UNIVERSIDAD AUTONÓMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Programación de Beacons para recomendación de POIs (Puntos de Interés)

Rubén Vela López
Tutor: Fernando Diez Rubio

Junio 2020

Programación de Beacons para recomendación de POIs (Puntos de Interés)

AUTOR: Rubén Vela López
TUTOR: Fernando Diez Rubio

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2020

Resumen

En las dos últimas décadas se ha producido un desarrollo impensable hace tiempo en todo lo referente a las tecnologías de la información y las comunicaciones. El desarrollo y la rápida evolución de la microelectrónica, han facilitado un avance tecnológico impredecible, que no sigue una evolución lineal, si no que se ha presentado de forma exponencial, dando a término con una tecnología más potente y con más capacidades para la interconexión de múltiples dispositivos, que se desarrollan a raíz de las necesidades que el propio auge de la tecnología que cada vez es más demandada.

Términos como Web 2.0, 3.0, 4.0, Internet de las cosas, digitalización, y tecnologías inalámbricas, como Bluetooth, GPS, NFC, etc. han pasado a formar parte de nuestro lenguaje cotidiano y por ende de nuestras vidas.

En este sentido, este Trabajo de Fin de Grado, consiste en la exploración, aproximación al conocimiento y uso de una nueva tecnología de comunicación basada en el principio de transmisión *Bluetooth Low Energy*. Esta tecnología nos permite una comunicación sencilla, de bajo consumo, entre unos dispositivos denominados Beacons y los dispositivos personales de los usuarios como por ejemplo *smartphones, tablets o smartwatchs*.

En el trabajo, se ha realizado un estudio de los beneficios que actualmente nos proporciona la plataforma del fabricante de Beacons *Estimote*, la cual permite la conexión de los Beacons con las aplicaciones, que tengamos desarrolladas para cumplir el objetivo que queremos proporcionar a este tipo de herramientas de comunicación.

Con este proyecto, aportamos conocimiento de esta tecnología, y a lo largo del mismo, poder mostrar mediante los distintos escenarios que hemos recreado con la aplicación de aproximación, creada a tal fin, las posibilidades que puede aportar la utilización de los dispositivos beacons en distintos ámbitos de desarrollo comercial.

Abstract (English)

In the last two decades, there has been an exponential development in information and communication technologies. This development is primarily due to the rapid evolution of microelectronics, resulting in a more powerful technology than increases the capacity for the interconnection of multiple devices. Also, this development comes from the increasing demand of customers that seek faster and user-friendly devices.

As an example of how technology is influencing our lives, we can observe how terms such as Bluetooth, GPS, or NFC have become part of our daily language.

This bachelor´s thesis aims to explore a new communication technology based on Bluethooth Low Energy transmisión, wich allows low consumption communication between devices called Beacons and personal devices such as smartphones, tablets, or smartwatches.

This work also describes a study of the main benefits that the Beacons manufacturer Estimote provides. This one allows the connection between the Beacons and the applications that we want to connect to.

Lastly, this work also mentions the most suitable scenarios where this technology could be implemented in order to provide substantial benefits to the users.

Palabras clave (castellano)

Beacons, Bluetooth, LTE, Android, Estimote, NFC, BLE, Aplicaciones móviles.

Keywords (inglés)

Beacons, Bluetooth, LTE, Android, Estimote, NFC, BLE, Mobile apps.

Agradecimientos

A mi familia por darme el apoyo constante y la ayuda necesaria para obtener una educación de grado, enseñarme a valorar lo obtenido con el esfuerzo y la superación del día a día, a no dejarse vencer por las adversidades que nos encontramos en el camino y apostar por la consecución de los retos que se proponen con dedicación y esfuerzo.

A mis amigos del colegio, los cuales todavía conservo, y que me han sido de gran apoyo en todos los años de esta carrera, dándome ánimos y estando siempre presentes cuando se necesitaban.

A mis compañeros de Universidad, por la inestimable ayuda y trato recibido durante todos estos años, por lo que ya los incluyo como mis amigos

A mi tutor Fernando, y a Alejandro por darme la oportunidad de conocer y trabajar con esta tecnología, darme el soporte necesario para avanzar y por su confianza para el desarrollo de este proyecto.

A los profesores del Grado por sus enseñanzas, que me han permitido tener los conocimientos necesarios para poder trabajar en el área siempre cambiante de las tecnologías de la información.

A Innova-TSN por confiar en mis capacidades y darme la oportunidad de trabajar con ellos, dándome la oportunidad de compatibilizar mis estudios de grado con el trabajo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE DIAGRAMAS

ÍNDICE DE CÓDIGOS

1 Introducción

En este capítulo se describe el marco general del proyecto y los objetivos propuestos.

1.1 Motivación

Esta memoria de TFG tiene el objetivo de conocer el funcionamiento de los dispositivos de corto alcance beacons, explorar e investigar su configuración y analizar las posibles aplicaciones en las que esta tecnología puede aplicarse.

En la actualidad tenemos poca información sobre esta tecnología y no hay grandes desarrollos que la contemplen, a pesar de ser una tecnología barata y de baja energía, aún no han alcanzado una comercialización que la extienda masivamente, es por ello que esta propuesta sea interesante con vistas al descubrimiento de nuevos usos y utilizaciones en sectores del marketing, entretenimiento, turismo, servicios, y otros muchos.

La motivación que nos hace elegir este proyecto, es la de utilizar tecnología de bajo consumo energético con alta capacidad de funcionamiento, en espacios interiores sin interferencias, y conseguir que este tecnología se conozca, y se pueda avanzar en ella para desarrollar proyectos que mejoren la calidad en los servicios de marketing de proximidad, geolocalización o posicionamiento en interiores.

Objetivos

Como consecuencia de esta motivación, los objetivos que nos hemos propuesto para este proyecto son los siguientes:

- O1: Conocer la tecnología beacons. Necesitamos conocer la tecnología para la aplicación de los protocolos entre los beacons y los dispositivos móviles.

- O2: Trabajar con el entorno de Estimote. Este objetivo implica conocer y experimentar con los recursos propuestos por el fabricante Estimote.

- O3: Interacción entre los dispositivos móviles y los beacons. Consiste en crear una aplicación para el dispositivo móvil que nos permita la interacción con los beacons.

- O4: Personalización de mensajes de notificación. El objetivo es poder mostrar mensajes diferentes para los distintos beacons detectados, así como su diferenciación en función de su posicionamiento en el rango de alcance.

Para el cumplimiento de los objetivos descritos, creamos una planificación dividida en dos fases, la primera fase se corresponde con la realización del proyecto, es decir, conocer la tecnología que vamos a utilizar y el trabajo de creación de la aplicación para interactuar con los beacons.

Por otro lado, en la segunda fase, se desarrolla esta memoria donde se han expuesto las ideas, los desarrollos necesarios para su funcionamiento y las pruebas realizadas durante este proyecto.

Una parte importante para la consecución de los objetivos, ha sido la aplicación de las enseñanzas que hemos recibido durante estos años en la Escuela, materias como, **Redes de comunicación,** nos han servido para conocer las interacciones con los beacons, **Análisis y diseño de software** nos ha permitido saber cómo disponer los casos de uso y la creación de diagramas que hemos aplicado en los escenarios de control, y lo aprendido en la asignatura de **Desarrollo de aplicaciones móviles,** ha sido fundamental para poder conseguir los objetivos propuestos, ya que esta tecnología se apoya básicamente en la realización de aplicaciones para móviles, y nos ha permitido tener el conocimiento para poder construir la aplicación de este proyecto.

Organización de la memoria

La memoria consta de 5 capítulos.

El primero es el correspondiente a esta **introducción**.

El estado del arte constituye el segundo capítulo, donde se contextualiza la tecnología beacon, frente a otras tecnologías de interconexión y la tipología de beacons que nos aporta el fabricante Estimote.

En el tercer capítulo se describe el **diseño y desarrollo** del proyecto, así como de la aplicación para móvil creada.

Pruebas y resultados es el cuarto capítulo, en el que mostramos las pruebas realizadas con los resultados obtenidos en cada una de ellas.

En el quinto capítulo, exponemos las **conclusiones** del trabajo realizado, y aspectos que se pueden tener en cuenta para tratar proyectos futuros.

2 Estado del arte

En nuestros tiempos, la tecnología evoluciona muy rápidamente ofreciéndonos cada día nuevos proyectos y oportunidades, cada vez es más habitual que utilicemos nuevos dispositivos y consultemos más información en la red. Durante muchos años se ha hablado del inicio de un mundo donde todo y todos estaremos conectados entre sí, donde muchos ya hemos escuchado y leído múltiples noticias sobre el "Internet de las Cosas" o sus siglas *IoT* (*Internet of Things*), lo cual nos permite la interconexión digital con objetos de uso cotidianos a través de aplicativos e internet.

La tecnología que presentamos en este capítulo facilita la interconexión de objetos, y en el mundo empresarial se está aplicando para el ámbito del marketing de proximidad.

2.1 ¿Qué es un Beacon?

Los Beacons son dispositivos basados en la tecnología de conectividad *bluetooth* de baja energía o BLE (por sus siglas en inglés de *Bluethoot Low Energy*), que emiten una señal única a cada dispositivo, tal y como lo hace una baliza o un faro. La señal que envía el beacon la reciben y procesan otros dispositivos. Como los beacons se pueden configurar para emitir señales a distintos alcances, el caso de uso más habitual, es que se conecten con los dispositivos más ampliamente comercializados, como son los *smartphones*,

Estos dispositivos se han convertido en tendencia en el ámbito del posicionamiento, contribuyendo de manera eficaz al ofrecimiento de servicios, o publicidad personalizada a aquellos dispositivos que se aproximan al beacon., coloquialmente llamadas experiencias de proximidad.

Figura 2.1 Beacon.

2.1.1 Tipos de Tecnologías

Tecnología RFID

Radio Frequency Identification (RFID) tecnología utilizada en etiquetas conteniendo un chip con identificador único, que permite su detección por un lector. La principal utilidad aplicada a esta tecnología es el control de stock, o la gestión de artículos en almacenes y tiendas.

Tecnología Wifi

La tecnología *Wifi (Wireless Fidelity)* es la más utilizada y más extendida en el ámbito de conexión inalámbrica que predomina para las redes de datos locales, y está generalizada en todos los móviles de última generación. La ventaja de esta tecnología para el uso en localización de interiores es la de no necesitar ninguna interacción ni instalación por parte del cliente, el sistema es capaz de identificar y triangular la posición del usuario y seguir sus movimientos. Por el contrario su mayor inconveniente es el elevado consumo eléctrico que utiliza, y que las señales que emiten tienen dificultad para transmitirse en superficies opacas y traspasar muros o paredes.

Tecnología NFC

NFC es la abreviatura de *Near Field Comunication* es un sistema de comunicación de corto alcance que utiliza la tecnología RFID para el intercambio de datos. Esta tecnología en la actualidad se aplica, principalmente, a los pagos mediante *contactless,* así como para la carga de la batería en los smartphones que contemplan esta opción.

Uno de los inconvenientes que nos encontramos al utilizar esta tecnología, es la necesidad de acercar el móvil a un beacon, pero sobre todo, esta tecnología por el momento no está muy extendida en móviles fuera de los ámbitos descritos anteriormente (pagos o carga inalámbrica).

2.2 Beacon Estimote

Estimote es una *start-up* que ha desarrollado una plataforma de análisis mediante la interacción de sensores, el principal área de consumo de la compañía son las tiendas minoristas y centros educativos. Actualmente envían “kits de desarrollo” a sus clientes para que prueben las nuevas tecnologías que desarrollan. Estos kits se basan en un dispositivo de muestra junto con el *Software Development Kit* (*SDK)* que permite a los desarrolladores de aplicaciones móviles añadir ubicación a las mismas.

Estimote comercializa diferentes tipos de beacons según el uso que deseamos efectuar. Disponen de dispositivos para la autenticación de presencia y proximidad, la localización de personas, el trazado de rutas en tiempo real, la personalización digital de productos y/o servicios, la interacción digital y el rastreo de activos y vehículos en interiores y exteriores.

2.3 Tipos de Beacons de Estimote

2.3.1 Dispositivos de Proximidad

Los beacons de proximidad nos permiten autenticar la presencia, enviar notificaciones y mostrar contenidos. Esto permite que los usuarios, cuando tienen el *bluetooth* del móvil activo, puedan recibir información más detallada sobre la ubicación donde se encuentran, y recibir notificaciones relativas a los contenidos dispuestos en ese espacio.

2.3.2 Dispositivos de Localización

Este tipo de dispositivos permite localizar a los usuarios con precisión, en la aplicación se muestra su posición en tiempo real sobre un plano, por ejemplo en un centro comercial es posible, con esta tecnología, visualizar en tu móvil que tiendas y pasillos se encuentran más concurridos. Además pueden indicar una ruta dando instrucciones al usuario para encontrar una dirección o lugar dentro de su rango y recopilar datos de asistencia.

2.3.3 Dispositivos de trazado en tiempo real

Los dispositivos de trazado en tiempo real permiten realizar un seguimiento de sus usuarios a medida que se desplazan de un lugar a otro, además de recoger datos del sensor como son cambios de temperatura y vibraciones.

2.3.4 Dispositivos de Personalización

Estos dispositivos permiten enviar contenido personalizado a las personas que visitan los lugares donde están conectados estos beacons. Estos dispositivos buscan señales de los teléfonos móviles de las personas cercanas a ellos y mostraran contenido personalizado en paneles o pantallas publicitarias a las que se aproximen.

2.3.5 Dispositivos LTE

Los dispositivos LTE (*Long Term Evolution*) son utilizados en objetos que precisan una alta trazabilidad, ya que gracias a este tipo de conectividad es posible enviar la información directamente a la nube. Los sectores que más utilizan estos dispositivos son empresas de alquiler de vehículos, ya que permiten localizar un vehículo en la aplicación y así poder alquilarlo, y las empresas de transporte que ofrecen el servicio de *Tracker* en sus envíos con el cual podemos visualizar en la aplicación del proveedor la situación actual del envío del producto.

2.4 Experiencias aplicadas con tecnología Beacon

En este sentido pasamos a exponer algunas experiencias que ya se están aplicando en sectores como el cultural, o el deportivo, entre otros.

En el sector cultural, museos como el Guggenheim de Nueva York, ha incorporado una de estas tecnologías en la aplicación de su galería, presentando a sus visitantes, cuando se aproximan a sus obras, información relativa a las mismas mediante videos, audios y textos, permitiendo al usuario obtener más información a través de su dispositivo móvil. [1].

En el ámbito deportivo la franquicia estadounidense de la NBA (*National Basketball Association*) *Golden State Warriors*, crearon una aplicación para que sus aficionados puedan seguir las estadísticas del partido, consultar eventos, y recibir notificaciones promocionales, en tiempo real, cuando están presentes en el estadio. [2].

En el sector turístico la tecnología beacons es utilizada para dar información al usuario. El aeropuerto de San Francisco ha logrado que sus clientes puedan consultar en sus dispositivos, en tiempo real, los menús diarios, notificaciones de las comodidades que el aeropuerto proporciona, o dar la opción de realizar el *check in* directamente por redes sociales.

Por otro lado el aeropuerto Hamad de Dubai ha implementado esta solución para facilitar a sus usuarios un desplazamiento más fácil por sus instalaciones. Gracias a la tecnología beacon, que permite localizar a los usuarios en espacios cerrados, ha sido posible en este aeropuerto dirigir a sus usuarios mediante las indicaciones proporcionadas, a las puertas de embarque de este enorme aeropuerto, e incluyendo información de vuelos. [3]

3 Diseño y Desarrollo

En este capítulo se desarrollan las ideas principales, y tratamos los procesos que hemos realizado para la configuración y adaptación a esta tecnología, con la que no habíamos trabajado anteriormente.

