UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA PARA LA MEJORA DE LA PRIVACIDAD Y LA SEGURIDAD EN LA FIDELIZACIÓN DE CLIENTES EN EL SECTOR RETAIL

GRADO EN INGENIERÍA TELEMÁTICA

JESSICA MENDOZA RUIZ
MÁLAGA, 2017

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA PARA LA MEJORA DE LA PRIVACIDAD Y LA SEGURIDAD EN LA FIDELIZACIÓN DE CLIENTES EN EL SECTOR RETAIL

GRADO EN INGENIERÍA
TELEMÁTICA

JESSICA MENDOZA RUIZ
MÁLAGA, 2017

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

Sistema para la mejora de la privacidad y la seguridad en la fidelización de clientes en el sector *retail*

Autor: Jessica Mendoza Ruiz

Tutor: Isaac Agudo Ruiz

Departamento: Lenguajes y Ciencias de la Computación

Titulación: Grado en Ingeniería Telemática

Palabras clave: seguridad, fidelización, Swift, centro comercial, baliza, Firebase, *retail,* privacidad, Bluetooth Low Energy

Resumen

En el presente TFG se ha desarrollado un sistema con el fin de adaptar la fidelización de clientes a las nuevas tecnologías y subsanar los problemas que presentan los actuales sistemas de fidelización.

El sistema se apoya en el uso de balizas BLE para realizar la localización de los usuarios. Debido a que estas balizas son fáciles de clonar, se ha utilizado un segundo tipo de baliza BLE, a la que llamamos baliza de seguridad. Esta baliza transmite un código pseudoaleatorio usando una clave secreta y el identificador de la baliza de posición a la que está asociada, lo que impide su clonación y permite localizar de forma segura a los usuarios.

La privacidad de los usuarios queda garantizada mediante el uso de pseudónimos y de un selector de tiendas, de manera que solo las tiendas que el usuario decida recibirán sus datos de localización.

El sistema funciona en segundo plano y utiliza notificaciones *Push* para el envío de ofertas personalizadas en función de la localización de los usuarios, una vez esta haya sido verificada de forma segura.

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

Security and Privacy enhanced system for customer loyalty in retail

Author: Jessica Mendoza Ruiz

Supervisor: Isaac Agudo Ruiz

Department: Computer Science

Degree: Degree in Telematics Engineering

Keywords: security, loyalty, Swift, shopping centre, beacon, Firebase, retail, privacy, Bluetooth Low Energy

Abstract

The target of the present project has been to adapt current loyalty systems used in retail scenarios to the challenges presented by the new technologies existing nowadays.

The system is based on the use of BLE beacons to perform the localization of users. Given that those beacons are easily clonable, in parallel to them we make use of a second kind of beacons, called security beacons, that continuously broadcast a pseudorandom number computed using a secret key and the id of the associated position beacon, allowing us to get better assurance on the location of the user.

The privacy of the users is guaranteed by the use of pseudonyms and a selector of stores, so that only the stores that the user decides will receive their location data.

Our system works in the background and uses Push notifications to send customized promotions based on the location of the users, once it has been securely verified.

Agradecimientos

Quiero dar las gracias en primer lugar a mi tutor, Isaac Agudo Ruiz, por sus grandes consejos y por todo el tiempo y dedicación entregados a este trabajo.

También me gustaría agradecer a mi familia su apoyo incondicional.

Contenido

Índice de figuras

Índice de tablas

Lista de Acrónimos

APNs	Apple Push Notifications service
BLE	Bluetooth Low Energy
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LBS	Location-Based service
MAC	Media Access Control
MAC	Message Authentication Code
MLA *opt-out*	Mobile Location Analytics *opt-out*
RSSI	Received Strength Signal Indication
SHA-1	Secure Hash Algorithm - 1
SMS	Short Message Service
TFG	Trabajo de Fin de Grado
UID	User Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

Capítulo 1. Introducción

En la actualidad, debido a la elevada competitividad en el mundo del *retail*, la captación de nuevos clientes y la fidelización de los mismos se han convertido en la base para la supervivencia de los diferentes comercios en este sector. Conocer las preferencias de los clientes puede ayudar a alcanzar estos objetivos. En el artículo [1], se lleva a cabo un análisis estadístico que revela cuales son los principales agentes influyentes en la elección de comercio por parte de los clientes. En dicho estudio se concluye que el uso de programas de fidelización es uno de los principales factores que tienen en cuenta los clientes a la hora de tomar su decisión, esto es debido a los beneficios que estos programas proporcionan a los clientes (promociones, acumulación de puntos, etc). El análisis también revela que el comportamiento de los clientes durante sus compras está directamente relacionado con las actuales ofertas.

Es importante resaltar el papel que juegan los programas de fidelización a la hora de recabar información sobre el comportamiento de los clientes, esta información es esencial para poder realizar patrones de conducta y estudios de mercado. Según Joseph Turow [2], estos datos no siempre son tomados con el conocimiento y consentimiento de los usuarios: “Traditionally stores used the programs to generate recurring purchases; now they also began to use them as bait to attract shoppers and encourage them to consider it perfectly normal and ordinary to give up information about themselves unselfconsciously.”

Recientes estudios realizados por la empresa The Nielsen Company [3] revelan que las tarjetas de fidelización son el método más extendido para proporcionar descuentos u otros tipos de beneficios a los clientes. Sin embargo, estas tarjetas presentan múltiples inconvenientes tanto para los comercios como para los consumidores [4].

Por el lado del cliente, el hecho de que cada comercio tenga su propia tarjeta hace incómodo el uso de esta tecnología, ya que necesitan mucho espacio para poder llevarlas todas. Además, en algunos casos la activación de estas tarjetas requiere que los usuarios rellenen formularios con sus datos personales.

Con el objetivo de que las empresas puedan realizar estudios de mercado robustos, es necesario que los clientes usen siempre durante sus compras las tarjetas de fidelización, en la mayoría de los casos las tarjetas son presentadas en cada compra, pero hay una parte de los clientes que no utilizan sus tarjetas con regularidad, por lo que la información recogida por los comercios no siempre es completa.

En la figura 1.1. se muestra el porcentaje de uso de los diferentes métodos de fidelización en *retail* en cada continente.

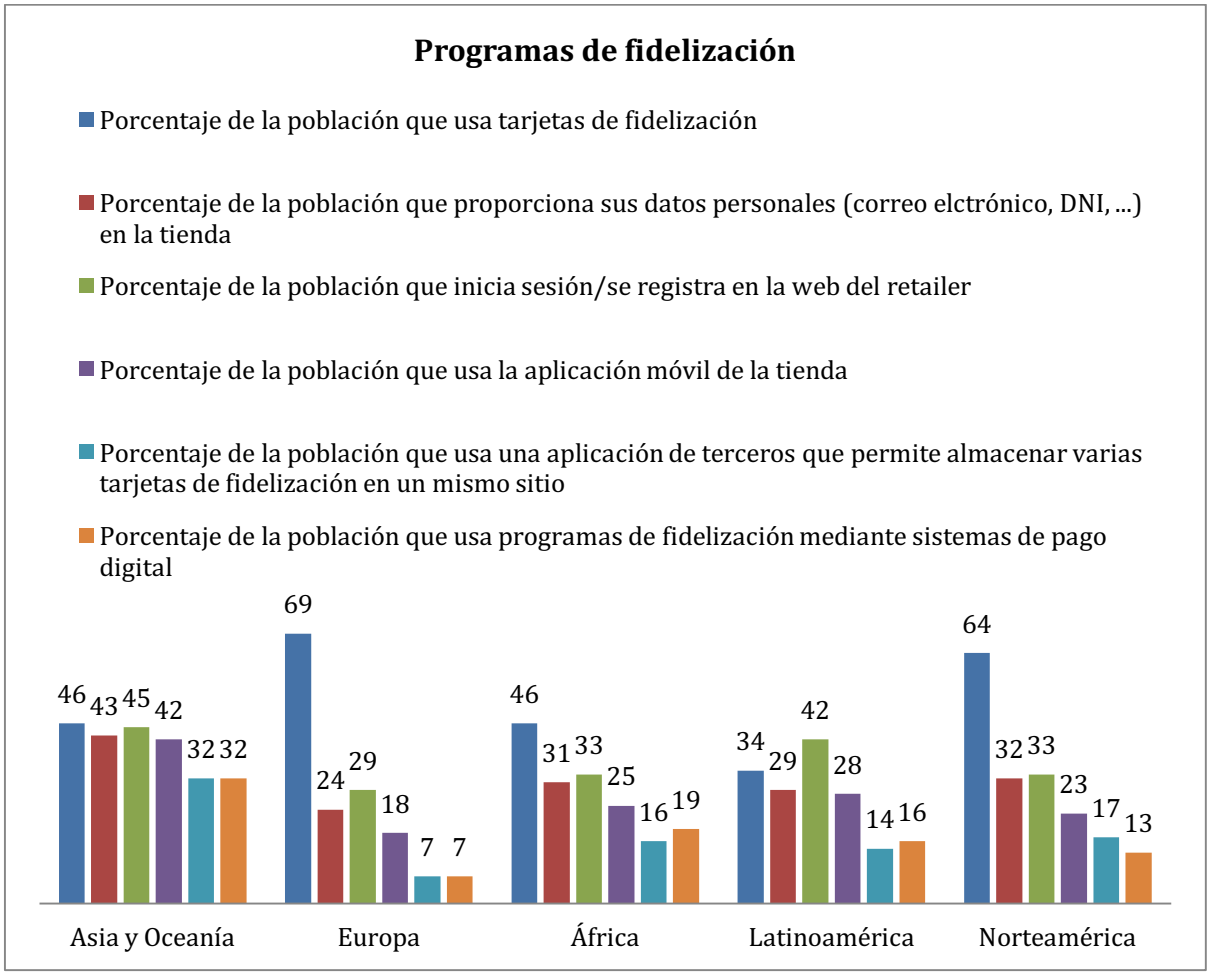


Figura 1.1. Uso de los diferentes programas de fidelización según continentes.

Además de las tradicionales tarjetas de fidelización, existen otros métodos para premiar la lealtad de los clientes y al mismo tiempo recolectar información sobre ellos. El envío de promociones a los teléfonos móviles de los clientes [5] se está convirtiendo en una

forma cada vez más popular de marketing. Estas promociones pueden adoptar múltiples formas: SMS (Short Message Service), correos electrónicos, notificaciones de tipo *Push* o *Pull*, etc; además estas ofertas les pueden llegar a los usuarios en cualquier momento durante el proceso de compra. Hoy en día, incluso existen patentes de sistemas de envío de incentivos a clientes por la red [6]. Para que los clientes reciban estas promociones en sus dispositivos móviles, estos deben identificarse de alguna forma ante los comercios.

Las aplicaciones móviles destinadas a la fidelización en *retail* siguen esta línea de trabajo, los clientes deben crear cuentas de usuario para poder recibir las actuales ofertas. Las aplicaciones además recolectan información de localización de los usuarios que se usa para analizar el comportamiento de estos ante diferentes circunstancias y conocer el impacto de las promociones en los clientes, pudiendo proporcionar una información más detallada que otras formas de fidelización.

Empresas como Euclid, InnoQuant y iPhonedroid se dedican a ofrecer este tipo de servicios a las grandes superficies, haciendo uso para ello de diferentes tecnologías, como WiFi, GPS (Global Positioning System) o BLE (Bluetooth Low Energy). Cabe destacar el papel de la tecnología BLE en el ámbito de los sistemas de fidelización en *retail*, ya que permite mejorar la precisión en la ubicación de los clientes [7]. Esta tecnología se puede usar en combinación con sistemas WiFi o como único método de localización [8].

Euclid Analytics [9] es la solución diseñada por la empresa Euclid que consiste en el uso de tecnología WiFi para realizar el seguimiento de los clientes dentro de la tienda.

MOCA [10] es una herramienta para el desarrollo de aplicaciones móviles creada por la empresa InnoQuant, que hace uso de unos dispositivos llamados balizas para determinar la posición exacta de los clientes dentro de las tiendas. Las balizas utilizan la tecnología BLE para transmitir una señal *broadcast*, esta señal es leída por receptores BLE (como teléfonos inteligentes) y enviada al servidor de la aplicación, el cual, según la información recibida, podrá determinar dónde se encuentra el usuario. MOCA también puede combinar el uso de WiFi o GPS, esta última tecnología se usa para localización en exteriores, ya que necesita de visión directa para el correcto emplazamiento de los dispositivos [11][12].

Por su parte la empresa iPhonedroid ha desarrollado el sistema Kappta [13] como solución. Kappta utiliza de manera combinada la tecnología GPS y las balizas BLE, empleando GPS para localizar al usuario dentro de un área relativamente amplia y las balizas para una localización más exacta.

En España, algunos ejemplos de instalaciones de redes de balizas se pueden encontrar en los aeropuertos de El Prat y Madid-Barajas, dónde se usan las balizas para enviar notificaciones *Push* a los pasajeros con información relativa a los vuelos; en el estadio de fútbol Camp Nou, dónde gracias al uso de una red de este tipo se ofrece a los aficionados diferentes tipos de ofertas y en el museo del Prado dónde en algunas exposiciones los visitantes han podido recibir información personalizada en tiempo real sobre la obra que estaban contemplando [14].

