Studio.

3.3.1. Android Studio

Android Studio es el entorno de desarrollo predeterminado para aplicaciones Android desde que se presentó su primera versión estable en diciembre de 2014, hasta entonces lo era Eclipse. Fue desarrollado tomando como base IntelliJ IDEA de JetBrains y es considerado un potente editor de códigos que cuenta con excelentes herramientas para desarrolladores. Destaca porque:

- Cuenta con un sistema de compilación basado en Gradle flexible.
- Es un entorno unificado en el que desarrollar aplicaciones para cualquier dispositivo Android.
- Posee una función que permite actualizar el código mientras se prueba la aplicación para no tener que cargar un nuevo APK.
- Dispone de múltiples plantillas de código para aplicaciones comunes, así como de integración con GitHub para importar fácilmente ejemplos de código.

- Incluye herramientas Lint para poder detectar problemas de rendimiento, usabilidad y compatibilidad de versiones entre otras cosas.
- Es compatible con C++ y NDK.
- Permite trabajar directamente y de manera sencilla con Google Cloud Messaging y App Engine gracias al soporte de Gloogle Cloud Platform que incorpora.

La estructura de un proyecto en Android Studio se subdivide en 3 módulos, los de la aplicación, los de las bibliotecas y los correspondientes a Google App Engine, ver Ilustración 22.

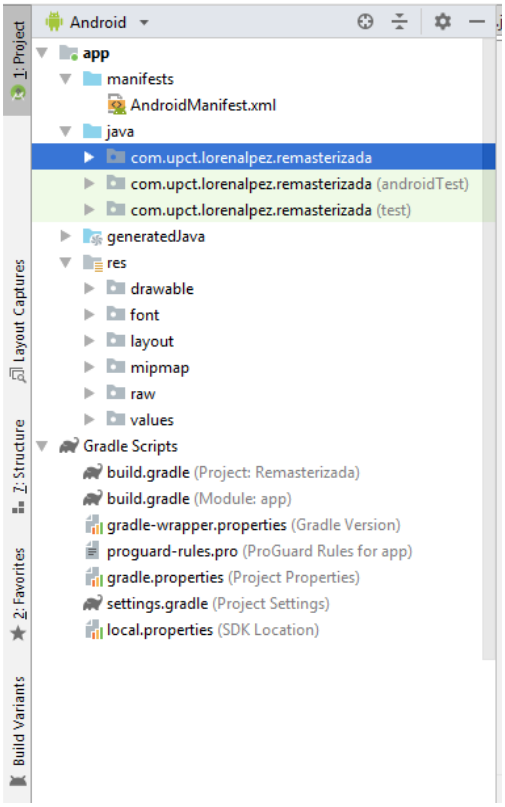

Ilustración 22. Estructura de un proyecto en Android Studio.

Cada módulo de aplicación consta de 3 principales carpetas:

- **manifest**: que contiene el archivo AndroidManifest.xml
- **java**: contiene los archivos del código fuente de las clases que se incluyan en cada proyecto
- **res**: que contiene todos los recursos que se usan en la aplicación, como pueden ser archivos de audio o imágenes, archivos xml como strings.xml, en el que se pueden definir cadenas de caracteres que muestren por pantalla o colors.xml donde se definen los colores que se usarán en la aplicación.

Dentro de la carpeta java se pueden encontrar varios tipos de archivos. En el caso que nos ocupa nos interesan dos, los Servicios y las Actividades.

Un **Servicio** es un componente de una aplicación que puede ejecutarse en segundo plano durante un largo periodo de tiempo y sin necesidad de interfaz de usuario.

Una **Actividad** es un componente de una aplicación que consta de una interfaz de usuario mediante la cual pueden introducirse distintos tipos de información que permitirán realizar distintas acciones en la aplicación

Otra importante parte de la estructura son los Gradle Scripts, archivos que definen la configuración de compilación que se aplica a cada módulo o a todos, según el archivo al que nos refiramos [18].

3.3.2. Actividades y servicios

Una vez descrito todo lo anterior se puede pasar a describir la aplicación en sí. Esta aplicación cuenta con varias Actividades, que son:

- Main Activity
- Modo Control
- Modo Registro
- Modo Alertas
- DataBaseHelper
- Lista Beacons
- Login
- RegistrationScreen

Para las notificaciones se usan IntentServices, que son un subtipo de los Servicios y hacen posible que se lancen en cualquier momento.

En cuanto a las tareas realizadas por los iBeacons son principalmente 2, la que permite identificar si se entra o se sale del rango establecido y la que escanea la zona para lanzar una lista de la que poder seleccionar el beacon que se desea enlazar a un determinado perfil.

La primera de ellas es la asociada a la plantilla de Estimote llamada remasterizada-proximidad. En este caso se hace uso del modo Monitoring de los iBeacons que, como ya se dijo, sirve para determinar si se está entrando o saliendo de una región preestablecida. Este método suele asociarse a un servicio para que pueda así ejecutarse en segundo plano, pero en el caso

que nos ocupa se colocó en la MainActivity por simplicidad con la conexión del resto de actividades y para hacer más fácil la validación del primer prototipo de aplicación. En sucesivas versiones y revisiones se planteará la posibilidad de usar un servicio en segundo plano como una mejora del sistema.

En cuanto a la lista de iBeacons, la plantilla de Estimote usada se llamó "listabeacons" y está basada en el método de Ranging de los beacons que, aunque permite extraer más características de los mismos, además de si se entra o se sale de un rango, consume también mucha más batería, es por eso que se usó en esta parte de la aplicación, a la que solamente será necesario acceder por un corto período de tiempo.

3.4. Interface de usuario.

En este apartado se describirán con más detalle las actividades de la aplicación de las que se ha hablado anteriormente, haciendo hincapié en todo lo relacionado con la interfaz de usuario. Se expondrán además los distintos archivos de layout asociados a ellas y se explicará la transición entre estas pantallas.

3.4.1. Storyboard navegacional.

Aquí se concretará la función básica de cada pantalla de la aplicación y se aporta también un grafo guiado con flechas que marcan la ruta que se sigue entre unas y otras a modo de storyboard navegacional, como paso previo al diseño final de las pantallas. En dicho grafo, cada letra numerada P_i hace referencia a la pantalla número i, ver Ilustración 23.

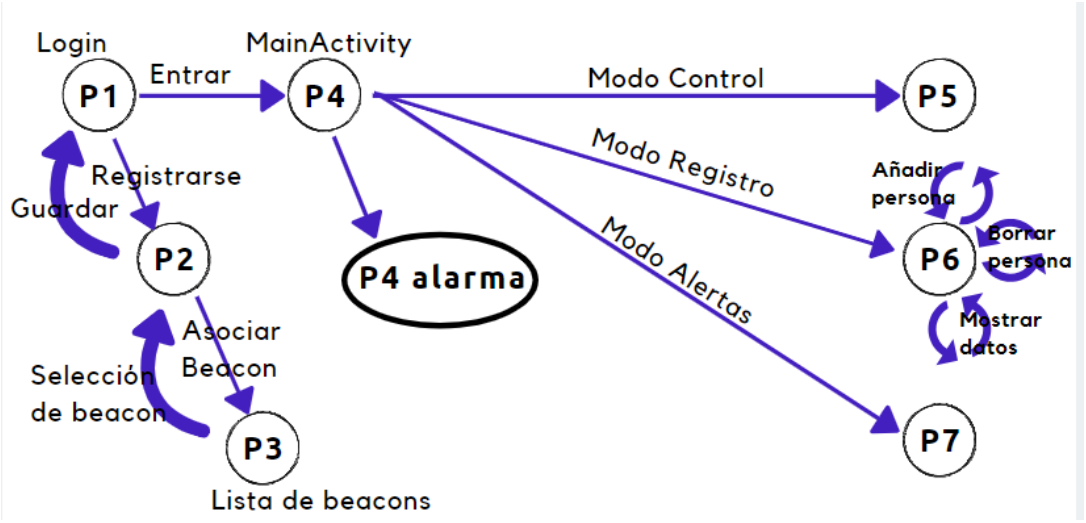

Ilustración 23. Grafo de navegación del storyboard.

