Trabajo de Fin de Grado

beTourist. Tecnología beacon en el turismo cultural

beTourist. Beacon technology in cultural tourism

Bianney Cabrera Delgado

La Laguna, 5 de septiembre de 2016

D. **José Luis González Ávila**, con N.I.F. 78.677.390-W profesor Asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"beTourist. Tecnología beacon en el turismo cultural."

Ha sido realizada bajo su dirección por Dña. **Bianney Cabrera Delgado**, con N.I.F. 78.626.143-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de septiembre de 2016.

Agradecimientos

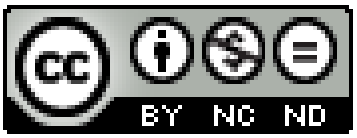
A D. José Luis González Ávila, por su labor como tutor de este trabajo, por brindarme su ayuda e indicaciones cuando las he necesitado.

A la Universidad de La Laguna, por ofrecerme una enseñanza de calidad y abrirme las puertas a un buen futuro profesional.

A mi familia, por tener paciencia, por no dejar que me rindiera y por enseñarme que nada cae del cielo.

A David Rodríguez Báez, mi pareja, por creer en mí cuando ni yo misma lo hacía. Por apoyarme y animarme cuando lo he necesitado y hacerme ver el mundo de otro color.

Licencia



Resumen

Este documento recoge aspectos relevantes del trabajo de investigación que ha llevado a cabo la alumna Bianney Cabrera Delgado durante el desarrollo de una aplicación para dispositivos móviles Android, que utiliza la tecnología beacons, una tecnología novedosa y poco conocida en nuestro país.

Al comienzo de este trabajo, la alumna poseía conocimientos de programación en Java, adquiridos en las asignaturas "Modelado de Sistemas Software" y "Diseño Arquitectónico y Patrones", asignaturas de tercero y cuarto curso del itinerario de Ingeniería del Software del Grado en Ingeniería Informática de la Universidad de La Laguna. Estas habilidades han sido útiles para el desarrollo de la aplicación móvil. Además, se han aplicado conocimientos sobre gestión de proyectos, adquiridos en la asignatura "Gestión de Proyectos Informáticos", y la capacidad para utilizar herramientas de desarrollo de software, obtenida en la asignatura "Laboratorio de Desarrollo y Herramientas".

La aplicación en cuestión, mostrará información relativa a monumentos históricos y naturales de la isla de Tenerife. Además, será capaz de establecer una conexión a dispositivos beacons, para mostrar información orientada a un perfil de usuario.

Palabras clave: Android, *beacon*, turismo inteligente, Internet of things, aplicación móvil.

Abstract

This document contains relevant aspects of the research that has been realized by the student Bianney Cabrera Delgado during development of an application for Android mobile devices, using the beacons technology, a new and little-known technology.

At the beginning of this work, the student had knowledge of programming in Java, acquired in the subjects "Modelado de Sistemas Software" and "Diseño Arquitectónico y Patrones" subjects of third and fourth year of the "Ingeniería del Software" itinerary of "Grado de Ingeniería Informática" on the University of La Laguna. These abilities have been useful for the development of the mobile application. They have also been applied project management abilities, acquired in the subject "Gestión de Proyectos Informáticos" and the ability to use software development tools obtained in the course "Laboratorio de Desarrollo y Herramientas".

La aplicación en cuestión, mostrará información relativa a monumentos históricos y naturales de la isla de Tenerife. Además, será capaz de establecer una conexión a dispositivos beacons, para mostrar información orientada a un perfil de usuario.

The application will display information of historical and natural monuments of the Tenerife Island. It will also be able to establish a connection to beacons to show custom information to a user profile.

Keywords: *Android, beacon, smart tourism, Internet of things, mobile application.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Este documento recoge aspectos relevantes del trabajo de investigación y desarrollo que ha llevado a cabo la alumna Bianney Cabrera Delgado y que representa su Trabajo de Fin de Grado del Grado en Ingeniería Informática, cursado en la Universidad de La Laguna.

En este capítulo se hará una breve introducción al tema que nos ocupa. Además, se situará al lector en el contexto pertinente, se introducirán algunos conceptos importantes y se expondrá el estado actual del asunto en cuestión.

1.1 Iniciativa

El uso de dispositivos inteligentes está tan extendido, que utilizarlos es una acción casi tan cotidiana como alimentarse. Mucha gente se levanta y se acuesta utilizando su *smartphone*. Además de la frecuencia y el alcance de éstos, también cabe destacar la temprana edad a la que se tiene acceso, por primera vez, a dichos dispositivos. Hoy en día, muchos niños, en los países desarrollados, aprenden a utilizar un *smartphone* incluso antes que a leer. Por otro lado, sabemos que cada año llegan a nuestra isla unos 5 millones de turistas, atraídos por nuestros paisajes, nuestra gente y cómo no, nuestro patrimonio histórico y natural. Como es lógico, la mayoría de ellos traerá consigo su *smartphone.*

Por estos y otros motivos, creo que debemos explotar aquellas posibilidades que los dispositivos inteligentes nos ofrecen, a la hora de conocer, y dar a conocer, los tesoros de los que disponemos en nuestro entorno. De esta forma podremos enriquecernos de nuestra cultura e historia, a la par que la acercamos a turistas y visitantes.

1.2 Antecedentes

En distintos lugares del mundo, se han llevado a cabo, exitosamente, algunos proyectos que pueden servir para poner al lector en situación. A continuación, se señalan algunos de ellos:

- **El estadio Citi Field de los New York Mets** (Cnet)**:** En este estadio se puso en marcha uno de los primeros proyectos basado en la tecnología *beacon*. Dicho proyecto consistió en la implantación de un sistema de *beacons* que consigue mejorar la experiencia de los visitantes. Entre otras funciones, estos dispositivos sirven para guíar a los usuarios hasta sus asientos, mostrarles puntos de interés del estadio y acercarles ofertas y promociones, todo ello directamente a sus *smartphones*.
- **Los cines Odeon de Inglaterra** (ibeacons.net)**:** El proyecto en el que se ha trabajado en estos cines de Inglaterra ha consistido en la implantación de *beacons* en sus salas de cine. En esta ocasión, las balizas son utilizadas para enviar a los espectadores información relevante, trailers de películas o notificaciones sobre el comienzo de las mismas. Además mediante los *beacons* son capaces de controlar el recorrido que efectúan los usuarios dentro de sus instalaciones, para así poder invertir en mejorarlas eficientemente.
- **Gafas Tzukuri** (Xataka)**:** La idea básica de este proyecto, desarrollado por un grupo de emprendedores, era encontrar una manera de localizar un objeto que se hubiera perdido. De esta forma diseñaron unas gafas con la tecnología iBeacon integrada. Estas gafas están conectadas con el iPhone mediante el *Bluetooth*, de manera que éste nos podría indicar el lugar en el que se encuentran, si se perdiesen.
- **El bar Kick en Londres** (9to5mac)**:** En este proyecto se ideó una manera para mejorar la experiencia de los clientes de este establecimiento, ofreciéndoles entretenimiento gratuito en forma de revistas digitales. Al hacer una consumición, mediante el uso de iBeacon, al cliente le entra una notificación en el iPad permitiéndole el acceso a la prensa que el establecimiento ofrezca. Una vez el cliente abandona el local, dichas publicaciones se vuelven a bloquear, ofreciéndole a los usuarios la opción de suscribirse si así lo desearan.

- **Be here** (Techcrunch)**:** Be here es una app desarrollada por un pequeño grupo de personas en Brasil. Esta aplicación utiliza la tecnología *beacon* para detectar automáticamente el acceso de los estudiantes al aula, ayudando de esta forma a controlar la asistencia a clase de los alumnos.

1.3 *Internet of things* y Tecnología *beacon*

El concepto de *internet of things* (Kontakt.io), o internet de las cosas, fue propuesto por Kevin Ashton (Wikipedia) en 1999 y se refiere a una gran variedad de maneras y tecnologías que permiten la conexión de "cosas" u objetos físicos a internet. De esta manera, cualquier objeto de la vida cotidiana podría adquirir la capacidad de ser inteligente y sensible al contexto en el que se encontrase. De esta forma, se conseguiría proporcionar experiencias atractivas y personalizadas a los usuarios y recopilar datos de interés de lo que ocurre en el entorno.

Una manera interesante de conectar objetos, del mundo real, con internet es la tecnología *beacon.*

Un *beacon*, faro o baliza, es un pequeño dispositivo que emite una señal periódica vía *Bluetooth.* Estas señales pueden ser detectadas por dispositivos móviles inteligentes y hacer que estos reaccionen ante la proximidad del *beacon,* proporcionando al usuario notificaciones o contenido extra (Kontakt.io).

La tecnología *beacon* es compatible con dispositivos móviles, *smartphones* y *tablets*, tanto Android como iOS, que dispongan de *Bluetooth* 4.0 o posterior.

Para poder realizar un proyecto que utilice esta tecnología, además de las balizas y los dispositivos móviles, es necesaria una aplicación móvil capaz de interpretar el contenido de las señales detectadas.

Más adelante se profundizará en este tema.

1.4 Estado del arte

Como ya se ha mencionado anteriormente, la tecnología *beacon* es una tecnología nueva y, mientras que en nuestro país aún está en fase de

conocimiento, en Estados Unidos está muy implantada en estrategias de marketing. En principio, se espera que a lo largo de este año, 2016, se generalice su uso en Europa.

En el primer apartado de este capítulo, se han señalado distintas razones que han motivado la propuesta y realización de este trabajo. A todas ellas, se le suma un interés en conocer y trabajar con una tecnología nueva y poco conocida y explotada, que puede tener un gran impacto en muchos ámbitos de la vida, debido a su gran variedad de aplicaciones.

1.5 Objetivos

Los objetivos principales que se pretenden conseguir con este TFG son los siguientes:

- Ampliar los conocimientos que se tiene con respecto al diseño y desarrollo de aplicaciones móviles para el sistema operativo Android.
- Investigar y conseguir conocimientos técnicos sobre una novedosa tecnología, la tecnología *beacon*.
- Ampliar los conocimientos y la destreza en el uso de distintas herramientas que facilitan las labores de desarrollo software, como pueden ser: GitHub, SonarQube, Gradle, etc.
- Realizar una aplicación móvil en la cual se apliquen los conocimientos obtenidos, con respecto a la tecnología *beacon*.

Capítulo 2. Tecnología beacon

En el capítulo anterior se ha introducido el término *beacon*, este capítulo se centrará en profundizar en este tema, en sus características y en los distintos protocolos que existen.

2.1 Qué es un *beacon*

Como se definió previamente, un *beacon* es un pequeño dispositivo que emite una señal periódica vía *Bluetooth* (Bluetooth Low Energy). Estos dispositivos actúan de manera "similar" a un faro. De la misma manera que un faro envía una señal lumínica repetitiva, a la espera de que los marineros la perciban, un *beacon* transmite una señal de radio, compuesta por una combinación de números y letras, a la espera de que un dispositivo equipado con *Bluetooth* 4.0 la detecte. Esta señal puede ser transmitida utilizando distintos protocolos de comunicación. Los dos protocolos más utilizados por las balizas son iBeacon y Eddystone. Una vez la señal ha sido detectada, y la aplicación ha reconocido el *beacon,* las acciones que se pueden ejecutar son muy diversas, entre ellas: mostrar alguna alerta, conectarse a un servidor, mostrar imágenes, audios, una página web, etc.

2.2 iBeacon vs Eddystone

En el apartado anterior, se mencionó que, tanto iBeacon como Eddystone son dos protocolos de comunicación utilizados por los *beacons.* A continuación, se detallará cada uno de ellos y se hará una comparativa.

2.2.1 iBeacon

iBeacon fue el primer protocolo desarrollado. En la actualidad es el más utilizado. Fue desarrollado por Apple en 2013 y está incluido de manera nativa en iOS7. iBeacon es compatible con dispositivos móviles, tanto iOS como Android, con *Bluetooth* 4.0 y posteriores y con ordenadors Macintosh

con OS X Mavericks utilizando la aplicación MacBeacon de Radius Networks (Wikipedia).

La señal de un faro iBeacon, contiene una serie de números y letras divididas en grupos con un significado concreto. Cada código reconoce unívocamente a cada baliza y la aplicación móvil sólo reaccionará ante la señal de un *beacon*, si dicha aplicación lo reconoce como un *beacon* al que debe escuchar, para ello deberá disponer de una lista de *beacons* a los que observar. Una señal iBeacon consta de cuatro partes de información:

- **UUID (16 bytes):** Identifica unívocamente al *beacon.*
- **Major (2 bytes):** Identifica un subgrupo de faros dentro de un grupo mayor.
- **Minor (2 bytes):** Identifica a un *beacon* específico.
- **Tx Power (1 byte):** Medida de intensidad, se utiliza para calcular la distancia a la que se encuentra el *beacon* del dispositivo receptor.