1.1. Objetivo

El objetivo del presente TFG (Trabajo de Fin de Grado) es el desarrollo de una aplicación de fidelización de clientes en grandes superficies que mediante el empleo de balizas BLE, permita recolectar información sobre las visitas que realizan los clientes a las diferentes tiendas que las componen. La aplicación está orientada a su uso en centros comerciales, centrándose en los siguientes aspectos diferenciadores:

- **Seguridad:** El consumidor recibirá incentivos por compartir información sobre su ubicación, para el correcto funcionamiento del sistema es imprescindible cerciorarse de que la información de localización enviada por los usuarios no es falsa. Para ello se usarán mecanismos de autenticación por proximidad que determinen de forma unívoca la ubicación de los usuarios, evitando así que el sistema pueda ser burlado por balizas clonadas.
- **Privacidad:** El consumidor debe ser capaz de elegir con qué comercios quiere compartir su información de localización, pudiendo ampliar o restringir su selección de tiendas en cualquier momento. Además, la privacidad de los clientes deberá quedar garantizada en todo momento.
- **Experiencia de usuario:** Una vez el usuario haya iniciado sesión y seleccionado sus preferencias en la aplicación, este no necesitará interactuar con la aplicación para el correcto funcionamiento de la misma. Tanto el envío de incentivos a los clientes como la recolección de datos del

usuario se realizará de manera totalmente transparente. Además, la aplicación no debe suponer una sobrecarga para el dispositivo del usuario, por lo que se deben utilizar tecnologías de bajo consumo.

- **Acceso de grano fino:** Los comercios deben ser capaces de ver solamente los datos que los consumidores hayan compartido en los términos que ellos hayan elegido. Debe ser posible definir políticas de acceso de grano que eviten el acceso a todo o nada tradicional en este tipo de esquemas.

Los dispositivos para los que se desarrollará la aplicación son aquellos que hacen uso del sistema operativo móvil iOS, en concreto aquellos dispositivos que dispongan de comunicación BLE.

1.2. Estado del arte

Como se menciona anteriormente, hoy en día ya existen aplicaciones destinadas a la fidelización de clientes en el sector *retail*, la principal diferencia existente entre estas aplicaciones es la tecnología que usan para autenticar y localizar a los usuarios, estas tecnologías suelen ser WiFi, GPS, Bluetooth o combinaciones de ellas. La tecnología GPS al proporcionar una menor precisión de la localización en entornos cerrados siempre se combina con alguna otra tecnología LBS (Location-Based service) para conseguir afinar la localización de los usuarios.

El objetivo principal de todas estas aplicaciones es, más que ofrecer promociones o beneficios a los usuarios, monitorizar el comportamiento de los usuarios en el centro comercial o en la tienda.

Aplicaciones que usan tecnología WiFi

Cuando un usuario con el WiFi de su dispositivo móvil activo entra en una tienda, el punto de acceso WiFi de la tienda leerá la dirección MAC (Media Access Control) del dispositivo (estas direcciones son únicas para cada dispositivo). El servidor de la aplicación de fidelización correspondiente codifica las direcciones MAC de los usuarios, para que no se pueda identificar a qué dispositivo pertenece cada una, y las guarda en su base de datos [15].

Estas aplicaciones se podrían considerar invasivas respecto a la privacidad del usuario ya que, aunque no se pueda saber a quién pertenece los datos registrados, en la mayoría de los casos los usuarios no tienen conciencia de que están siendo "vigilados".

Si el usuario quiere evitar que su posición sea rastreada debe realizar lo que se conoce como MLA *opt-out* (Mobile Location Analytics *opt-out)* [16], de esta forma su dirección MAC no podrá ser seguida por este tipo de aplicaciones. Los dispositivos de Apple que usan iOS 8, generan direcciones MAC dinámicas, por tanto, el uso de MLA *opt-out* no les sirve como solución. Los usuarios también pueden desactivar el WiFi de sus dispositivos móviles para evitar ser localizados.

Estos métodos para evitar la monitorización tienen algunos problemas. En el caso de MLA *opt-out*, se puede afirmar que este no es un método deductivo o simple para aquellos usuarios que no tengan conocimientos básicos de informática. Por otro lado, el hecho de que un usuario tenga que desactivar el WiFi de su dispositivo móvil para evitar el seguimiento tampoco parece una solución acertada, ya que los usuarios pueden desear tener el WiFi activo por otros motivos.

En este tipo de soluciones, como en el caso de Euclid Analytics, si los clientes quieren recibir promociones o beneficios de las tiendas, estos deben registrarse en el servicio Euclid proporcionando datos como su dirección de correo electrónico, nombre, género, edad, etc. Si los clientes no han realizado el registro ni tampoco el *opt-out*, las tiendas obtendrán información sobre los usuarios sin darles nada a cambio.

Aplicaciones que usan tecnología Bluetooth (balizas BLE) y GPS (geofencing)

En el caso de las aplicaciones que utilizan este tipo de tecnología los usuarios inicialmente deben registrarse. Tras haber completado el registro, la información sobre el comportamiento del usuario será enviada al servidor de la aplicación sin que el usuario pueda tener ningún tipo de control sobre los datos transmitidos.

La tecnología BLE presenta problemas relacionados con la seguridad y la privacidad de los datos transmitidos [17], los estándares básicos de balizas BLE carecen de capas de seguridad, por lo que los datos que transmiten pueden ser leídos por cualquier receptor BLE. Esta información puede usarse para copiar balizas y confundir el sistema, haciendo creer que un usuario está en una posición errónea.

1.3. Contenido de la memoria

La memoria consta de los siguientes capítulos:

Introducción

Es el presente capítulo, en él se hace una aproximación inicial al contexto en el que se desarrolla el TFG marcando de manera precisa los objetivos.

Tecnologías usadas

En él se describen las tecnologías que han sido usadas para el desarrollo del sistema, así como los lenguajes de programación y técnicas de seguridad utilizadas.

Diseño de la solución

En este capítulo se definen los casos de uso del sistema, además se hace referencia a cómo se han usado las tecnologías descritas en el capítulo anterior y se explica porque han sido usadas estas tecnologías y no otras.

Implementación

En la implementación se expone paso a paso los procedimientos que han sido necesarios para el desarrollo del sistema.

Conclusiones y líneas de trabajo futuro

Por último, se realiza una conclusión en vista de los resultados obtenidos durante el desarrollo del trabajo, haciendo referencia a los objetivos alcanzados y proponiendo líneas de trabajo futuro consideradas interesantes y que ayudarían a obtener un sistema más completo.

Capítulo 2. Tecnologías usadas

Las tecnologías descritas en el presente capítulo son aquellas que han sido seleccionadas para el desarrollo del sistema de fidelización en *retail*. En el capítulo 3 se presentan cuáles son los motivos por los que se usan estas tecnologías y no otras que podrían realizar funciones similares.

2.1. Firebase

Firebase es una plataforma de desarrollo móvil en la nube de Google, disponible para diferentes plataformas (iOS, Android, web). Su principal función es la de facilitar el desarrollo de aplicaciones web.

Las principales características de Firebase son [18]:

- **Analíticas:** gracias al servicio Analytics podemos obtener medidas sobre el uso de nuestra aplicación, además Firebase te permite crear tus propios eventos, para así tener una mejor aproximación de cómo funciona la aplicación y del uso que los usuarios hacen de ella.
- **Desarrollo:** servicios como almacenamiento, mensajería en la nube, autenticación y testeo entre otros, permiten a los desarrolladores delegar en Firebase determinadas operaciones. De esta forma se consigue reducir el tiempo de desarrollo de la aplicación, evitar errores y alcanzar un nivel de calidad aceptable.
- **Poder de crecimiento:** la plataforma también ofrece una gestión sencilla de los usuarios y la posibilidad de captar nuevos usuarios con sus servicios de envío de notificaciones e invitaciones.
- **Monetización:** AdMob es un servicio proporcionado por Firebase el cuál te permite ganar dinero mediante la publicación de anuncios en tu aplicación.

Para el desarrollo de la aplicación asociada al presente TFG se han usado los servicios [19] que se describen en los siguientes apartados.

2.1.1. Analytics

El servicio Analytics de Firebase genera automáticamente estadísticas sobre el uso que los usuarios hacen de nuestra aplicación. Estas estadísticas se basan en el número de veces que son ejecutadas determinadas acciones por los usuarios.

Este servicio consta de una serie de eventos que, por defecto, son usados en todas las aplicaciones sin necesidad de escribir ninguna línea de código. Acciones como la primera vez que el usuario abre la aplicación tras haberla instalado por primera vez o reinstalado, el inicio de sesión o la actualización de la versión de la aplicación están asociadas a eventos automáticos.

Firebase pone a disposición de los usuarios una lista de eventos predefinidos para su libre uso en las aplicaciones. Además, existe la posibilidad de crear nuestros propios eventos, y de esta forma tener un mayor control de nuestra aplicación.

2.1.2. Authentication

La autenticación proporcionada por Firebase nos permite conocer la identidad de los usuarios de la aplicación, esta autenticación se puede llevar a cabo mediante contraseña, de manera anónima o con el uso de autoridades de identidades federadas como Facebook, Google o Twitter. También se permite el uso de un sistema personalizado de autenticación.

Firebase asigna un identificador único a cada usuario registrado en una determinada aplicación, a este identificador se le conoce como UID (User Identifier).

Además del registro de nuevos usuarios y el inicio de sesión por parte de los usuarios ya registrados, este servicio permite realizar acciones como la recuperación de contraseña en caso de olvido, la actualización del perfil de usuario o el cambio de contraseña.

2.1.3. Realtime Database

Se trata de una base de datos de tipo NoSQL que permite el almacenamiento de datos y la sincronización de los mismo en tiempo real con todos los usuarios conectados a ella. El formato de almacenamiento de los datos es JSON (JavaScript Object Notation).

Para el correcto funcionamiento de las aplicaciones aun cuando un dispositivo pierda la conexión, Firebase permite habilitar capacidades sin conexión, gracias a las cuales los datos de los usuarios persisten de manera local en sus dispositivos móviles. Cuando el terminal recupera la conexión, la Realtime Database sincroniza los datos de usuario.

También es posible definir reglas para limitar el acceso a los datos, indicando quién y de qué forma puede acceder a los mismos. Para ello Firebase hace uso de unas expresiones llamadas Security Rules. Estas reglas se definen con una sintaxis parecida a JavaScript. Por defecto, no se permite la lectura o escritura en la base de datos a usuarios que no estén autenticados:

```
{
      "rules": {
            ".read": "auth != null",
            ".write": "auth != null"
      }
}
```

2.1.4. Cloud Functions

Son funciones que se ejecutan automáticamente en *backend* cuando se produce una determinada acción, como por ejemplo la escritura de una nueva entrada en la base de datos de Firebase, el registro de un nuevo usuario en la aplicación, el envío de un cierto evento o el acceso a una URL (Uniform Resource Locator) específica. Las respuestas a estas acciones también pueden ser muy variadas, tal como se representa en la figura 2.1.

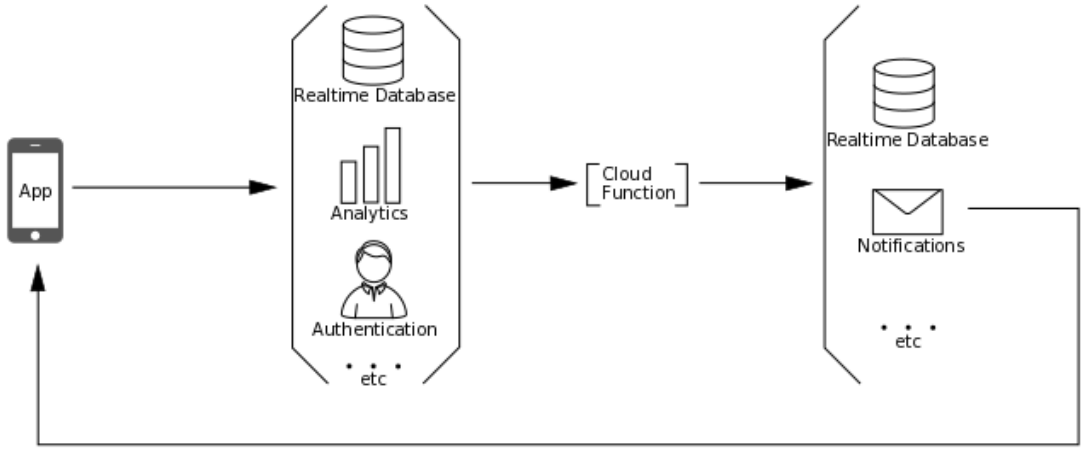


Figura 2.1. Funcionamiento general de Firebase Cloud Functions.

Una vez que la función es creada y se ha realizado el despliegue de la misma, el servidor de Google será el encargado de gestionar la función, escuchando los eventos y lanzando la función cuando sea necesario.

El lenguaje de programación en el que se escriben estas funciones es Node.js, este es un lenguaje de código abierto que destaca por ser asíncrono y por tener una arquitectura orientada a eventos.