La pantalla genérica de una aplicación típica se muestra en la Ilustración 24.

Ilustración 24. Pantalla genérica de una aplicación típica.

Las pantallas Pi se muestran en la Ilustración 25.

Ilustración 25. Pantallas del storyboard navegacional.

Cada una de estas pantallas está relacionada con una o varias actividades que se describen en las secciones siguientes.

3.4.2. Actividades.

Las diferentes actividades que permiten implementar el funcionamiento de cada una de las pantallas del storyboard navegacional diseñado se describen en las secciones siguientes.

3.4.2.1 Login.

Esta es la actividad relacionada con la P1. Es la primera que se encuentra una vez se ha abierto la aplicación. En ella se pide introducir la matrícula del coche en el que se encuentra la sillita de bebé que se quiere asociar al beacon y una contraseña. Que se guarde la información es posible gracias a SharedPreferences(). Que permite asociar una serie de características a un perfil y guardarlas para posteriores veces en que se use la aplicación, para ello archiva estas preferencias en un archivo .xml. Además, puede accederse a ellas desde cualquier otra actividad.

Extracto de código fuente:

```
SharedPreferences preferences = getSharedPreferences("MyPREFS",
MODE_PRIVATE);

String userDetails = preferences.getString( plate + password +
"data", "Usuario o contraseña incorrectos" );
SharedPreferences.Editor editor = preferences.edit();
editor.putString( "display", userDetails );
editor.commit();
```

Por otra parte, en esta actividad de Login, si se pulsa el botón " Entrar" estando ya registrado se lanzará la actividad principal P4 y un Toast que indica la matrícula y el email asociado a ella. Mientras que, si se pulsa el botón "Registrarse", se lanzará automáticamente la P2.

3.4.2.2 RegistrationScreen

A esta actividad se accede desde la actividad de Login tras pulsar el botón "Registrarse", como se acaba de indicar. En ella se encuentran varios cuadros de texto para rellenar y un par de botones. Si se pulsa el botón "Asociar Beacon" la aplicación conduce directamente a la P3, que es la lista de beacons de la que se tendrá que seleccionar cuál se pretende asociar al presente perfil. Si tras esto se rellenan los datos y se pulsa el botón "Guardar", se llamará a SharedPreferences(), se almacenarán los datos del usuario como se explicaba arriba y se retornará a la P1.

Extracto de código fuente:

```
SharedPreferences.Editor editor = preferences.edit();

editor.putString(newPlate + newPassword + "data", newPlate + "\n"+
```

```
newEmail);
editor.putString( "tag", tag );
editor.putString( "plate", newPlate );
editor.putString( "user", newUser );
editor.putString( "password", newPassword );
editor.putString( "email", newEmail );
editor.commit();

Intent loginScreeen = new Intent( RegistrationScreen.this,
Login.class );
startActivity( loginScreeen );
```

3.4.2.3 Lista Beacons

A esta otra actividad se accede a través de P2 y tras haber pulsado en esta el botón "Asociar Beacon", como se explicaba en el punto anterior. Una vez aquí se despliega una lista de todos los beacons cercanos con sus identificadores y colores característicos. Esto se realiza mediante el método Ranging de los beacons como ya se aclaró en otro apartado. Al pulsar sobre cualquiera de ellos queda seleccionado y se envía está información a la actividad P2 lanzándose esta de nuevo.

Para hacer posible este proceso se necesitan varias actividades auxiliares, como son:

- **Proximity Content Manager**: es en la que se gestiona el contenido de proximidad recibido por parte de los beacons
- **Proximity Content Adapter**: es la que recibe de Proximity Content Manager los datos y los adecúa al formato en el que se quieren mostrar en pantalla.
- **Proximity Content**: es la que hace posible recibir el contenido de proximidad de cada beacon.
- **Utils**: en esta actividad están contenido los colores que se mostrarán según el beacon que se detecte y la forma en la que aparecerá por pantalla el identificador de cada uno.

3.4.2.4 Main Activity

Es la actividad principal P4, se accede a ella tras introducir unos datos de Login correctos. - En su pantalla se encuentran 3 botones que dan acceso a otras 3 actividades secundarias, Modo Control P5, Modo Alertas P7 y Modo Registro P6. Cuando se pulsan los distintos botones de los modos se aprovecha para pasar también algunos parámetros a las actividades de destino.

Además de esto, la actividad principal realiza varias tareas. Una de ellas es la más importante de esta aplicación y la referente a los iBeacons. Básicamente lo que permite esta actividad es

detectar cuando se entra o se sale de un rango previamente establecido y utiliza esos datos para enviar una notificación cuando esto ocurre y una alarma también en el caso de que estemos saliendo. Cuando se sale de rango y la notificación es enviada, se activa un temporizador que dura 20 minutos, y al finalizar la cuenta atrás de este se le pasan varios parámetros a la actividad P7 entre los que está un "flag" que indica que el temporizador ya ha acabado.

Otra de las cosas que se realizan en esta actividad es la comprobación de permisos, tanto de Bluetooth, como de Ubicación y de envío de SMS.

Además contiene un método específico que controla la puesta en marcha del temporizador y las acciones a realizar cuando este acaba.

3.4.2.5 Modo Control

Se puede acceder a esta pantalla pulsando el botón adecuado en la clase Main. Una vez dentro lo que se ve es información acerca del beacon y la persona a la que se avisará en el caso de que no se pare la alarma al sonar, entre otras cosas. Esta información es actualizada cada vez que se pulsa el botón en la actividad P1, pues para conocer los parámetros que nos permiten mostrarla se han de recibir de esta primero.

Extracto de código fuente:

```
String kidName = getIntent().getExtras().getString( "kid" );
boolean bandera = getIntent().getExtras().getBoolean( "bandera" );
String contad = getIntent().getExtras().getString( "cont" );
String idBeac = getIntent().getExtras().getString( "beacon" );
```

3.4.2.6 Modo Registro

Para acceder al Modo Registro P6 se debe pulsar el botón para tal fin presente en la pantalla principal. Su layout muestra una serie de cuadros de texto que se han de rellenar con los datos que solicita y cuatro botones, uno para añadir a una persona de confianza, un par para borrar o actualizar a cualquier persona ya introducida y uno más para mostrar todos los datos que se han introducido desde que se instaló la app. Todo esto es posible mediante una base de datos: SQLite y una clase auxiliar llamada DataBaseHelper.

La actividad del Modo Registro P6 es la que lanza los mensajes tipo Toast que indican las acciones que se han ido produciendo tras pulsar los botones además de llamar a DataBaseHelper, que es la que contiene el código que se usa para añadir, actualizar o borrar a cualquier persona de la base de datos.

Esta base de datos es útil para nuestra aplicación a la hora de nombrar a personas de nuestra confianza a las que se les enviará un SMS en el caso de que en el momento de la alerta el padre en cuestión no responda.

Extracto de código fuente:

```
public boolean addData(String name, String email, String Tlf ){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues(  );
    contentValues.put(COL2, name);
    contentValues.put(COL3, email);
    contentValues.put(COL4, Tlf);

    long result = db.insert( NOMBRE_TABLA, null, contentValues );
    if(result == -1){
        return false;
    }
    else {
        return true;
    }
}

public boolean update_Data(String id, String Name, String Email,
String Tlf){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put( COL1, id );
    contentValues.put( COL2, Name );
    contentValues.put( COL3, Email );
    contentValues.put( COL4, Tlf );
    db.update( NOMBRE_TABLA, contentValues, "ID = ?", new String []
{id} );
    return true;

}

public Integer deleteData(String id){
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete( NOMBRE_TABLA, "ID = ?", new String[] {id} );

}
```

Estos son los métodos presentes en la clase DataBaseHelper que se usan para añadir, borrar o actualizar los datos de una persona.