En la siguiente imagen se resume el protocolo iBeacon y el contenido y significado de la señal que envía:

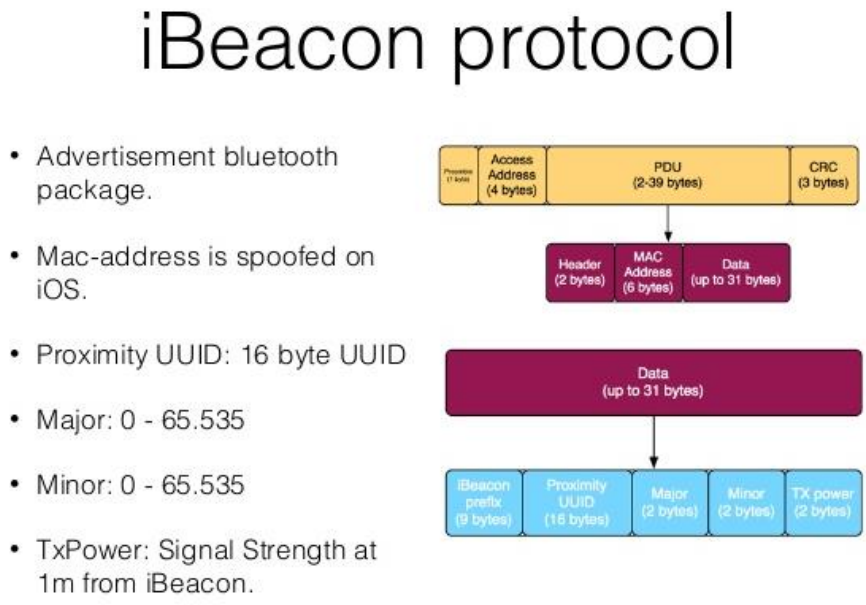


Figura 2.1. Protocolo iBeacon (Slide Share)

2.2.2 Eddystone

Eddystone es un protocolo de comunicación nuevo y abierto. Ha sido desarrollado por Google, enfocado a los usuarios de Android, aunque compatible también con dispositivos iOS.

Su funcionamiento es similar al de iBeacon, pero tiene funcionalidades añadidas. Eddystone tiene 3 tipos de protocolo diferentes:

- **Eddystone-UID:** Tiene un funcionamiento similar al de iBeacon. Emite un código en un intervalo regular.
- **Eddystone-URL:** Emite una dirección URL que puede ser vista por cualquier *smartphone*, tenga o no instalada la aplicación.
- **Eddystone-TLM:** Transmite datos telemétricos relativos a sensores que pueden ser conectados a los beacons (Kontakt.io).

Seguidamente se muestra una imagen en la cual se resume el protocolo Eddystone y el contenido y significado de su señal:

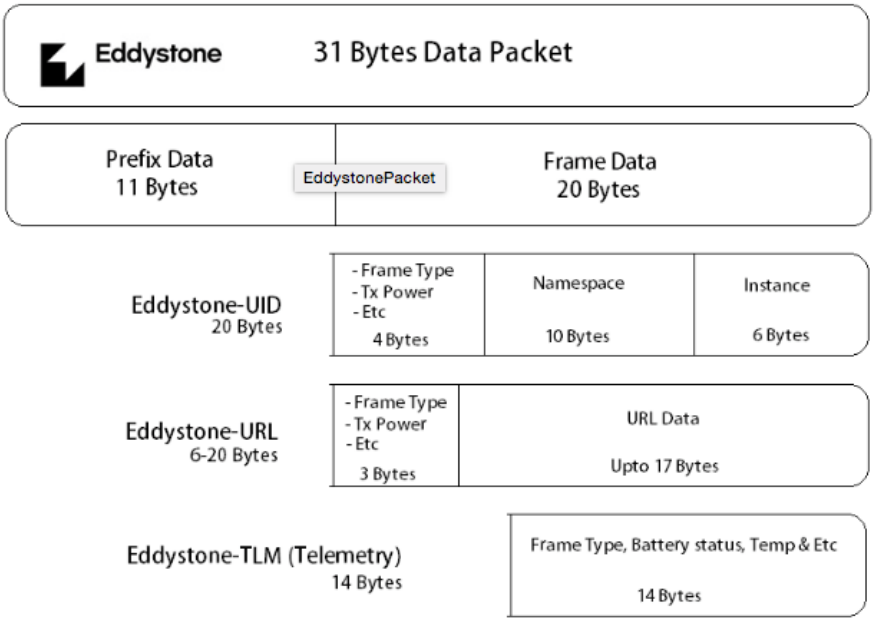


Figura 2.2. Protocolo Eddystone (IOSandi)

2.2.3 iBeacon vs Eddystone

Para decidir el perfil de *beacon* que será utilizado, se analizó la comparativa entre ambos protocolos. Para este proyecto, se ha decidido utilizar el protocolo de comunicación Eddystone, debido a que ofrece una

mayor flexibilidad y que está desarrollado especialmente para dispositivos Android. A continuación se mostrará una imagen que compara ambos protocolos.

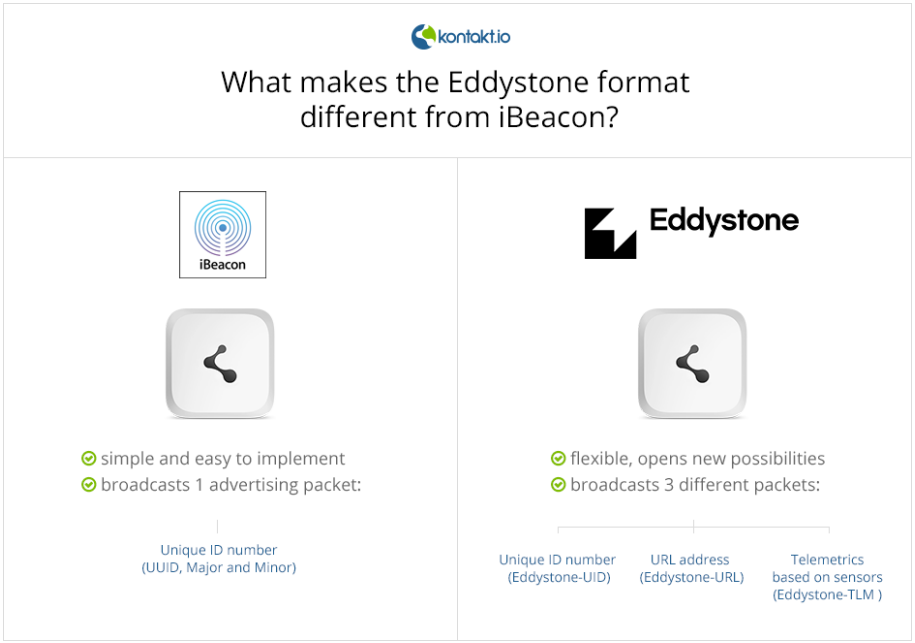


Figura 2.3 Comparativa iBeacon y Eddystone (Using beacons)

2.3 Elección de dispositivo

Debido a que no se disponía de ningún dispositivo *beacon*, se tuvo que realizar una pequeña investigación y análisis de la oferta de los distintos fabricantes. Se valoró la calidad y características físicas en relación al precio al que ofrecían su producto. A continuación, se muestran las dos opciones más importantes que se barajaron:

- **Estimote** (Estimote)**:**
  - 2 años de duración media de la batería. Máximo 6 años y 1 año como iBeacon.
  - Procesador ARM® Cortex™ M4F.
  - Dispone de BLE, sensores de movimiento y temperatura.
  - Compatible con iBeacon y Eddystone.
  - Hasta 70m de alcance.
  - SDK propia.

- **Kontakt** (Kontakt.io)**:**
    - 2 años de duración media de la batería.
    - Procesador ARM® Cortex™ M0 CPU core.
    - Dispone de BLE.
    - Compatible con iBeacon y Eddystone.
    - Entre 50 y 70m de alcance.
    - SDK propia.
    - Panel de desarrollador.

Después de analizar estas opciones, se optó por la solución de Kontakt, ya que se consideró que ofrecían unas muy buenas características a un precio un poco más asequible, además de un servicio y atención a desarrolladores bastante completo.

Seguidamente, se profundizará un poco más en el hardware y software proporcionado por Kontatk.io.

2.3.1 Hardware

Se optó por adquirir los *Smart Beacon* de kontakt. En el apartado anterior se dieron algunas características de estos dispositivos. A continuación, se entrará más en detalle en la especificación técnica (Kontakt.io) de ellos.

- **Dimensiones y peso:**
    - **Dimensiones:** 15 x 55 x 56 mm.
    - **Peso:** 23 gr.

Figura 2.4. Dimensiones *Smart Beacon* de Kontakt (Kontakt.io).

- **Procesador:**
    - 32-bit ARM® Cortex™ M0 CPU core.

    - **Bluetooth:** Nordic nRF51822.
    - **Tasas de transferencia de datos:** 250kBs, 1Mbs, and 2Mbs.
    - **Memoria:** 256KB flash 16KB RAM.
- **Batería:**
    - 1 x 1,000mAh CR2477.
    - **Tipo:** Celda, reemplazable.
- **Carcasa:**
    - **Material:** LUPOY GN5001RFG.
    - **Resistencia al fuego:** Seguro – Clase de inflamabilidad V0, extinción del fuego en menos de 10 seg. Goteo de partículas permitidos, siempre que no se inflamen (Wikipedia).
    - **Grado de protección:** IP-57, protección al polvo e inmersión en agua a 1 m durante 30 min. (Wikipedia).
- **Comunicaciones:**
    - **Bluetooth:** Bluetooth Low Energy wireless technology 2.4GHz RF.
- **Poder de transmisión:**
    - **Bluetooth:** -30 dBm to 4 dBm.
- **Sensibilidad:**
    - **Bluetooth:** -93 dBm.
- **Batería:**
    - Hasta 2 años, con un intervalo de transmisión de 350 ms.
    - Hasta 6 meses, con un intervalo de 100 ms.
- **Requisitos medioambientales:**
    - **Temperatura:** Desde -20ºC hasta 60ºC.
    - **Humedad:** desde 0% hasta 100%.

A continuación se muestra la imagen del despiece de un *Smart Beacon* de Kontakt:

Figura 2.5. Despiece *Smart beacon* de Kontakt (Kontakt.io)

2.3.2 Software

Al adquirir los *beacons* de Kontakt, fue proporcionada una aplicación móvil que permite consultar y realizar cambios en la configuración de los *beacons*, acceso a un panel de control en el que poder consultar y gestionar algunos aspectos de los dispositivos y la utilización de su SDK propia.

- **App móvil Kontakt** (Google Play)**:** Como ya se mencionó, desde la aplicación móvil de Kontakt se puede modificar la configuración de los *beacons*. Además, permite actualizar el firmware y establecer el protocolo de comunicación, iBeacon o Eddystone, que utilizará el faro. A continuación, se muestran varias imágenes obtenidas de la aplicación:

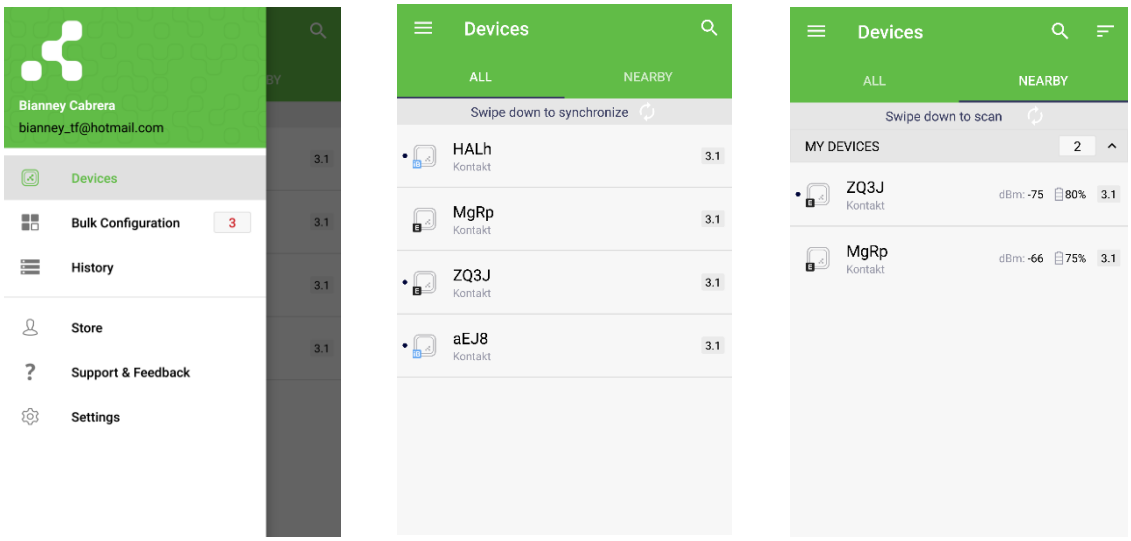


Figura 2.6. Menú de la app, pestaña con todos los *beacons* asociados a la cuenta y pestaña con los *beacons* cercanos.