En el siguiente código corresponde a una Cloud Function que es accionada cuando el usuario escribe en una determinada referencia su nombre, y que devuelve como resultado una entrada en otro directorio de la base de datos.

```
exports.Mensaje = functions.database.ref('app/{PushId}/nombre')
.onWrite(event=>{
      const nombre = event.data.val();
      const mensaje = 'Bienvenido' + nombre;
      return event.data.ref.parent.child('mensaje').set(mensaje);
});
```

2.1.5. Notifications

Este servicio permite el envío de notificaciones de tipo *Push* a los usuarios de tu aplicación. Los usuarios destinatarios de estas notificaciones pueden ser todos los registrados en la aplicación, aquellos suscritos a un determinado tema o un usuario en concreto.

Apple Push Notifications service (APNs) [20]

Servicio proporcionado por Apple que permite la propagación de notificaciones tipo *Push* a los dispositivos de la marca de una forma segura y eficiente.

Para que se pueda realizar en envío de notificaciones, es necesario el establecimiento de dos conexiones persistentes. La primera, que conecta la aplicación del usuario con el APNs, se establece cuando el usuario abre la aplicación por primera vez, sobre esta conexión el usuario podrá configurar si desea recibir o no notificaciones. La segunda conexión es la que une el APNs con el servidor del proveedor de la aplicación web. Para el establecimiento de esta conexión será necesario realizar algunos ajustes en la cuenta de desarrollador de Apple del proveedor, así como el uso de un certificado criptográfico proporcionado por Apple.

Una vez las dos conexiones se hayan establecido satisfactoriamente, el servidor de la aplicación podrá enviar solicitudes de notificaciones hacia el APNs y este se las entregará al dispositivo adecuado, y es en el mismo dispositivo donde se selecciona la aplicación destino.

En la figura 2.2. se muestra cual sería el camino recorrido por una notificación enviada desde el servidor hasta una aplicación en concreto.

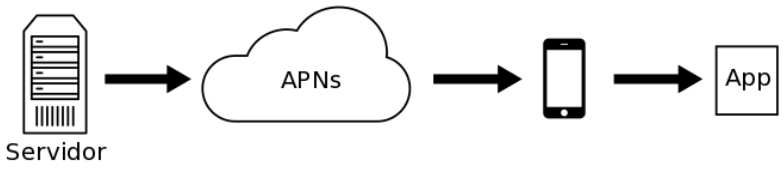


Figura 2.2. Envío de notificación tipo Push desde un servidor remoto a una aplicación mediante APNs.

APNs se caracteriza por tener una arquitectura segura basada en el uso de dos niveles de confianza:

- **Conexión:** para que un proveedor o dispositivo se pueda conectar al APNs debe estar autorizado, esta autorización se gestiona mediante el uso de certificados.
- ***Token* de dispositivo:** para cada notificación enviada se establece una conexión extremo a extremo entre el proveedor y el dispositivo final. Los *tokens* de dispositivos son unos identificadores únicos para cada aplicación de cada dispositivo proporcionados por el APNs en el momento del registro, una vez recibido el *token*, la aplicación lo envía hacia su servidor para que este le pueda

enviar notificaciones. El valor del *token* puede cambiar durante la vida de la aplicación.

En la figura 2.3. se representa con detalle cómo se realiza la actualización de *tokens* desde el terminal móvil al servidor y cómo se lleva a cabo el envío de una notificación del servidor a la aplicación destino.

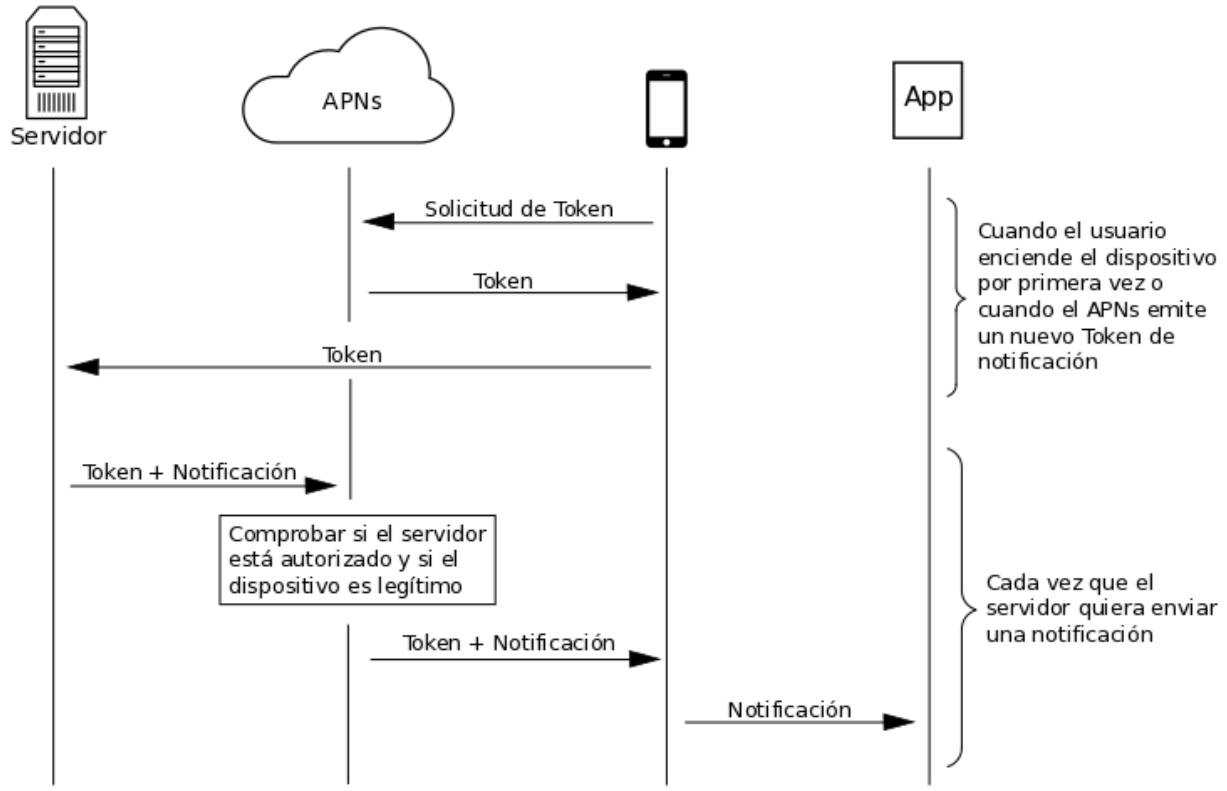


Figura 2.3. Diagrama envío de notificaciones Push.

2.2. Balizas BLE

Las balizas BLE [21] son pequeños dispositivos que hacen uso de la tecnología BLE para emitir señales *broadcast* de onda corta, estas señales pueden alcanzar unos 40 metros y pueden ser captadas por otros dispositivos que dispongan de la misma tecnología.

Una de las principales características de estos dispositivos es su bajo consumo de energía, proporcionándoles una elevada tasa de autonomía. Normalmente se alimentan con una pila de botón que puede durar años.

En la actualidad existen diferentes protocolos de balizas, cada uno de ellos define una estructura diferente para los datos transmitidos por estos dispositivos.

2.2.1. iBeacons

Es un protocolo creado por Apple que utiliza BLE para transmitir la información representada en la trama de datos de la figura 2.4. [22][23]:

- **UUID** (Universally Unique Identifier)**:** dieciséis bytes que sirven para identificar de manera única y global a un conjunto de iBeacons.
- **Major:** dos bytes que identifican a un subconjunto de iBeacons dentro del grupo formado por todos los iBeacons que comparten UUID.
- **Minor:** dos bytes que identifican a un iBeacon concreto dentro del subconjunto definido por el major.
- **Tx Power:** byte extra donde se envía la información sobre el RSSI (Received Strength Signal Indication) a un metro de distancia del iBeacon. El RSSI mide la intensidad con la que un teléfono móvil recibe la señal, la intensidad varía según la distancia a la que se encuentre el terminal de la baliza y de cómo se hayan configurado los iBeacons, ya que estos pueden transmitir más o menos potencia dependiendo de su implementación. Este hecho hace que el RSSI recibido no sea un factor determinante a la hora de calcular la distancia a la que se encuentra el usuario, por ello es necesario que cuando se configure un iBeacon se establezca el valor que este transmitirá como Tx Power. La librería Core Location de Apple hace uso de este valor y del RSSI verdaderamente recibido por el usuario para determinar a qué distancia del iBeacon se encuentra.

iBeacon Data (31Bytes)				
iBeacon Prefix 9Bytes	UUID 16Bytes	Major 2Bytes	Minor 2Bytes	Tx Power 1Byte

Figura 2.4. Paquete de datos iBeacon.

2.2.2. Eddystone

Protocolo creado por Google que al igual que iBeacon hace uso de las comunicaciones BLE para transmitir información. El formato de trama en este tipo de protocolo responde a la estructura representada en la figura 2.5. [24]:

- Once bytes de prefijo.

- Trama de datos: el tamaño de la trama de datos al igual que su contenido puede variar dependiendo del tipo de paquete de datos que se esté utilizando.

Eddystone Data (31Bytes)									
Len 1Byte	Type 1Byte	Flags 1Byte	Len 1Byte	Type 1Byte	Eddystone UUID 2Bytes	Len 1Byte	Type 1Byte	Eddystone UUID 2Bytes	Eddystone Frame 20Bytes

Figura 2.5. Paquete de datos Eddystone.

Tipos de paquetes:

- **Eddystone-UID:** en este tipo de paquete se envía un identificador único global, es lo más parecido a un iBeacon usando el protocolo de Google. El identificador se compone de dos partes: namespace e instance. El campo Tx Power se usa con el mismo propósito que el protocolo iBeacon, a diferencia de que en las balizas Eddystone, el valor de referencia del RSSI es el medido a cero metros. Si el paquete transmitido es de este tipo, la trama de datos ocupará veinte bytes. En la figura 2.6. se muestra la trama de datos de una baliza Eddystone-UID.

Eddystone-UID Frame (20Bytes)				
Type 1Byte	Tx Power 1Byte	Namespace ID 10Bytes	Instance ID 6Bytes	RFU 2Bytes

Figura 2.6. Trama Eddystone-UID.

- **Eddystone-URL:** envía una URL. Cuando un usuario pase cerca de una baliza y reciba una trama de este tipo, podrá interactuar con la URL (suponiendo que su teléfono móvil dispone de las funcionalidades necesarias). La trama de datos puede ocupar de seis a veinte bytes y seguirá la forma de la trama mostrada en la figura 2.7.

Eddystone-URL Frame (20Bytes)			
Type 1Byte	Tx Power 1Byte	URL Scheme Prefix 1Byte	Encoded URL 0-17Bytes

Figura 2.7. Trama Eddystone-URL.

- **Eddystone-TLM:** este paquete se emite junto a uno de los dos anteriores. La información que contiene está relacionada con el estado de la propia baliza: potencia de la batería, temperatura de la baliza, número de anuncios transmitidos y tiempo desde que se encendió la baliza. Para este tipo de paquete de datos la trama de datos sólo ocupa catorce bytes. En la figura 2.8. se representa la estructura de la trama de datos enviada por una baliza Eddystone-TLM.

Eddystone-TLM Frame (14Bytes)					
Type 1Byte	Version 1Byte	Battery Voltage (mV) 2Bytes	Beacon Temp 2Bytes	Advertisement PDU Count since boot 4Bytes	Time Since Boot 4Bytes

Figura 2.8. Trama Eddystone-TLM.

- **Eddystone-EID:** emite un identificador aleatorio que va cambiando periódicamente, este identificador se resuelve enviando una petición a la nube de Google. Proporciona seguridad a los sistemas, evitando ataques de tipo *Hijacking/Piggybacking* (atacante usa las balizas para transmitir sus propios mensajes) y *Spoofing* (atacante copia la baliza) [25]. La trama de datos será de diez bytes y seguirá la estructura mostrada en la figura 2.9.

Eddystone-EID Frame (10Bytes)		
Type 1Byte	Version 1Byte	Ephemeral Identifier 8Bytes

Figura 2.9. Trama Eddystone-EID.

2.3. Xcode

IDE (Integrated Development Environment) creado por la empresa Apple. Xcode proporciona todas las herramientas necesarias para el desarrollo de aplicaciones para macOS, iOS, watchOS y tvOS, entre las herramientas proporcionadas se incluyen todas aquellas asociadas a la escritura del código, la compilación, la carga del software creado en un dispositivo, la depuración, etc.

Tradicionalmente las aplicaciones para dispositivos de Apple han sido programadas en Objective-C, que es un lenguaje de programación orientado a objetos caracterizado por ser un superconjunto de C. En junio de 2014 [26] Apple lanzó la primera versión de

Swift, ofreciendo así una nueva herramienta más simple y amigable para el desarrollo de aplicaciones.

Un aspecto importante es la compatibilidad entre los dos lenguajes, independientemente del lenguaje en el que estemos programando, podremos usar librerías y API (Application Programming Interface) escritas en Swift ó Objective-C.

Entre los cambios y mejoras introducidos por Swift destacan el uso de un lenguaje mucho más claro, conciso e intuitivo y el uso de un gestor de memoria, evitando así el uso de punteros. Estas características reducen tanto los errores como el tiempo de desarrollo de las aplicaciones, por lo que se podría decir que Swift aumenta la productividad de los desarrolladores.