Para este proyecto, se han utilizado dispositivos beacon de proximidad, de la compañía Estimote, en el departamento contamos con 4 beacons del tipo de proximidad, que se identifican por el color de la carcasa y por una etiqueta de denominación, en nuestro caso se ha trabajado con 2 de ellos, uno de color azul, con etiqueta *ice*, y otro blanco, etiquetado como *coconut*.

Al principio de este proyecto, se propusieron las líneas básicas del mismo, además de la búsqueda de información para comprensión y motivación del trabajo con esta tecnología, que nunca antes habíamos conocido y/o desarrollado. Esta tecnología es utilizada y se desarrolla en gran mayoría para uso en aplicaciones móviles, fue en sus orígenes, en el año 2016, que la compañía Apple incorporaba esta tecnología a sus dispositivos, llamándola *iBeacon*. Por otro lado, la compañía Google, creó otra tecnología utilizando beacons llamada *Eddystone*, que será la rival directa de la tecnología desarrollada por Apple.

Mientras que *iBeacon*, únicamente es compatible con el sistema operativo *iOS*, que utilizan todos los dispositivos de la marca Apple, *Eddystone* es compatible con todos los dispositivos *Android* e *iOS*, lo que permite que la tecnología de Google sea más robusta, completa y de código libre, lo que significa que cualquier usuario puede utilizar el código y realizar los cambios que necesite, y de ese modo mejorar el desarrollo En nuestro caso utilizamos beacons de Estimote, que son compatibles con ambos sistemas, además la compañía aporta un "kit de desarrollo", *SDK* para facilitar la creación de la aplicación móvil necesaria para la interacción con los beacons.

Por lo tanto, nuestra propuesta fue utilizar la plataforma de Estimote para crear la aplicación de aproximación diseñada para este proyecto, utilizando las plantillas de código que nos ofrece dicha plataforma, utilizando el código básico, y añadiendo nuestro propio código para la consecución de los objetivos.

Los primeros pasos fueron dirigidos a conocer los recursos que nos ofrece Estimote en su nube, para conseguir la configuración y desarrollo de las aplicaciones que interactúan con los beacons. Adquirido el conocimiento necesario, realizamos la creación de la aplicación para la detección de los dispositivos y las correspondientes pruebas de funcionamiento en su detección. Una vez que las pruebas fueron correctas procedemos a modificar el layout para que envíe información personalizada al dispositivo.

Avanzamos en el desarrollo incorporando un nuevo dispositivo beacon al proyecto, realizando los correspondientes ajustes en la configuración y pruebas para comprobar el funcionamiento en la detección de más de un dispositivo beacon y su forma de actuar.

Cuando las pruebas fueron correctas, personalizamos de nuevo el layout para que la información fuera distinta en función del beacon detectado.

El siguiente paso en nuestro proyecto, es la incorporación de las notificaciones automáticas, que se envían a los dispositivos móviles en la detección y salida del rango del beacon. Una vez realizada la configuración y pruebas pertinentes, avanzamos incorporando un segundo beacon a las notificaciones y personalizando cada una de ellas, de tal manera que en cada acceso y salida del rango de cada uno de los beacons el mensaje fuera distinto y personalizado.

Por último se implementó un layout distinto para cada beacon, configurando y realizando las pruebas necesarias, para que en cada detección de un beacon mostrara un layout personalizado en el dispositivo móvil.

Desarrollamos el proyecto en el ámbito de la detección de proximidad debido, a que, consideramos que este campo tendrá un mayor desarrollo para aplicaciones comerciales.

3.1 Requisitos Funcionales

En este apartado indicamos los requisitos funcionales que se deben cumplir para alcanzar los objetivos propuestos.

- RF01: La aplicación debe estar activa o en segundo plano en el dispositivo móvil.
- RF02: La aplicación debe tener configurados los beacons a utilizar con su identificador único.
- RF03: La aplicación debe reconocer por el rango de proximidad a los beacons contenidos en su configuración.
- RF04: La aplicación enviará un mensaje por notificación a la entrada y salida de rango de proximidad.
- RF05: La aplicación enviará un mensaje por notificación a la salida de rango de proximidad
- RF06: Las notificaciones de aproximación deben ser diferentes para cada acción y beacon detectado.
- RF07: La aplicación mostrará un layout una vez tengamos el beacon en nuestro rango de acción.
- RF08: El layout mostrado debe ser distinto y diferenciado por cada beacon detectado.

3.2 Requisitos no funcionales

En este apartado relacionamos los requisitos no funcionales asociados a la aplicación.

- RNF01: La conectividad *bluetooth* debe estar activa en el dispositivo móvil.
- RNF02: La aplicación móvil tendrá compatibilidad con el sistema operativo *Android*.
- RNF03: La aplicación se adaptará a las resoluciones de pantalla disponibles en formato vertical.

3.3 Arquitectura del Software

La aplicación sigue un patrón llamado *Beacon Action Manager*, este patrón se diseñó para determinar el comportamiento de las clases de una aplicación que interactúa con estos dispositivos, ejecutando diversas acciones al entrar o salir del rango de proximidad del beacon. En este patrón se definen los objetos enunciados a continuación.

- Beacons son los dispositivos físicos que se detectan en los métodos de las clases `ContentAdapter`, es decir, todos los beacons que interactúan con la aplicación.
- `ContentManager` o `ManagerController` es la clase que inicia el escaneo de los dispositivos beacons y que a su vez, llama a las clases de tipo `ContentAdapter` o `Action`, que van a manejar los eventos que se produzcan en la aplicación cuando comience a interactuar con los beacons.
- `ContentAdapter` se encarga de conectar los eventos que envía el controlador y se muestra en la vista del dispositivo móvil según el objeto que se está identificando.

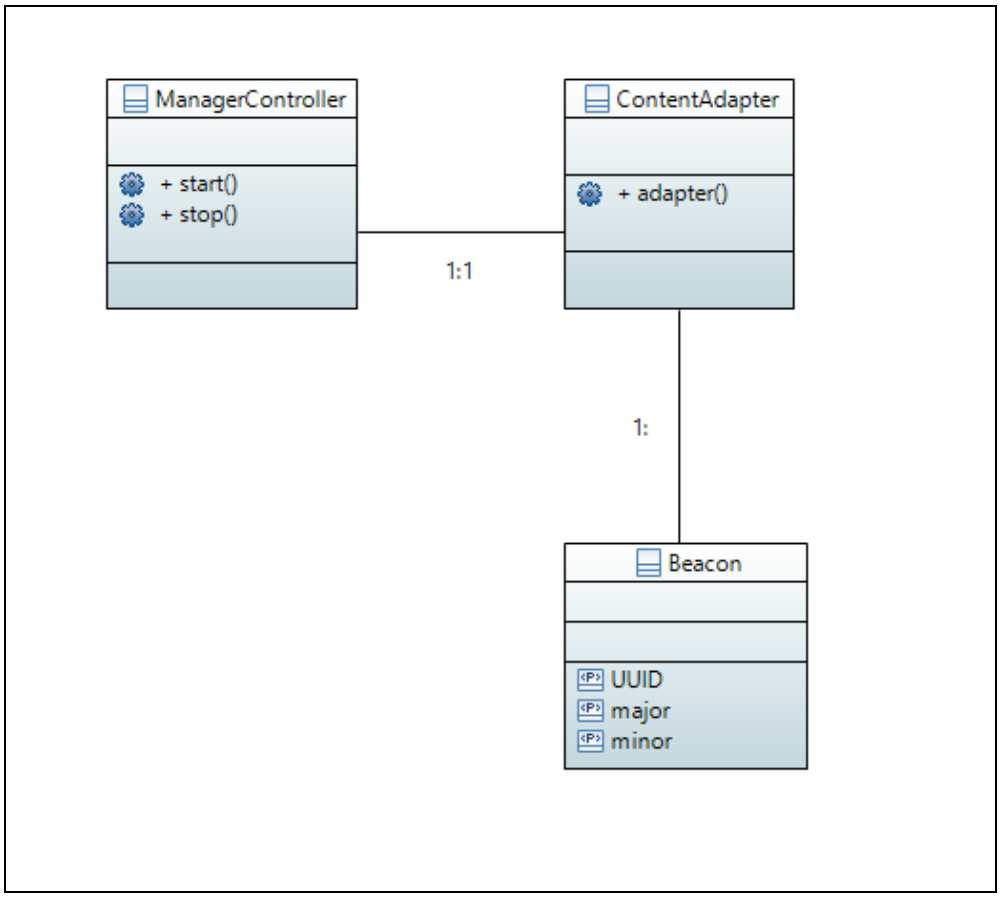

Diagrama 3.1 Patrón Beacon Action Manager.

Este patrón genera una acción dentro de la aplicación cuando un dispositivo móvil se aproxima al rango de un beacon específico, es por ello necesario que cada beacon tenga un identificador único, por el cual la aplicación va a interactuar con ese beacon para realizar una acción diferente en función del dispositivo beacon al que se aproxime. Este patrón también se utiliza para enviar notificaciones al entrar o salir del alcance de un beacon.

Este patrón es necesario y se invoca cuando la aplicación necesita realizar alguna de los siguientes casos:

- Generar una acción dentro de la aplicación cuando un dispositivo móvil se aproxima al rango de un beacon.
- Interactuar con el beacon para realizar una acción diferente en función del dispositivo al que se aproxime.
- Notificar al entrar o salir del alcance de un beacon

3.4 Gestión de conectividad

La aplicación se conecta al beacon mediante la tecnología *bluetooth* de baja intensidad BLE, que permite que el beacon envíe señales constantes a los dispositivos que se encuentren dentro del alcance de su rango operativo.

- La aplicación se conecta por medio de los métodos de la clase `ProximityContentManager`, donde en función de la variable `proximityObserverHandler`, nos permite iniciar una escucha constante mientras la aplicación se encuentre en funcionamiento.

```
class ProximityContentManager(private val context: Context) {

    private var proximityObserverHandler: ProximityObserver.Handler? = null
    private val notificationManager : NotificationManager  = context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
```

Código 3.1 Clase proximityContentManager.

```
fun start() {

    val notificationId = 1
    val proximityObserver = ProximityObserverBuilder(context, (context.applicationContext as MyApplication).cloudCredentials)
            .withTelemetryReportingDisabled()
            .withEstimoteSecureMonitoringDisabled()
            .onError { throwable ->
                Log.e( tag: "app",  msg: "proximity observer error: $throwable")
            }
            .withBalancedPowerMode()
            .build()

    val zone = ProximityZoneBuilder()
            .forTag( tag: "fernando-diez-s-proximity--9eb")
            .inNearRange()
            .onEnter {val title: String = it.attachments["fernando-diez-s-proximity--9eb/title"] ?: "unknown"
                val helloNotification = buildNotification(getproximityTitle(title), getproximityTextEntrada(title))
                notificationManager.notify(notificationId,helloNotification )}
            .onContextChange { contexts ->
                val nearbyContent = ArrayList<ProximityContent>(contexts.size)
                for (context in contexts) {
                    val title: String = context.attachments["fernando-diez-s-proximity--9eb/title"] ?: "unknown"
                    val subtitle = Utils.getShortIdentifier(context.deviceId)
                    nearbyContent.add(ProximityContent(title, subtitle))
                }
                (context as MainActivity).setNearbyContent(nearbyContent)
            }
            .onExit { val title: String = it.attachments["fernando-diez-s-proximity--9eb/title"] ?: "unknown"
                val goodbyeNotification = buildNotification(getproximityTitle(title), getproximityTextSalida(title))
                notificationManager.notify(notificationId, goodbyeNotification) }
            .build()

    proximityObserverHandler = proximityObserver.startObserving(zone)
}
```

Código 3.2 Función start()

En la función *start()* construimos el observador de la aplicación, utilizando las credenciales que nos aporta Estimote en la creación de la app y también en esta misma función configuramos la zona de proximidad, aplicando diferentes acciones en función de la localización del dispositivo móvil.

- .OnEnter: Detecta la entrada del dispositivo que contempla la aplicación en el rango de emisión de señal del beacon.

- .onContextChange: Detecta los cambios que se producen mientras la aplicación permanece encendida, creando una lista con todos los beacons que emiten señales y son detectados por la aplicación. Para añadir el dispositivo beacon a dicha lista es necesario recoger el nombre y el ID, los cuales están definidos en los *attachments* de la aplicación.

- .onExit: Detecta la salida de la aplicación de la zona de proximidad del beacon.

Por otro lado la función `stop()` nos permite detener el escuchador de la aplicación cuando se cierra completamente.

```
fun stop() {
    proximityObserverHandler?.stop()
}
```

Código 3.3 Función stop()

3.5 Gestión de la API

Una vez explicada la funcionalidad de la clase `ProximityContentManager`, entramos en detalle para definir la creación de las vistas que se muestran en la interfaz de la aplicación.

Si accedemos al menú de nuestro proyecto en *Android* Studio, podemos encontrar en la carpeta res/layout los ficheros XML *activity_main.xml* y el *content_view.xml*, que contienen el código de la plantilla que aparecerá en la vista que queremos mostrar en la aplicación.

Una vez abierto cualquiera de estos archivos XML, tendremos dos modos para diseñar un layout, seleccionando la pestaña de gráfico o texto. En el modo gráfico podemos arrastrar y posicionar los elementos que deseamos en nuestra pantalla de una forma visual y en el modo texto codificaríamos con código XML los distintos elementos a mostrar.

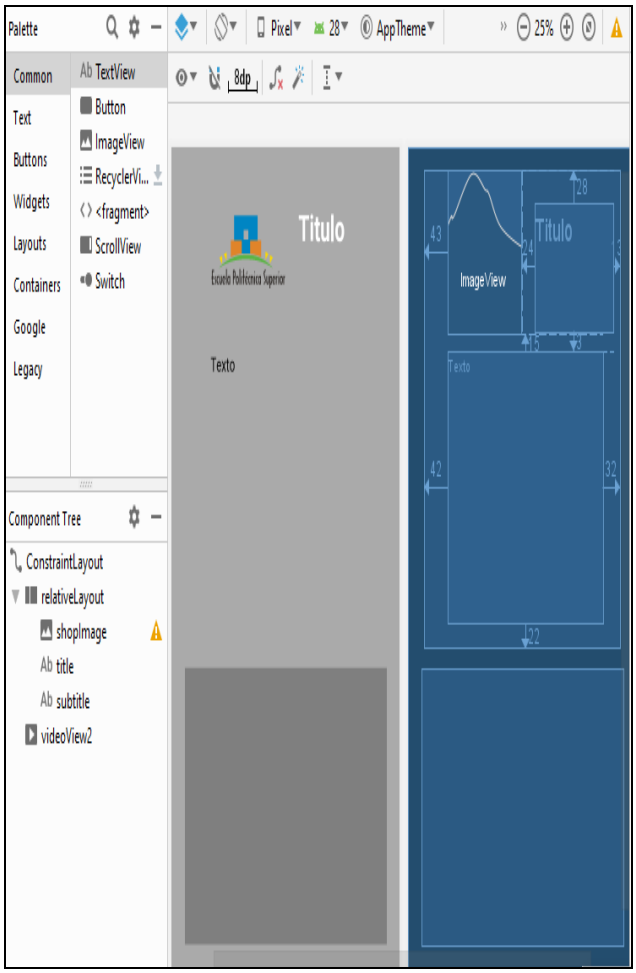

Figura 3.1 Modo diseño.

```
<RelativeLayout
    android:id="@+id/relativeLayout"
    android:layout_width="354dp"
    android:layout_height="418dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <ImageView
        android:id="@+id/shopImage"
        android:layout_width="133dp"
        android:layout_height="143dp"
        android:layout_alignParentStart="true"
        android:layout_marginStart="43dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.029"
        app:srcCompat="@drawable/logoeps" />

    <TextView
        android:id="@+id/title"
        android:layout_width="157dp"
        android:layout_height="112dp"
        android:layout_above="@+id/subtitle"
        android:layout_alignTop="@+id/shopImage"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="24dp"
        android:layout_marginTop="28dp"
        android:layout_marginEnd="13dp"
```

Código 3.4 Layout XML.