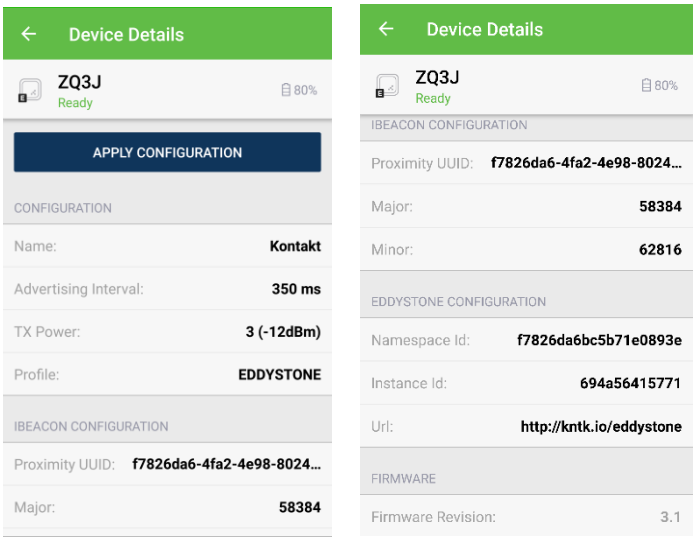


Figura 2.7. Detalle configuración de un *beacon*

- **Panel de control** (Kontakt.io)**:** Para poder acceder a este panel web, es necesario iniciar sesión con una cuenta de desarrollador que proporciona Kontakt. Este panel ofrece distintas funcionalidades, repartidas en 3 pestañas que se detallan a continuación:
  - ***Devices*:** En esta pestaña, se pueden observar los distintos dispositivos que se tienen vinculados a la cuenta.

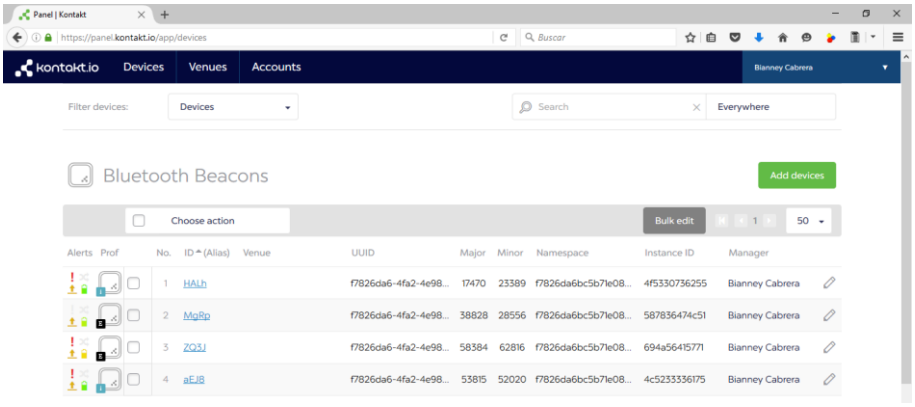


Figura 2.8. Vista de los dispositivos

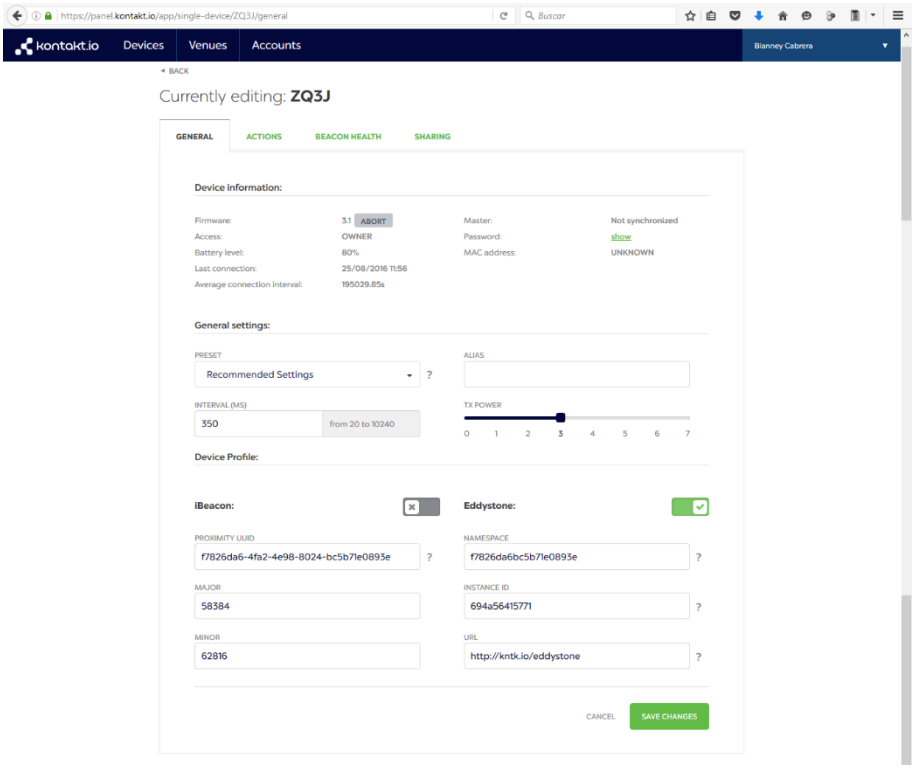


Figura 2.9. Detalle configuración de un *beacon*

- ***Venues*:** Un *venue* (Kontakt) es una agrupación de *beacons*. Estos grupos suelen representar los lugares físicos donde se encuentran las balizas. Podría ser cualquier lugar concreto, una habitación, planta, edificio, o en nuestro caso, un monumento concreto. En este apartado se pueden crear y configurar estos grupos. A continuación, se muestra un ejemplo de la creación de un grupo denominado Monumento X que contiene dos *beacons*.

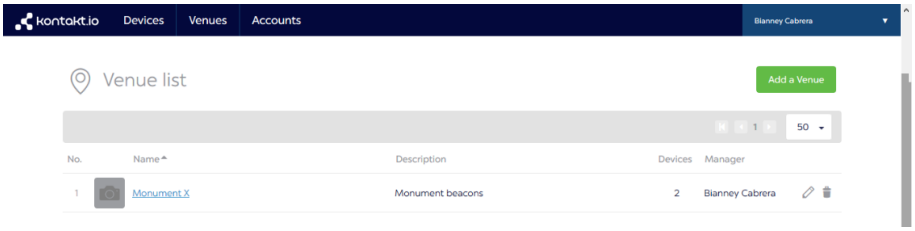


Figura 2.10. *Venue list*

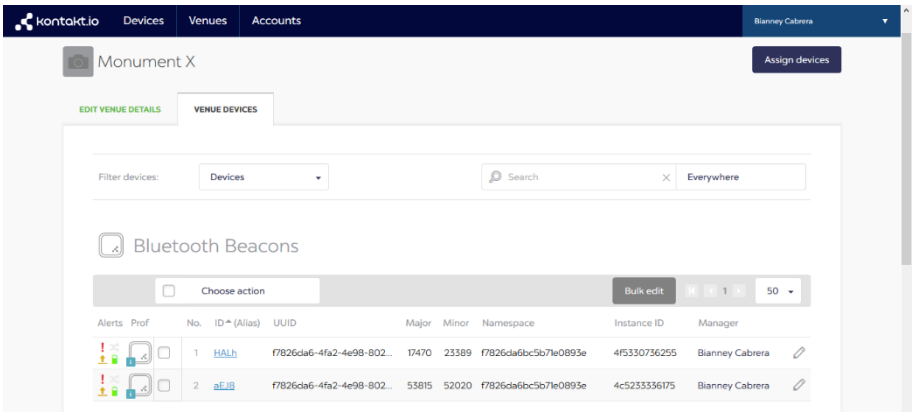


Figura 2.11. Detalle de un grupo

- Accounts: En este apartado se pueden añadir cuentas de otros usuarios y asignarlos como administradores de faros o grupos concretos. Esta funcionalidad es muy útil a la hora de trabajar con un grupo de desarrolladores.

- **SDK:** La SDK, o Kit de Desarrollo de Software, ofrece al programador las herramientas necesarias para integrar la funcionalidad de los *beacons* a cualquier aplicación móvil existente. Esto ha sido vital para la elaboración de este trabajo. Más adelante, se entrará más en detalle respecto a esto.

Capítulo 3.
Análisis

En este capítulo se realizará un análisis previo de la aplicación. En primer lugar, se establecerán los requisitos que deberá cumplir el proyecto. Seguidamente, se presentarán los diagramas de casos de uso y a continuación se especificará cada uno de ellos con más detalle.

3.1 Requisitos

3.1.1 Aplicación servidor

- **Requisitos funcionales:**
  - **Importación del conjunto de datos:** Deberá poder acceder y descargar los datos relativos a los monumentos de Tenerife. Dichos datos están alojados en la web de Open Data Canarias (Open Data Canarias), en un fichero en formato .json. Además se deberá realizar un parseo de dichos datos y almacenarlos en una base de datos.

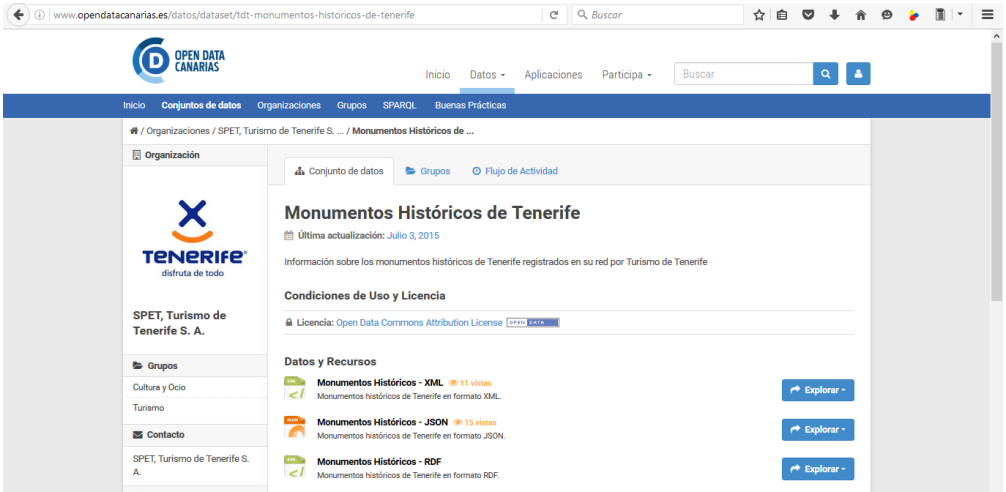


Figura 3.1. Captura de la localización de los datos relativos a los monumentos históricos (Open Data Canarias)

  - Facilitará el acceso a los datos a la aplicación móvil.

- **Requisitos no funcionales:**
    - Deberá estar disponible las 24h.

3.1.2 Aplicación móvil

- **Requisitos funcionales:**
    - **Importación del conjunto de datos**: Podrá acceder a los datos relativos a los monumentos facilitados por la aplicación servidora y almacenarlos en una base de datos en el dispositivo móvil.
    - **Lista de monumentos:** Mostrará una lista de monumentos. En la lista únicamente se detallará el nombre y el municipio en el que se encuentra localizado dicho monumento.
    - **Búsqueda:** Ofrecerá la posibilidad de realizar búsquedas de monumentos concretos, según su nombre o el municipio en el que esté localizado.
    - **Localización:** Mostrará la localización de los monumentos mediante un mapa basado en la API de Google Maps.
    - **Cómo llegar:** Ofrecerá el servicio *Cómo llegar* a un monumento deseado, desde la posición actual del usuario.
    - **Menú de configuración:** Dispondrá de un menú de configuración que deberá tener, los siguientes apartados:
        - **Perfil:** En esta sección el usuario podrá establecer su perfil de usuario. Este perfil se utilizará para personalizar la información, de los monumentos, mostrada, utilizando un cierto grado de inteligencia artificial.
        - **Idioma:** Se permitirá elegir el idioma de la aplicación (español o inglés).
    - **Tecnología *beacon*:** Tendrá que percibir e identificar una serie de dispositivos *beacons* e interpretar las señales enviadas por ellos.
- **Requisitos no funcionales:**
    - Deberá estar disponible para dispositivos Android con *Bluetooth* 4.0 y Android 4.3 o posterior.