Los estudios realizados por la empresa TIOBE en el TIOBE Index de Julio de 2017 [27], que indican la popularidad de los diferentes lenguajes de programación, posicionan al lenguaje de programación Swift como el doceavo de la lista, mientras que Objective-C queda seis puestos más abajo. Además, si observamos la evolución de la popularidad de ambos lenguajes en las figuras 2.10. y 2.11., se puede apreciar como la popularidad de Swift va aumentando progresivamente mientras que la de Objective-C ha sufrido una brutal caída durante los últimos años.

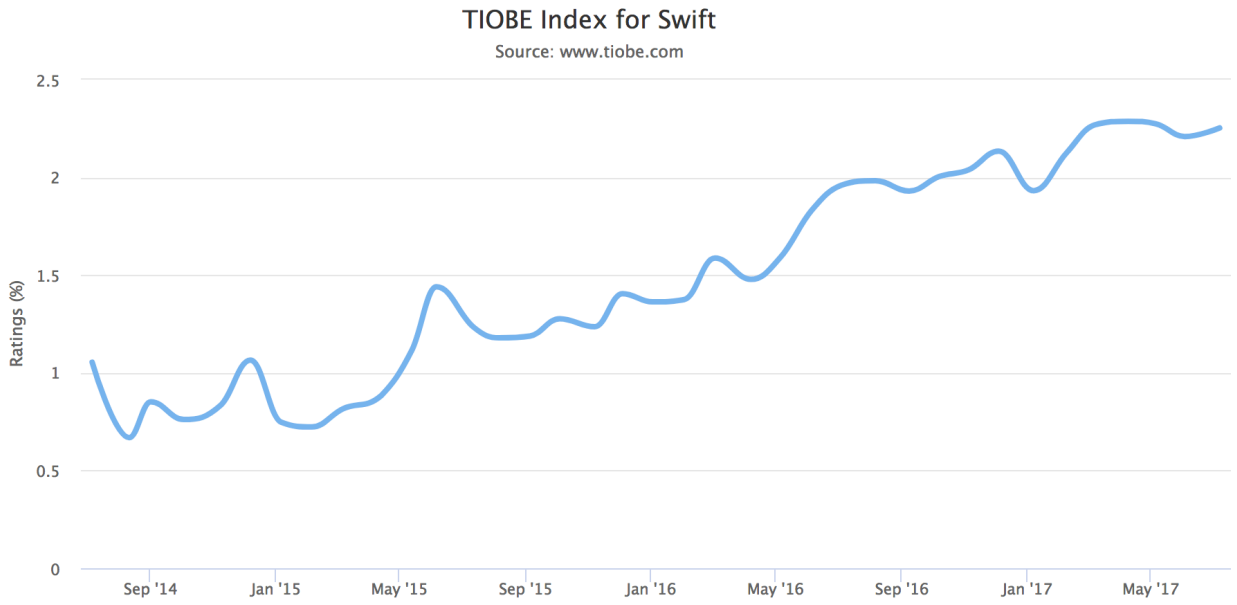


Figura 2.10. Evolución en la popularidad de Swift.

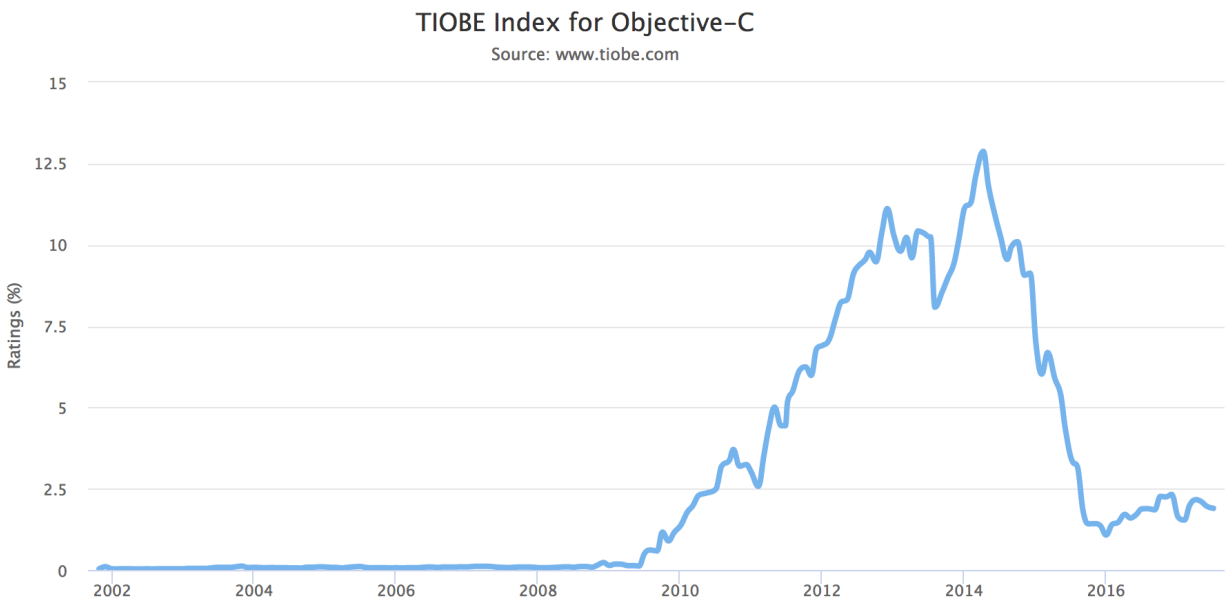


Figura 2.11. Evolución en la popularidad de Objective-C.

Para poder añadir librerías externas (que no son propias de Apple) en un proyecto de Xcode, es necesario usar el gestor de dependencias a nivel de aplicación CocoaPods, que permite mantener actualizadas las librerías de nuestro proyecto.

2.4. HMAC (Hash-based Message Authentication Code)

Como se indica en [28], es una función que calcula el valor MAC (Message Authentication Code) de un mensaje mediante el uso de una función hash criptográfica y una clave criptográfica secreta. Su objetivo es autenticar y proteger la integridad de los datos transmitidos.

Para el correcto funcionamiento de HMAC, el cliente y el servidor deben compartir una clave secreta, la cual solo es conocida por las dos partes. Cuando el cliente quiere enviar un mensaje hacia el servidor, este calculará el valor HMAC de dicho mensaje y lo enviará junto al mensaje hacia el servidor, tal como se muestra en la figura 2.12. Cuando el servidor recibe los datos, calcula el valor HMAC y compara los dos valores HMAC (el calculado por el cliente y el suyo propio), si los valores coinciden los datos quedan autenticados y la integridad de estos verificada, evitando que otra entidad pueda impersonar al cliente legítimo.

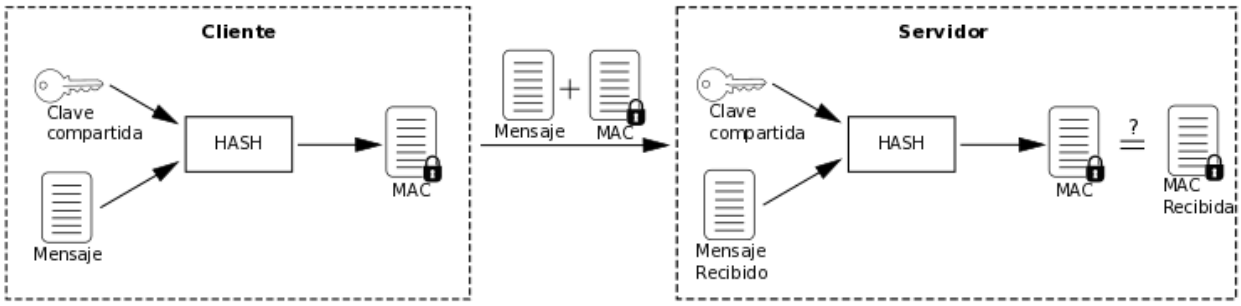


Figura 2.12. HMAC.

Los códigos HMAC también se pueden usar para la autenticación de usuarios en los protocolos de Reto-Respuesta (Challenge-Response). Como se puede observar en la figura 2.13., el servidor envía al cliente un reto (un número aleatorio) y este debe ser capaz de calcular el código HMAC con la clave que ambos comparten.

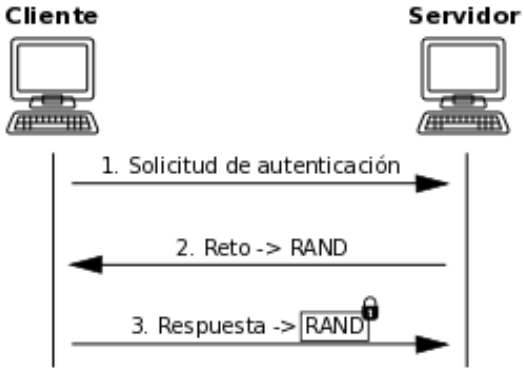


Figura 2.13. Protocolo Reto-Respuesta.

La fuerza criptográfica de HMAC depende de la función hash criptográfica utilizada y del tamaño y calidad de la clave criptográfica secreta.

Capítulo 3. Diseño de la solución

Para cumplir con los objetivos marcados en el capítulo 1, se desarrollará una aplicación móvil que será capaz de leer y procesar la información recibida de las balizas BLE para así conocer la ubicación del cliente, este sistema debe ser seguro y garantizar la privacidad de los datos del usuario, a la vez que le otorga a este el poder de decisión acerca de qué datos y de qué forma quiere compartir. El sistema desarrollado se divide en tres bloques: infraestructura, aplicación móvil y servicio de soporte. Esta división en bloques queda reflejada en la figura 3.1.

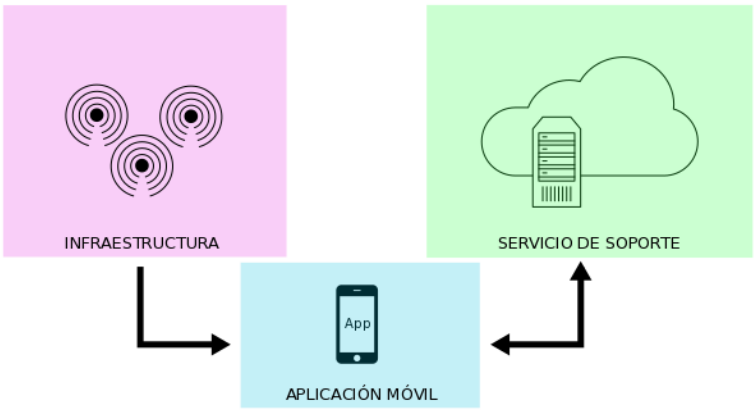


Figura 3.1. Esquema de división en bloques del sistema.

3.1. Casos de uso

Los principales casos de uso del sistema a desarrollar serán:

- **Caso de uso 1: Gestión de la sesión.** Este primer caso de uso recoge todas las acciones relacionadas con la gestión del perfil del usuario, de esta forma se podría diferenciar dos sub-casos:

- **Caso de uso 1.a: Iniciar de sesión y registrarse en la aplicación.** Se consideran estas dos acciones dentro de un mismo caso de uso ya que el procedimiento en ambos casos sería el mostrado en la figura 3.2. Cada vez que el usuario inicie sesión en la aplicación se le proporcionará un nuevo pseudónimo.

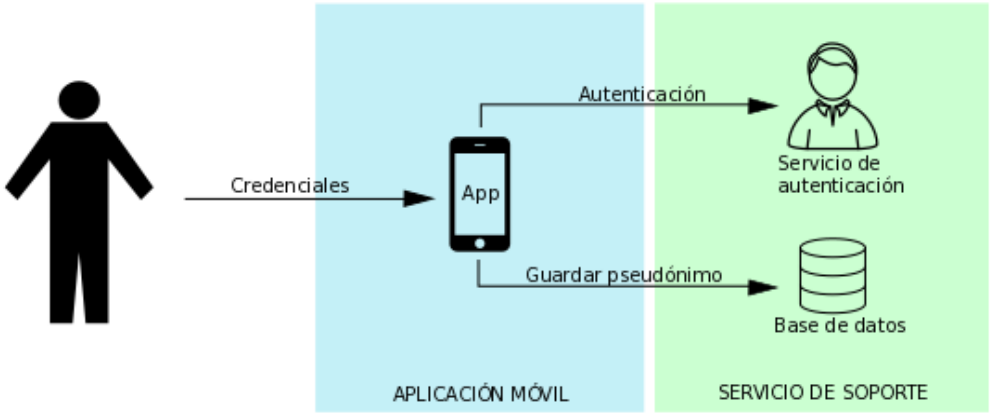


Figura 3.2. Caso de uso 1.a: Iniciar sesión y registrar en la aplicación.

- **Caso de uso 1.b: Cerrar sesión.** El usuario podrá cerrar sesión en la aplicación cada vez que lo considere oportuno. Para ello se seguirá en diagrama de la figura 3.3.

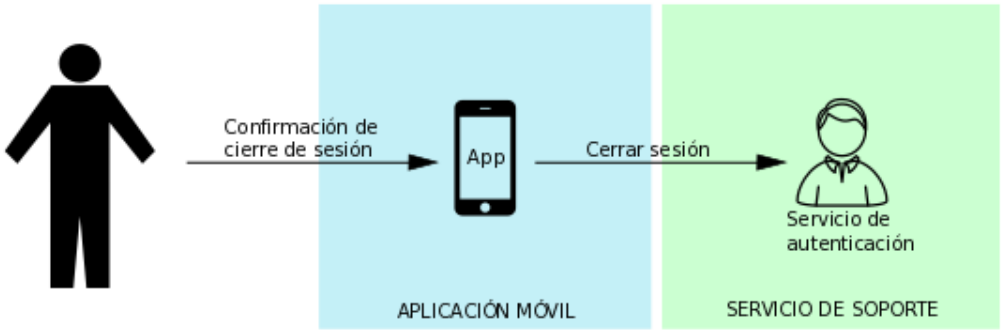


Figura 3.3. Caso de uso 1.b: Cerrar sesión.