3.4.2.7 Modo Alertas.

Esta última actividad es la asociada a la pantalla P7. En ella solamente aparece un botón que permite parar la alarma en caso de que haya desaparecido el pequeño diálogo (P4 alertas) que se lanza a la vez que la alarma, sin embargo, esta no es la única tarea que realiza.

A la vez que se lanza la alarma indicadora de que se ha salido del rango con es Smartphone se inicia un contador de 20 minutos. Trascurrido este tiempo se lanzará la pantalla P7 a la

que se le pasará un parámetro que indica que el temporizador ha llegado a su fin. Esto lo usará la actividad para extraer el número de teléfono de la última persona introducida en la base de datos, el nombre de la misma y el nombre de quien aparece registrado en la aplicación. Con todo ello procederá a enviar un SMS a la persona encontrada en la base de datos pidiéndole que contacte con el padre en cuestión y se asegure de que su hijo está bien.

El método usado para enviar SMS está dentro de una estructura tipo try{}catch{}

Extracto de código fuente:

```
try{
    //usamos la clase smsmanager//
    SmsManager sms = SmsManager.getDefault();
    //insertamos los datos para enviar el sms//
    sms.sendTextMessage( telefonillo, null, datosbuenos, null, null
);
//Mostramos un mensaje con el toast para verificar que se ha enviado
    Toast.makeText( getApplicationContext(),"Mensaje enviado
correctamente!", Toast.LENGTH_LONG ).show();

}
catch(Exception e){
    //mostramos Toast
    Toast.makeText( getApplicationContext(),"Mensaje no enviado :(
Verifique permisos", Toast.LENGTH_LONG ).show();
    e.printStackTrace();
}
```

3.4.3. Uso de la aplicación. Manual de Usuario.

Una vez se ha realizado la configuración básica imprescindible desde la página de Estimote Cloud ya se puede usar la aplicación con normalidad.

3.4.3.1 Registro

Lo primero que se debe hacer la primera vez que se accede a la aplicación es registrarse. Para ello en la pantalla P1 Login se ha de pulsar el botón "Registrarse". Este conduce a la pantalla P2 Registration Screen y allí lo primero será asociar un iBeacon a nuestro perfil. Para hacer esto se pulsa el botón "Asociar Beacon" y tras conceder los permisos de Bluetooth y Ubicación se despliega una lista de entre la que se debe seleccionar un dispositivo. Tras pinchar sobre uno, se laza de nuevo la pantalla de registro que se deberá completar con los datos pertienentes para pulsar al acabar el botón de "Guardar". Se lanzaría entonces de nuevo la pantalla de registro y una vez ahí, al introducir la matrícula registrada y la constraseña elegida se da paso a la aplicación en sí.

En el caso de que la matrícula o la constraseña no fuesen las correctas se lanza un mensaje que lo indica. Se muestran a continuación todas estas diferentes pantallas por las que pasa la aplicación en orden de aparición (ver Ilustración 26).

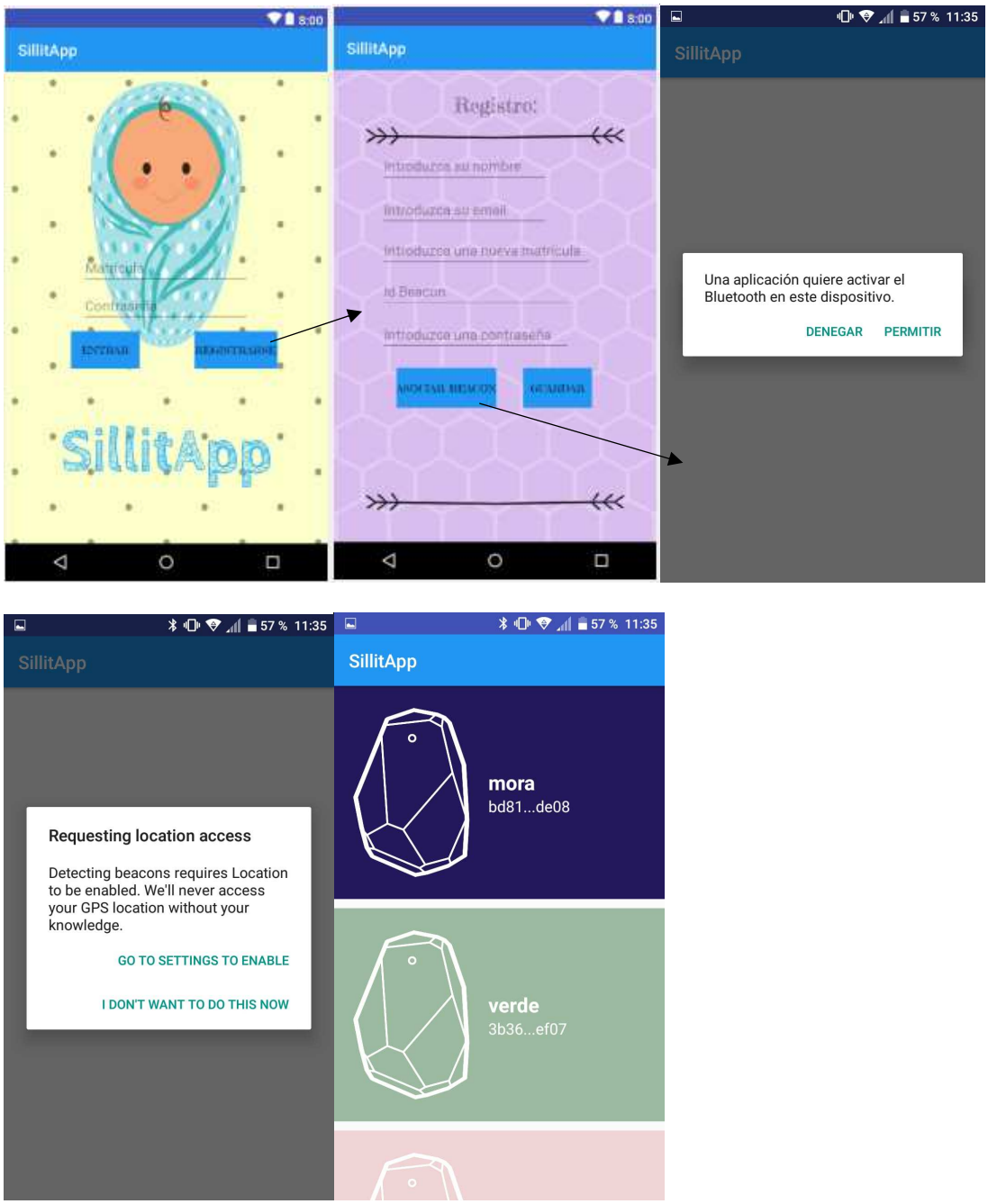

Ilustración 26. Pantallas de la aplicación diseñada conforme al diseño del storyboard navegacional.

3.4.3.2 Manejo básico

Una vez en la página principal se verán tres botones distintos que dan acceso a cada una de las actividades que ya han sido descritas. El que aparece en primer lugar es el del Modo Control, luego el del Modo Registro y por último el del Modo Alertas, ver Ilustración 27.

Ilustración 27. Pantalla de selección de modo.

Si se hace clic sobre el primero se accederá a la pantalla del modo de control, que muestra la temperatura del lugar en el que se encuentra el iBeacon, con cuál de los iBeacons está asociado el perfil actual, si el iBeacon se encuentra dentro o fuera del rango establecido, a quién se enviará el SMS de ser necesario y cuántas veces se ha salido del rango desde que se activó, ver Ilustración 28.

Ilustración 28. Pantalla de información transmitida desde el beacon y datos de configuración de acciones.