Desde la actividad principal `MainActivity` llamamos al layout activity_main para que al abrir la aplicación se muestre esta vista, a continuación se llama al Proximity Manager para ejecutar la función start() que inicia el observador.

Con anterioridad hemos descrito la clase `ProximityContentManager` que es la funcionalidad intermediaria entre los beacons y la clase `ProximityContentAdapter` en la cual se personaliza el contenido de la vista según la señal recibida que identifica nuestra aplicación.

La clase `ProximityContentAdapter` hereda propiedades de la clase `BaseAdapter` por ello se deben sobreescribir varios métodos propios de esa clase con la instrucción *override*.

```
class ProximityContentAdapter(private val context: Context) : BaseAdapter() {

    private var nearbyContent: List<ProximityContent> = ArrayList()


    fun setNearbyContent(nearbyContent: List<ProximityContent>) {
        this.nearbyContent = nearbyContent
    }

    override fun getCount(): Int {
        return nearbyContent.size
    }

    override fun getItem(position: Int): Any {
        return nearbyContent[position]
    }

    override fun getItemId(position: Int): Long {
        return 0
    }
```

Código 3.5 Funciones de ProximityContentAdapter

La función central de esta clase es `getView()`, en la cual inflamos la vista y además se personalizan los objetos que la componen. Para ello tenemos que buscar en la lista `nearbyContent`, en la posición que se pasa por parámetro a la función, el beacon detectado que está guardado en dicha posición de la lista, a continuación se van modificando los diferentes componentes de la vista, en nuestro caso en la vista tenemos textos imágenes y videos, que se modifican según el beacon detectado, es por ello, que debemos recoger el id del objeto utilizando la función `findById<>` de la vista que se pasa por parametro, y una vez encontrado ese identificador nos mostrara los objetos personalizados.

Para personalizar un objeto de video es necesario la dirección del archivo que queremos reproducir, en nuestro caso los videos que hemos utilizado se han guardado en rutas locales, para utilizar URLs de internet se deben habilitar los permisos de usuario hacia internet que permite que la aplicación pueda utilizar videos externos de la red. Despues convertirlo en una URI para que la función de video lo pueda reproducir, además se ha añadido un controlador de video que nos permite manejar las opciones de reproducción (retroceder, pausar, avanzar o reproducir).

Las imágenes se personalizan mediante la función `setImageResource()` de la propia biblioteca de la clase ImageView, enviando por parámetro la imagen que nos devuelve la función `getImage()`, todas las imágenes de la aplicación estan guardadas en la carpeta *drawable* por lo que se puede acceder directamente al objeto.

Las variables de texto de la vista tienen una variable interna llamada *text*, la cual se debe modificar para que en el componente de texto de cada TextView se muestre personalizado.

Para todos estos componentes hemos creado funciones que permiten simplificar la personalizarción, en concreto en el caso de los objetos de video, a la función *getVideo*() le pasamos por parámetro el nombre del beacon detectado, lo cual nos permite completar la dirección URI descrita anteiormente.

```
override fun getView(position: Int, convertView1: View?, parent: ViewGroup): View {
    var convertView : View? = convertView1
    if (convertView == null) {
        val inflater : LayoutInflater = context.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater

        convertView = inflater.inflate(R.layout.content_view, parent, attachToRoot: false)
    }

    val content : ProximityContent = nearbyContent[position]

    val videoView2 : VideoView! = convertView!!.findViewById<VideoView>(R.id.videoView2)
    var dir : String = "android.resource://"+ context.packageName + "/"+ getVideo(content.title)
    val uri : Uri! = Uri.parse(dir)
    val control=MediaController(context)
    if (videoView2!= null){
        Log.d( tag: "tag", msg: "mensaje URI"+ uri)
        videoView2.setVideoURI(uri)
        videoView2.start()
        //Controlador de video
        videoView2.setMediaController(control)
        control.setAnchorView(videoView2)

    }
    //modificamos la imagen según el beacon
    val img : ImageView! =convertView.findViewById<ImageView>(R.id.shopImage)
    img.setImageResource(getImage(content.title))


  val title : TextView! = convertView.findViewById<TextView>(R.id.title)
    val subtitle : TextView! = convertView.findViewById<TextView>(R.id.subtitle)

    title.text = getTitle(content.title)//"Oferta de Abrigo"
    subtitle.text = getSubTitle(content.title)

    convertView.setBackgroundColor(Utils.getEstimoteColor(content.title))
    //convertView = getbeacon(content.title,parent,convertView1)

    return convertView
}
```

Código 3.6 Función getView()

```
fun getTitle(beacon:String): String {
    val lista: List<String> = listOf("EPS", "UAM","Economicas","Derecho","Garaje","Teraza","No se encuentra el beacon")

    return when (beacon) {

    "ice" ->lista.get(0)

    "coconut" -> lista.get(1)

    "candy" -> lista.get(2)

    "mint" -> lista.get(3)

    "blueberry" -> lista.get(4)

    "lemon" -> lista.get(5)

    else -> lista.get(6)

}
```

Código 3.7 Función getTitle()

Las funciones de personalización que se han descrito anteriormente tienen la misma estructura que la función `getTitle()`, modificando los valores de la lista inicial.

3.6 Gestión de notificaciones

Una notificación es el envío de un mensaje al usuario, se muestran generalmente en la parte superior del dispositivo al que se envía y tienen como finalidad informar o alertar de eventos proporcionados por las aplicaciones destinadas para tal fin.

La gestión de las notificaciones se ha implementado en la clase `ProximityContentManager` debido a la facilidad de conexión que ofrece entre el observador y el envío de las notificaciones al realizar una acción. En nuestro caso, las notificaciones se envían a los terminales móviles, cuando entran a la zona de rango del beacon y también a su salida.

Si la aplicación se encuentra en segundo plano y detecta la entrada o la salida de la zona de un dispositivo beacon, la aplicación nos mostrara la notificación correspondiente al beacon detectado. Por ello es importante saber que las notificaciones se envían aunque la aplicación no se encuentre en primer plano.

En el código de la aplicación, se crea una variable de tipo `NotificationManager` para enviar la notificación como se muestra en la imagen de la clase `proximityContentManager` (Código 3.1). A continuación implementamos la función `buildNotification` para construir la notificación, personalizando la notificación mediante las siguientes funciones:

- `.setSmallIcon()`: Añade un icono a la notificación.
- `.setStyle()`: Da estilo a la notificación.

- `.setContenTitle()`: Añade el título a la notificación.
- `.setContenText()`: Añade el texto a la notificación.
- `.setContentIntent()`: Crea el intento para mostrar la notificación.
- `.setPriority()`: Permite decir la prioridad de la notificación.

```
private fun buildNotification(title: String, text: String): Notification {

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        notificationManager.createNotificationChannel(NotificationChannel(
                id: "content_channel", name: "Things near you", NotificationManager.IMPORTANCE_HIGH))
    }

    return NotificationCompat.Builder(context, channelId: "content_channel")
            .setSmallIcon(android.R.drawable.ic_dialog_info)
            .setStyle(NotificationCompat.DecoratedCustomViewStyle())
            .setContentTitle(title)
            .setContentText(text)
            .setContentIntent(PendingIntent.getActivity(context, requestCode: 0,
                    Intent(context, MainActivity::class.java), PendingIntent.FLAG_UPDATE_CURRENT))
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .build()
}
```

Código 3.8 Función buildNotification()

La función *buildNotification* a la que se le pasa un título y un texto es llamada en *.OnEnter* y *.OnExit* de la función *start()*, lo que nos permite inicializar una variable que contendrá un título y un texto personalizado según el beacon que se detecta.

```
.onEnter {val title: String = it.attachments["fernando-diez-s-proximity--9eb/title"] ?: "unknown"
    val helloNotification : Notification = buildNotification(getproximityTitle(title), getproximityTextEntrada(title))
    notificationManager.notify(notificationId,helloNotification )}
```

Código 3.9 OnEnter{} notificación

```
.onExit { val title: String = it.attachments["fernando-diez-s-proximity--9eb/title"] ?: "unknown"
    val goodbyeNotification : Notification = buildNotification(getproximityTitle(title), getproximityTextSalida(title))
    notificationManager.notify(notificationId, goodbyeNotification) }
```

Código 3.10 OnExit{} notificación

Para personalizar estas notificaciones se han creado tres funciones auxiliares, *getproxymityTitle*(beacon), *getproximityTextEntrada*(beacon) y *getproximityTextSalida*(beacon). Estas funciones llaman internamente a otras de la clase *proximityContentAdapter* que permiten diferenciar los dispositivos beacons y devolver el texto específico del beacon correspondiente.

```
fun getproximityTitle(beacon:String):String {
    val proximityContentAdapter = ProximityContentAdapter(context)

    val titulo :String  = proximityContentAdapter.getTitleNotification(beacon)

    return titulo
}
```

Código 3.11 Función getproximityTitle()

```
fun getTitleNotification(beacon:String): String {
    val lista: List<String> = listOf("EPS", "UAM","Economicas","Derecho","Filosofía","Ciencias","No se encuentra el beacon")

    return when (beacon) {

        "ice" ->lista.get(0)

        "coconut" -> lista.get(1)

        "candy" -> lista.get(2)

        "mint" -> lista.get(3)

        "blueberry" -> lista.get(4)

        "lemon" -> lista.get(5)

        else -> lista.get(6)

    }
}
```

Código 3.12 Función getTitleNotification

```
fun getproximityTextEntrada(beacon:String):String {
    var proximityContentAdapter = ProximityContentAdapter(context)

    val texto :String  = proximityContentAdapter.getTextNotificationEntrada(beacon)

    return texto
}
```

Código 3.13 Función getproximityTextEntrada()

```
fun getproximityTextSalida(beacon:String):String {
    var proximityContentAdapter = ProximityContentAdapter(context)

    val texto :String  = proximityContentAdapter.getTextNotificationSalida(beacon)

    return texto
}
```

Código 3.14 Función getproximityTextSalida()

4 Pruebas y resultados

Para familiarizarse con la tecnología, el entorno de programación y la configuración de Estimote, las primeras semanas se realizaron pruebas que sirvieron para la toma de contacto. Al ser una tecnología nueva y poco utilizada por el momento, se realizó una búsqueda de información sobre su funcionamiento y los posibles usos en los que se puede aplicar esta tecnología.

Todas las pruebas han sido realizadas con un dispositivo móvil marca y modelo "OPPO RX17 pro" las características de este dispositivo se describen en el Anexo B.

Comenzamos las pruebas utilizando la aplicación para configurar los beacons de Estimote.

4.1 Prueba de Telemetría.

Los Beacons de proximidad que disponemos para el proyecto tienen la opción de medir la temperatura, por lo que una de las primeras pruebas realizadas fue la de medir la temperatura en distintas ubicaciones, interiores y exteriores.

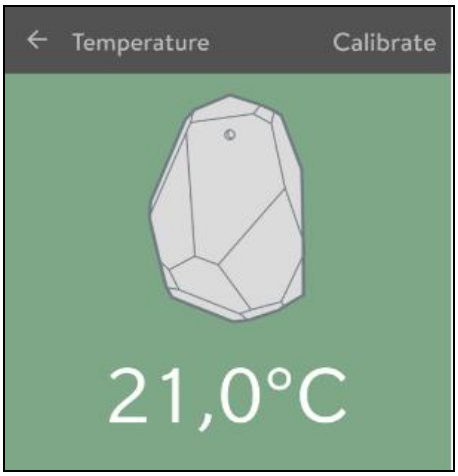

Figura 4.1 Temperatura interior.

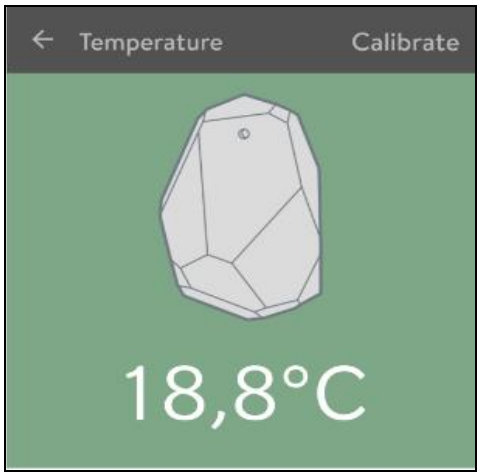

Figura 4.2 Temperatura exterior.

Se puede apreciar por ambas figuras el cambio de temperatura obtenido al medir en interiores y exteriores resultando una variación en ese momento de 2ºC aproximadamente, hay que remarcar que la telemetría de los beacons no es del todo precisa, ya que están cubiertos de silicona, material que aísla el sensor, y por ello la detección y variación de temperatura se produce lentamente.

4.2 Prueba de Movimiento y Alcance.

Otra de las funciones que tienen estos dispositivos es la de movimiento, por la cual nos permite saber, siempre que estemos dentro del rango de alcance, si el beacon se encuentra en movimiento. Al mover el beacon la señal será detectada por la aplicación de configuración, lo cual provoca que se inicie la vibración del dispositivo móvil que tiene activa la aplicación.

Por otro lado se han realizado pruebas de alcance de rango, para determinar el radio de alcance de la señal del beacon.

4.3 Prueba de Detección de un único beacon y cambio de objetos en los layouts.

Con nuestra modificación de la plantilla de Estimote conseguimos crear nuestra aplicación para probar, que recoge las señales de un único beacon, y con ello poder probar los distintos elementos en el layout del beacon. Al principio una de las pruebas fue modificar ese layout para mostrar una imagen y un texto, los siguientes pasos fueron añadir un elemento de video y probar que más elementos se podrían añadir a este layout.

Los objetos probados han sido:

- Textos.
- Videos.
- Imágenes.
- WebView- Este elemento finalmente no se añadió a los layouts debido a la complejidad de uso para el usuario, ya que no se acopla a un espacio determinado y su manipulación por éste no resulta sencilla.

Figura 4.3 Layout con WebView.

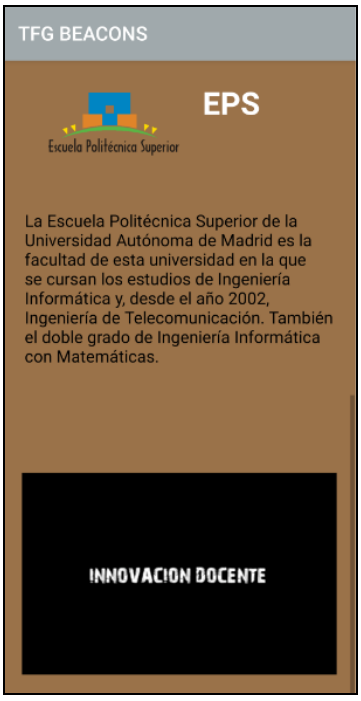

Figura 4.4 Layout EPS, único beacon.

Con esta prueba conseguimos utilizar y moldear cada layout con diferentes objetos.