- **Caso de uso 2: Selección de preferencias de seguridad.** Como se menciona con anterioridad, en la aplicación el usuario podrá elegir las tiendas a las que quiere enviar sus datos. Las preferencias del usuario quedarán guardadas en la base de datos del servidor, como se muestra en la figura 3.4.

Figura 3.4. Caso de uso 2: Selección de preferencias de seguridad.

- **Caso de uso 3: Localización del usuario.** Para el correcto funcionamiento del sistema será necesario poder ubicar al usuario dentro del centro comercial a medida que este se va moviendo por el mismo. Para cumplir con las políticas de privacidad descritas en los objetivos del trabajo, solo las tiendas previamente seleccionas por los usuarios obtendrán la información de la posición de los usuarios. En la figura 3.5. se muestra un diagrama de cómo se llevaría a cabo la localización de los usuarios.

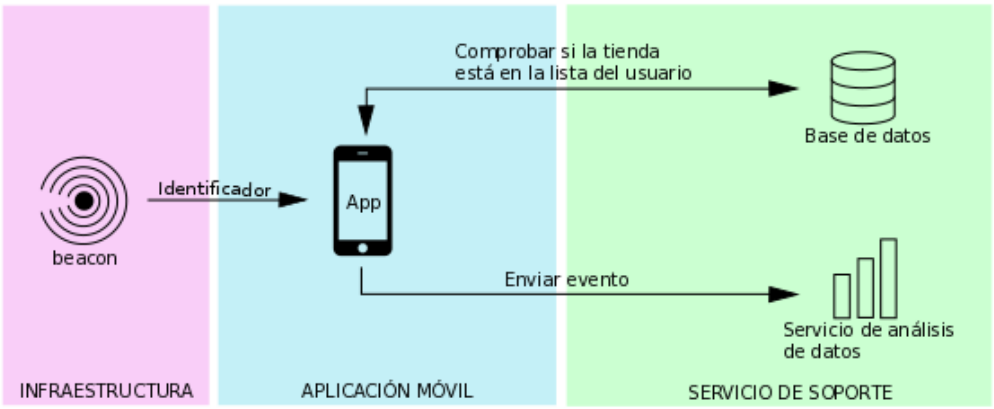


Figura 3.5. Caso de uso 3: Localización del usuario.

- **Caso de uso 4: Validación de la posición del usuario.** Antes de que el usuario pueda recibir la promoción ofertada por la tienda que esté visitando su posición debe ser verificada. Las posiciones validadas deberán quedar registradas para poder realizar de manera más exacta los estudios de mercado. En la figura 3.6. se representa de manera esquemática como se llevaría a cabo la validación de la posición de los usuarios y el envío de ofertas en su caso.

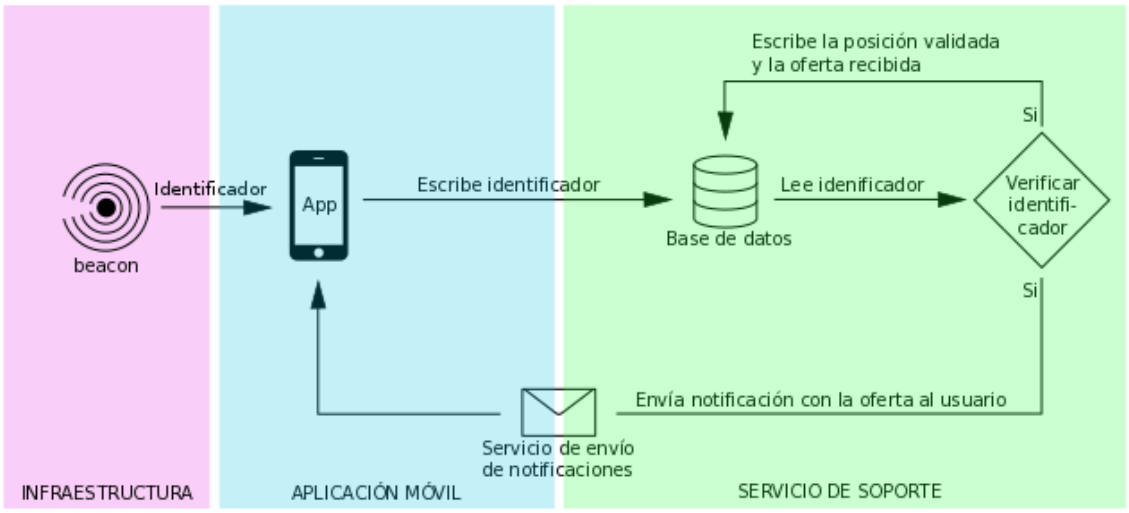


Figura 3.6. Caso de uso 4: Validación de la posición del usuario.

- **Caso de uso 5: Consulta de ofertas obtenidas por el usuario.** Las ofertas acumuladas por los usuarios quedarán registradas la base de datos del servidor para que puedan ser consultarlas posteriormente, de manera similar a como se muestra en la figura 3.7.

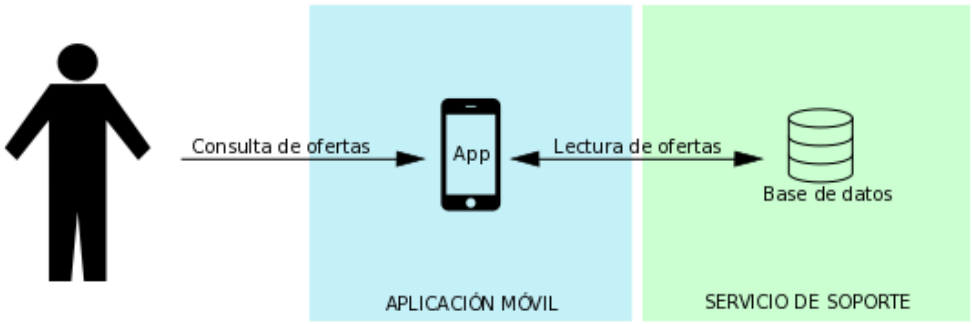


Figura 3.7. Caso de uso 5: Consulta de ofertas obtenidas por el usuario.

3.2. Infraestructura de localización

Compuesta por las balizas BLE distribuidas por el centro comercial. Se tendrán dos tipos de balizas, diferenciadas por su función dentro del sistema y por el protocolo que utilizan, de esta forma se podrá distinguir entre balizas de posición y balizas de seguridad.

3.2.1. Balizas de posición

De entre las diferentes tecnologías que pueden ser usadas para la localización de usuarios en el desarrollo de sistemas de fidelización en *retail*, en la solución

diseñada se propone el uso de dispositivos BLE. Las principales ventajas del uso de esta tecnología frente a WiFi o GPS son [29][30]:

- Mayor precisión en la localización de los usuarios.
- Menor consumo de batería en los dispositivos móviles.
- Para que el sistema pueda recabar información sobre los usuarios, es necesario que los usuarios tengan instalada una aplicación en sus teléfonos móviles. Este punto, que es visto como un inconveniente por algunos autores, se considera una ventaja en el ámbito del presente TFG, ya que de esta forma los usuarios al instalar la aplicación y aceptar sus términos de uso están dando su consentimiento al envío de información sobre su posición.

Ya que la aplicación móvil se desarrollará para dispositivos iOS, el protocolo de balizas BLE seleccionado para realizar la localización de los usuarios será iBeacon, esta decisión se debe a cuestiones de compatibilidad, ya que tanto el protocolo iBeacon como el sistema operativo iOS han sido desarrollados por la marca Apple. Las balizas Eddystone desarrolladas por Google también son compatibles con la plataforma iOS, pero la detección y lectura de estas resulta más compleja en el desarrollo de aplicaciones iOS, al no estar integradas en el Core del sistema operativo.

Como se menciona en el capítulo anterior, las balizas que utilizan el protocolo iBeacon transmiten un identificar único (formado por el UUID, el major y el minor), con lo cual cuando la aplicación lea este identificador podrá saber dónde se encuentra el usuario.

Será necesario disponer de múltiples iBeacons distribuidos por el centro comercial. El reparto de estos dispositivos dentro del área comercial se hará de la siguiente forma: se colocará un iBeacon en cada acceso al centro comercial y otro en los escaparates de cada tienda. Dentro de las tiendas las balizas se pueden distribuir de diferente forma según qué áreas de la tienda queramos tener controladas, se considera oportuno tener al menos una justo al entrar en la tienda, otra en la caja y una última en la zona de probadores. En la figura 3.8. se muestra un ejemplo de cómo podrían ir distribuidos los iBeacons dentro de una tienda.

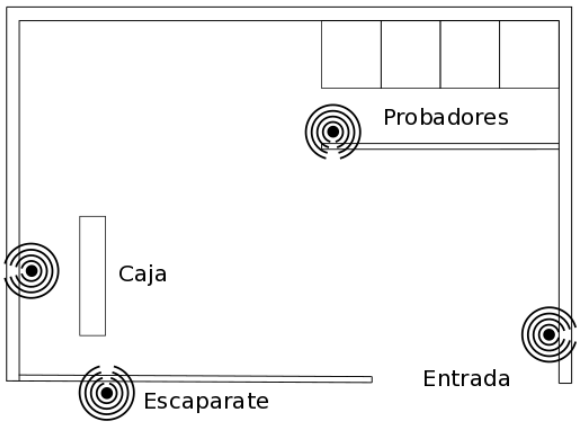


Figura 3.8. Esquema de situación de iBeacons dentro de una tienda.

En el desarrollo del sistema se ha utilizado el UUID como identificador de todos los iBeacons asociados a un centro comercial, el major se refiere al subconjunto de iBeacons relacionados con una tienda y el minor hace referencia a un iBeacon en particular dentro de una tienda. En la Tabla 3.1. se muestra un ejemplo de asignación de identificadores a diferentes iBeacons de varias tiendas.

Centro Comercial	**UUID**	**Tienda**	**Major**	**Zona de la tienda**	**Minor**
Vialia	00000000-0000-0000-0000-000000000001	Zara	100	Escaparate	0
				Puerta	1
				Cajas	2
				Probadores	3
		Mango	101	Escaparate	0
				Puerta	1
				Cajas	2
				Probadores	3

Tabla 3.1. Ejemplo asignación UUID, major y minor.

3.2.2. Balizas de seguridad

La seguridad del sistema se sustenta en el uso de balizas Eddystone. Estas balizas están destinadas a verificar la posición de los usuarios, proporcionando la seguridad necesaria al sistema para evitar que en el caso de que alguna baliza de posición haya sido copiada, el atacante pueda engañar el sistema para recibir las ofertas desde otro sitio que no sea la propia tienda. De esta forma también se obtendrán estadísticas más fiables del uso que los usuarios hacen del centro comercial.

El funcionamiento de las balizas de seguridad será similar las balizas Eddystone-EID de Google [17]: en lugar de enviar un mismo identificador constante durante todo el tiempo de funcionamiento, estas balizas van a mandar un identificador diferente en cada trama, de forma que solo conociendo una clave secreta, una tercera parte sería capaz de comprobar que todos los identificadores corresponden a la misma baliza. Sin embargo, se ha descartado el uso de esta tecnología ya que la validación de los datos enviados por el usuario depende de un servicio Google y en el ámbito del presente TFG se prefiere tener una solución independiente.

No se ha podido usar el protocolo iBeacon para la seguridad debido a que el formato de la información transmitida en este protocolo es mucho más limitado (solo permite el envío de identificadores) que el usado en las tramas del protocolo Eddystone.

Para conseguir una solución independiente al servicio de Google, se ha definido un formato propio que sigue la misma filosofía de Eddystone-EID. Para ello a cada baliza de seguridad se le asocia un iBeacon en particular, y transmitirá el resultado de aplicar la función MAC al identificador del iBeacon.

Las balizas de seguridad podrán ir colocadas junto a los iBeacons situados en los escaparates de las tiendas, aunque también se podrían usar dispositivos que transmitieran de manera alterna tramas de iBeacon y tramas de Eddystone, de esta forma cuando un usuario llegue a una tienda, su teléfono recibirá ambas tramas y se llevará a cabo el proceso de autenticación por cada iBeacon. En la figura 3.9. se muestra un ejemplo de cómo podrían ir situadas las balizas de seguridad y de posición en los escaparates de las tiendas.