- **Multi-idioma:** Debido a que la aplicación estará orientada, en gran medida, al turismo, deberá estar disponible en español e inglés.
- **Interfaz:** Deberá tener una interfaz atractiva visualmente, intuitiva y de fácil manejo.

3.2 Aplicación Servidora

3.2.1 Importación de los conjuntos de datos

- **Descarga de los conjuntos de datos**

Como se dijo anteriormente, los datos serán obtenidos de la página web de OpenDataCanarias. En primer lugar, la aplicación deberá descargar los ficheros *.json,* correspondientes a los monumentos históricos (Open Data Canarias) y naturales (Open Data Canarias). Estos archivos tendrán la siguiente estructura:


```
{
  "help": "Listado de monumentos históricos de Tenerife",
  "listado": [
    {
      "ows_LinkTitle": "Parroquia de Santo Domingo de Guzmán ",
      "ows_Zona": "Norte",
      "ows_Georeferencia": "(28.378118, -16.670812999999953)",
      "ows_CodigoPostal": "",
      "ows_Direccion": "La Costa de la Guancha ",
      "ows_Telefono": "",
      "ows_Fax": "",
      "ows_Web": "http://www.obispadodetenerife.es",
      "ows_FechaActualizacion": "",
      "ows_Situacion": "Abierto",
      "ows_ObservacionesSituacion": "",
      "ows_DescripcionEspanol": "",
      "ows_DescripcionIngles": "",
      "ows_DescripcionAleman": "",
      "ows_TipoActividad": "Naturaleza y Monumentos",
      "ows_Actividad": "Fichas Temáticas",
      "ows_Email": "obispado@obispadodetenerife.es",
      "ows_Municipio": "Guancha (La)",
      "ows_Recomendado": "",
      "ows_DescripcionRuso": "",
      "ows_DescripcionItaliano": "",
      "ows_Dificultad": "Bajo",
      "ows_Peligrosidad": "Bajo",
      "ows_ComoLlegarEspanol": "",
      "ows_ComoLlegarIngles": "",
      "ows_ComoLlegarAleman": "",
      "ows_ComoLlegarEspanol": "",
      "ows_ComoLlegarIngles": "",
      "ows_ComoLlegarAleman": "",
      "ows_ComoLlegarFrances": "",
      "ows_ComoLlegarItaliano": ""
    }
```


Figura 3.2. Fragmento del fichero de datos de monumentos históricos

- **Parseo de los datos**

Una vez descargados los datos, se debe proceder a realizar una lectura de los ficheros, un *parseo* de dichos datos y un adecuado almacenamiento de

ellos. El *parseo* se realiza para comprobar que el esquema del fichero no ha cambiado, con respecto al que se tenía al desarrollar la aplicación. Previamente, se debe haber creado una buena estructura de clases para almacenar toda la información útil de manera eficaz.

3.3 Casos de uso

3.3.1 Diagramas de casos de uso

- **Aplicación servidor:**

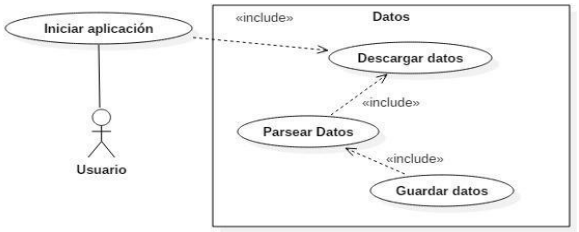


Figura 3.3. Casos de uso Aplicación servidor

- **Aplicación móvil:**

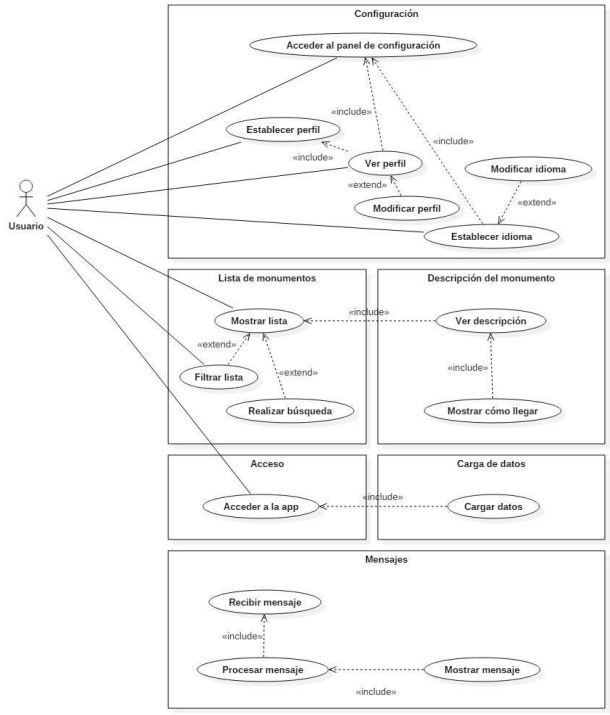


Figura 3.4. Casos de uso de la aplicación móvil

3.3.2 Especificación de casos de uso

En la especificación de casos de uso se proporcionan detalles textuales de cada uno de los casos de uso. Para este proyecto, está representada en las tablas que se pueden encontrar en el apéndice A de este documento.

Capítulo 4.
Diseño

En este capítulo se hará una primera aproximación al diseño de la aplicación. Además se mostrará una serie de vistas preliminares, para así tener una cierta idea de cómo puede ser la interfaz gráfica de la aplicación.

4.1 Prototipo

Para el diseño del primer prototipo de las vistas de la aplicación, se ha utilizado la herramienta de prototipado *Pencil* (Pencil) y el programa de edición de imágenes *Adobe Photoshop* (Adobe).

4.1.1 Pantalla de inicio

Al ejecutar la aplicación, se mostrará una pantalla de inicio como la mostrada en la siguiente imagen. Esta pantalla permanecerá hasta que la aplicación cargue todos los datos y opciones necesarias.

Figura 4.1. Vista inicial

4.1.2 Vistas principales (monumentos)

Una vez iniciada la aplicación, se cargará la vista de los monumentos históricos (Figura 4.2). En estas vistas se representa una lista de todos los monumentos disponibles en los datos, separados en *históricos* y *naturales*, como se mostrará a continuación, además de tres botones (en la esquina superior izquierda un botón para acceder al menú, mientras que en la derecha dos botones, el primero aplicar un filtro a la lista y el segundo para realizar una búsqueda).

Figura 4.2. Vistas principales

4.1.3 Menú

La siguiente imagen representa la vista de un menú lateral, en el que estarán disponibles los distintos apartados de la aplicación. Podrá accederse a dicho menú pulsando el botón de menú o arrastrando la pantalla desde la izquierda a la derecha. En este menú se presentan las siguientes opciones: Históricos, Naturales, Mapa, Configuración y Acerca de.

Figura 4.3. Menú

4.1.4 Configuración

En esta vista se podrá encontrar el panel de configuración. El panel está compuesto de dos botones, desde los cuales se podrá acceder al perfil de usuario y a la elección del idioma, respectivamente, según se muestra en la siguiente imagen:

Figura 4.4. Menú de configuración

4.1.5 Visualización y establecimiento del perfil

A continuación, se muestran las dos vistas relativas al perfil de usuario. Serán accesibles desde el menú de configuración. En la primera, se mostrará el perfil de usuario, mientras que en la segunda, se podrá modificar.

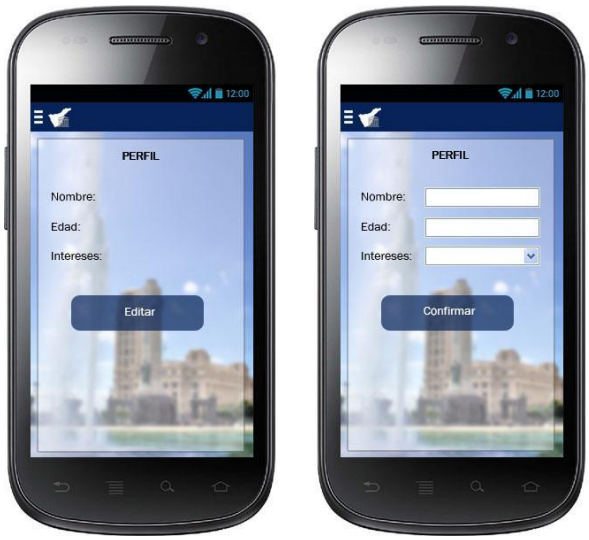


Figura 4.5. Visualización y establecimiento de perfil

4.1.6 Establecimiento del idioma

En esta vista se podrá elegir el idioma en el que se desea que se presente la aplicación. En principio, estará disponible en español e inglés.

Figura 4.6. Establecimiento del idioma

4.1.7 Detalle de monumento

En la vista que se muestra a continuación, se presentan datos de interés del monumento. A ella se accede seleccionando algún monumento desde la lista. Además, se muestra un mapa con su localización.

Figura 4.7. Detalle del monumento

4.1.8 Acerca de

Seguidamente, se muestra una sencilla vista del acerca de. Esta vista deberá contener información relativa a la aplicación.

Figura 4.8. Acerca de

Capítulo 5. Desarrollo

En este capítulo se detallarán aspectos relativos a las distintas fases del desarrollo llevado a cabo.

5.1 Aplicación servidor

En primer lugar se ha desarrollado una sencilla aplicación servidor. Esta aplicación se ha desarrollado utilizando el lenguaje de programación Java en el IDE de Eclipse.

La aplicación se encarga de descargar periódicamente los ficheros de los datos, de monumentos históricos y naturales, de la web de Open Data Canarias, realizar un *parseo* de dichos datos y comprobar que encajan en la estructura de clases creada para la aplicación móvil. Además, aloja estos ficheros de datos en un servicio de alojamiento en la nube.

A continuación, se muestra un fragmento de código de la función *parser()*. En esta función, se comprueba que cada uno de los campos necesarios, introducidos previamente en la lista *fieldRequired*, existe en el objeto *obj*, de tipo *JSONObject*, que contiene los datos de cada monumento.

```
public boolean parser (JSONArray monuments, int i) throws JSONException{
    JSONObject obj = monuments.getJSONObject(i);
    List<String> keys = (List<String>) obj.keys();
    for (int j = 0; j < fieldRequired.size(); j++){
       if (!keys.contains(fieldRequired.get(j))){
           return false;
       }
    }
    return true;
}
```

Figura 5.1. Función que comprueba la existencia, en el .JSON, de los datos necesarios.

5.2 Aplicación móvil

La aplicación móvil ha sido desarrollada en Android Studio, utilizando la SDK de Android.

5.2.1 Lectura de los datos

En primer lugar, se ha realizado un preprocesado de los datos. Para ello, se ha debido acceder a los datos relativos a los dos tipos de monumentos, históricos y naturales, alojados en el servidor, leerlos y almacenar la información en una estructura de clases que los represente.

```
public class Monument implements IMonument{

    private String name;
    private String zone;
    private String address;
    private String cp;
    private String town;
    private double longitude;
    private double latitude;
    private String phoneNumber;
    private String web;
    private String situation;
    private String spanishDescription;
    private String englishDescription;
    private String germanDescription;
    private String update;
    private String dificulty;
    private String danger;
    private String image;
```

Figura 5.2. Especificación de la clase Monument.

La interfaz *IMonument* define las funciones que debe implementar la clase *Monument*, de la cual heredan las clases Historical y Natural, que representan los dos tipos de monumentos existentes.