4.4 Prueba de detección de múltiples beacons.

Después de conseguir detectar un único beacon, para lo cual utilizamos la plantilla de single beacon proporcionada por Estimote, continuamos probando la plantilla multibeacon para las pruebas con más de un dispositivo beacon. Una vez descargada y abierta al examinar el código que nos aporta Estimote vemos que las aplicaciones de proximidad de beacons individuales y multibeacon tienen el mismo código inicial, y que en este código no se detalla el número de beacons a utilizar.

Al realizar esta prueba entrando en la nube de Estimote, comprobamos que en la configuración de los beacons, concretamente en los *Tags* de cada uno de ellos, nos muestra las aplicaciones que se conectan al beacon, por lo que si queremos conectar un nuevo beacon a una aplicación, tan solo es necesario añadir el identificador de la aplicación en los Tags de los beacons.

Con respecto a la detección de varios beacons los Escenarios realizados han sido las siguientes:

- Escenario 1: Comenzar dentro del rango de uno de los beacons, la aplicación lo detecta dentro del rango y muestra la información definida en los layouts, a continuación al salir del rango del beacon y no detectar ningún otro nos dejaría de mostrar el layout anterior, por último nos aproximamos a un nuevo beacon, y en cuanto la aplicación detecta la señal del nuevo beacon nos muestra la información definida para este beacon en la aplicación.

- Escenario 2: Otra de las pruebas realizadas fue comenzar fuera del rango de los beacons, aproximándonos al primero de ellos, para a continuación salirnos del rango y volver al alcance de ese mismo beacon, esta prueba nos servirá para estudiar el tiempo de reacción de la señal y tiempo de reacción entre la aplicación y los beacons. Podemos afirmar que en la mayoría de las veces la recepción de la señal cuando el dispositivo se encuentra dentro del rango de un beacon es rápida, exceptuando aquellas ocasiones en las que la conectividad con los beacons no es detectada al instante, como puede ser al inicio de la aplicación que nos hace esperar unos segundos hasta que la señal es recibida y la aplicación reacciona.

- Escenario 3: Una de las pruebas más interesantes de detección de varios beacons, es dar la posibilidad que el usuario se encuentre entre dos o más dispositivos, por lo que la aplicación detectará los distintos beacons determinando que se encuentra dentro de todos los rangos, mostrando todos los layouts definidos en cada de uno de ellos. En la aplicación debemos hacer *scroll* para ver las plantillas definidas de cada uno de ellos.

4.5 Personalización de los objetos según el beacon.

El siguiente paso a objeto de prueba fue que cada presentación del beacon fuera diferente. La primera personalización que se realiza, es modificar la imagen y los textos que componen los layouts, comprobamos que el fondo de éstos se puede personalizar en cada beacon, por tanto a partir del esqueleto de esa función, se crean diferentes funciones que nos permiten personalizar las imágenes y textos según la detección de los diferentes dispositivos.

A continuación realizamos la personalización de los videos en función del beacon detectado, los archivos de vídeo deben almacenarse en una carpeta creada a tal efecto en el proyecto, en nuestro caso la nombramos como RAW, a continuación creamos una nueva función en la aplicación, para que cargue el video en función del identificador del beacon que detecta.

Esto permite personalizar nuestros layouts de una forma rápida y sencilla, ya que lo único que debemos hacer es modificar el listado de cada función para cambiar los textos, videos o imágenes a mostrar.

El resultado de este proceso es el que mostramos en las siguientes figuras que presenta dos entonos diferentes en función del beacon detectado.

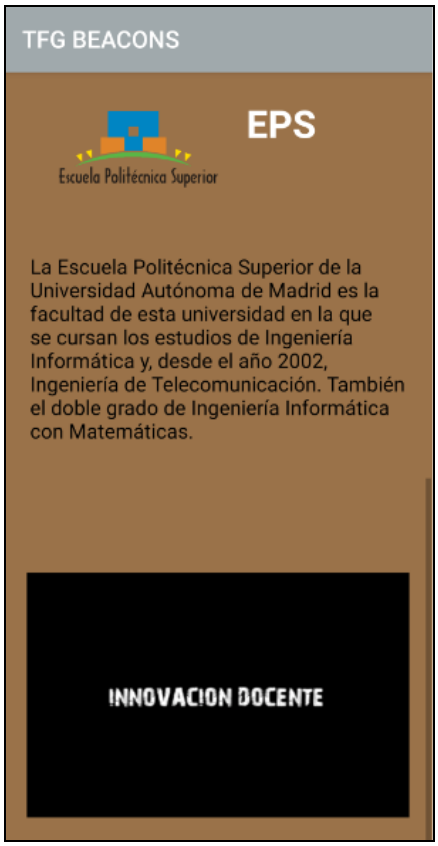

Figura 4.5 Layout EPS MultiBeacon

Figura 4.6 Layout UAM MultiBeacon.

4.6 Notificaciones de entrada y salida.

Una de las primeras pruebas realizadas en el ámbito de las notificaciones fue el envío de notificación estándar al detectar cualquier beacon, es decir, enviamos la misma notificación al encontrar distintos beacons.

Después de comprobar que al entrar en la zona de cualquier dispositivo con la aplicación encendida en primer plano se recibía la notificación, el siguiente paso fue comprobar que cuando se detecte que el usuario ha salido del rango de cualquier beacon también se reciba la notificación.

Además se realizaron pruebas para diferenciar la entrada y salida del rango de cada beacon, personalizando mediante textos diferentes las notificaciones de entrada de las de salida de los beacons.

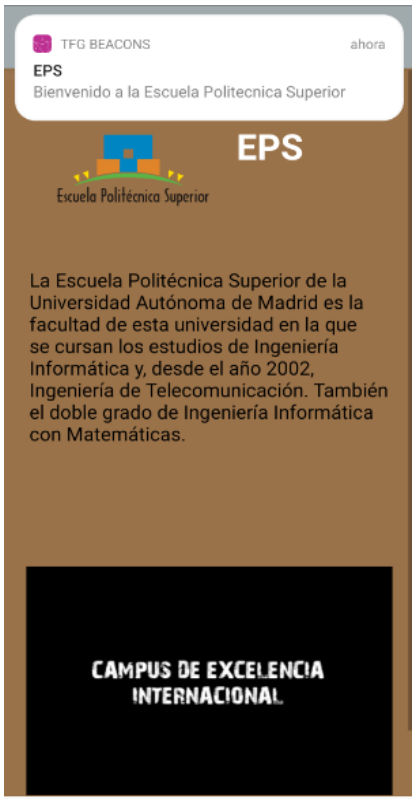

Figura 4.7 Notificación en la app layout EPS

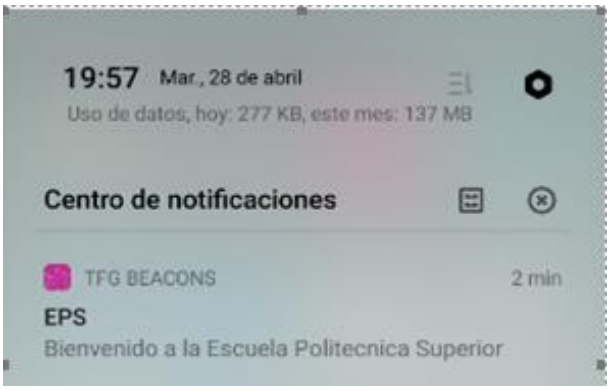

Figura 4.8 Notificación EPS en el menú

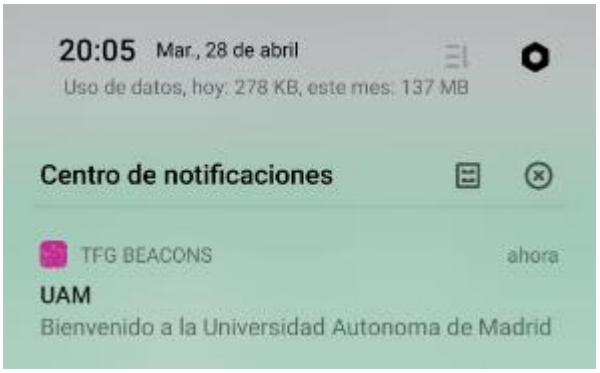

Figura 4.9 Notificación UAM en el menú.

En nuestras pruebas detectamos que cuando estamos dentro del rango de dos o más beacons solo envía el mensaje del primer beacon que detecta.

Una vez comprobada y verificada la personalización de las notificaciones, hemos realizado la prueba dejando la aplicación en segundo plano.

Al entrar en los rangos de los beacons con el dispositivo móvil, la aplicación nos envía el mensaje y nos notifica con el texto definido para la entrada en la zona del beacon, lo que hace muy interesante el uso de esta tecnología en diferentes sectores como pueden ser el marketing o turismo, ya que con esta aplicación en segundo plano, un usuario recibirá notificaciones cuando se aproxime a un beacon, con lo que podemos hacer entrega de un avance de la información disponible en ese lugar o espacio determinado.

.

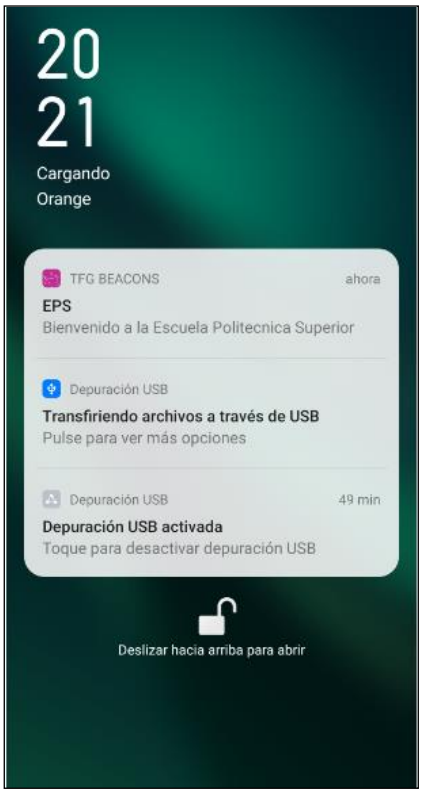

Figura 4.10 Notificación con el movil bloqueado.

4.7 Diferencia de layouts según el beacon.

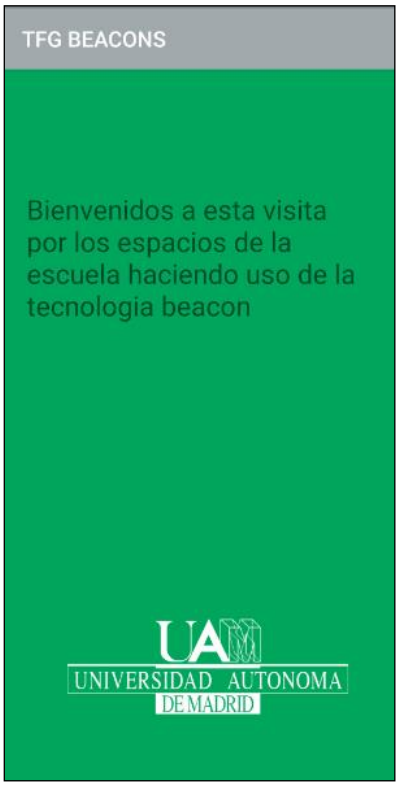

Figura 4.11 Layout de Bienvenida

Una nueva prueba de diferenciación ha sido la creación de diferentes layouts, dentro del mismo proyecto de aplicación móvil. Es decir, que la composición de los layouts sea totalmente diferente por cada beacon, por lo tanto podremos definir en uno de los beacons solamente una imagen y un texto y en otros la composición que teníamos en un principio, el logo con un título, un texto y un video.

Para la realización de estas pruebas se han creado los diferentes escenarios que describen a continuación, y que muestran la situación tratada y el resultado obtenido.

Escenario 1- El dispositivo entra en la zona de rango del beacon azul

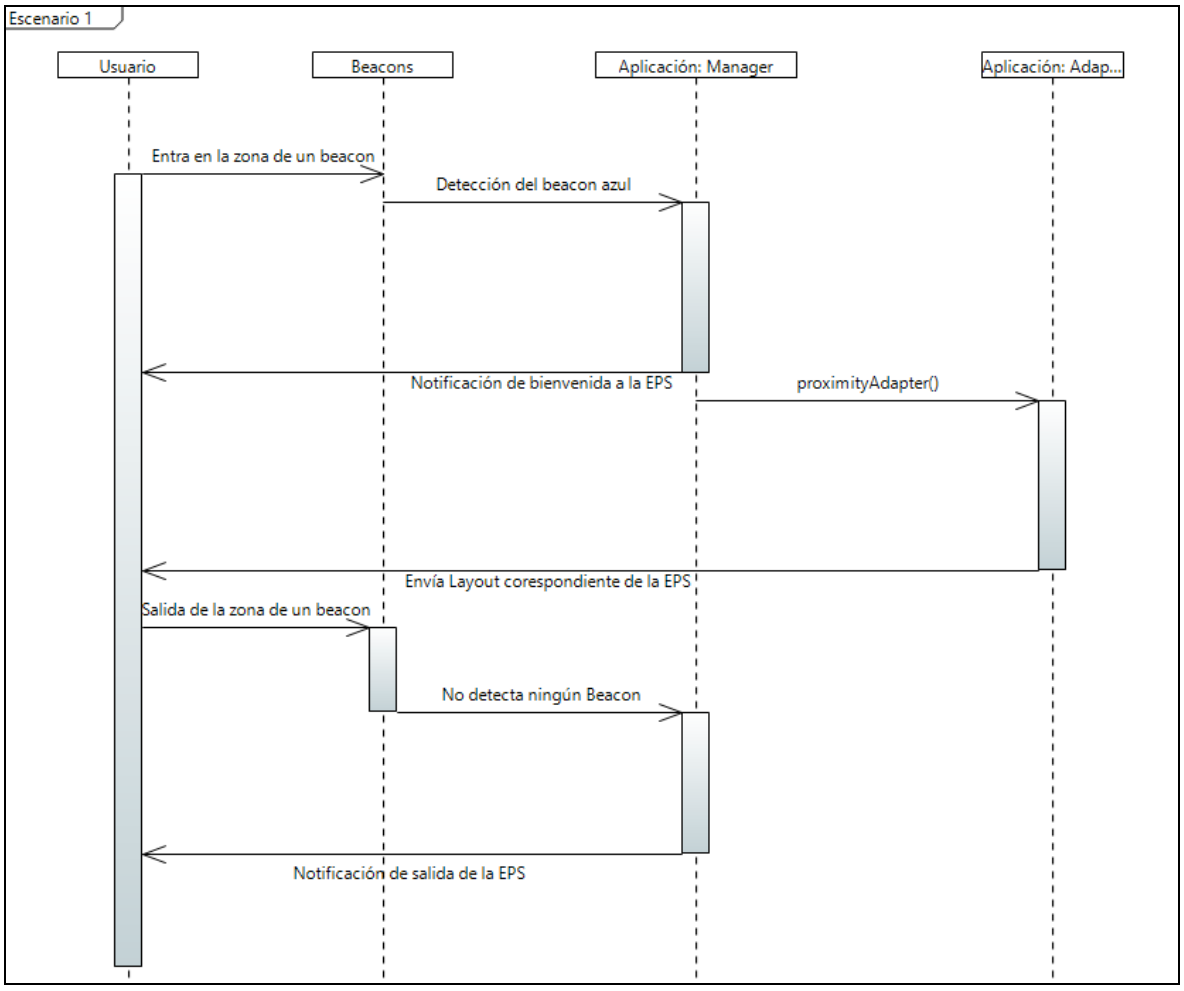

Diagrama 4.1 Escenario 1

En el diagrama se describe la situación de un usuario que accede a la Escuela Politecnica Superior, EPS, y que entra en el rango de alcance de uno de los beacons.