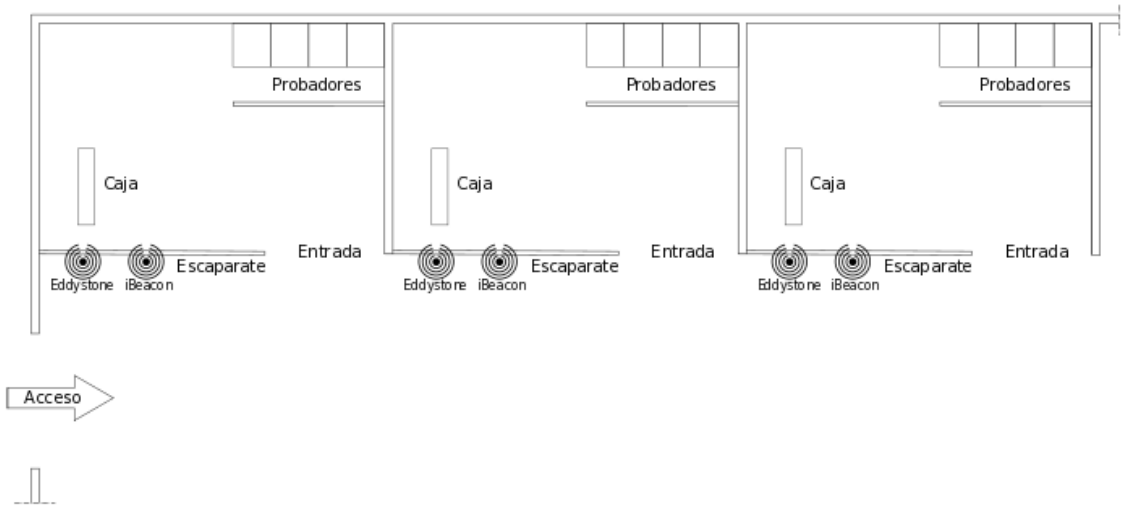


Figura 3.9. Esquema de situación de Eddystone en el centro comercial.

3.3. Servicio de soporte

Es el servidor de la aplicación móvil, en él se realizan las funciones de almacenamiento y gestión de los datos del usuario, así como realización de estadísticos y analíticas del uso de la aplicación y funciones en *backend*.

3.3.1. Analíticas y estadísticas de uso de la aplicación

Para la recolección de datos de posición se han creado dos eventos personalizados, estos eventos están asociados a cuando un usuario entra o sale del área de cobertura de una baliza de posición que corresponda a alguna de las tiendas de su lista. De esta forma se podrá llevar un control sobre las visitas que reciben las diferentes tiendas y cómo se mueven los usuarios dentro de las tiendas, pudiendo determinar cómo afectan las diferentes campañas de ofertas al comportamiento de los usuarios. Para respetar la privacidad de los usuarios los eventos no identificarán a un usuario en concreto, sino que usarán pseudónimos.

Las estadísticas generadas a partir del recuento de estos eventos pueden no ser cien por cien fiables, esto se debe a la relativa facilidad con la que un usuario con conocimientos de informática podría falsificar un iBeacon, y por tanto engañar al sistema, por ello es necesario contrastar la información recogida por estos eventos con los registros de posiciones validadas.

3.3.2. Funciones del *backend*

Como se explica con anterioridad, la seguridad del sistema viene dada por el uso de una segunda baliza, la cual envía como datos a la aplicación el resultado de calcular una función MAC al identificador del iBeacon al que está asociado. El teléfono móvil al entrar en el área de dicha baliza escribirá una nueva entrada en la base de datos en la que indicará el valor de los datos recibidos. Este nuevo registro lanzará una función en *backend*, en la que se comprueba la veracidad de la información enviada por el usuario. Si la información enviada por el usuario es correcta la posición del usuario queda verificada y por tanto se registrará en la base de datos, además el usuario recibirá la actual oferta de la tienda.

3.3.3. Base de datos

En la base de datos será necesario guardar información relativa a diferentes entidades. En la figura 3.10. se muestra un diagrama donde aparecen todas las entidades que necesitan ser almacenadas en la base de datos, así como sus campos y las relaciones entre ellas.

- Usuario: la entidad “Usuario” consta de dos campos: el campo “pseudonimo”, que guardará el pseudónimo usado por el usuario para el envío de eventos, este cambiará cada vez que el usuario inicie una nueva sesión en la aplicación; y el campo “username”, que guardará el nombre de usuario, este puede ser cambiado por el mismo en cualquier momento desde la aplicación. En cuanto a las relaciones con otras entidades, podemos ver que la entidad “Usuario” está relacionada con las entidades “Tienda”, “Oferta” y “Posicion_Validada”, estas relaciones indican que un usuario puede tener ninguna o varias tiendas seleccionadas con las que compartirá sus datos y recibirá sus ofertas, ninguna o varias ofertas acumuladas y ninguna o varias posiciones validadas.
- Tienda: esta entidad consta de un único campo, “nombre”, en él se guardará el nombre de la tienda. La entidad “Tienda” está relacionada con las entidades “Beacon”, indicando en su relación que una tienda

puede tener una o muchas balizas; y “Oferta”, una tienda puede tener ninguna o muchas ofertas.

- Beacon: se compone de dos campos: “major” y “minor”, estos campos almacenarán el major y el minor de una baliza de posición respectivamente.
- Oferta: la entidad “Oferta” tiene tres campos: “descripcion” en el que guardará los detalles de la oferta, “periodo_de_validez” en el que se indica hasta que fecha será válida la oferta y “franja_horaria_de_envio” en el que se guarda en qué horario debe ser enviada la oferta a los clientes.
- Posición validada: esta última entidad se compone de un solo campo, “fecha”, en el que se guardará la fecha y hora en la que la posición fue validada, además esta entidad tiene una relación del tipo uno a uno con la entidad “Beacon”, lo que quiere decir que cada posición validada estará asociada a una baliza.

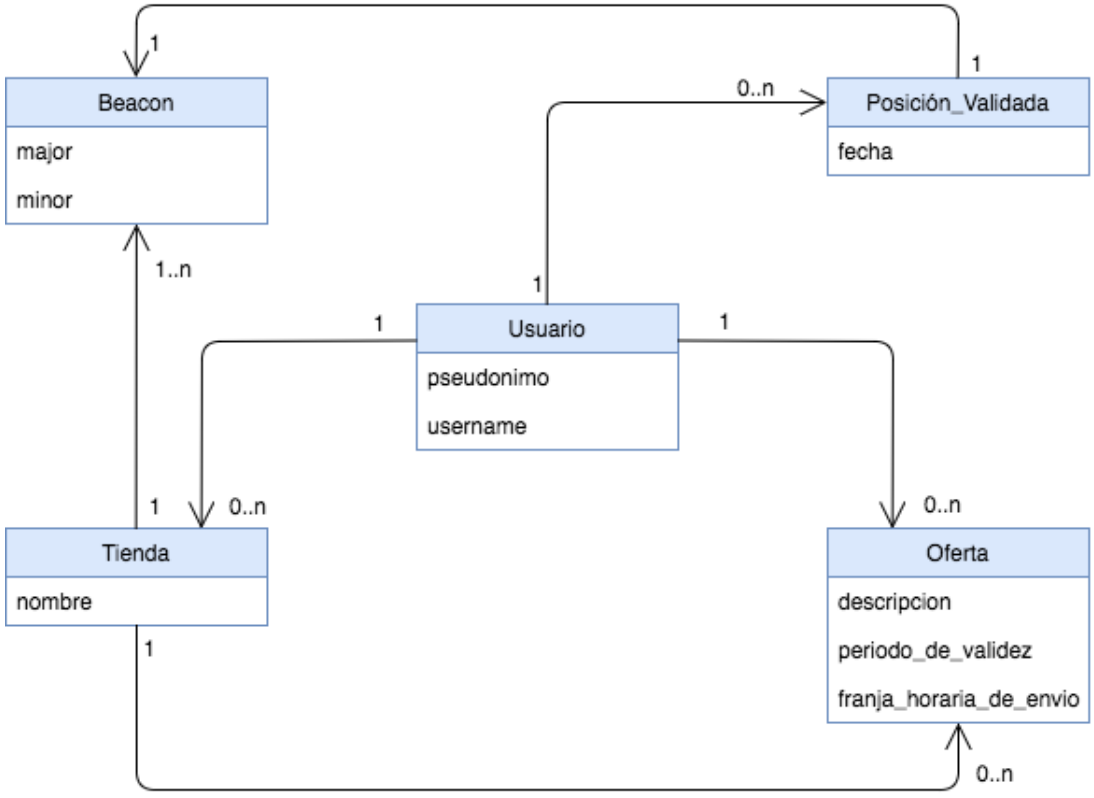


Figura 3.10. Diagrama entidad/relación de la base de datos.

3.4. Aplicación móvil

Es la interfaz con la que interactúa el usuario, a través de esta interfaz podrá seleccionar sus preferencias en cuanto a comercios con los que quiere colaborar y a políticas de acceso de grano, además podrán ver desde esta aplicación los beneficios proporcionados por las tiendas y hacer diferentes gestiones de su propia cuenta de usuario.

El registro e inicio de sesión en la aplicación se podrán realizar mediante diferentes métodos: autenticación por correo electrónico y contraseña, autenticación por cuenta de Google y autenticación por cuenta Facebook. Además, la aplicación contará con un servicio de restablecimiento de contraseña para que en caso de olvido los usuarios puedan recuperar sus cuentas.

Se establecerá en la aplicación el uso de pseudónimos para evitar de esta forma que los datos compartidos por los usuarios lleven su propio nombre o dirección de correo electrónico. Los pseudónimos serán cadenas de caracteres elegidas de manera aleatoria por la aplicación. Cada vez que un usuario reinicie su sesión se le proporcionará un pseudónimo nuevo, garantizando así la privacidad de los usuarios. Cuanto más frecuente sea el cambio de pseudónimo de un usuario, menos información podrán obtener de él los comerciantes, ya que de cara a ellos cada vez que un usuario cambia de pseudónimo este se convierte en un nuevo usuario del que no se tienen datos anteriores.

La aplicación contará con una lista de selección donde el usuario podrá indicar cuales son las tiendas con las que quiere compartir sus datos y por tanto recibir ofertas.

Las promociones de las tiendas llegarán a la aplicación del usuario en forma de notificaciones *Push*, y además serán almacenadas en la base de datos para que los usuarios puedan consultarlas en cualquier momento.

3.4.1. Interfaz gráfica de la aplicación

La aplicación consta de diferentes vistas:

- En la figura 3.11. se muestra la vista inicial, en ella los usuarios pueden registrarse o iniciar sesión una vez registrados, estas acciones se podrán llevar a cabo de tres maneras diferentes: autenticación por correo y contraseña, autenticación mediante Google o autenticación mediante Facebook. En esta vista los usuarios además tienen la opción de poder recuperar sus contraseñas en caso de olvido.

Figura 3.11. Vista inicial.

- Una vez el usuario ha iniciado sesión, accederá a la vista principal, que es la que se muestra en la figura 3.12. En esta vista se muestra información del usuario como su foto, su nombre de usuario y su dirección de correo electrónico, además también se muestra información acerca de la posición del usuario en relación al centro comercial, es decir, si esta fuera o dentro de este, dentro de alguna tienda concreta, etc. Las acciones que el usuario puede realizar en esta vista son: cambio del nombre de usuario, actualización de la posición actual y cerrar sesión.

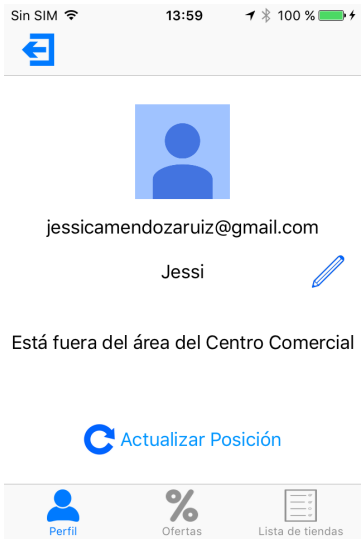


Figura 3.12. Vista principal.

- La vista de ofertas, tal como se muestra en la figura 3.13., contiene una lista con los descuentos disponibles que tiene el usuario. Los descuentos podrán ser eliminados por el usuario en cualquier momento. En esta vista el usuario también tiene la opción de cerrar sesión.

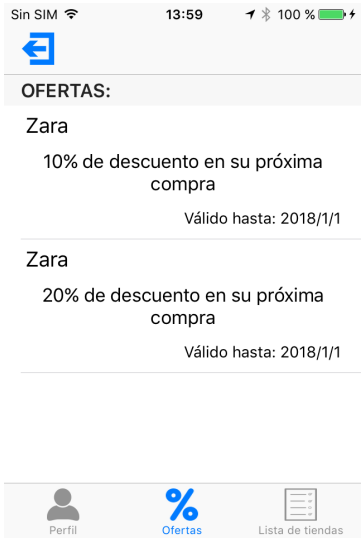


Figura 3.13. Vista de ofertas.

- La vista de tiendas tiene una lista de todas las tiendas que hay en el centro comercial, el usuario podrá seleccionar en esta lista las tiendas de las cuales quiere recibir ofertas y por tanto enviarles sus datos de posición, además de cerrar sesión. En la figura 3.14. se muestra la vista de tiendas.

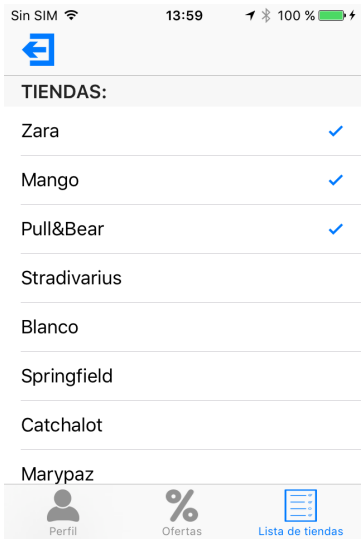


Figura 3.14. Vista de tiendas.