Si se pulsa sobre el último aparecerá la pantalla del modo alertas con un botón central que permite parar la alarma en caso de que haya desaparecido el dialogo que se lanza al activarse esta.

Por último, si se pulsa en el botón central se entra en el modo registro, que es en el que se decide a quien se avisará en caso de emergencia. Este modo permite agregar, modificar o eliminar a una persona de la lista de personas de confianza.

Para agregar a una nueva persona, solamente han de introducirse los datos y pulsar el botón "Agregar Persona". En el caso de que se quiera eliminar o modificar una persona ya presente en la base de datos se debe introducir el número ID de la misma que se encuentra sobre el nombre de cada perfil en la lista de personas de confianza y pulsar el botón correspondiente después. Esta lista puede consultarse pulsando el botón "Mostrar Datos". Se pueden ver a continuación capturas de dichas pantallas, ver Ilustración 29.

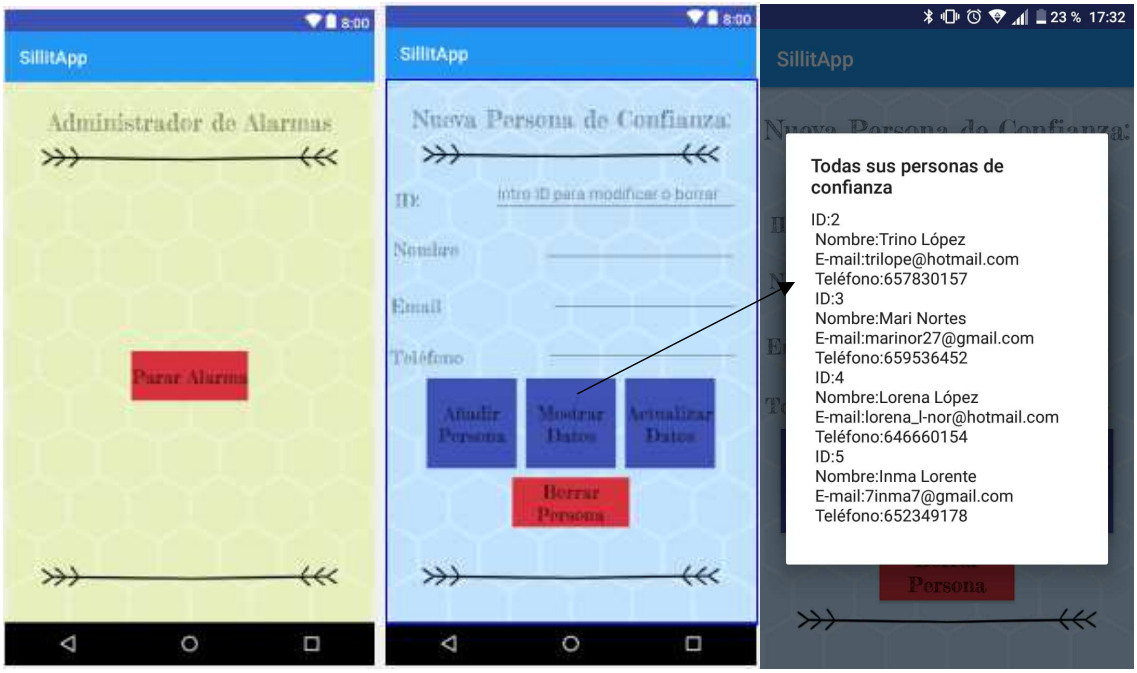

Ilustración 29. Diferentes pantallas de la aplicación relacionadas con la gestión de la alarma y la lista de personas de confianza.

En el caso de que, al sonar la alarma, inmediatamente el padre o madre del bebé se dé cuenta y vuelva a por él, la alarma puede pararse pulsando el botón aceptar del diálogo que se lanza a la vez que ésta, ver Ilustración 30.

Ilustración 30. Diálogo de parada de alarma.

Capítulo 4. Implantación y prueba del sistema.

Una vez expuesto todo lo anterior y con la aplicación en marcha es el momento de diseñar una pequeña batería de pruebas que permita garantizar que la aplicación diseñada cumple con su propósito y funciona como era de esperar.

4.1. Diseño de la batería de pruebas.

Las pruebas que se le realizarán a la aplicación consisten en la propuesta de diversas situaciones comunes a las que se espera que se enfrente realmente la aplicación y el estudio de los resultados de las mismas con el fin de futuras mejoras.

4.1.1. Prueba básica de arranque

Esta prueba básica consiste en comprobar que todo funciona correctamente a la hora de arrancar la aplicación por primera vez y que lanza las correspondientes advertencias en caso de que no haya datos que mostrar.

4.1.2. Prueba de Login incorrecto

Esta prueba como su nombre indica consiste simplemente en tratar de acceder a la aplicación con unas credenciales que no se hayan asociado a un perfil previamente y ver si se puede entrar o si de lo contrario se obtiene algún mensaje informativo del motivo.

4.1.3. Prueba de diagnóstico de funcionamiento de proximidad

Este test se divide en dos partes, una en la que se va a comprobar que una vez hemos salido del rango se envía una notificación con vibración y sonido y aparece una ventana emergente que nos permite parar la alarma, y otra en la que se comprueba que al entrar en el rango se envía otra notificación que lo corrobora.

En este test se pretendía verificar que al salir del rango efectivamente se ponía en marcha la alarma, saltaba una ventana emergente que nos permitía silenciarla y se enviaba una notificación acompañada de vibración al teléfono.

4.1.4. Prueba de aplicación en teléfono distinto y de funcionamiento de envío de SMS

En este caso lo que se pretende comprobar es que la aplicación funciona correctamente en otro teléfono distinto al de la primera prueba y a su vez asegurar que en caso de que tras 20

minutos no se haya parado la alarma se envía un SMS a la última persona de confianza que figure en la base de datos del Modo Registro.

4.2. Resultados obtenidos.

Se exponen en este apartado los resultados obtenidos en la realización de las pruebas más arriba descritas.

4.2.1. Resultado de la prueba simple de arranque

Se instaló la app en un dispositivo Android que contaba con la versión 6.0 y API 23. Una vez instalada se abrió y se procedió con el registro tal y como se explicaba más arriba. Se concedieron los permisos pertinentes, el de enviar SMS, el de Bluetooth y el de Ubicación. Una vez hecho esto se desplegó la lista de iBeacons, de entre ellos se escogió uno. Se rellenaron entonces los datos pertinentes y se pulsó el botón de "guardar". Una vez hecho esto y de vuelta en la pantalla de Login se introdujeron los datos y se accedió a la pantalla principal. En ella, en el modo Control, el campo de la persona a la que se debía avisar aparecía vacío y un Toast al acceder a este modo nos avisaba de ello. Tras ir al modo Registro, si se pulsa el botón "mostrar datos" nos saltaba un mensaje que indicaba que no había datos aún. Ambos errores se solucionaron añadiendo una persona de confianza. Cuando se seleccionó una vez más el modo control ya no apareció ningún Toast mostrando error y sí el nombre de la persona a la que acudir en el campo correspondiente.

4.2.2. Resultado de la prueba de Login incorrecto.

Se procede en esta prueba a rellenar los campos de contraseña y matrícula con datos aleatorios y se pulsa el botón de "entrar", tras esto se lanza un Toast que reza "Usuario o contraseña incorrectos". Lo mismo ocurre si se intenta entrar con la matricula asociada a un perfil previamente registrado, pero con una contraseña aleatoria.