La aplicación lo detecta y recibe el identificador del beacon, enviando al usuario una notificación de bienvenida a la escuela, secuencialmente se envía la información del beacon detectado a la clase `ProximiyAdapter`, lo que permite personalizar el layout y enviar al usuario el correspondiente a la Escuela, monstrandolo en la aplicación del usuario.

Cuando el usuario sale del rango de señal del beacon, la clase Manager no detecta ningún beacon y envía la notificación de salida de la escuela.

Escenario 2- El dispositivo entra en la zona de rango del beacon blanco

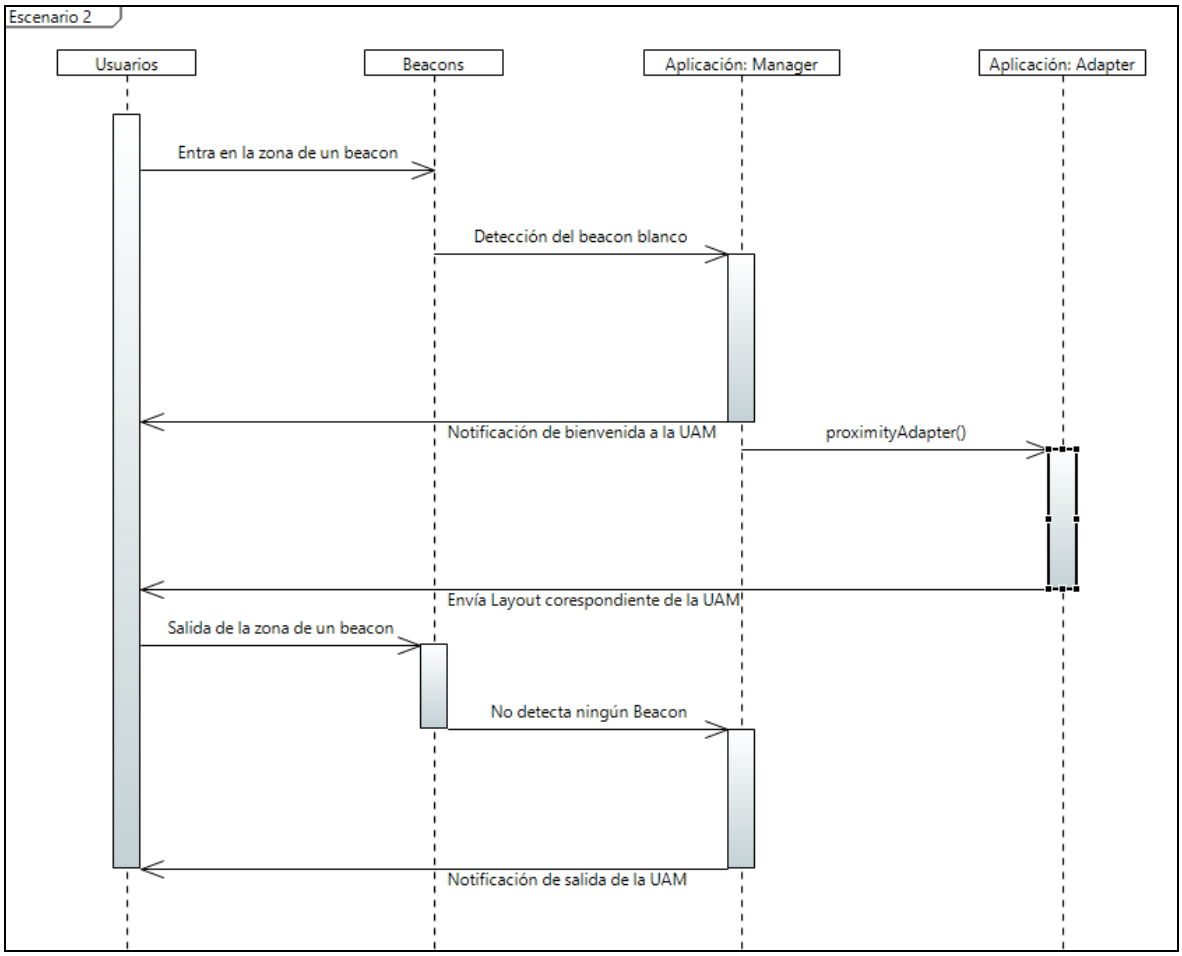

Diagrama 4.2 Escenario 2

Este segundo escenario describe la interacción del usuario con el beacon blanco.

El usuario realiza los mismos pasos que hemos descrito en el escenario 1, teniendo en cuenta, que el identificador que envía el beacon a la aplicación es el de color blanco, realizando para este caso el envío de notificaciones de la Universidad Autonoma de Madrid, UAM , y su correspondiente layout.

Escenario 3-Entra en la zona del beacon azul y a continuación también detecta la señal del beacon blanco, permaneciendo en el rango de emisión del beacon azul.

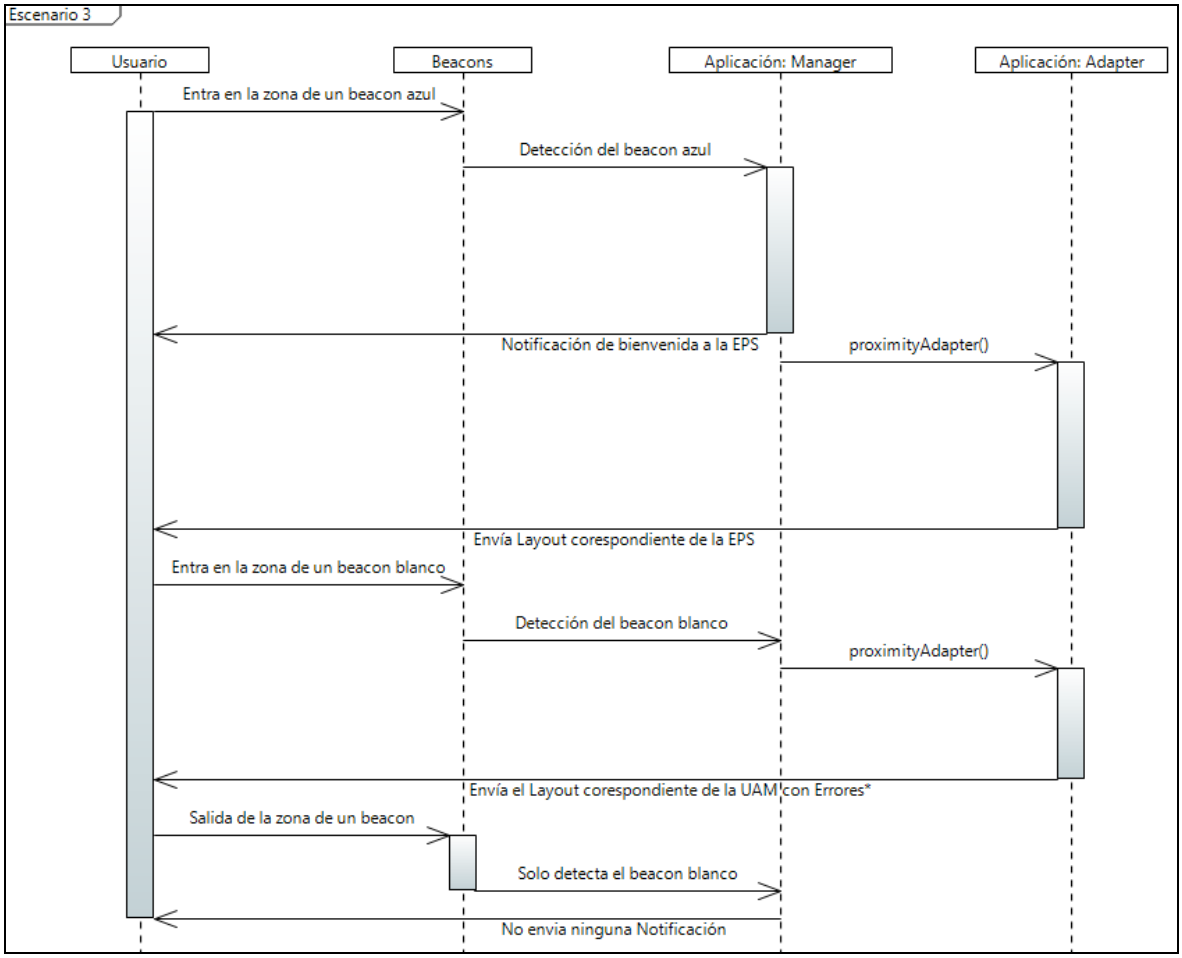

Diagrama 4.3 Escenario 3

El diagrama describe el proceso por el cual el usuario detecta diferentes beacons en un orden específico. Primero es detectado el beacon azul realizando el mismo proceso del escenario 1 y, a continuación, sin salir del rango de emision del beacon azul, detecta otro nuevo beacon, en nuestro caso el blanco. Al detectar el beacon blanco no envía la notificación corespondiente ya que aún nos encontramos dentro del rango del primero, por lo que solo nos muestra el layout. Este layout muestra el fondo del beacon correspondiente a la UAM pero los objetos de este son los correspondientes al beacon de la EPS.

El error se debe a que la aplicación recoge los dos identificadores, pero prevalece el primer identificador con respecto al segundo, en este caso correspondiente al beacon azul.

Escenario 4- Entra en el rango de dos beacons. Primero detecta el blanco y sin salir de su zona detecta un segundo beacon, el azul, permaneciendo en el rango de emisión del beacon blanco.

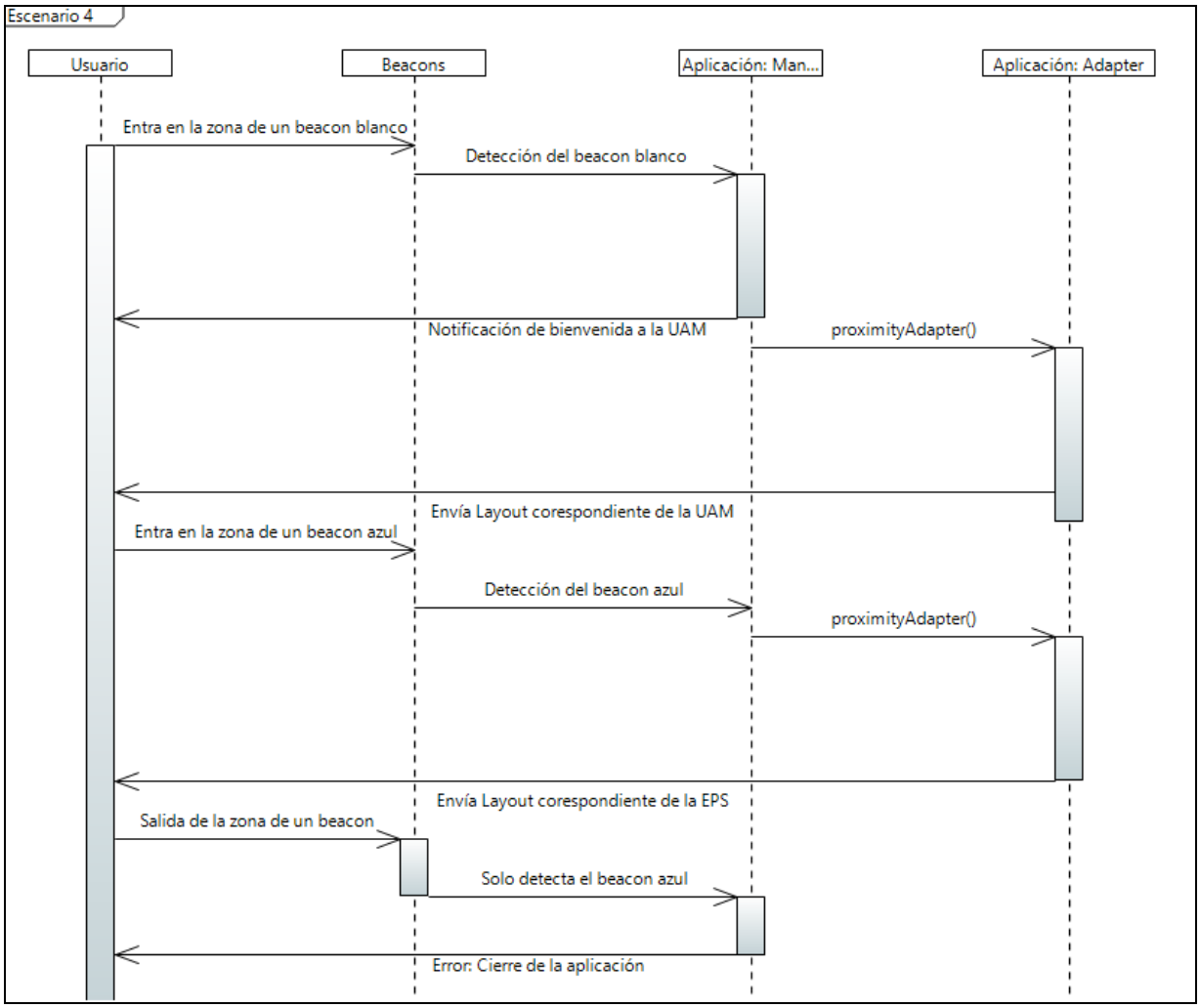

Diagrama 4.4 Escenario 4

En el escenario 4 se describe el proceso contrario al escenario 3 en este caso primero se detecta el beacon blanco y se envía correctamente las notificaciones y layout correspondiente. A continuación es detectado el beacon azul y se muestran los dos layouts corectamente en la aplicación del usuario.

Una vez que sales del rango de emisión del beacon blanco solo es detectado el azul y la aplicación crea un error y se cierra automáticamente. Esto se debe a que el contenido del layout blanco no contiene los mismos objetos que el otro layout intentando cargar en el layout objetos que no estan disponibles.

5 Conclusiones y trabajo futuro

En este capítulo se exponen las conclusiones basadas en el trabajo realizado y las experiencias recogidas a lo largo del desarrollo del proyecto, incorporando además propuestas de mejora que ayuden a la realización de trabajos futuros.

5.1 Conclusiones

La tecnología beacon se basa en la conectividad *bluetooth* de baja energía, lo cual nos permite la localización en lugares de interior más precisa que otras tecnologías, así como un menor consumo, por lo que consideramos que su estudio o exploración es importante para afrontar nuevos proyectos centrados en la geolocalización , proximidad, y notificación en espacios cerrados.

La idea principal del proyecto era conocer y trabajar sobre esta nueva tecnología.

El proyecto se inicia sin conocimientos previos de uso de esta tecnología, y a base de investigación y realizando pruebas funcionales, hemos conseguido conocer más el producto y sus funcionalidades y de este modo poder poner en práctica el conocimiento adquirido para futuros proyectos.

Con este trabajo hemos logrado conocer los dispositivos y su funcionamiento, configurar y adaptarlos a nuestros propósitos, conseguir la interacción entre los beacons y los dispositivos móviles, y su personalización en la actuación de las acciones que deben realizarse cuando entren en contacto con el rango de cada uno de los beacons utilizados.

También se han realizado diferentes escenarios de pruebas y se ha conseguido configurar diferentes layouts para cada beacon detectado.

Por el estudio realizado con estos dispositivos, hemos tomado conciencia del amplio mercado que pueden albergar en aplicaciones tales como:

- Envío de ofertas comerciales o anuncios.
- Información sobre productos al alcance del usuario.
- Puntos de referencia y localización
- Navegación en interiores cuando la conexión vía GPS no sea factible.
- Creación de analíticas para su estudio, como obtener información de cuáles son los lugares más concurridos o visitados, aplicables a centros comerciales o ciudades.
- Integración de la información obtenida por estos dispositivos en redes sociales
- Envío de mensajes de audio, videos o imágenes.