Capítulo 4. Implementación

La implementación del sistema de fidelización en *retail* diseñado se divide en tres bloques. Por un lado, se encuentra la configuración y el despliegue de la infraestructura, es decir, la configuración de las balizas de posición y seguridad; en segundo lugar, se encuentran las configuraciones realizadas en el servidor, como la creación de la base de datos o la definición de las funciones a realizar en el *backend*; y por último el desarrollo de la aplicación.

4.1. Configuración y despliegue de la infraestructura de locaclización

Debido a los diferentes tipos de protocolos que se utilizarán para las balizas de posición y para las balizas de seguridad, las configuraciones de estas se llevarán a cabo de diferente forma.

4.1.1. Balizas de posición

Para la simulación de las balizas de posición se ha empleado una aplicación móvil llamada Beacon Toy. Esta aplicación hace uso del Bluetooth del teléfono móvil para transmitir las tramas *broadcast* de las balizas configuradas y activadas, permitiendo que varias balizas estén activas al mismo tiempo. Para que la aplicación pueda funcionar de forma correcta el teléfono móvil debe disponer del modo periférico BLE.

En la imagen 4.1. se muestra una captura de la aplicación en la que se pueden ver los diferentes formatos de balizas que soporta, las balizas de posición usarán el protocolo iBeacon tal como se explica en el apartado 3.2.1.

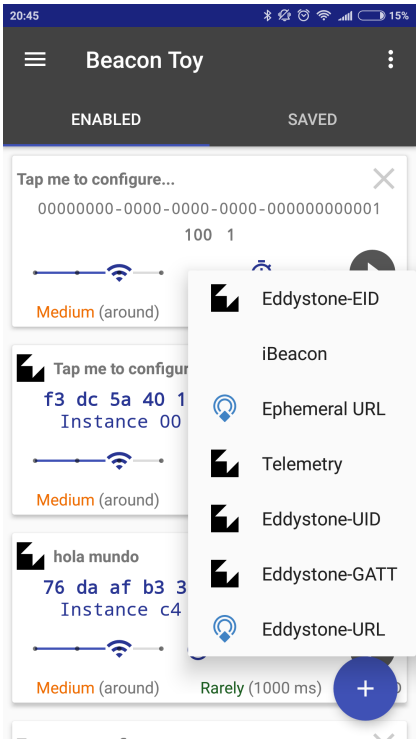


Figura 4.1. Formatos de balizas BLE soportados por Beacon Toy.

En la imagen 4.2. se muestra una captura de la pantalla de configuración de iBeacon de la aplicación Beacon Toy, en ella se puede apreciar los campos UUID, major y minor, además de la frecuencia a la que se enviarán las tramas y la potencia de transmisión de la señal BLE.

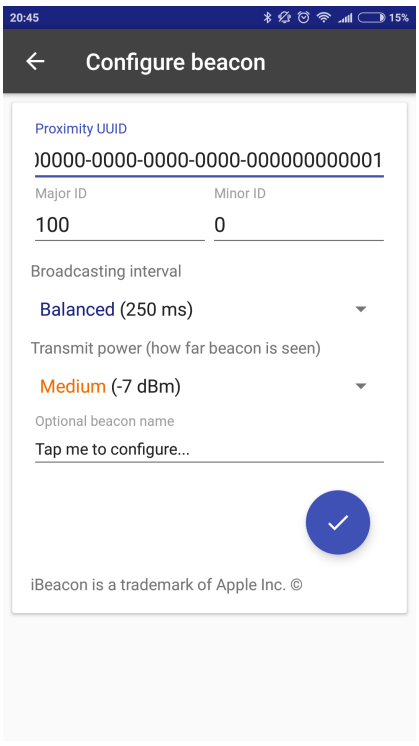


Figura 4.2. Configuración de iBeacon en Beacon Toy.

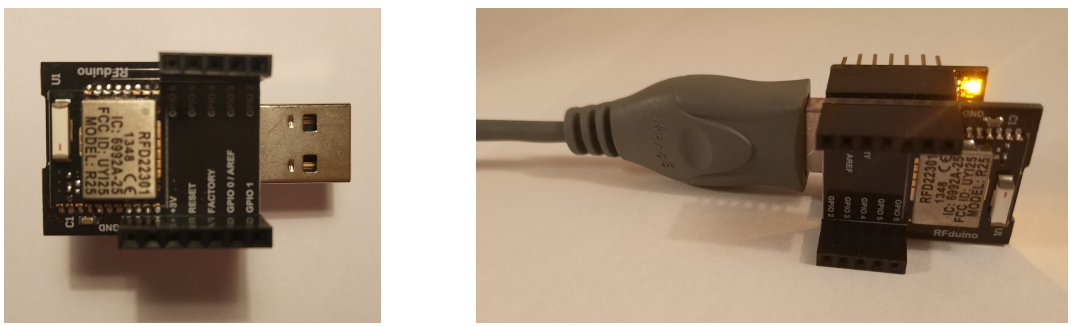
4.1.2. Balizas de seguridad

Las balizas de seguridad usarán un formato propio basado en Eddystone-EID, como se explica en el apartado 3.2.2., cada baliza de seguridad estará asociada a una baliza de posición concreta, y transmitirá el resultado de aplicar la función HMAC a un mensaje compuesto por: el major del iBeacon al que está asociado la baliza de seguridad y la fecha y hora actual. El valor transmitido por el dispositivo Eddystone, se actualizará cada minuto, evitando de esta forma que las balizas de seguridad puedan ser copiadas. Esta información será enviada en los veinte bytes de la trama de datos que permite Eddystone.

La función hash criptográfica usada para el cálculo del valor HMAC será SHA-1 (Secure Hash Algorithm - 1), esto se debe a que esta función produce una salida de longitud fija igual a veinte bytes, que es la longitud de la trama de datos de las balizas Eddystone. Los datos de entrada a la función pueden tener una longitud de hasta 2^{64} bits.

Para la simulación de las balizas de seguridad se ha utilizado un RFduino, que es un microcontrolador que dispone de tecnología Bluetooth 4.0 y por tanto es compatible con BLE. El RFduino utilizado ya había sido previamente programado para calcular la función HMAC del major de una baliza dada y la fecha y hora actual, actualizando el valor transmitido cada minuto.

Para la correcta sincronización de las balizas de seguridad con la función que valida la posición de los usuarios es necesario saber la clave utilizada en la función HMAC que ejecuta la baliza de seguridad. En la figura 4.3. se muestras dos imágenes del RFduino utilizado para el trabajo.



a) b)

Figura 4.3. RFduino.

4.2. Configuración y despliegue de los servicios de soporte

La aplicación desarrollada está destina a móviles que hagan uso del sistema operativo iOS. Por ello, lo primero que se necesita es un ID de app, para conseguir este identificador se ha creado una cuenta Apple Developer. Es muy importante recordar que el Bundle ID elegido al crear el ID de app será el que luego debemos usar en el proyecto de Xcode, figura 4.4. Si queremos que nuestra aplicación soporte el envío de notificaciones *Push*, como es el caso del presente TFG, deberemos marcar la casilla correspondiente en la creación del ID de app.

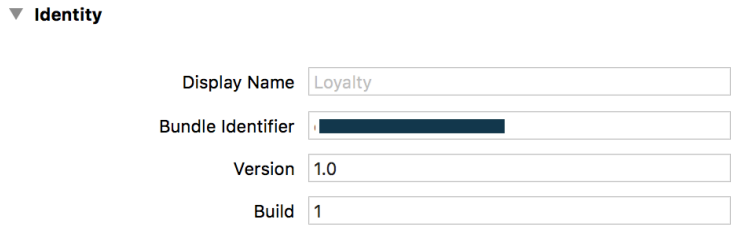


Figura 4.4. Bundle ID en Xcode.

Para probar la aplicación en un móvil real necesitaremos un perfil de aprovisionamiento para desarrollo. Este perfil se genera accediendo a nuestra cuenta de Apple Developer, al crearlo debemos seleccionar el ID de app de la aplicación que se desea probar. Una vez creado el perfil de aprovisionamiento, debemos descargarlo y haciendo doble *click* sobre el instalarlo en el ordenador donde se desarrollará la aplicación. Este perfil será luego utilizado en Xcode, como se muestra en la figura 4.5.

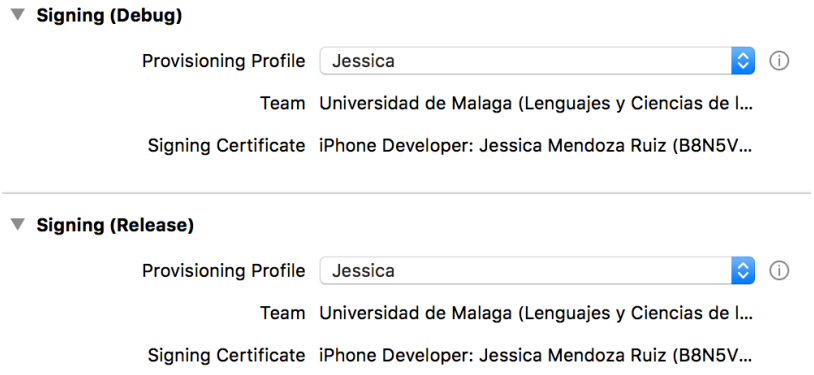


Figura 4.5. Perfil de aprovisionamiento para desarrollo en Xcode.

Dado que el servidor utilizado para la aplicación móvil es Firebase de Google, será necesario vincular el proyecto de Xcode en el que se desarrollará la aplicación con dicho servidor, para ello se han seguido los pasos indicados en la documentación de Firebase [31].

4.2.1. Configuración de los métodos de autenticación

La autenticación en la aplicación se lleva a cabo mediante el servicio Authentication de Firebase, dentro de los diferentes métodos que este servicio permite habilitar para el inicio de sesión, se han decido utilizar, tal como se indica en el apartado 3.4., la autenticación mediante correo electrónico y contraseña, la autenticación mediante cuenta de Google y la autenticación mediante cuenta de Facebook.

Para activar el inicio de sesión mediante autenticación por correo electrónico y contraseña, bastará con entrar en la consola de Firebase y habilitar la opción, como se muestra en la figura 4.6.

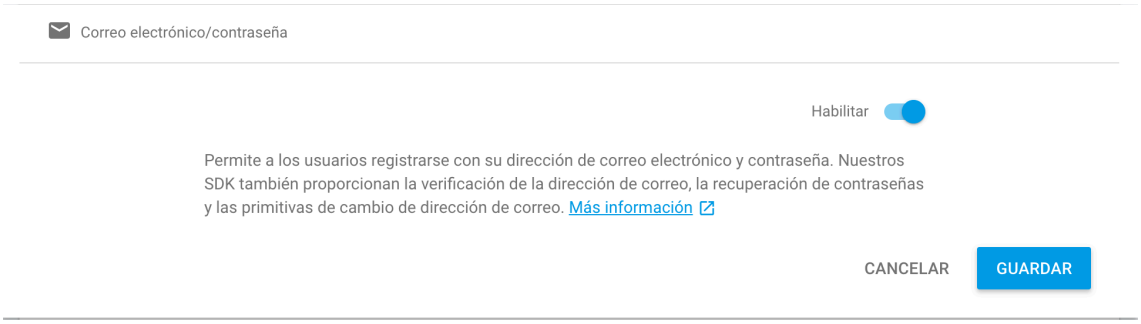


Figura 4.6. Habilitar método de autenticación por correo electrónico y contraseña.

La activación del inicio de sesión mediante cuenta de Google requiere de un paso más, este consiste en agregar un esquema de URL a nuestro proyecto de Firebase. Para ello basta con copiar el valor que aparece en el campo RESERVED_CLIENT_ID, figura 4.7., del archivo de configuración GoogleService-Info.plist en un nuevo tipo de URL, figura 4.8.

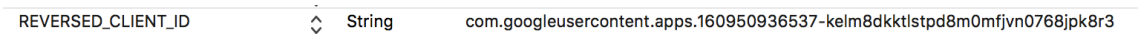


Figura 4.7. RESERVED_CLIENT_ID.

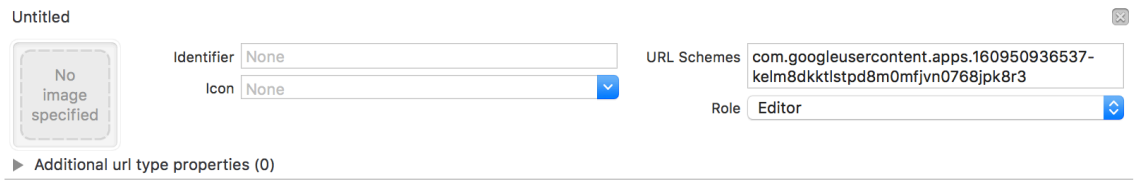


Figura 4.8. Tipos de URL en Xcode.

Por último, para habilitar el inicio de sesión mediante cuenta de Facebook, será necesario entrar en la web de Facebook Developers y crear en ella una aplicación asociada a nuestro proyecto de Xcode, de la cual necesitaremos el ID y el secreto. En el campo de URI de redireccionamiento de OAth válidos de la configuración de la aplicación de Facebook deberemos poner la que nos aparece en la consola de Firebase, esa URI es la mostrada en la figura 4.9.