4.2.3. Resultado de la prueba de diagnóstico de funcionamiento de proximidad.

Para llevarla a cabo se pone en marcha la aplicación y se le conceden los permisos, pero al entrar no consigue detectar ningún beacon en el rango, a pesar de haberlo asociado previamente. Se cierra y se vuelve a abrir la aplicación y esta vez, al arrancar con los permisos previamente concedidos, sí que funciona correctamente. Se recibe entonces una notificación indicando que nos encontramos dentro del rango establecido. Puede comprobarse también

desde el modo Control. Tras esperar durante cinco minutos y con el móvil bloqueado se procede a salir del rango del beacon. Cuando se está a más de 3 metros, que es el rango que se ha establecido previamente comienza a sonar una alarma, al desbloquearlo se ve una notificación que indica que se está olvidando al niño en el coche y una ventana emergente que permite parar la alarma pulsando el botón aceptar, y eso es lo que se hace. Tras esto se permanece durante otros cinco minutos fuera del rango, y nada inesperado ocurre. Todo funciona correctamente.

4.2.4. Resultado de la prueba de aplicación en teléfono distinto y de funcionamiento de envío de SMS.

Se instala para ello la aplicación en otro terminal Android, en este caso 7.0 API 24 y se introduce un número de teléfono al que se tiene acceso en la base de datos. Se procede entonces al registro para comprobar que todo funciona correctamente también en este móvil y tras verificar que lo hace y esperar de nuevo cinco minutos se procede a salir del área delimitada por el beacon. De nuevo, con el teléfono bloqueado suena una alarma y una notificación aparece en la pantalla, pero esta vez simplemente se silencia para no escucharla durante los siguientes 20 minutos y se espera a que pasen. Trascurrido ese tiempo se recibe un SMS en el teléfono introducido en la base de datos.

Capítulo 5. Conclusiones y trabajo futuro.

En este capítulo final se hablará de las conclusiones a las que se ha llegado tras la realización del presente trabajo y se propondrán posibles mejoras y ampliaciones de la aplicación.

5.1. Conclusiones.

Lo que se pretendía en este trabajo era estudiar los sistemas existentes en el mercado destinados a evitar el olvido de bebés en los coches de sus padres. También se ha hablado de las distintas alternativas de tecnologías de comunicación inalámbrica y de localización en interiores y exteriores. Tras ir seleccionando de entre las propuestas las que mejor se adecuaban al fin perseguido se introdujeron las balizas de BLE iBeacon como posible solución, ya que implementaban la tecnología de comunicación inalámbrica que se estimó más adecuada. Finalmente, se comentó también el sistema operativo móvil que se creyó más conveniente usar y como el desarrollo de una aplicación en este en combinación con el iBeacon iba a conseguir dar solución al problema en un principio planteado.

Es cierto que algunos de los sistemas existentes en el mercado que se encontraron podrían solucionar de manera más sofisticada el olvido de bebés, pero también es cierto que son en la mayoría de los casos más costosos o menos simples a la hora de usarlos.

El sistema que se ha desarrollado permite satisfacer esa necesidad planteada de una forma eficaz y fácil de usar a la par que económica. Solo consta de la pequeña baliza BLE y un smartphone y tiene funcionalidad suficiente como para indicarnos no solamente que se está dejando olvidado al bebé sino también para mostrar cierta información útil acerca del entorno cercano y comunicar a alguien externo la emergencia en caso de que fuese necesario.

5.2. Trabajo futuro.

Como mejoras para el sistema, se podría implementar la aplicación en conjunto con un sensor de peso que detectase si el bebé está sentado sobre la sillita o no para evitar tener que mover el iBeacon cuando salimos del coche y poder dejarlo fijo en la sillita.

También se podría ampliar la aplicación conectándola con un servicio de mensajería en la nube del tipo Firebase de Google que permitiese enviar una notificación de que hay un bebé cercano en peligro a cualquier usuario que la tuviese descargada.

Por otra parte, se podría también implementar el modo Monitoring de los beacons en un Servicio en lugar de en una Actividad de modo que este siguiese detectando si se entra o se sale del rango incluso en segundo plano.

Además, podría desarrollarse la aplicación para el sistema operativo móvil iOS, que como se estudió, es otro de los más importantes.

ANEXO

En este Anexo se muestran los elementos más significativos del código fuente de la aplicación diseñada.

Login.

```
package com.upct.lorenalpez.remasterizada;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.util.Collections;
import java.util.Set;

public class Login extends AppCompatActivity {

    String title3;
    //¿como marco que ya he pasaso por el registro una vez?. A lo
mejor veindo que las sharedpreferences no están vacías?

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_login );
        final EditText etName = (EditText)findViewById( R.id.etName
);
        final EditText etPassword = (EditText)findViewById(
R.id.etPassword );
        Button btnLogin = (Button)findViewById( R.id.btn_entrar );
        Button btnRegistration = (Button)findViewById(
R.id.btn_registrarse );



        btnLogin.setOnClickListener( new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String plate = etName.getText().toString();
            String password = etPassword.getText().toString();


            SharedPreferences preferences =
getSharedPreferences("MyPREFS", MODE_PRIVATE);

            String userDetails = preferences.getString( plate +
password + "data", "Usuario o contraseña incorrectos" );
            SharedPreferences.Editor editor = preferences.edit();
            editor.putString( "display", userDetails );
            editor.commit();
            if(userDetails=="Usuario o contraseña incorrectos" )
            {
```

```
                Toast.makeText(Login.this, userDetails,
Toast.LENGTH_LONG).show();
            }
            else{
            Intent displayScreen = new Intent( Login.this ,
MainActivity.class);
            startActivity( displayScreen );}
        }
    } );

    btnRegistration.setOnClickListener( new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent registerScreen = new Intent( Login.this ,
RegistrationScreen.class);
            startActivity( registerScreen );
        }
    } );



    }


}
```

RegistrationScreen

```
package com.upct.lorenalpez.remasterizada;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class RegistrationScreen extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.registration_screen );

        final EditText userName = (EditText)findViewById(
R.id.etNewName );
        final EditText password = (EditText)findViewById(
R.id.etNewPassword );
        final EditText plate = (EditText)findViewById(
R.id.etNewPlate );
        final EditText email = (EditText)findViewById(
R.id.etNewEmail );
        final EditText beacon = (EditText)findViewById(
R.id.etNewBeacon );
        Button btnRegistro =(Button)findViewById( R.id.btn_guardar
);
```

```
        Button btnAsociar = (Button)findViewById(
R.id.btn_asociarbeacon );


        btnRegistro.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences preferences =
getSharedPreferences( "MyPREFS", MODE_PRIVATE );
                String newPlate = plate.getText().toString();
                String newUser =userName.getText().toString();
                String newPassword =password.getText().toString();
                String newEmail =email.getText().toString();
                String title2 = "na";
                title2 = getIntent().getExtras().getString(
"newbeacon" );
                if(title2 != null){ beacon.setText( title2 );}
                String tag = beacon.getText().toString();



                SharedPreferences.Editor editor =
preferences.edit();

                editor.putString(newPlate + newPassword + "data",
newPlate + "\n"+ newEmail);
                editor.putString( "tag", tag );
                //editor.putString( "plate", newPlate );
                editor.putString( "user", newUser );
                //editor.putString( "password", newPassword );
                editor.putString( "email", newEmail );
                editor.commit();

                Intent loginScreeen = new Intent(
RegistrationScreen.this, Login.class );
                startActivity( loginScreeen );


            }
        } );

        btnAsociar.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View vi) {
                Intent vengaAsocia= new
Intent(RegistrationScreen.this, ListaBeacons.class);
                startActivity( vengaAsocia );
            }
        } );


    }

}
```