- Mejoras en la experiencia del cliente, para hacer más personal la relación entre el cliente y la empresa.

5.2 Trabajo futuro

Dado que esta tecnología es novedosa y poco utilizada por el momento, el objetivo a futuro es que los avances y conocimientos logrados en este Trabajo de Fin de Grado sirvan de apoyo y asesoramiento para avanzar en el conocimiento de estos dispositivos y sus posteriores aplicaciones.

5.2.1 Caso de uso en la Escuela Politécnica Superior

En la actualidad se dispone de cuatro beacons. Nuestra propuesta de uso consistiría en la creación de una aplicación que permita conocer la ubicación de los distintos espacios comunes de la Escuela y que incluya un guiado que permita a un usuario desplazarse por la Escuela mediante indicaciones de localización enviadas por los beacons utilizando la información posicional del lugar en el que se encuentra.

Otra utilidad que proponemos es la utilización de los beacons para enviar notificaciones en todo el espacio de la Escuela, tales como eventos, información sobre los laboratorios disponibles para su uso, información detallada de los elementos que componen el museo, el menú de la cafetería o incluso enviar avisos generales a los usuarios presentes en ese momento en la Escuela.

Como ejemplo de desarrollo futuro, consideremos a un usuario entrando por el hall de la Escuela, uno de los beacons le envía una notificación de bienvenida, además presentamos un layout con opciones mostrando los lugares más utilizados de la Escuela, como pueden ser la cafetería, biblioteca o laboratorios, y pulsando una de las opciones indicar al usuario el recorrido para llegar al destino seleccionado.

Otro trabajo futuro para ampliar nuevos campos sería la de incorporar una base de datos a la aplicación, para lo cual implementaremos el módulo de análisis de Estimote que permite almacenar y tratar los datos recibidos por los beacons.

Esta posibilidad permitiría el control de aforo a nuestro centro, contabilizando el número de usuarios que visitan la escuela, o la concurrencia de usuarios en los laboratorios.

5.2.2 Control de presencia

Cada vez es más necesario el fichaje o control de presencia en empresas e instituciones. Actualmente métodos como el uso biométrico por huella digital, se pueden encontrar en desuso motivado a los cambios que en medida de sanidad por transmisión de virus se puedan poner en vigor, en este contexto nuestra propuesta de control de presencia y fichaje, sería la del diseño e implementación de una aplicación utilizando la tecnología de los beacons, que nos proporciona seguridad al no necesitar contacto físico, por otro lado al estar necesariamente próximo en alcance, evitaría los falsos fichajes y el absentismo laboral, realizando automáticamente el registro de la jornada laboral de los trabajadores.

Para desarrollar este proyecto será necesario la creación de una base de datos conectada a los beacons y la utilización de estos dispositivos creando interface que añada registros en la base de datos por cada iteración de presencia con el beacon.

5.2.3 Aplicación para IOS

Otro de los trabajos futuros utilizando esta tecnología novedosa, que hemos conocido y tenido la oportunidad de trabajar con ella, es la creación de una aplicación para dispositivos móviles con sistema operativo *iOS*. Para ello se podría crear una aplicación únicamente utilizando la tecnología *iBeacon* o utilizar *Eddystone* de acuerdo con el blog "*usingbeacons*" [5] o desde la página de desarrolladores de google [6] en la que indican que es compatible con *iOS* y *Android*.

Por otro lado también de acuerdo con el blog "*beaconzone*" [7] que los dispositivos móviles de Apple *iOS*, sólo son compatibles con beacons con tecnología *iBeacon*, por ello creemos interesante la realización del estudio para aplicaciones *iOS* para testar ambas teorías.

5.2.4 Experimentar con diferentes aplicaciones de Estimote

Otro de los trabajos a realizar sería la experimentación con las diferentes plantillas de aplicación que nos ofrece la plataforma de Estimote, en ella existen opciones de aplicación como la interacción de un dispositivo móvil con los beacons mediante la tecnología NFC, o la intercomunicación por *Wireless* con otros dispositivos, crear tu propia aplicación de configuración para tus beacons, o la creación de analíticas asociadas al uso del beacon.

Referencias

[1] Guggenheim App Adds Feature to Highlight Artworks Near Users, Solomon R Guggenheim Museum.(2015), https://www.guggenheim.org/news/guggenheim-app-adds-feature-to-highlight-artworks-near-users (último acceso 03/06/2020)

[2] Rebeca Borison, Golden State Warriors enhance game day with beacon technology, https://www.retaildive.com/ex/mobilecommercedaily/golden-state-warriors-enhance-game-day-with-beacon-technology (último acceso 03/06/2020)

[3] Beacons para aeropuertos , enriqueciendo la experiencia de viaje, motti.com, 17 Mayo 2018, https://motti.com/es/beacons-para-aeropuertos-enriqueciendo-la-experiencia-de-viaje/ (último acceso 03/06/2020)

[4] El mercado de los beacons se doblará este año, Redacción Channel Partner, 20/05/2016, https://www.redestelecom.es/conectividad/noticias/1089626051003/el-mercado-de-los-beacons-se-doblara-este-ano.1.html (último acceso 03/06/2020)

[5] Foro de Estimote, https://forums.estimote.com/ (último acceso 03/06/2020)

[6] Usingbeacons, http://www.usingbeacons.com/principales-caracteristicas-de-google-eddystone/ (último acceso 03/06/2020)

[7] Página de desarrolladores de Google, https://developers.google.com/beacons/eddystone (último acceso 03/06/2020)

[8] Beaconzone, https://www.beaconzone.co.uk/blog/beacon-compatibility-with-ios-and-android/ (último acceso 03/06/2020)

[9] Estimote, https://estimote.com/ (último acceso 03/06/2020)

[10] Características del OPPO RX17 PRO, https://www.xataka.com/analisis/oppo-rx17-pro-analisis-caracteristicas-precio-especificaciones (último acceso 03/06/2020)

Glosario

API	Application Programming Interface.
Android	Sistema operativo para dispositivos móviles.
Android Studio	Entorno de desarrollo para aplicaciones *Android.*
SDK	Kit de desarrollo de software.
Beacon	Dispositivo transmisor que se utiliza para emitir una señal bluetooth de baja intensidad.
LTE	Long Term Evolution, tecnología para comunicación inalambrica para la transmission de datos de alta velocidad.
Kotlin	Lenguaje de programación usado en el desarrollo de aplicaciones *Android*.
Observer	Interfaz implementable cuando se quiere que una clase envie notificaciones sobre distintos cambios.
Adapter	Interfaz implementable cuando se utiliza un layout de tipo Grind Layout.
Aplicación Móvil	Programas diseñados para ser ejecutados en telefono, tablets y otros dispositivos móviles.

Anexos

A Configuración

A.1 Configuración de los Beacons

Una vez que hemos recibido los dispositivos la configuración de los beacons se puede realizar de dos maneras:

- Mediante el acceso que nos proporciona Estimote al hacer login en su web.
- Y otra a través de la aplicación para *smartphones* (app) que dispone Estimote (*iOS* app store y *Android* play store)

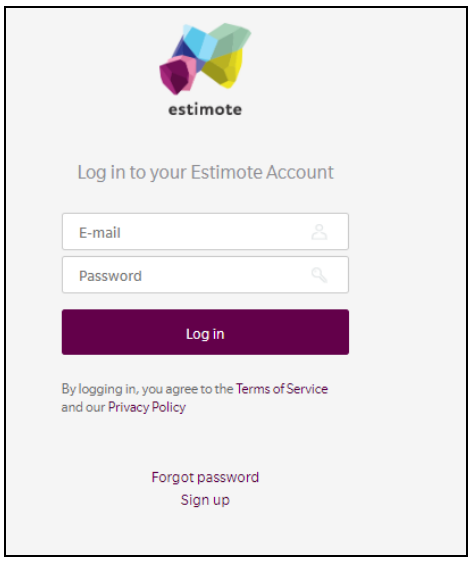

Figura A.1 Login web

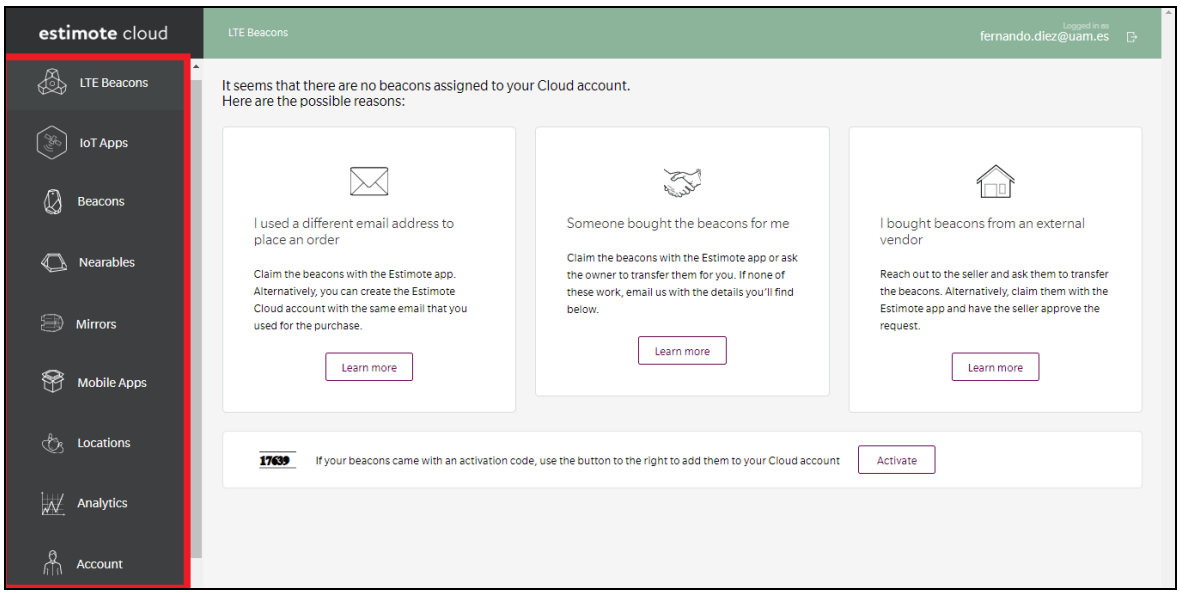

Figura A.2 panel de Estimote cloud

Una vez nos logueamos la web nos presenta una página dividida en tres zonas.

En la parte izquierda nos encontramos el menú, marcado en rojo en la figura anterior, donde nos muestra todos los tipos de beacons que nos ofrece la página de Estimote además de aplicaciones para el móvil, analíticas e información de tu perfil.

En la parte superior nos muestra la cabecera con el usuario logueado y la página de la sección en la que nos encontramos.

Y por último la sección central o principal de la página que irá mostrando la información relativa a las distintas opciones del menú que estemos seleccionando.

Si pulsamos en el menú en el icono de nuestros dispositivos, en nuestro caso pulsamos en Beacons, en este panel central aparecen todos los beacons de proximidad donde podemos configurar cada beacon. Cada beacon tiene un nombre, y cuatro campos de referencia:

- Identificador- Código para identificar el beacon en las diversas aplicaciones
- *Tags*- Estas etiquetas corresponden a las aplicaciones que reciben las señales del beacon correspondiente.
- Paquetes- Te muestra los paquetes que están activos para ese beacons.
- Localización- Si esta activada la localización del beacon nos mostrara el rango que en el que puede enviar las ondas *bluetooth*.

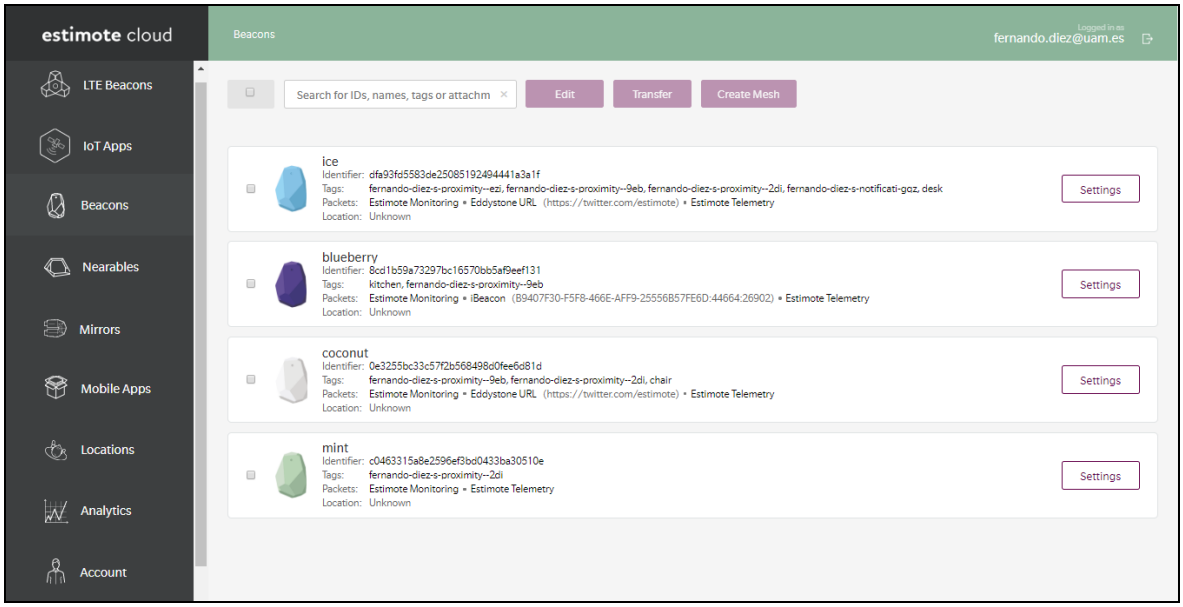

Figura A.3 Pantalla de Beacons.

Para configurar el dispositivo debemos entrar en Settings del beacon correspondiente y nos mostrara con mas detalle la configuración y los distintos campos que el beacon nos aporta que podemos editar.

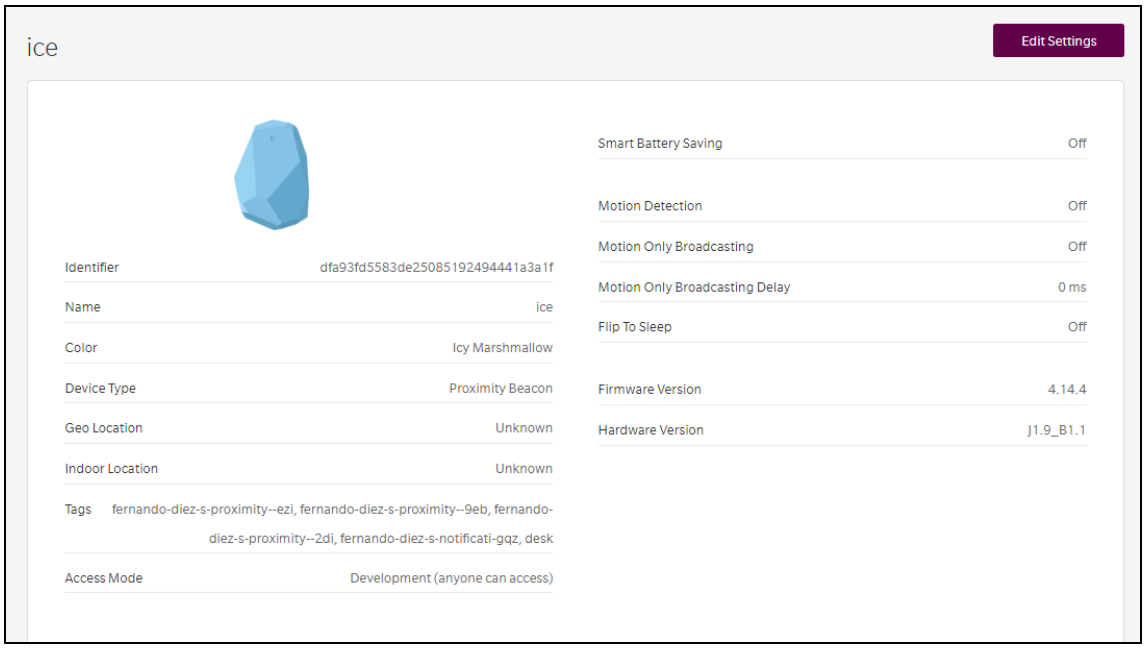

Figura A.4 Principal de la configuración del beacon.