5.2.2 Representación de los datos

Las listas de monumentos han sido separadas en dos *sliding tabs,* monumentos históricos y naturales, donde se muestra el nombre y el municipio donde se encuentra cada uno de ellos. En la siguiente imagen, se puede observar una vista de ambas listas de monumentos:

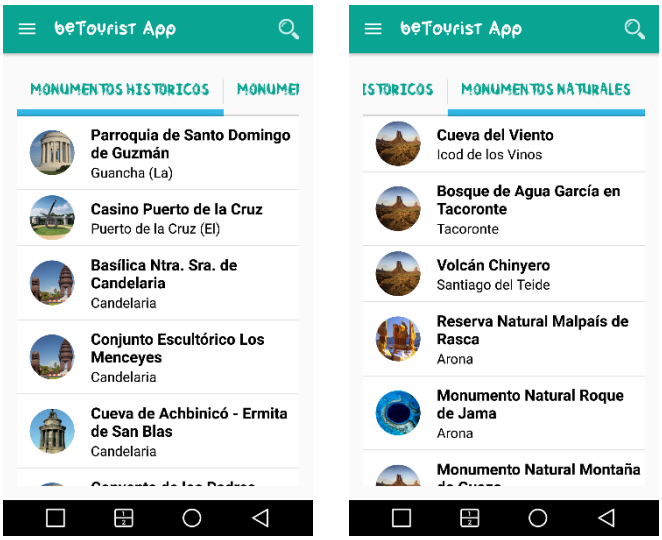


Figura 5.3. Vistas de las listas de monumentos, históricos y naturales.

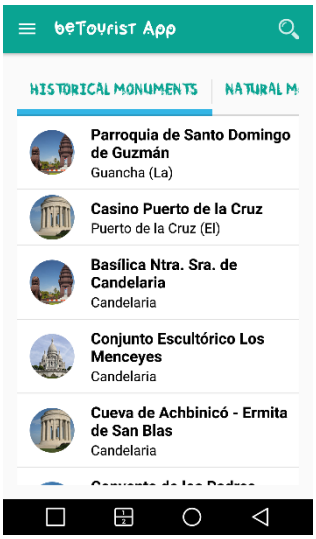


Figura 5.4. Vista de la lista de monumentos históricos, en inglés.

Debido a que no se disponía de una imagen de cada monumento, se decidió utilizar imágenes de prueba, cargadas aleatoriamente, a modo de ejemplo.

Al seleccionar alguno de los monumentos, se accede a la actividad *HistoricalActivity.java* o *NaturalActivity.java*, según corresponda. En esta vista, se muestra una imagen, información más detallada y la localización, mediante un marcador en un mapa, del monumento seleccionado. En dicho mapa, al seleccionar el marcador, se ofrece la posibilidad de mostrar *cómo llegar* al monumento o de abrir dicha localización en Google Maps (Google), en el siguiente punto, en el apartado *Mapa,* se verá esta opción. A continuación, se muestra una vista de esta actividad.

Figura 5.5. Vista del detalle de los monumentos, en la primera un monumento histórico y en la segunda un monumento natural.

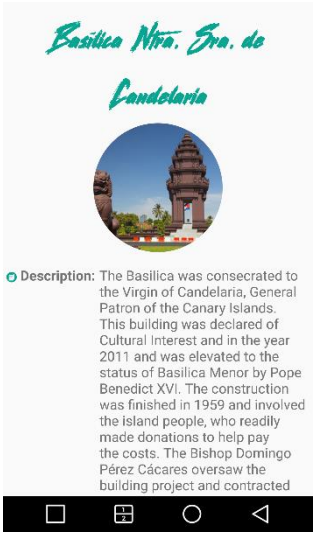


Figura 5.6. Vista del detalle de un monumento histórico, en inglés

5.2.3 Función buscar

En la vista principal de la aplicación, se ha proporcionado la posibilidad de buscar un monumento, tanto histórico como natural, según su nombre o la localización en la cual esté ubicado. Para acceder a dicha función, se debe seleccionar el icono de la lupa, situado en la barra de la aplicación, e

introducir la palabra deseada. A continuación, se muestra un ejemplo en el cual se desea conseguir una lista con todos los monumentos relacionados con Candelaria:

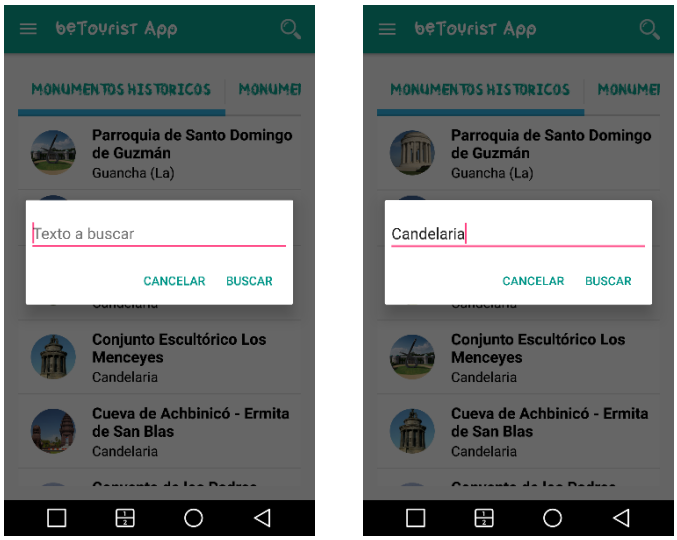


Figura 5.7. Ejemplo de búsqueda de monumentos, en Candelaria.

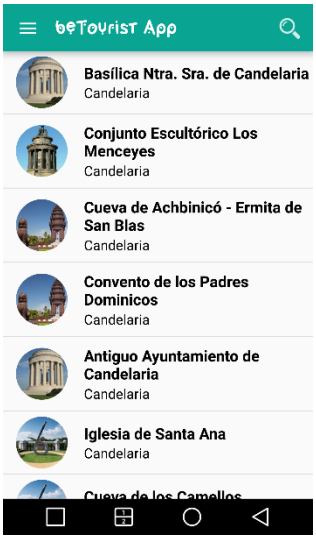


Figura 5.8. Resultado de la búsqueda.

La siguiente imagen muestra la función *search*. Esta función recorre las listas de monumentos, históricos y naturales, y devuelve otra lista filtrada que contiene los monumentos cuyo nombre o ciudad de localización contienen el texto buscado:

```
public List<Monument> search (String word){
    List<Monument> monumentsSearch = new ArrayList<Monument>();
    for (int i = 0; i < Monuments.getInstance().getHistorical().size(); i++){
        Monument monument = Monuments.getInstance().getHistorical().get(i);
        if ((monument.getName().contains(word))||(monument.getTown().contains(word))){
            monumentsSearch.add(monument);
        }
    }
    for (int i = 0; i < Monuments.getInstance().getNatural().size(); i++){
        Monument monument = Monuments.getInstance().getNatural().get(i);
        if ((monument.getName().toLowerCase().contains(word))||(monument.getTown().toLowerCase().contains(word))){
            monumentsSearch.add(monument);
        }
    }
    return monumentsSearch;
}
```

Figura 5.9. Función que filtra las listas de monumentos según el texto introducido.

5.2.4 Menú

Al igual que en el prototipo, se ha provisto a la aplicación de un *slide menu*. Se puede observar en la siguiente imagen:

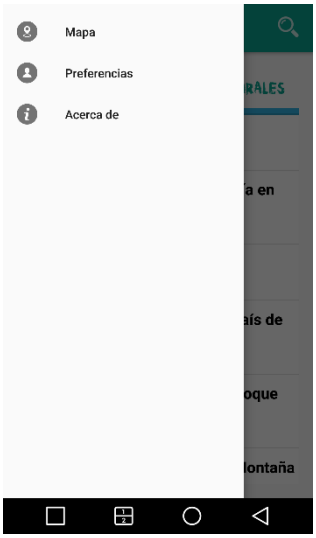


Figura 5.10. Menú lateral de la aplicación.

Como se puede apreciar, el menú cuenta con las siguientes opciones:

- **Mapa:** Ejecuta la actividad *MapActivity*. En esta actividad se muestra un mapa con la localización de todos los monumentos, tanto históricos como naturales, distinguiéndose por el color del marcador, naranja para monumentos históricos y verde para naturales, según se muestra a continuación:

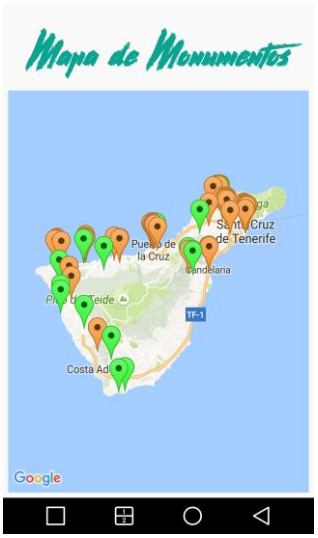


Figura 5.11. Vista del mapa de monumentos.

Al seleccionar un marcador, se muestra el nombre del monumento y, en la esquina inferior derecha, dos botones. El primer botón ejecuta el servicio *Cómo llegar* de Google Maps, mientras que el segundo muestra, también en Google Maps, la localización seleccionada. Además, si se selecciona el nombre del monumento, se abre la actividad que muestra el detalle del mismo.

Figura 5.12. Mapa de monumentos.

- **Preferencias:** Lanza la actividad *ConfigurationActivity*. En esta actividad se recogen y almacenan las preferencias, en cuanto a las características de los monumentos, del usuario. Para esta implementación se ha utilizado la interfaz *SharedPreferences* (Android) de Android. Estas preferencias podrán ser modificadas en el momento

en el que el usuario lo desee. Las siguientes imágenes muestran las vistas de esta opción del menú:

Figura 5.13. Vistas de las preferencias

Los distintos iconos representan diferentes características que pueden tener los monumentos. De izquierda a derecha, representan las siguientes categorías: Arte, mar y costas, historia, miradores, montaña y religión. El icono se muestra en gris, si la preferencia se encuentra desactivada y en color si se selecciona como de interés.

En los siguientes fragmentos de código, se puede observar cómo se utiliza la memoria interna del dispositivo para almacenar las preferencias del usuario. Además, el controlador del evento *onClick*, asociado a la imagen de cada preferencia, consultará el estado actual, *enable* o *disable*, de la preferencia y la actualizará, cada vez que se haga click en dicha imagen.

```
final SharedPreferences.Editor editor = getSharedPreferences("BeTouristAppUserPreference", MODE_PRIVATE).edit();
final SharedPreferences prefs = getSharedPreferences("BeTouristAppUserPreference", MODE_PRIVATE);

String name = prefs.getString(ARTINTEREST, DISABLE);//"disable" is the default value.
setImage(name, artInterestImg, R.drawable.interes_arte_disable, R.drawable.interes_arte);

artInterestImg.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String name = prefs.getString(ARTINTEREST, DISABLE);//"disable" is the default value.
        changeImage(name, ARTINTEREST, editor, artInterestImg, R.drawable.interes_arte_disable, R.drawable.interes_arte);
    }
});
```

Figura 5.14. Establecimiento de las preferencias

```
public void setImage(String name, ImageView image, int imageNumber, int image2Number){
    if (!name.equals(ENABLE)){
        image.setImageResource(imageNumber);
    }
    else{
        image.setImageResource(image2Number);
    }
}
```

Figura 5.15. Función que establece la imagen, según la preferencia.

Al observar la vista del prototipo, se consideró que la información útil del perfil era, únicamente, las preferencias del usuario. Por otro lado, se prefirió que el idioma de la aplicación se estableciera, automáticamente, según el idioma predefinido del dispositivo. De esta manera, se consigue otorgar limpieza, usabilidad y sencillez a la interfaz. Por todo esto, se decidió que la opción "Configuración", pasase a ser "Preferencias".

- **Acerca de:** Lanza la actividad *AboutActivity*. En esta actividad se muestra una vista con información relativa a la aplicación y a la finalidad de la misma. Además, se proporciona un correo de contacto.

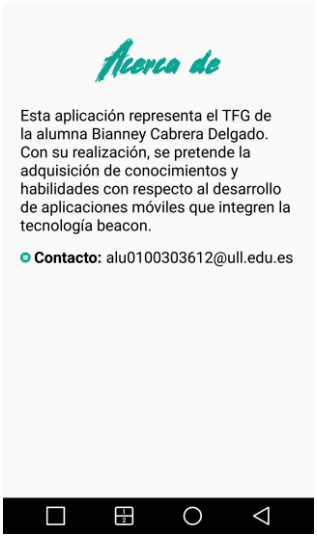


Figura 5.16. Vista del Acerca de

Se decidió eliminar del menú las opciones "Históricos" y "naturales" que aparecían en las vistas del primer prototipo, ya que se consideró que era una redundancia.

5.2.5 Integración de la tecnología *beacon*

Para poder integrar la tecnología *beacon* a la aplicación, en primer lugar, se debe cumplir con los siguientes requisitos:

- Estar en disposición de una cuenta de acceso al panel web de Kontakt.
- Android 2.3 (API 9) o superior.
- Java 6 o superior.