ListaBeacons

```
package com.upct.lorenalpez.remasterizada;

import [..]

//
// Running into any issues? Drop us an email to:
contact@estimote.com
//

public class ListaBeacons extends AppCompatActivity implements
AdapterView.OnItemClickListener{

    private ProximityContentManager proximityContentManager;
    private ProximityContentAdapter proximityContentAdapter;




    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.lista_beacons_activity);

        proximityContentAdapter = new ProximityContentAdapter(this);
        GridView gridView = findViewById(R.id.gridView);
        gridView.setAdapter(proximityContentAdapter);
        gridView.setOnItemClickListener( this );


        RequirementsWizardFactory
                .createEstimoteRequirementsWizard()
                .fulfillRequirements(this,
                        new Function0<Unit>() {
                            @Override
                            public Unit invoke() {
                                Log.d("app", "requirements
fulfilled");
                                startProximityContentManager();
                                return null;
                            }
                        },
                        new Function1<List<? extends Requirement>,
Unit>() {
                            @Override
                            public Unit invoke(List<? extends
Requirement> requirements) {
                                Log.e("app", "requirements missing:
" + requirements);
                                return null;
                            }
                        },
                        new Function1<Throwable, Unit>() {
                            @Override
                            public Unit invoke(Throwable throwable)
{
                                Log.e("app", "requirements error: "
```

```
+ throwable);
                                    return null;
                                }
                            });
    }

    private void startProximityContentManager() {
         EstimoteCloudCredentials cloudCredentials =
                 new EstimoteCloudCredentials("listadebeacons-cnm",
"ab5a1b3653a9e73dc1dafed7af7c9d4d");
        proximityContentManager = new ProximityContentManager(this,
proximityContentAdapter, cloudCredentials);
        proximityContentManager.start();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (proximityContentManager != null)
            proximityContentManager.stop();
    }

    @Override
    public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {

            ProximityContent content = (ProximityContent)
adapterView.getItemAtPosition( position );
            String title = content.getTitle();

            Toast.makeText( this,  title, Toast.LENGTH_SHORT
).show();

            Intent deVuelta = new Intent( ListaBeacons.this,
RegistrationScreen.class );
            deVuelta.putExtra( "newbeacon", title );
            startActivity( deVuelta );


    }

}
```

5.2.1. Main Activity

```
package com.upct.lorenalpez.remasterizada;


import […]


public class MainActivity extends AppCompatActivity {

    private Button btn_ModoControl;
    private Button btn_ModoRegistro;
    private Button btn_ModoAlertas;

    public boolean bandera, flag, boton;
    public int brewTime, mosquito = 5, cont=0;
```

```
    private boolean working;
    private CountDownTimer timer;
    public String tvTime, conti;
    public String deskOwner;
    public static String taggi;




    //prox
    private ProximityObserver proximityObserver ;

    Uri uri = RingtoneManager.getDefaultUri(R.raw.bright_morning);
    long[] voltage ={500,1000};
    public MediaPlayer mp;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setTheme( R.style.AppTheme );
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_main );


        SharedPreferences preferences = getSharedPreferences(
"MyPREFS", MODE_PRIVATE );
        String display = preferences.getString( "display", "" );
         taggi = preferences.getString( "tag", "multicolor" );


       Toast.makeText(MainActivity.this, display,
Toast.LENGTH_LONG).show();

        //verificar si tenemos permisos para enviar sms
        int permissionCheck = ContextCompat.checkSelfPermission(
this, Manifest.permission.SEND_SMS );
        if(permissionCheck!= PackageManager.PERMISSION_GRANTED){
            Toast.makeText( getApplicationContext(), "No se tiene
permiso para enviar SMS", Toast.LENGTH_LONG ).show();
            ActivityCompat.requestPermissions( this, new
String[]{Manifest.permission.SEND_SMS}, 225 );
        }
        else{
            Log.i("Mensaje", "Se tiene permiso para enviar SMS");
        }







        brewTime=1;
        working = false;
        boton = false;
        flag = false;
```

```
        //Base de datos
        btn_ModoControl =(Button)findViewById(R.id.btn_ModoControl);
        btn_ModoRegistro
=(Button)findViewById(R.id.btn_ModoRegistro);
        btn_ModoAlertas =(Button)findViewById(R.id.btn_ModoAlertas);

        mp = MediaPlayer.create(this,R.raw.bright_morning);


        btn_ModoControl.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
Modo_Control.class);
                intent.putExtra( "kid", deskOwner );
                intent.putExtra( "bandera", bandera );
                intent.putExtra( "cont", conti );
                intent.putExtra( "beacon", taggi );
                startActivity(intent);
            }
        });
        btn_ModoRegistro.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent2 = new Intent(MainActivity.this,
Modo_Registro.class);
                startActivity(intent2);
            }
        });

        btn_ModoAlertas.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent3 = new Intent(MainActivity.this,
Modo_Alertas.class);
                intent3.putExtra( "tiempo", tvTime );
                intent3.putExtra( "bandera", bandera );
                intent3.putExtra( "flag", flag );
                intent3.putExtra( "boton", boton );
                intent3.putExtra( "kid", deskOwner );
                startActivity(intent3);
            }
        });


        //Proximidad

        EstimoteCloudCredentials cloudCredentials =
                new EstimoteCloudCredentials("remasterizada-
proximidad-2bc", "6691d6ea37e81e18540f7db4bef501d1");
        this.proximityObserver = new
ProximityObserverBuilder(getApplicationContext(),
cloudCredentials).onError( new Function1<Throwable, Unit>() {
            @Override
            public Unit invoke(Throwable throwable) {
                Log.e("app", "proximity observer error:"
+throwable);
                return null;
```

```
            }
        }).withBalancedPowerMode().build();


        final ProximityZone zone = new ProximityZoneBuilder()
                .forTag(taggi)
                .inCustomRange(2.0)
                .onEnter(new Function1<ProximityZoneContext, Unit>()
{
                    @Override
                    public Unit invoke(ProximityZoneContext context)
{
                        bandera = true;
                        deskOwner =
context.getAttachments().get("desk-owner");
                       Log.d("app", "Welcome to " + deskOwner + "'s
desk");
                        NotificationCompat.Builder notif= new
NotificationCompat.Builder(getApplicationContext());

notif.setSmallIcon(android.R.drawable.sym_def_app_icon).setContentTi
tle("HOLA").setContentText("Bienvenido al coche de "+deskOwner)

.setTicker("Notificacion").setColor(52);
                        Toast.makeText(MainActivity.this,
"Bienvenid@ al coche de "+ deskOwner, Toast.LENGTH_LONG).show();
                        NotificationManager man
=(NotificationManager)getApplicationContext().getSystemService(
Context.NOTIFICATION_SERVICE);
                        man.notify(1,notif.build());
                        return null;
                    }
                })
                .onExit(new Function1<ProximityZoneContext, Unit>()
{
                    @Override
                    public Unit invoke(ProximityZoneContext context)
{
                        bandera = false;
                        if(bandera==false){
                            cont++;
                            conti = String.valueOf( cont );
                            mp.start();
                            aTrabajar();
                            // Creamos el Dialog que se mostrará
para poder pausar la
                            // reproducción del sonido.
                            AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
                            // Le damos un texto al título del
diálogo.
                            builder.setTitle("Alarma");
                            // Establecemos un mensaje a mostrar al
usuario.
                            builder.setMessage("Desea parar la
alarma?");
                            // Creamos un botón y escribimos su
evento &amp;quot;onClick&amp;quot;.
                            builder.setPositiveButton("Aceptar", new
DialogInterface.OnClickListener() {
```

```
                                @Override
                                public void onClick(DialogInterface
dialog, int which) {
                            // Cuando se pulse el botón, se parará
la reproducción del archivo.
                                    mp.pause();
                                    boton = true;

                                }
                            });
                            // Una vez creado el Dialog, lo
mostramos.
                            builder.show();
                        }
                        Log.d("app", "Adevertencia!!!");
                        NotificationCompat.Builder note= new
NotificationCompat.Builder(getApplicationContext());

note.setSmallIcon(android.R.drawable.alert_dark_frame).setContentTit
le("Es peligroso dejar solos a los niños en el
coche").setContentText("Se está olvidando a su hijo!!!")
                                .setTicker("Alarma").setColor(
Color.blue(155)).setVibrate(voltage);
                        Toast.makeText(MainActivity.this, "Se está
olvidado de su hijo!!!", Toast.LENGTH_LONG).show();
                        NotificationManager man
=(NotificationManager)getApplicationContext().getSystemService(Conte
xt.NOTIFICATION_SERVICE);
                        man.notify(1,note.build());
                        return null;
                    }
                })
                .build();




        RequirementsWizardFactory.createEstimoteRequirementsWizard()
                .fulfillRequirements(this, new Function0<Unit>() {
                            @Override public Unit invoke() {
                                Log.d("app", "requirements
fulfilled");

proximityObserver.startObserving(zone);
                                return null;
                            }
                        },
                        // onRequirementsMissing
                        new Function1<List<? extends Requirement>,
Unit>() {
                            @Override public Unit invoke(List<?
extends Requirement> requirements) {
                                Log.e("app", "requirements missing:
" + requirements);
                                return null;
                            }
                        },
                        // onError
                        new Function1<Throwable, Unit>() {
                            @Override public Unit invoke(Throwable
```

```
throwable) {
                                    Log.e("app", "requirements error: "
+ throwable);
                                    return null;
                                }
                            });




        }



    private void aTrabajar(){
        working = true;
        timer = new CountDownTimer(brewTime*60*1000, 100 ) {
            @Override
            public void onTick(long millisUntilFinished) {
                tvTime =String.valueOf(millisUntilFinished/1000) +
"s";

            }

            @Override
            public void onFinish() {
                flag = true;
                working= false;
                if(!boton){
                Intent finTempo = new Intent( MainActivity.this,
Modo_Alertas.class );
                finTempo.putExtra( "bandera", bandera );
                finTempo.putExtra( "flag", flag );
                finTempo.putExtra( "boton", boton );
                finTempo.putExtra( "tiempo", tvTime );
                finTempo.putExtra( "kid", deskOwner );
                startActivity( finTempo );}


            }
        };
        timer.start();
    }
}
```