En la imagen anterior nos muestra los diferentes campos asociados a ese dispositivo, como los descritos anteriormente, identificador, *Tags*, nombre del Beacon y además de esos campos nos muestra otros como pueden ser el tipo de dispositivo, en este caso concreto es un beacon de proximidad, la versión de *firewall* y de *hardware*, telemetría y otros.

Una vez configurado el dispositivo vía web o aplicación para móvil, debemos proceder a conectar los dispositivos con un *smartphone* que obligatoriamente debe contener y estar logueado a la aplicación de Estimote para móvil, de ese modo el dispositivo actualizará su configuración y quedará listo para su uso.
Para proceder a la conexión entre beacon y *smartphone* tenemos dos posibilidades.

Acceso desde la aplicación de móvil, y desde el menú de configuración seleccionar el beacon que aparecerá en el radar de dispositivos al alcance, O bien desde acceso por conectividad NFC, activando dicha funcionalidad del *smartphone* y una vez activo acercándolo al beacon que necesitamos configurar o actualizar, ya que este modo nos asocia directamente y aparece en la aplicación la pantalla de configuración del dispositivo seleccionado.

En el caso de proceder a la configuración como se ha descrito anteriormente a través de la aplicación móvil, los pasos son los siguientes:

1. Descargar la aplicación de Estimote

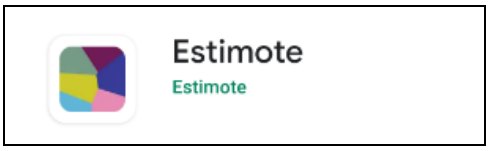

Figura A.5 Icono aplicación Estimote

2. Iniciar sesion en la aplicación.

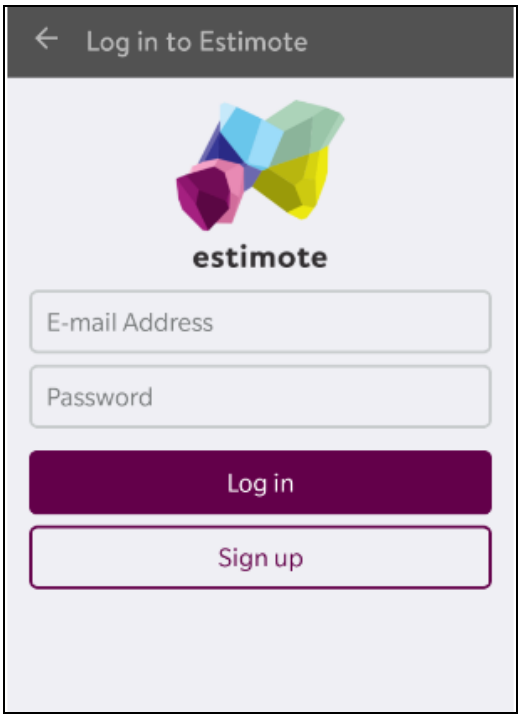

Figura A.6 Login de la aplicación móvil.

3. En el menú principal seleccionamos configuración.

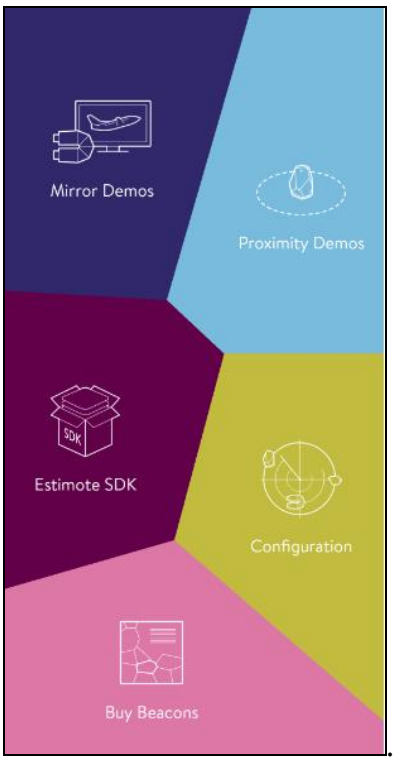

Figura A.7 Menú principal

4. En el menu de configuración nos permite detectar los beacons mediante tres formas, un radar una lista de beacons o directamente como hemos descrito utilizando tecnología NFC.

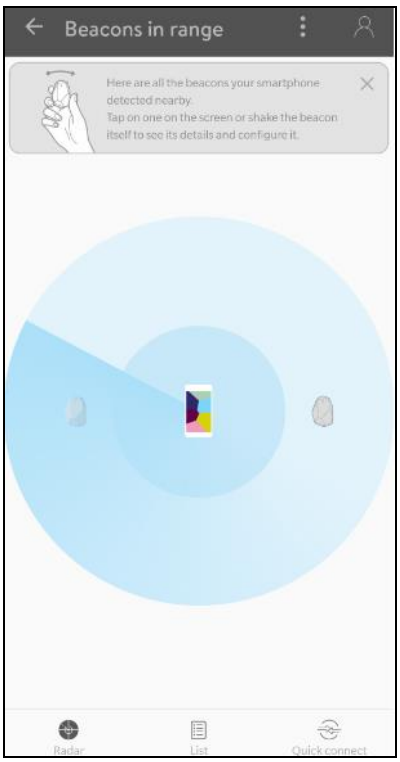

Figura A.8 Menú principal.

En esta imagen se muestra la pantalla de configuración en modo radar, en ella nos muestra los Beacons que estan cerca de nuestro dispositivo movil, para acceder a la configuración de cualquiera de los beacons solo es necesario pulsar encima del beacon que queramos configurar y entrariamos en la siguiente pantalla.

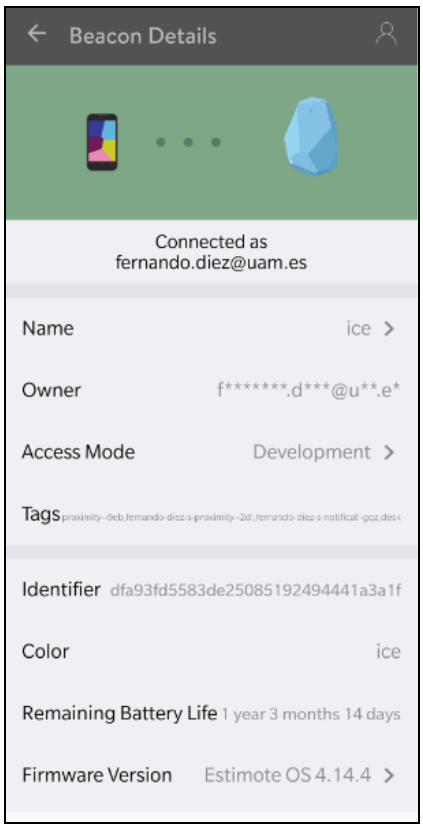

Figura A.9 Configuración de la aplicación.

A.2 Configuración de aplicación Android

Después de saber cómo se configuran los beacons describiremos como se crea una aplicación *Android* desde la página de Estimote.

En el menú principal debemos seleccionar "*Mobiles Apps*", donde encontraremos todas las aplicaciones móviles que hemos creado. Si es la primera vez aparecerá vacía.

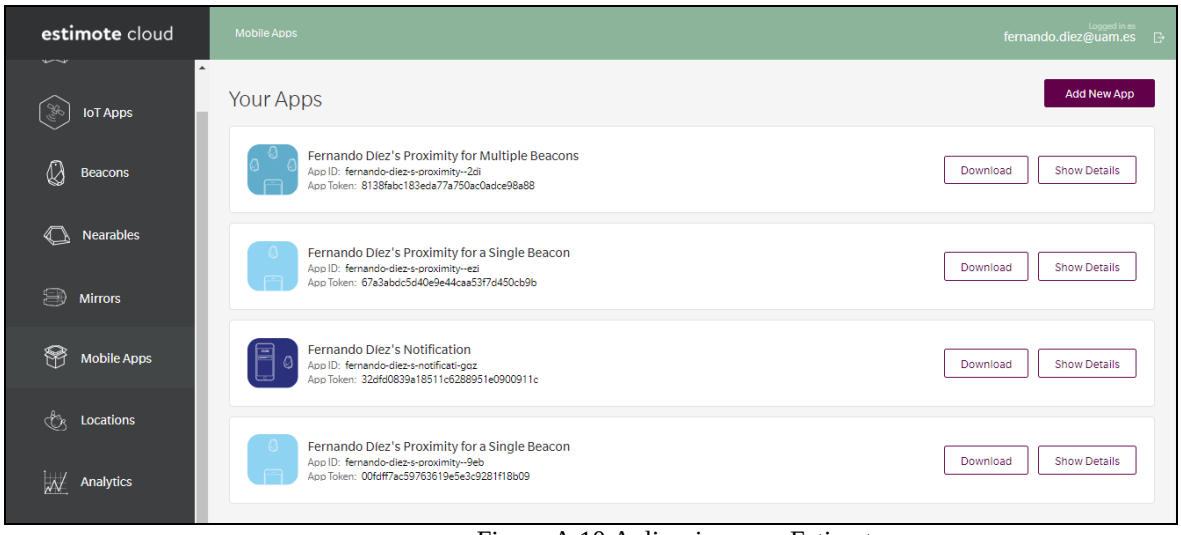

Figura A.10 Aplicaciones en Estimote.

Para crear una nueva aplicación pulsamos el boton "*Add New App*", nos aparecera una ventana emergente (*pop up*), que nos da a elegir distintas plantillas de aplicación para que elijamos las que mas se ajuste a las necesidades que queremos desarollar. En este proyecto hemos desarrollado a raiz de la plantilla de proximidad de un solo beacon, multiples beacons y notificaciones. En esta misma vista también nos ofrece otras aplicaciones, como la de localización, configuración o incluso crear tu propia aplicación desde cero si no se ajusta a ninguna de las funcionalidades que nos ofrece Estimote, y si se dispone de una aplicación ya creada, la puedes integrar con la opcion "*Get Estimote Cloud*", ya que este método te proporciona las credenciales de la nube de Estimote para que puedas realizarlo.

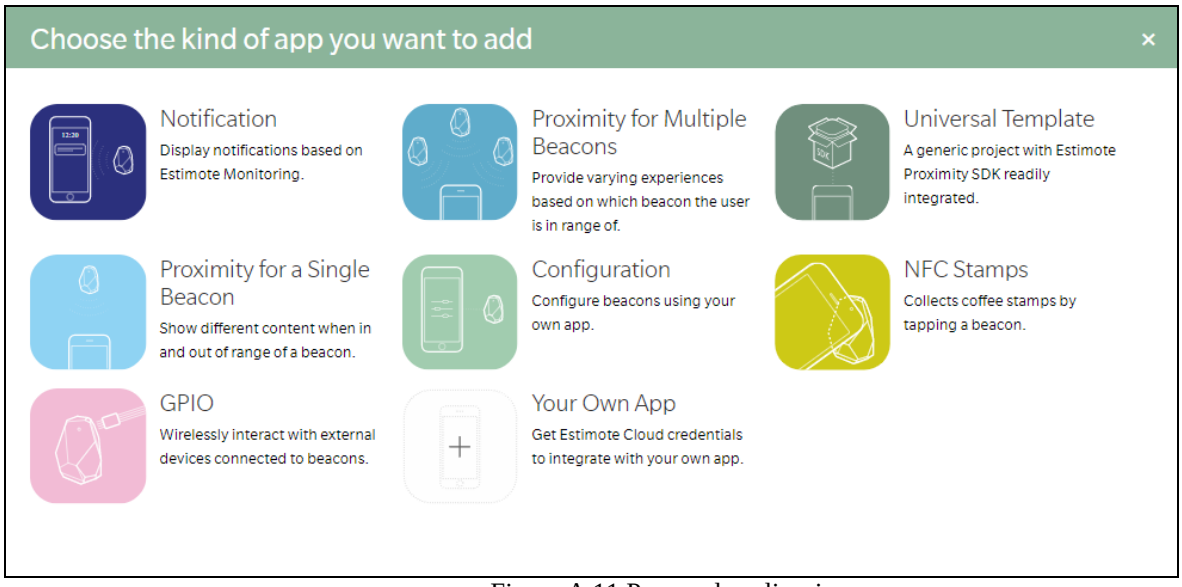

Figura A.11 Pop up de aplicaciones.

Si hemos accedido a través de un *template* tendremos que confirmar que queremos utilizar el "layout" que nos proporciona Estimote y luego en qué plataforma o lenguaje de programación se va a desarrollar la aplicación, y finalizamos esta sección pulsando el botón de continuar.

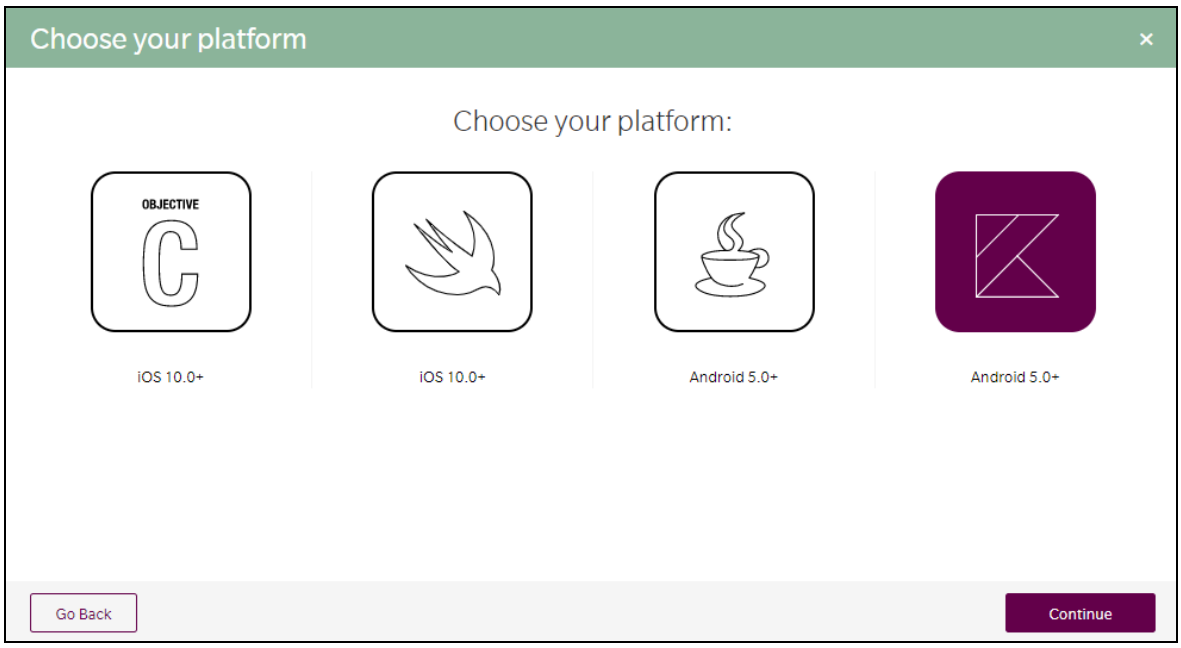

Figura A.12 Tipo de lenguajes.