Una vez cumplidos estos requisitos, se tiene que preparar el proyecto para poder utilizar la SDK de Kontakt, para ello se siguieron los siguientes pasos:

- **Paso 1:** El primer paso es añadir la dependencia a la SDK de Kontakt al proyecto. Gracias a Gradle este paso es tan sencillo como añadir la línea al módulo `dependencies` del fichero `app/build.gradle.`

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    provided 'com.googlecode.json-simple:json-simple:1.1.1'
    compile 'com.android.support:appcompat-v7:23.4.0'
    compile 'com.android.support:support-v4:23.4.0'
    compile 'com.android.support:design:23.4.0'
    compile 'com.kontaktio:sdk:3.0.0'
    compile 'com.google.android.gms:play-services:9.2.1'
}
```

Figura 5.17. Añadida dependencia SDK Kontakt

- **Paso 2:** En segundo lugar, se debe añadir los siguientes permisos, necesarios para interactuar con los *beacons*:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 5.18. Permisos necesarios para interactuar con los *beacons*

  - BLUETOOTH: Es necesario para buscar y monitorizar los *beacons.*
  - BLUETOOTH_ADMIN: Es necesario para realizar la conexión con los dispositivos *beacon.*
  - INTERNET & ACCESS NETWORK STATE: Es necesario para conectarse a Kontakt Cloud.
  - ACCESS_COARSE_LOCATION y ACCESS_FINE_LOCATION: Son requeridos en Android 6.0, ya que, en esta versión, realizar una búsqueda con BLE requiere permisos de localización.

Debido a que en Android 6.0, Marshmallow, los permisos se aceptan en tiempo de ejecución, se ha tenido que seguir una guía (Android) en la

cual se especifica cómo deben ser solicitados. En este caso, estos permisos deben ser pedidos nada más ejecutarse la aplicación, ya que son necesarios desde el primer momento. En la siguiente imagen se muestra como se solicita, en tiempo de ejecución, el acceso al usuario:

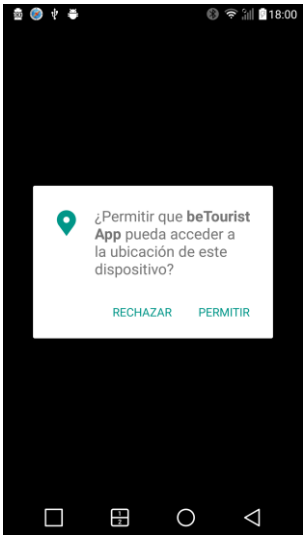


Figura 5.19. Solicitud de permisos al iniciar, por primera vez, la aplicación.

A continuación, se muestra la función que solicita los permisos necesarios:

```
@TargetApi(Build.VERSION_CODES.M)
private void requestPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_COARSE_LOCATION)) {
        showPermissionsSnackBar();
    } else {
        requestPermissions(new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                MY_WRITE_EXTERNAL_STORAGE);
    }
}
```

Figura 5.20. Función *requestPermission()*. Solicita los permisos necesarios.

- **Paso 3:** Activar el servicio SDK *ProximityService* en el manifest.

```
<service
    android:name="com.kontakt.sdk.android.ble.service.ProximityService"
    android:exported="false" />
```

Figura 5.21. Activar el *ProximityService* en el Manifest.

- **Paso 4:** Añadir las siguientes reglas al archivo `proguard-rules.pro`:

```
-dontwarn okio.**
-dontwarn retrofit2.**
-keep class retrofit2.** { *; }
-keepattributes Signature
-keepattributes Exceptions

-keepclasseswithmembers class * {
    @retrofit2.http.* <methods>;
}

-keep class com.kontakt.sdk.** { *; }
-keep interface com.kontakt.sdk.** { *; }
```

Figura 5.22. Contenido del fichero `proguard-rules.pro`.

- **Paso 5:** Proporcionar la contraseña de la API de Kontakt. Esta contraseña es enviada por el equipo de Kontakt, una vez se ha adquirido su producto. Se debe establecer de la siguiente manera:

```
KontaktSDK.initialize(String.valueOf(R.string.kontakt_io_api_key));
```

Figura 5.23. Establecimiento de la contraseña de la API, en el *MainActivity*.

Una vez realizado cada uno de los pasos anteriores, se ha procedido a implementar el establecimiento de la conexión con uno de los *beacons*. Para ello, en primer lugar se ha activado el rastreo de dispositivos, en la Actividad principal de la aplicación.

Como se puede observar en las siguientes imágenes, se ha utilizado mensajes tipo *Toast* para informar el estado de la búsqueda de *beacons*.

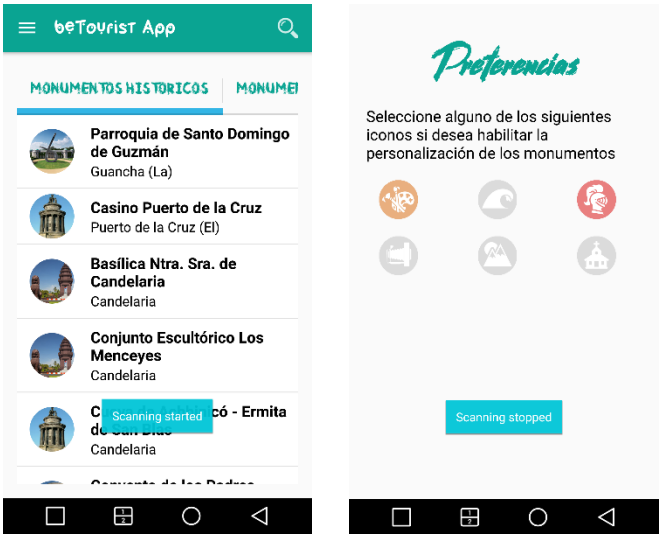


Figura 5.24. *Toasts del* estado de la búsqueda

Una vez se encuentra un dispositivo *beacon*, se intenta realizar una conexión con él. En el siguiente fragmento de código, se muestra la función *createEddystoneListener*(). En dicha función se comprueba si dicho *beacon* detectado pertenece a la lista de *beacons* que la aplicación debe reconocer. Si

el *beacon* es un dispositivo "conocido", se intenta establecer una conexión con él, ejecutándose la actividad *ConnectActivity.*

```
private EddystoneListener createEddystoneListener() {
    return new SimpleEddystoneListener() {
        @Override
        public void onEddystoneDiscovered(IEddystoneDevice eddystone, IEddystoneNamespace namespace) {
            Log.d("MyBeacon", "Eddystone discovered: " + eddystone.getName());
            for (int i = 0; i < MyBeacons.getInstance().getList().size(); i++){
                if ((eddystone.getName().equals(MyBeacons.getInstance().getList().get(i).getName())) &&
                    (!MyBeacons.getInstance().getList().get(i).getViewed())){
                        Intent intent = new Intent(MainActivity.this, ConnectActivity.class);
                        intent.putExtra("beacon", eddystone);
                        startActivity(intent);
                }
            }
        }
```

Figura 5.25. Función createEddystoneListener().

Para realizar dicha conexión, es necesario conocer previamente una contraseña, única para cada *beacon.* Si la conexión se establece, se llevan a cabo los siguientes pasos:

- En primer lugar, se recoge la información del *beacon.*
- Se relaciona el dispositivo con el monumento al que corresponde.
- Se consulta las preferencias que el usuario ha seleccionado previamente.
- Se muestra un *Dialog* con información personalizada del monumento, según las preferencias que el usuario ha seleccionado.
- Se cierra la conexión con el faro, para evitar el sobregasto de energía.

A continuación, se muestran imágenes en las que se puede observar un ejemplo llevado a cabo. En la primera, se ve como el usuario escoge, como preferencias, las categorías de arte e historia. En la segunda, el *Dialog* que aparece al conectarse con un *beacon,* que se corresponde con la Basílica de Ntra. Sra. De Candelaria, como se dijo anteriormente, las imágenes de los monumentos no son reales.

Figura 5.26. Imágenes del ejemplo del Dialog mostrado al conectarse a un *beacon.*

En los siguientes fragmentos de código, se muestran funciones relevantes en el proceso de establecimiento de una conexión con un *beacon.* En la primera imagen, se muestra la función que inicia el proceso de conexión. Mientras que la segunda, muestra la función que ejecuta las distintas acciones que se quieren realizar cuando la autenticación del *beacon* es satisfactoria.

```
private void connect(RemoteBluetoothDevice remoteBluetoothDevice) {
    kontaktDeviceConnection = new KontaktDeviceConnection(this, remoteBluetoothDevice, connectionListener);
    kontaktDeviceConnection.connect();
}
```

Figura 5.27. Función *connect().*

```
@Override
public void onAuthenticationSuccess(RemoteBluetoothDevice.Characteristics characteristics) {
    MyBeacons.getInstance().getList().get(position).setViewed(true);
    showDialog();
    printCharacteristic(characteristics);
    disconnect();
}
```

Figura 5.28. Función *onAuthenticationSuccess().*

Capítulo 6. Problemas o dificultades encontradas

En este capítulo se detallarán algunos de los problemas más importantes que se han encontrado a la hora de desarrollar el proyecto.

6.1 Tecnología *beacon*

Sin lugar a dudas, la primera dificultad que se encontró fue la falta de conocimientos que se tenía de la tecnología *beacon.* A esto se le añadió la poca información disponible, debido a la novedad de dicha tecnología. En un principio, se partió de ideas que, más tarde se supo, eran erróneas, por lo que el entendimiento del funcionamiento de los *beacons* conllevó un mayor esfuerzo del esperado.

6.2 Android

Una dificultad añadida fue la poca experiencia que se tenía en desarrollo para dispositivos móviles, en este caso Android. Además de esto, la poca experiencia previa que se tenía, era para otras versiones de Android más antiguas, lo que conllevó la necesidad de una puesta al día. Esto nos hace recordar el continuo reciclado y actualización que exige nuestra profesión.

6.3 Android Studio

Se ha tenido algunos problemas al utilizar Android Studio, entre ellos se pueden encontrar los siguientes:

- En primer lugar, la gran cantidad de recursos que consume. Al ser un entorno tan potente, consume muchos recursos del ordenador, lo que ocasionó lentitud, y algunos bloqueos en la máquina.

- Numerosas actualizaciones. Algunos plug-ins han dejado de funcionar por no actualizarse, sin ningún tipo de aviso. Por ejemplo, un plug-in para incluir imágenes *.svg* como archivos *.xml,* dejó de funcionar. Al intentar añadir la imagen, se mostraba un error de "*imagen no válida".* Se buscó en internet y la solución fue realizar una actualización completa de Android Studio.
- En ocasiones, tarda bastante en realizar la construcción del proyecto. Esto ha ocasionado que, a lo largo de todo el desarrollo, la suma del tiempo que se ha perdido esperando a la compilación sea elevada.

Estas dificultades han ocasionado retrasos a la hora de llevar a cabo la ejecución del proyecto.

Capítulo 7. Herramientas Software

En este capítulo se presentarán las distintas herramientas software que se han utilizado durante la elaboración de este trabajo.

7.1 Android Studio

Android Studio (Android) ha sido el entorno de desarrollo elegido para desarrollar la aplicación móvil, ya que es el IDE oficial para el desarrollo de aplicaciones para Android. Está basado en IntelliJ IDEA (Jet Brains). Además de todo lo que ofrece Intellig IDEA, Android Studio aumenta su funcionalidad, añadiendo utilidades que ayudan a aumentar la productividad de los desarrolladores, entre ellas se pueden encontrar las siguientes:

- Un sistema de compilación flexible, basado en Gradle (Gradle). Además, Android Studio permite configurar algunos aspectos de la compilación (Android). Esta funcionalidad ha sido de mucha utilidad a la hora de añadir las dependencias a las distintas librerías que han sido necesarias, además de para la construcción del proyecto.
- Un emulador rápido. El emulador no ha sido muy utilizado, ya que la aplicación hace necesario el uso del *bluetooth* para realizar la detección y recepción de la señal de los *beacons*.
- *Instant Run,* una funcionalidad que permite aplicar cambios mientras la aplicación se ejecuta, sin tener la necesidad de generar una nueva APK.
- Integración de GitHub.
- Herramientas de depuración.

7.2 Eclipse

Para desarrollar la aplicación servidor, se ha utilizado el IDE para Java de Eclipse (Eclipse). Este IDE es el más utilizado para desarrollar aplicaciones en

Java. Ofrece multitud de herramientas para facilitar el desarrollo en este lenguaje, entre otras, se puede encontrar las siguientes:

- Integración con Git.
- Integración con Maven.
- Herramientas de desarrollo para Java.
- Posibilidad de incluir multitud de plug-ins.

7.3 Gradle

Como se dijo en el apartado 5.1, la herramienta Gradle es utilizada por Android Studio para realizar la compilación del proyecto. Gradle es una herramienta de construcción, basada en el concepto de Apache Ant y Apache Maven.

El uso de este tipo de herramientas ofrece una multitud de ventajas durante el desarrollo de un proyecto, entre ellas se pueden encontrar las siguientes:

- Facilitar la construcción, compilación, prueba y ejecución del software, permitiendo un ahorro en tiempo y esfuerzos.
- Incrementar la eficiencia debido a que únicamente son reconstruidos aquellos archivos, cuyos ficheros fuente han sido modificados.
- Facilidad a la hora de establecer las dependencias entre los ficheros fuente y bibliotecas externas.

7.4 GitHub

GitHub es una plataforma de desarrollo colaborativo, que aloja proyectos software, utilizando el sistema de control de versiones Git (Git), diseñado por Linus Torvalds. GitHub utiliza el *framework* Ruby on Rails (Ruby on Rails).

Si se utiliza una cuenta gratuita de GitHub, el código es almacenado de manera pública, para hacerlo de forma privada, sería necesario utilizar una cuenta de pago.

Durante la realización de este proyecto, se ha utilizado esta plataforma para llevar un control de la evolución del mismo. Además, ha otorgado confianza en el desarrollo, al ofrecer la posibilidad de recuperar las distintas versiones del código ante la ocurrencia de cualquier tipo de imprevisto.

Gracias al uso de esta herramienta, se ha conseguido ampliar los conocimientos en Git, obtenidos en las asignaturas "Lenguajes y Paradigmas de la Programación" y "Laboratorio de Desarrollo y Herramientas". Además se ha obtenido un cierto grado de destreza a la hora de emplear las funcionalidades que esta ofrece.

7.5 SonarQube

SonarQube (SonarQube) es una plataforma abierta para gestionar la calidad del código. Integra distintas herramientas que cubren la medición de los 7 ejes de la calidad del software, que se muestran a continuación:

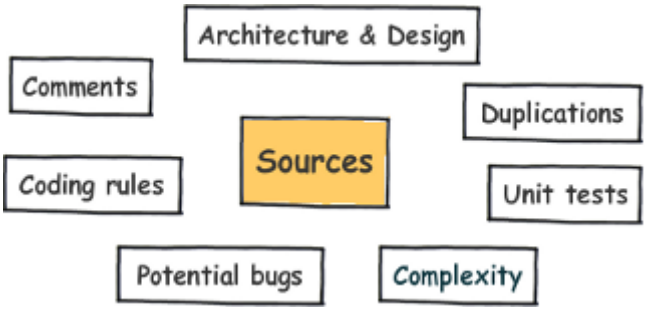


Figura 7.1. Ejes de la calidad del software (SonarQube).

Para configurar el sonarQube para este proyecto, se generó el archivo sonar-project.properties de la siguiente manera:

```
# Required metadata
sonar.projectKey=TFG
sonar.projectName=TFG
sonar.projectVersion=1.0

# Comma-separated paths to directories with sources (required)
sonar.sources=app/src

#Language
sonar.language=java

# Encoding of the source files
sonar.sourceEncoding=UTF-8

sonar.scm.disabled=true
```

Figura 7.2. Contenido del fichero sonar-project.properties.

Una vez finalizado el desarrollo del proyecto, se han obtenido los resultados que se pueden observar en la siguiente imagen:

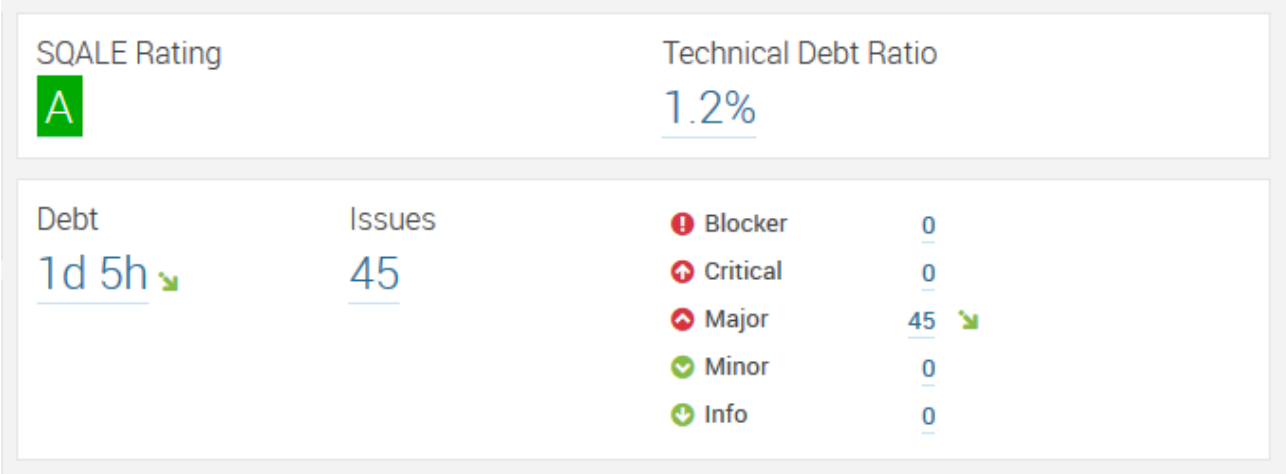


Figura 7.3. Resultados análisis con SonarQube.

Se puede apreciar que existe una deuda técnica d 1 día y 5 horas. Esta deuda técnica se corresponde con el sobreesfuerzo que conllevaría continuar con el proyecto, sin resolver o corregir las 45 evidencias mayores que se han encontrado. Las evidencias mayores son defectos de calidad que podrían tener un alto impacto en la productividad.

Gracias al uso de esta herramienta, se han logrado resolver 256 evidencias, por lo tanto, se ha conseguido un software de mayor calidad.

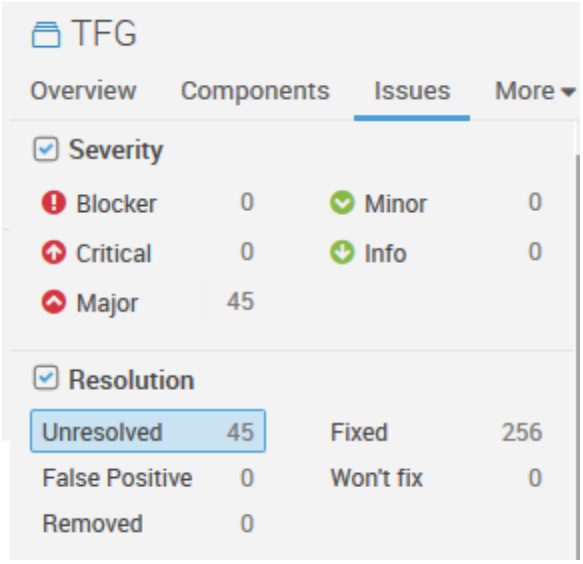


Figura 7.4. Evidencias resueltas.

Capítulo 8. Conclusiones y líneas futuras

En este capítulo se presentarán las conclusiones obtenidas una vez se ha realizado el trabajo. Seguidamente, se detallarán una serie de posibles líneas de trabajo para un futuro.

8.1 Conclusiones

Como resultado del trabajo realizado, tanto de investigación como de desarrollo, se puede concluir que la tecnología *beacon*, es una tecnología que aún tiene mucho potencial de explotación. Es una tecnología nueva, aplicable a una multitud de campos que abarcan diferentes ámbitos de la vida.

Por otro lado, a nivel de implementación, se puede concluir que se ha desarrollado, con éxito, una aplicación móvil capaz de utilizar la tecnología *beacon.* Se han conseguido los objetivos marcados y cumplido los requisitos planteados previamente.

8.2 Líneas futuras

Como trabajo futuro se podrían añadir distintas características a la aplicación, que aporten funcionalidad y atractivo. Entre otras cosas, se podría incluir:

- **Incluir un módulo de realidad aumentada**: Esta opción se estuvo investigando en el trabajo. Se barajó la posibilidad de utilizar la librería de realidad aumentada ARToolkit (ARToolkit) a la hora de mostrar información de los monumentos. De esta manera, se hace más atractiva la experiencia al usuario. Por falta de tiempo esta funcionalidad no pudo ser desarrollada.
- Mejorar la IA asociada a las preferencias del usuario.

- **Incluir gamificación a la aplicación:** Incluir un sistema de gamificación que aporte atractivo a la app.
- Mejorar la información de cada monumento.
- Incluir imágenes reales de los monumentos.

Capítulo 9.
Summary and Conclusions

This chapter will present the conclusions obtained once the work was done.

9.1 Conclusions

As a result of work done, both research and development, it can be concluded that the *beacon* technology, is a technology that still has much exploitation potential. It is a new technology applicable to a wide range of fields covering different areas of life.

On the other hand, at implementation level, it can be concluded that it has successfully developed a mobile application, capable of using beacon technology. They have achieved the goals marked and fulfilled the requirements previously raised.

Capítulo 10. Presupuesto

En este capítulo se hará una estimación del coste que conllevaría la puesta en marcha de este proyecto, tanto en recursos como en horas de trabajo invertidas. Se separará en dos apartados, el coste del proyecto en el estado actual

10.1 Presupuesto del trabajo realizado

En esta sección se presenta el presupuesto del trabajo elaborado y los recursos que se han invertido hasta el momento.

- **Tiempo invertido:** Se estima que han podido dedicarse 300 horas al desarrollo de este trabajo de fin de grado. Este número de horas se corresponde con el exigido en la guía docente de esta asignatura. A continuación, se muestra un desglose del tiempo invertido:

Tareas	Horas	Precio por hora	Total
Análisis	70	20	1400
Diseño	30	20	600
Desarrollo	170	15	2550
Pruebas	30	15	450

Tabla 1. Presupuesto del tiempo invertido.

- **Equipo y dispositivos necesarios:** Debido a que no se disponía de ningún dispositivo *beacon,* ha sido necesario adquirir el siguiente hardware:

Dispositivo	Descripción	Precio
Beacons x3	Dispositivos *beacon* de Kontakt	80

Tabla 2. Presupuesto de dispositivos.

- **Herramientas:**

Software	Descripción	Precio

SDK de Kontakt	Librerías de Kontakt	Incluido en los *beacons*

Tabla 3. Presupuesto de herramientas.

Apéndice A. Especificación de casos de uso

A.1. Especificación de casos de uso

- ### Aplicación servidor

Nombre	Iniciar la aplicación
Descripción	Este caso de uso describe el conjunto de acciones necesarias para iniciar la aplicación.
Actores	Usuario
Precondiciones	Servidor encendido
Flujo normal	1. Click en el icono referente a la aplicación. 2. Fin del flujo.
Flujo alternativo	
Notas	

  - ### Datos

Nombre	Descargar datos
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo por el sistema para descargar los datos del sitio web.
Actores	
Precondiciones	Disponer de conexión a internet.
Flujo normal	1. Conectarse a la dirección web pertinente. 2. Descargar los dos ficheros correspondientes a los monumentos. 3. Fin del flujo.
Flujo alternativo	1. Conectarse a la dirección web pertinente. 2. Error al intentar descargar los datos. Mostrar mensaje de error. 3. Fin del flujo.
Notas	

Nombre	*Parsear* datos
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo por el sistema para realizar un *parseo* de los datos descargados.
Actores	
Precondiciones	
Flujo normal	1. Leer línea del fichero. 2. Transformar los datos a los tipos necesario. 3. Almacenar los datos útiles. 4. Fin de flujo.
Flujo	

alternativo	
Notas	

Nombre	Guardar datos
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo para por el sistema para guardar los datos en un nuevo fichero.
Actores	Usuario
Precondiciones	
Flujo normal	1. Crear fichero. 2. Abrir fichero. 3. Escribir datos en el fichero. 4. Cerrar fichero. 5. Fin del flujo.
Flujo alternativo	
Notas	

▪ Aplicación móvil

○ Acceso

Nombre	Acceder a la app
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo para acceder a la aplicación móvil
Actores	Usuario
Precondiciones	
Flujo normal	1. El usuario accede a la aplicación móvil seleccionando en el dispositivo móvil el icono correspondiente a la aplicación. 2. Mientras carga la aplicación se mostrará la imagen de inicio. Una vez cargada se mostrará, la interfaz que se muestra en la imagen XXXX. 3. Fin del flujo.
Flujo alternativo	1. El usuario accede a la aplicación móvil seleccionando en el dispositivo móvil el icono correspondiente a la aplicación. 2. Ocurre un error a la hora de ejecutarse y se detiene la app. 3. Se muestra un mensaje de error. 3. Fin del flujo.
Notas	

○ Carga de datos

Nombre	Cargar datos
Descripción	Este caso de uso describe la secuencia de acciones llevadas a cabo por el sistema para cargar los datos necesarios.
Actores	
Precondiciones	1. El servidor debe estar arrancado. 2. El fichero debe estar creado correctamente en el servidor. 3. Debe disponerse de conexión a internet.
Flujo normal	1. La aplicación se conecta al servidor. 2. Acceder al fichero con los datos necesarios.

	3. Leer datos del fichero. 4. Almacenar los datos en local. 5. Fin del flujo.
Flujo alternativo	1. La aplicación intenta conectarse al servidor. 2. No se consigue descargar correctamente los datos. 3. Se muestra un mensaje de error. 4. Fin del flujo.
Notas	

- ## Lista de monumentos

Nombre	Mostrar lista
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo para mostrar la lista de los monumentos.
Actores	Usuario
Precondiciones	Los datos deben estar correctamente cargados en el sistema.
Flujo normal	1. Abrir el menú lateral (seleccionando el icono de menú, situado en la parte superior izquierda o arrastrando la parte izquierda de la pantalla hacia la derecha). 2. Seleccionar alguno de los iconos de monumentos. 3. Fin del flujo.
Flujo alternativo	Si ya se está mostrando alguna lista de monumentos: 1. Arrastrar la pantalla a derecha o izquierda. 2. Fin del flujo.
Notas	

Nombre	Filtrar lista
Descripción	Este caso de uso se describe las acciones necesarias para aplicar un filtro a las listas de monumentos.
Actores	Usuario
Precondiciones	Se debe estar en alguna de las pantallas de las listas de monumentos.
Flujo normal	1. Se selecciona el icono de filtro, situado arriba a la derecha. 2. Se elige el criterio por el que se quiere filtrar. 3. El sistema genera una lista de monumentos que concuerde con el filtro aplicado. 4. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Realizar búsqueda
Descripción	Permite al usuario realizar una búsqueda en la lista de los monumentos.
Actores	Usuario
Precondiciones	Se debe estar en alguna de las pantallas de las listas de monumentos.