Modo_Control

```
package com.upct.lorenalpez.remasterizada;

import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;
```

```
import android.widget.Toast;

public class Modo_Control extends AppCompatActivity {

    String Name;



    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_modo_control );
        String kidName = getIntent().getExtras().getString( "kid" );
        boolean bandera = getIntent().getExtras().getBoolean(
"bandera" );
        String contad = getIntent().getExtras().getString( "cont" );
        String idBeac = getIntent().getExtras().getString( "beacon"
);


        TextView tvNombre = (TextView)findViewById( R.id.tvKidname
);
        TextView temper = (TextView)findViewById( R.id.etTemper);
        TextView dist = (TextView)findViewById( R.id.etDist);
        TextView rango = (TextView)findViewById( R.id.etRango);
        TextView persona = (TextView)findViewById( R.id.etPersona);
        TextView contador = (TextView)findViewById(
R.id.etContador);

        DataBaseHelper myDataBaseHelper = new DataBaseHelper( this
);
        Cursor cursor = myDataBaseHelper.showData();
        if(cursor.getCount() == 0){
            Toast.makeText( this, "ERROR FALTA Datos",
Toast.LENGTH_LONG ).show();
        }
        else{
            while(cursor.moveToNext()){
                Name = cursor.getString( 1 );

            }
        }




        tvNombre.setText( kidName );
        persona.setText( Name );
        if(bandera)rango.setText( "Si" );
        else rango.setText( "No" );
        contador.setText( contad );
        temper.setText( "22°C" );
        dist.setText( idBeac );

    }
}
```

Modo_Registro

```
package com.upct.lorenalpez.remasterizada;

import android.content.Intent;
import android.database.Cursor;
import android.graphics.Typeface;
import android.os.Bundle;
import android.os.PersistableBundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Modo_Registro extends AppCompatActivity {

    DataBaseHelper peopleDB;

    Button btnAddData, btnViewData, btnUpdateData, btnDelete;
    EditText etName, etEmail, etTlf, etID;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_modo_registro );

        peopleDB = new DataBaseHelper( this );

        etName = (EditText) findViewById( R.id.etNombre );
        etEmail = (EditText) findViewById( R.id.etEmail );
        etTlf = (EditText) findViewById( R.id.etTelefono );
        etID = (EditText) findViewById( R.id.etID );
        btnAddData = (Button) findViewById( R.id.btn_addpersona );
        btnViewData = (Button) findViewById( R.id.btn_showdata );
        btnUpdateData = (Button) findViewById( R.id.btn_updatedata
);
        btnDelete = (Button) findViewById( R.id.btn_deletedata );

        AddData();
        ViewData();
        UpdateData();
        DeleteData();

    }

    public void AddData (){

        btnAddData.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = etName.getText().toString();
                String email = etEmail.getText().toString();
                String Tlf = etTlf.getText().toString();


                boolean insertData = peopleDB.addData( name, email,
Tlf );

                if(insertData == true){
                    Toast.makeText( Modo_Registro.this, "Datos
```

```
correctamente introducidos!", Toast.LENGTH_LONG).show();
                }
                else{
                    Toast.makeText( Modo_Registro.this, "Algo ha ido
mal :(", Toast.LENGTH_LONG).show();

                }
            }
        } );
    }

    public void ViewData(){

        btnViewData.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Cursor data = peopleDB.showData();
                if(data.getCount()==0){
                    //message
                    display( "Error", "No se encontraron datos." );
                    return;

                }
                StringBuffer buffer = new StringBuffer();
                while(data.moveToNext()){
                    buffer.append("ID:" + data.getString( 0 )+ "\n
");
                    buffer.append("Nombre:" + data.getString( 1 )+
"\n ");
                    buffer.append("E-mail:" + data.getString( 2 )+
"\n ");
                    buffer.append("Teléfono:" + data.getString( 3 )+
"\n ");


                    //display message
                    display( "Todas sus personas de confianza",
buffer.toString() );

                }
            }
        } );
    }

    public void display(String title, String message){
        AlertDialog.Builder builder = new AlertDialog.Builder( this
);
        builder.setCancelable( true );
        builder.setTitle( title );
        builder.setMessage( message );
        builder.show();
    }

    public void UpdateData(){
        btnUpdateData.setOnClickListener( new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                int temp = etID.getText().toString().length();
                if(temp > 0){
                    boolean update = peopleDB.update_Data(
```

```
etID.getText().toString(),etName.getText().toString(),etEmail.getTex
t().toString(),etTlf.getText().toString() );
                    if( update == true){
                        Toast.makeText( Modo_Registro.this, "Datos
actualizados con éxito!", Toast.LENGTH_LONG ).show();
}

                    else{
                        Toast.makeText( Modo_Registro.this, "Algo ha
ido mal :(", Toast.LENGTH_LONG ).show();

                    }}
                else {
                    Toast.makeText( Modo_Registro.this, "Debes
introducir un ID para poder actualizar los datos", Toast.LENGTH_LONG
).show();
                }
            }
        } );

    }

    public void DeleteData(){
        btnDelete.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int temp = etID.getText().toString().length();
                if(temp > 0){
                    Integer deleteRow = peopleDB.deleteData(
etID.getText().toString());
                    if(deleteRow > 0){
                        Toast.makeText( Modo_Registro.this, "Datos
borrados correctamente! ", Toast.LENGTH_LONG ).show();

                    }
                    else {
                        Toast.makeText( Modo_Registro.this, "Algo ha
idomal :( ", Toast.LENGTH_LONG ).show();

                    }
                }
                else{
                    Toast.makeText( Modo_Registro.this, "Debes
introducir un ID para poder actualizar los datos ",
Toast.LENGTH_LONG ).show();

                }
            }
        } );
    }



}
```