Por último nos aparece otra pantalla para rellenar el nombre de la aplicación, una descripción y la asignación de uno o varios beacons según la aplicación elegida.

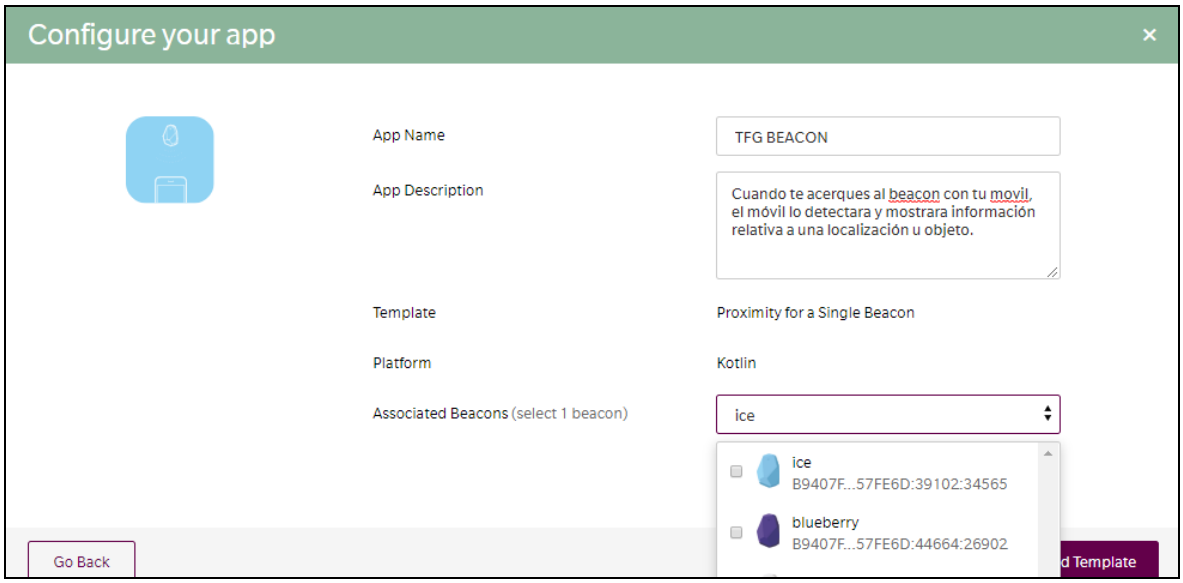

Figura A.13 Última ventana de configuracion de la aplicación.

Una vez elegida la aplicación que queremos para nuestro negocio, desde la página principal de aplicaciones nos aparece la opción de descargar la aplicación creada.

Para iniciar nuestro entorno de desarrollo, en nuestro caso Android Studio, abrimos el proyecto que nos hemos descargado anteriormente.

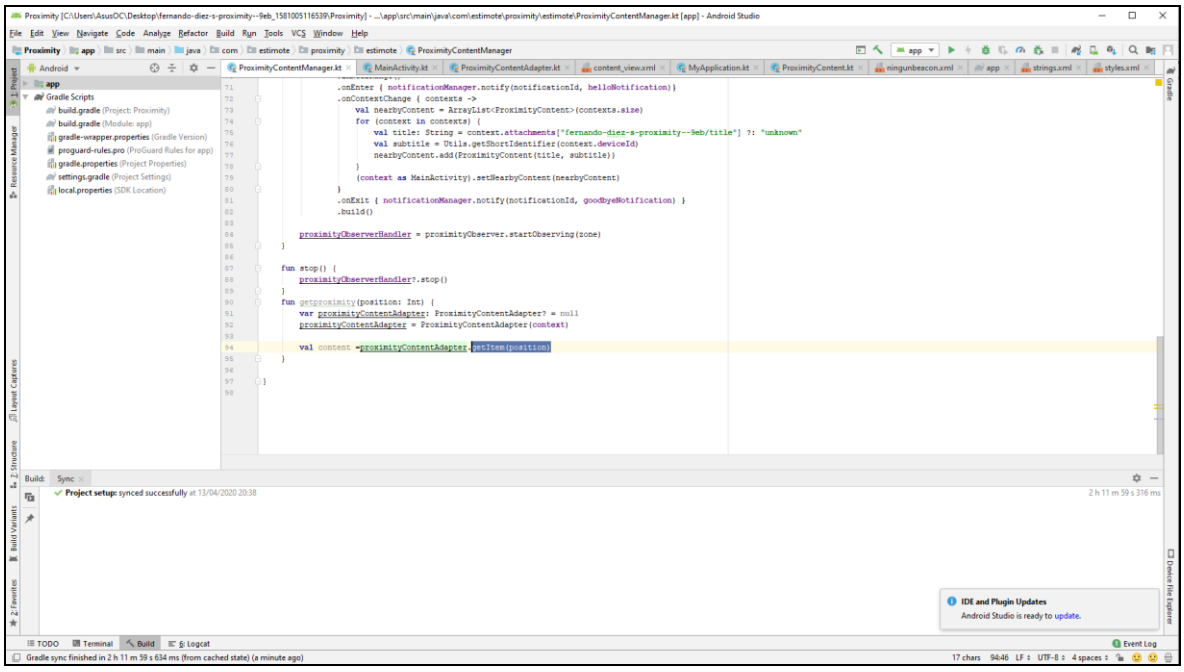

Figura A.14 Android Studio.

Para abrir el proyecto tendremos que ir a File >> Open, y se nos abre un seleccionador de archivos donde tendremos seleccionaremos nuestro proyecto.

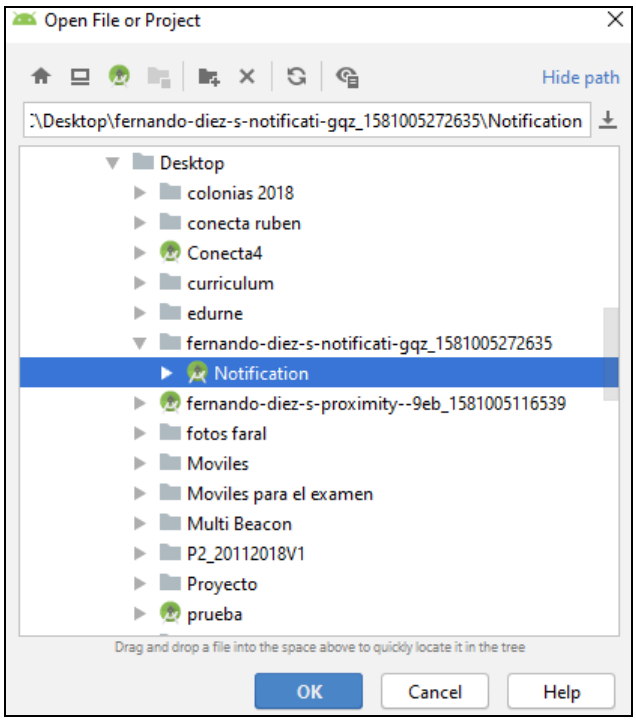

Figura A.15 Seleccionador de ficheros.

Lo primero que realiza Android Studio al abrir el proyecto es compilarlo, para detectar posibles errores de sintaxis y verificar que el *gradle* de la aplicación y el proyecto sean compatibles.

Si no compila y aparece un error de archivo corrupto se debe a que el *gradle* del proyecto no es el mismo que usa el programa, en este caso debemos ir al archivo *gradle-wrapper.properties* de nuestro proyecto para ver que versión de *gradle* va a utilizar y en su caso corregirlo, para nuestro proyecto utilizamos la siguiente distribución:

```
distributionUrl=https\://services.gradle.org/distributions/gradle-4.1-all.zip
```

Como se aprecia utilizamos "Gradle-4.1-all.zip", si no disponemos de este archivo en nuestro equipo, tenemos la posibilidad de descargarlo desde la siguiente dirección URL https://services.gradle.org/distributions/.

Una vez descargado el fichero comprimido debemos copiarlo a la carpeta *gradle* de Android, con el archivo zip descargado en su ruta anteriormente de dicha carpeta volvemos al programa *Android Studio* para terminar de asignar el *gradle* al proyecto.

El próximo paso que debemos realizar es abrir la configuración de *Android Studio* e ir a "*Build, Execution, Deployment*".

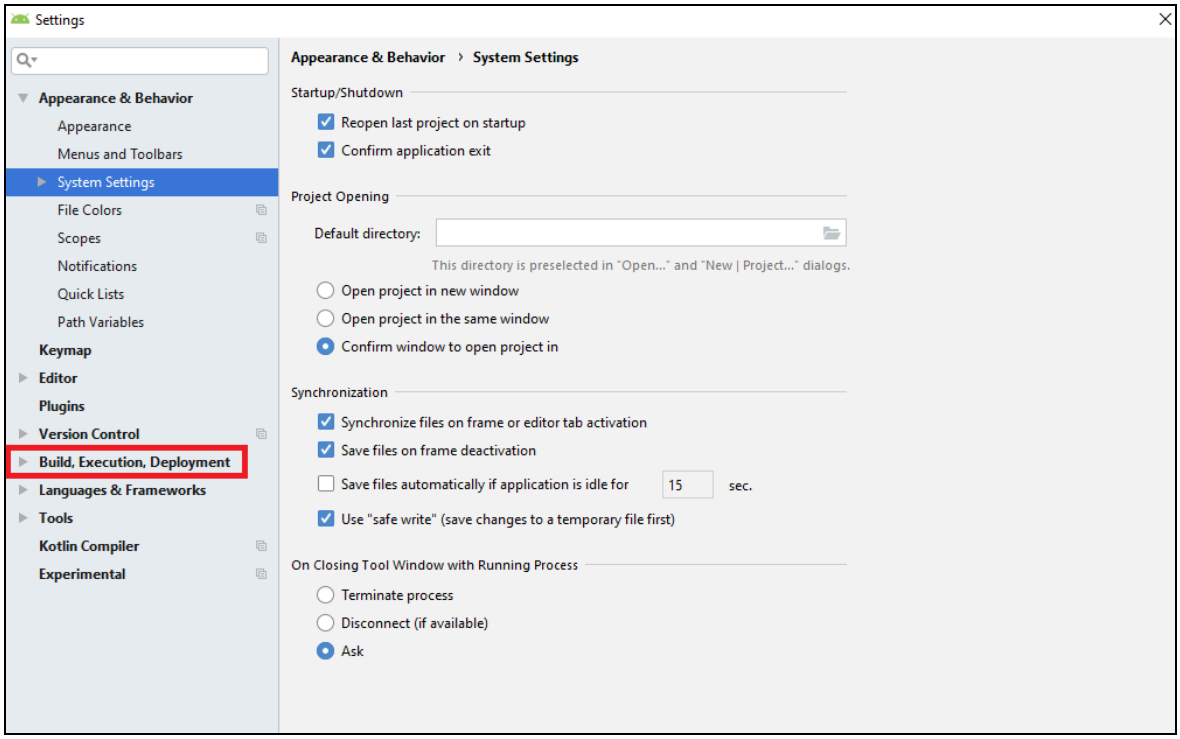

Figura A.16 Settings.

A continuación pulsamos en la opción Gradle y entramos en la configuración de compilación de la aplicación, aquí debemos habilitar "*Use local gradle distribution*" y seleccionar la carpeta donde hemos extraído el archivo comprimido descargado anteriormente. Para terminar y guardar aplicamos los cambios y aceptamos. Lo último que nos queda es volver a sincronizar la aplicación para compilar el proyecto.

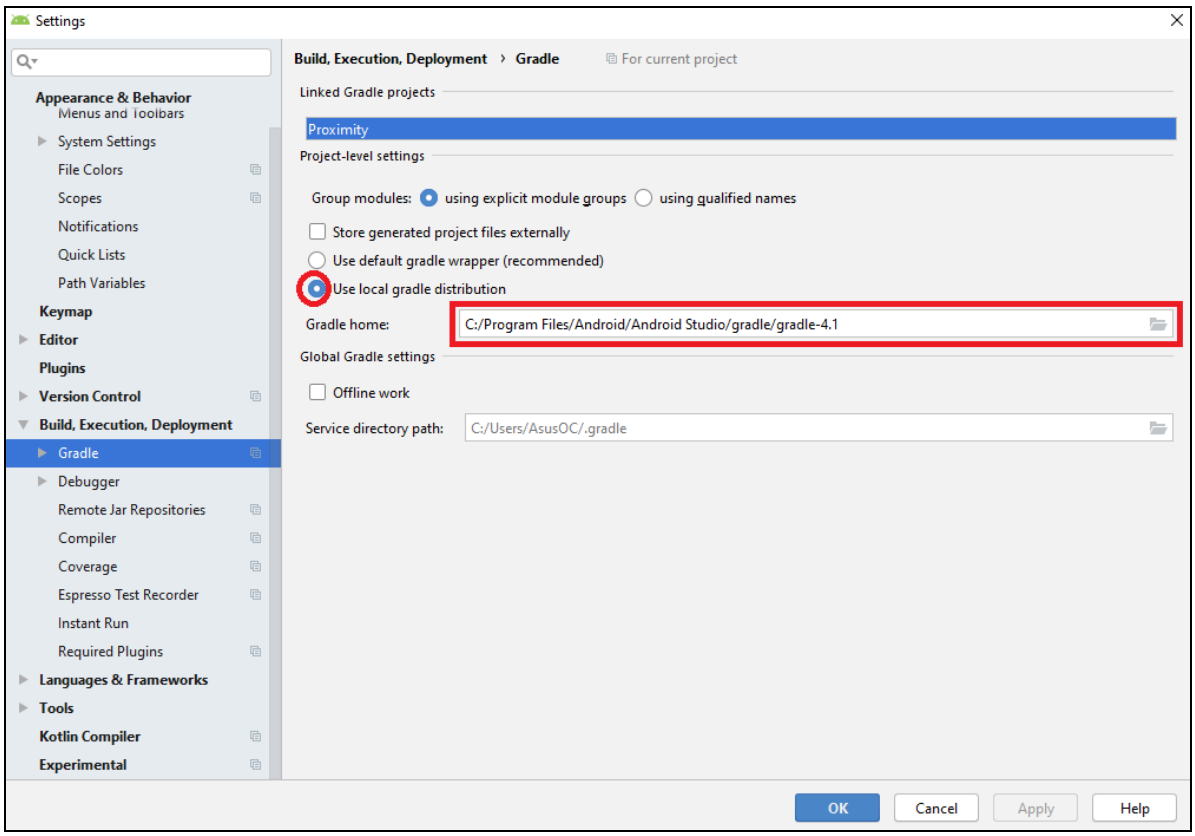

Figura A.17 Configuración del gradle.

B Características del Dispositivo móvil de pruebas

Las características del móvil con el que se han realizado las pruebas se han recogido de la página web indicada en la referencia [10]

OPPO RX17 PRO-

PANTALLA	**6,4" AMOLED** **FulHD+ 2.340 x 1.080, 402 ppp**
PROCESADOR	Snapdragon 710, octa core a 2,2 GHz GPU Adreno 616
RAM	6 GB
MEMORIA	128 GB
DISEÑO	157.6 x 74.6 x 7.9 mm 183 g
SOFTWARE	ColorOS 5.2 Android 8.1 Oreo
BATERÍA	3.700 mAh con carga Super VOOC
OTROS	Lector de huellas bajo la pantalla, dual SIM, 4G, NFC, BT 5.0, WiFi ac, GPS, USB-C