Flujo normal	1. Se selecciona el icono de la lupa, situado en la parte superior derecha de la pantalla. 2. Se introduce la/s palabra/s a buscar. 3. Se confirma la búsqueda. 4. El sistema genera una lista de monumentos que concuerde con la búsqueda realizada. 5. Fin del flujo.
Flujo alternativo	1. Se selecciona el icono de la lupa, situado en la parte superior derecha de la pantalla. 2. Se introduce la/s palabra/s a buscar. 3. Se confirma la búsqueda.

	4. La búsqueda realizada no genera ningún resultado. El sistema genera un mensaje. 5. Fin del flujo.
Notas	

◦ Descripción del monumento

Nombre	Ver descripción
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo para acceder a la descripción de un monumento.
Actores	Usuario
Precondiciones	Se debe estar en alguna de las pantallas de las listas de monumentos.
Flujo normal	1. Se selecciona el monumento del cual se quiere ver su descripción. 2. El sistema genera la vista correspondiente con la descripción del monumento. 3. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Mostrar cómo llegar
Descripción	Este caso de uso describe las actividades realizadas para mostrar "cómo llegar" a un monumento.
Actores	Usuario
Precondiciones	Se debe estar en la vista de descripción del monumento. Se debe tener activado en el dispositivo el servicio de ubicación y tener acceso a internet.
Flujo normal	1. Se selecciona el botón *cómo llegar*. 2. El sistema calcula e indica la ruta al usuario, mediante la API de Google. 3. Fin del flujo
Flujo alternativo	
Notas	

◦ Configuración

Nombre	Acceder al panel de configuración
Descripción	Este caso de uso muestra el conjunto de acciones necesarias para acceder al panel de configuración.
Actores	Usuario
Precondiciones	
Flujo normal	1. Abrir el menú lateral (seleccionando el icono de menú, situado en la parte superior izquierda o arrastrando la parte izquierda de la pantalla hacia la derecha). 2. Seleccionar el icono de configuración. 3. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Establecer perfil
Descripción	Describe el conjunto de actividades realizadas para establecer el perfil de usuario.
Actores	Usuario

Precondiciones	Estar iniciando por primera vez la aplicación.
Flujo normal	1. Se iniciará la vista de establecimiento del perfil. 2. Rellenar los campos necesarios. 3. Confirmar. 4. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Ver perfil
Descripción	Establece las acciones que se realizan para visualizar el perfil de usuario.
Actores	Usuario
Precondiciones	
Flujo normal	1. Abrir el menú lateral (seleccionando el icono de menú, situado en la parte superior izquierda o arrastrando la parte izquierda de la pantalla hacia la derecha). 2. Seleccionar el icono de configuración. 3. Seleccionar el icono de perfil. 4. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Modificar perfil
Descripción	Establece las acciones que se realizan para modificar el perfil de usuario.
Actores	Usuario
Precondiciones	
Flujo normal	1. Abrir el menú lateral (seleccionando el icono de menú, situado en la parte superior izquierda o arrastrando la parte izquierda de la pantalla hacia la derecha). 2. Seleccionar el icono de configuración. 3. Seleccionar el icono de perfil. 4. Seleccionar el botón modificar. 5. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Establecer idioma
Descripción	Recoge las acciones que se realizan para establecer el idioma de la aplicación.
Actores	Usuario
Precondiciones	Estar iniciando por primera vez la aplicación.
Flujo normal	1. Se iniciará la vista de selección de idioma. 2. Elegir el idioma preferido. 3. Confirmar.
Flujo alternativo	
Notas	

Nombre	Modificar idioma

Descripción	Recoge las acciones que se realizan para modificar el idioma de la aplicación.
Actores	Usuario
Precondiciones	
Flujo normal	1. Abrir el menú lateral (seleccionando el icono de menú, situado en la parte superior izquierda o arrastrando la parte izquierda de la pantalla hacia la derecha). 2. Seleccionar el icono de configuración. 3. Seleccionar el botón de idioma. 4. Elegir el idioma preferido. 5. Confirmar.
Flujo alternativo	
Notas	

○ Mensajes

Nombre	Recibir mensaje
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo por el sistema a la hora de recibir los mensajes.
Actores	
Precondiciones	Tener activado el *bluetooth* en el dispositivo móvil y estar al alcance de algún *iBeacon.*
Flujo normal	1. Recibir el mensaje. 2. Almacenar el mensaje. 3. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Procesar mensaje
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo por el sistema para procesar los mensajes recibidos.
Actores	
Precondiciones	
Flujo normal	1. Consultar perfil. 2. Establecer el mensaje que se desea mostrar, utilizando cierto grado de inteligencia artificial, según el perfil del usuario. 3. Almacenar mensaje. 4. Fin del flujo.
Flujo alternativo	
Notas	

Nombre	Mostrar mensaje
Descripción	Este caso de uso describe el conjunto de acciones llevadas a cabo por el sistema para mostrar al usuario un mensaje recibido.
Actores	
Precondiciones	
Flujo normal	
Flujo alternativo	

Notas	

Bibliografía

[1] *9to5mac*. s.f. <http://9to5mac.com/2013/12/04/imaginative-use-of-ibeacon-gives-barpatrons-free-access-to-newsstand-magazines/>.

[2] *Adobe*. s.f. <http://www.adobe.com/es/products/photoshop.html>.

[3] *Android*. s.f. <https://developer.android.com/studio/index.html>.

[4] *Android*. s.f. <https://developer.android.com/studio/build/index.html#build-config>.

[5] *Android*. s.f. <https://developer.android.com/training/permissions/requesting.html>.

[6] *Android*. s.f. <https://developer.android.com/reference/android/content/SharedPreferences.html>.

[7] *ARToolkit*. s.f. <www.artoolkit.org/dist/artoolkit5/5.3/ARToolKit5-bin-5.3.2-Android.zip>.

[8] *Beaconstac*. s.f. <http://blog.beaconstac.com/2016/01/ibeacon-vs-eddystone/>.

[9] *Cnet*. s.f. <http://www.cnet.com/videos/mlb-tests-apples-ibeacon-at-citi-field/>.

[10] «Easy Context.» s.f. <https://www.easycontext.com/wp-content/uploads/2016/06/ebook-easy-context-beacons.pdf>.

[11] *Easy Context*. s.f. <https://www.easycontext.com/beacons-en-espana-tecnologia-proximidad/#.>.

[12] *Eclipse*. s.f. <https://eclipse.org/>.

[13] *Estimote*. s.f. <http://estimote.com/#jump-to-products>.

[14] *Estimote*. s.f. <http://estimote.com/#jump-to-products>.

[15] Gast, Matthew S. *Building Applications with iBeacon* . O'Reilly Media, s.f.

[16] *Git*. s.f. <https://git-scm.com/>.

[17] *GitHub*. s.f. <https://github.com/SonarQubeCommunity/sonar-android>.

[18] *Google*. s.f. <https://www.google.es/maps/>.

[19] *Google Play*. s.f. <https://play.google.com/store/apps/details?id=com.kontakt.app&hl=es>.

[20] *Gradle*. s.f. <https://gradle.org/>.

[21] *Gradle*. s.f. <https://gradle.org/>.

[22] *ibeacons.net*. s.f. <http://ibeacons.net.au/uk-cinemas-odeon-cinema-to-test-ibeacons/>.

[23] *IOSandi*. s.f. <http://iosandi.blogspot.com.es/2015/12/in-addition-to-my-previous-post-i-have.html>.

[24] *Jet Brains*. s.f. <https://www.jetbrains.com/idea/>.

[25] *Kontakt*. s.f. <https://support.kontakt.io/hc/en-gb/articles/201628101-What-is-a-beacon->.

[26] *Kontakt.io*. s.f. <https://kontakt.io/beacon-basics/what-is-proximity-and-iot/>.

[27] *Kontakt.io*. s.f. <https://kontakt.io/beacon-basics/what-is-a-beacon/>.

[28] *Kontakt.io*. s.f. <https://kontakt.io/beacon-basics/ibeacon-and-eddystone/>.

[29] *Kontakt.io*. s.f. <https://store.kontakt.io/our-products/27-bluetooth-beacon.html>.

[30] *Kontakt.io*. s.f. <https://support.kontakt.io/hc/en-gb/articles/201628101-What-is-a-beacon->.

[31] *Kontakt.io*. s.f. <https://store.kontakt.io/our-products/27-bluetooth-beacon.html>.

[32] *Kontakt.io*. s.f. <https://panel.kontakt.io>.

[33] *Open Data Canarias*. s.f. <http://www.opendatacanarias.es/datos/dataset/tdt-monumentos-historicos-de-tenerife>.

[34] *Open Data Canarias*. s.f. <http://opendatacanarias.es/>.

[35] *Open Data Canarias*. s.f. <http://www.opendatacanarias.es/datos/dataset/tdt-monumentos-historicos-de-tenerife>.

[36] *Open Data Canarias*. s.f. <http://www.opendatacanarias.es/datos/dataset/tdt-monumentos-naturales-de-tenerife>.

[37] *Pencil*. s.f. <http://pencil.evolus.vn/>.

[38] *Ruby on Rails*. s.f. <http://rubyonrails.org/>.

[39] *Slide Share*. s.f. <http://www.slideshare.net/ChristianMelchior/trifork-ibeacon-demo-lunch-talk>.

[40] *SonarQube*. s.f. <http://www.sonarqube.org/>.

[41] *Techcrunch*. s.f. <http://techcrunch.com/2014/03/28/behere-lets-teachers-take-attendance-using-ibeacon/>.

[42] *Using beacons*. s.f. <http://www.usingbeacons.com/wp-content/uploads/2015/09/what_makes_eddystone_different.png>.

[43] *Wikipedia*. s.f. <https://en.wikipedia.org/wiki/Kevin_Ashton>.

[44] *Wikipedia*. s.f. <https://es.wikipedia.org/wiki/IBeacon#Dispositivos_compatibles>.

[45] *Wikipedia*. s.f. <https://en.wikipedia.org/wiki/UL_94>.

[46] *Wikipedia*. s.f. <https://es.wikipedia.org/wiki/Grado_de_protecci%C3%B3n_IP>.

[47] *Xataka*. s.f. <http://www.xataka.com/wearables/tzukuri-las-gafas-con-ibeacon-que-nunca-perderas>.