Modo_Alertas

```
package com.upct.lorenalpez.remasterizada;
```

```
import […]

public class Modo_Alertas extends AppCompatActivity {

    DataBaseHelper mDataBaseHelper;
    MediaPlayer mMediaPlayer = new MediaPlayer();



    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_modo_alertas );
        Intent incomingData = getIntent();
        String tempo = getIntent().getExtras().getString( "tiempo"
);
        String kidName = getIntent().getExtras().getString( "kid" );
        boolean band =incomingData.getExtras().getBoolean( "bandera"
);
        boolean flag = incomingData.getExtras().getBoolean( "flag" )
;
        boolean boton = incomingData.getExtras().getBoolean( "boton"
) ;
        String nombrecillo="Raul";
        String telefonillo="659536452" ;
        SharedPreferences preferences = getSharedPreferences(
"MyPREFS", MODE_PRIVATE );
        String padrecillo =preferences.getString( "user", "Julian" )
;
        TextView tvTime = (TextView)findViewById( R.id.tvTime );
        tvTime.setText( tempo );


        mDataBaseHelper = new DataBaseHelper( this );
        Cursor cursor = mDataBaseHelper.showData();
        if(cursor.getCount() == 0){
            Toast.makeText( this, "ERROR FALTAN Datos",
Toast.LENGTH_LONG ).show();
        }
        else{
            while(cursor.moveToNext()){
                nombrecillo =cursor.getString( 1 );
                telefonillo = cursor.getString( 3 ) ;
            }
        }




//        String datos = "holi";
        String datosbuenos = "Hola! "+ nombrecillo +"\n ¿Podrías
hablar con " + padrecillo +" y asegurarte de que su hijo"+ kidName+
"\n está bien?";

        if(band ==true) {
             Toast.makeText(Modo_Alertas.this, "Tu bandera! \n"+
telefonillo, Toast.LENGTH_LONG).show();
         }

        if(flag == true && boton == false  ){
```

```
            /* NotificationCompat.Builder notif= new
NotificationCompat.Builder(getApplicationContext());

notif.setSmallIcon(android.R.drawable.sym_def_app_icon).setContentTi
tle("HOLA").setContentText("Bienvenido al coche ")
                    .setTicker("Notificacion").setColor(52);
             Toast.makeText( Modo_Alertas.this, "Hola! "+ nombrecillo
+"\n ¿Podrías hablar con " + padrecillo +" y segurarte que su hijo "
+ kidName+ "\n está bien?", Toast.LENGTH_LONG ).show();
             NotificationManager man
=(NotificationManager)getApplicationContext().getSystemService(
Context.NOTIFICATION_SERVICE);
             man.notify(1,notif.build());*/
             mMediaPlayer.stop();




             //metodo enviar mensaje

             try{
                 //usamos la clase smsmanager//
                 SmsManager sms = SmsManager.getDefault();
                 //insertamos los datos para enviar el sms//
                 sms.sendTextMessage( telefonillo, null, datosbuenos,
null, null );
                 //Mostramos un mensaje con el toast para verificar
que se ha enviado
                 Toast.makeText( getApplicationContext(),"Mensaje
enviado correctamente!", Toast.LENGTH_LONG ).show();

             }
             catch(Exception e){
                 //mostramos Toast
                 Toast.makeText( getApplicationContext(),"Mensaje no
enviado :(  Verifique permisos", Toast.LENGTH_LONG ).show();
                 e.printStackTrace();



             }

         }



    }

    public void paro(View view) {
        mMediaPlayer.pause();
    }
}
```

Referencias.

[1] "10 dispositivos para que nadie olvide a los niños en el coche." [Online]. Available: https://motor.elpais.com/tecnologia/10-dispositivos-olvidar-ninos-coche/. [Accessed: 19-Feb-2019].

[2] "Windows Phone - Qué es y Definición 2019." [Online]. Available: https://conceptodefinicion.de/windows-phone/. [Accessed: 24-Mar-2019].

[3] "Windows Phone OS - definition - GSMArena.com." [Online]. Available: https://www.gsmarena.com/glossary.php3?term=windows-phone-os. [Accessed: 24-Mar-2019].

[4] "▷ ¿Qué es iOS? significado, definición, siglas y evolución 【 2019 】." [Online]. Available: https://swiftlatino.com/ios/. [Accessed: 24-Mar-2019].

[5] "¿Qué es Android?" [Online]. Available: https://www.xatakandroid.com/sistema-operativo/que-es-android. [Accessed: 24-Mar-2019].

[6] "Características y hardware de los dispositivos móviles - Tecnologias para el desarrollo de aplicaciones móviles." [Online]. Available: https://mastermoviles.gitbook.io/tecnologias2/caracteristicas-y-hardware-de-los-dispositivos-moviles. [Accessed: 24-Mar-2019].

[7] "BLE (Bluetooth Low Energy) | Elt." [Online]. Available: https://www.elt.es/ble-bluetooth-low-energy. [Accessed: 22-Mar-2019].

[8] "¿Qué novedades trae el nuevo Bluetooth 5.0?" [Online]. Available: https://wikiversus.com/informatica/bluetooth-5-novedades-velocidad/. [Accessed: 22-Mar-2019].

[9] J. Salazar, "REDES INALÁMBRICAS."

[10] "¿Cuáles son los principales protocolos Wifi? Todo lo que necesitas saber." [Online]. Available: https://www.profesionalreview.com/2017/11/18/cuales-principales-protocolos-wifi/. [Accessed: 23-Mar-2019].

[11] "Sistemas de Localización – AGPS." [Online]. Available: https://www.agps.es/sistemas-de-localizacion/. [Accessed: 25-Jan-2019].

[12] A. Pozo-Ruz, A. Ribeiro, M. C. García-Alegre, L. García, D. Guinea, and F. Sandoval, "SISTEMA DE POSICIONAMIENTO GLOBAL (GPS): DESCRIPCIÓN,

ANÁLISIS DE ERRORES, APLICACIONES Y FUTURO."

[13] "Tecnologías Localización Tiempo Real (RTLS): RFID vs BLE vs UWB – Blog de Sofia2 IoT Platform." [Online]. Available: https://about.sofia2.com/2017/05/25/sistemas-de-localizacion-en-tiempo-real-rtls-rfid-vs-ble-vs-uwb-tags/. [Accessed: 25-Jan-2019].

[14] "Siete gadgets para conseguir el sueño de encontrar los objetos importantes a la primera (o no perderlos)." [Online]. Available: https://www.xataka.com/accesorios/siete-gadgets-para-conseguir-el-sueno-de-encontrar-los-objetos-importantes-a-la-primera-o-no-perderlos. [Accessed: 24-Jan-2019].

[15] "What is Apple iBeacon? Here's what you need to know | ZDNet." [Online]. Available: https://www.zdnet.com/article/what-is-apple-ibeacon-heres-what-you-need-to-know/. [Accessed: 12-Feb-2019].

[16] "9 usos reales para comprender qué son los 'beacons' - Nobbot." [Online]. Available: https://www.nobbot.com/redes/9-usos-reales-comprender-los-beacons/. [Accessed: 19-Feb-2019].

[17] "What is a Beacon? - Beacon Basics - Kontakt.io." [Online]. Available: https://kontakt.io/beacon-basics/what-is-a-beacon/. [Accessed: 24-Mar-2019].

[18] "Conoce Android Studio | Android Developers." [Online]. Available: https://developer.android.com/studio/intro?hl=es-419. [Accessed: 30-Mar-2019].

[19] "Posición y contexto con Beacons – BEEVA Labs." [Online]. Available: https://labs.beeva.com/posición-y-contexto-con-beacons-15e662cfacf2. [Accessed: 31-Mar-2019].

[20] "Máster en Desarrollo de Aplicaciones Android - Arquitectura de Android." [Online]. Available: http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android. [Accessed: 31-Mar-2019].

[21] "Las piezas clave que hacen que el sistema Android funcione." [Online]. Available: https://elandroidelibre.elespanol.com/2016/11/arquitectura-sistema-android.html. [Accessed: 31-Mar-2019].

[22] "Aprendiendo Sobre La Arquitectura De Android." [Online]. Available:

Referencias.

http://www.hermosaprogramacion.com/2014/08/aprendiendo-la-arquitectura-de-android/. [Accessed: 31-Mar-2019].