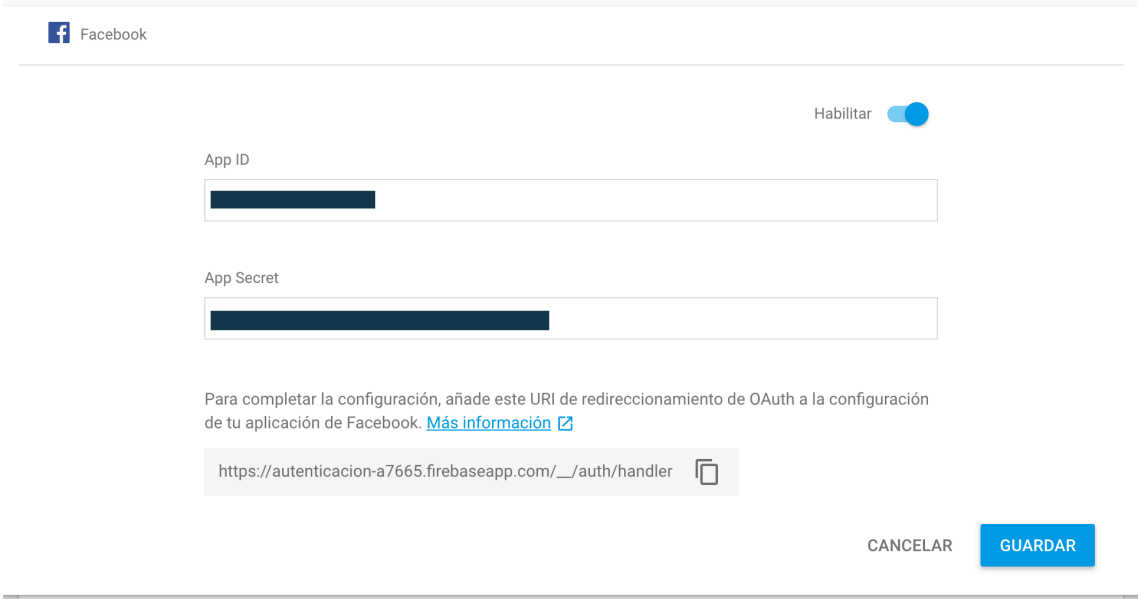


Figura 4.9. Habilitar método de autenticación por cuenta de Facebook.

4.2.2. Estructura de la base de datos

La base de datos utilizada para guardar toda la información proporcionada por la aplicación móvil es la Realtime Database de Firebase. El acceso a la base de datos de la aplicación quedará restringido a aquellos usuarios previamente autenticados.

En el apartado 3.3.3. se citaban las entidades que, a priori, deberían almacenarse en la base de datos, finalmente, la estructura resultante de la base de datos se divide en cuatro partes, como se muestra en la figura 4.10.

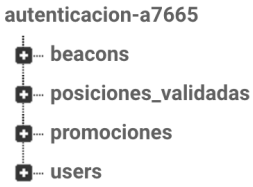


Figura 4.10. Base de datos.

Al comparar las entidades citadas en el apartado 3.3.3. con las ramas finalmente creadas en la base de datos, se puede observar que falta una rama para la entidad “Tienda”, la decisión de no crear tal rama viene motivada por varios factores. Por un lado, desde la aplicación se necesita la información de las tiendas de forma continua con lo que para evitar constantes consultas a la base de datos del servidor se ha decido guardar esta entidad como una constante en el código de la aplicación. Por otro lado, los contenidos relacionados con las tiendas son estáticos y están asociados a la aplicación. Además, esto también nos permitió experimentar con dos tipos de persistencia dentro de la aplicación.

A continuación, se describe la información almacenada en cada rama de la base de datos:

1. La rama de “beacons” mantiene una lista de las balizas instaladas en las tiendas para que la aplicación pueda monitorizarlos. Son datos que la aplicación en ningún momento puede modificar, solo consultarlos. Debido a las limitaciones del proceso de monitorización en segundo plano de iOS (ver apartado 4.3.3.) nos hemos visto obligados a almacenar, no solo la lista de balizas, sino la lista de balizas vecinas para cada baliza (regiones vecinas). En la figura 4.11. se muestra parte de la base de datos referente a la rama “beacons”.

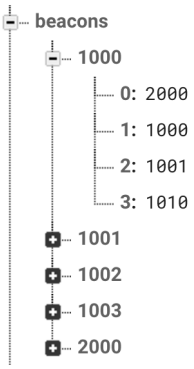


Figura 4.11. Sección “beacons” de la base de datos.

2. En la segunda rama se almacenan las posiciones validadas, tal como se muestra en la figura 4.12., para proteger la identidad de los usuarios se usarán pseudónimos al igual que en el registro de eventos, de esta forma se podrá saber que un usuario ha estado en una tienda determinada pero no quien es el usuario. Esta segunda parte tampoco podrá ser modificada por la aplicación móvil.

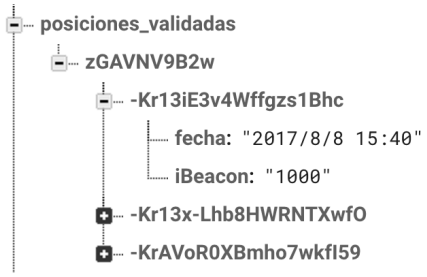


Figura 4.12. Sección "posiciones_validadas" de la base de datos.

3. La rama "promociones" guarda las actuales ofertas de las diferentes tiendas, estos datos tampoco podrán ser modificados por la aplicación del usuario. Para cada tienda (identificada con su major) se guarda la oferta, la fecha de vencimiento de la misma y una etiqueta que indica cuándo se ofrecerán los descuentos, estos campos se pueden apreciar en la figura 4.13.

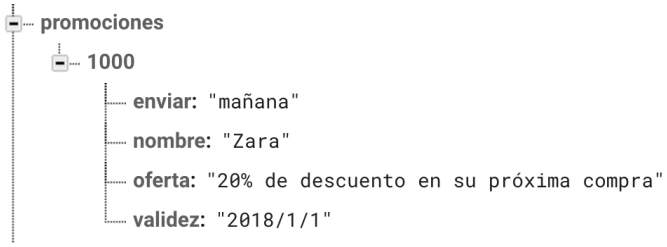


Figura 4.13. Sección "promociones" de la base de datos.

4. La cuarta rama de la base de datos contiene información específica de los usuarios. Cada usuario dentro de la base de datos se identifica por el UID asignado por Firebase en el momento del registro. La información almacenada para cada usuario será su nombre, la lista de tiendas de las cuales se desea recibir ofertas, el *token* actual que usa para recibir notificaciones, el MAC de la última baliza de seguridad detectada, el major del último iBeacon detectado, el pseudónimo que utiliza el usuario y la lista de ofertas que tiene acumuladas el usuario. Estos datos si pueden ser modificados por la aplicación de usuario. En la figura 4.14.

se muestra la rama de la base de datos dónde se guarda la información relativa a un usuario.

Figura 4.14. Sección "users" de la base de datos.

4.2.3. Función de verificación de la posición de los usuarios

Para verificar la posición de los usuarios se usa el servicio Cloud Function de Firebase. Esta función además también será la encarga de, una vez validada la posición, enviar al usuario la oferta actual de la tienda, en caso de que haya alguna.

La función será lanzada cuando la aplicación móvil, tras detectar una baliza de seguridad y comprobar que el iBeacon al que está asociado pertenece a alguna de las tiendas de la lista del usuario, escribe en la base de datos del usuario el valor HMAC recibido. Esta función realizará dos comprobaciones:

1. Comprobar si el valor HMAC es el correcto para una tienda y momento determinados: la función Cloud Function calculará el HMAC para el major dado y tomando la hora actual del sistema. Para la realización de

esta comprobación es necesario que la baliza de seguridad y Firebase estén sincronizados, además tendrán que compartir una clave criptográfica secreta y usar una función hash común (en este caso SHA-1). De esta forma queda validada la posición del usuario, y se registrará la misma en la Realtime Database:

```
exports.validarUsuario =
functions.database.ref('/users/{PushId}/seguridad').onWrite(event => {
      const original = event.data.val();
      console.log('UID', event.params.PushId, 'HMAC enviado por la
      app', original);

      const uid = event.params.PushId;
      //Eliminar los espacios de la cadena original
      var orig = original.replace(/\s+/g, '');
      //Calcular función HMAC
      return calcularHMAC(uid).then(resul => {
            //Si MAC calculado y MAC original son iguales
            if (resul.trim() === orig.trim()){
                  console.log('Coinciden');
                  //Enviar cupón
                  return sendCouponViaFCM(uid);
            }else{ //Si MAC calculado y MAC original no son iguales
                  console.log('No coinciden, hacer segunda prueba',
                  'calculado', resul, 'original', orig);
                  //Esperar 3 segundos y volver a comprobar
                  sleep(3000, function() {
                        //Coger valor HMAC de la base de datos
                        return admin.database().ref(`/users/${uid}
                        /seguridad`).once("value").then(function(snapsh
                        ot) {
                              var original2 = snapshot.val();
                              //Eliminar los espacios de la cadena
                              original
                              var orig2 = original2.replace(/\s+/g,
                              '');
                              //Vuelve a calcular HMAC
                              return calcularHMAC(uid).then(resul2 =>{
                                    //Si MAC calculado y MAC original
                                    son iguales
                                    if (resul2.trim() ===
                                    orig2.trim()){
                                          console.log('Coinciden en la
                                          segunda prueba');
                                          //Enviar cupón
                                          return sendCouponViaFCM(uid);
                                    }else{
                                          console.log('No coinciden',
                                          'calculado', resul2,
                                          'original', orig2);
                                    }
                              });

                        });

                  });
            } }); });
```

2. Comprobar si el comercio en el que se encuentra el usuario está ofreciendo alguna promoción y si el usuario aun no la tiene en su lista de ofertas:

```
//Solo lo debería enviar si la tienda está en la lista de tiendas que
el usuario quiere ver y si no ha recibido ya esta oferta
function sendCouponViaFCM(uid) {
      //Mirar si el iBeacon ultimo está en la lista de tiendas del
      usuario
    return admin.database().ref(`/users/${uid}/iBeacon`)
      .once("value").then(function(snapshot) { //Coge el iBeacon
            var iBeacon = snapshot.val();
            //Comprobar si el iBeacon se corresponde a alguna tienda
            de la lista
            return comprobarLista(uid,iBeacon).then(pertenece => {
                  if (pertenece == true){
                        //Registrar posición validada
                        registrarPosicion(uid,iBeacon);
                        //Comporbar si la tienda ofrece alguna
                        promoción
                        return comprobarOferta(uid,iBeacon).then(hay
                        =>{
                              if (hay == true){
                                    //Mirar si el usuario tiene la
                                    oferta, en caso negativo enviar
                                    notificación
                                    return
                                    comprobarSiUsuarioTieneOferta(uid,
                                    iBeacon);
                              }else{
                                    console.log('No hay oferta');
                              }
                        });
                  }else{
                        console.log('Tienda no está en la lista',
                        iBeacon);
                  }
            });
      });
}
```

Si las dos condiciones se cumplen, se envía al usuario una notificación con la oferta disponible en la tienda, además dicha oferta será añadida a su base de datos para que el usuario pueda hacer uso de ella en cualquier momento.

4.2.4. Configuración de las notificaciones

Como se menciona en el apartado 2.1.5., el envío de notificaciones tipo *Push* a los dispositivos de Apple hacen uso del servicio APNs, siendo necesario obtener un certificado criptográfico que nos autentica ante este servicio. Este certificado lo podremos obtener entrando en nuestra cuenta de Apple Developer[32][33].

Una vez obtenido el certificado, será necesario importarlo a las configuraciones de la aplicación en la consola de Firebase, como se muestra en la figura 4.15.

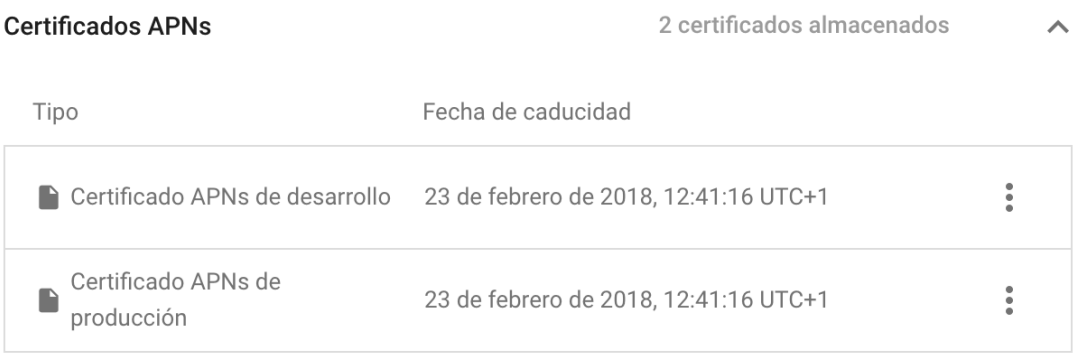


Figura 4.15. Añadir certificado de APNs a Firebase.

4.3. Desarrollo de la aplicación

Para completar el sistema propuesto se ha desarrollado una aplicación móvil para clientes en la cual estos pueden administrar sus perfiles de usuario, elegir las tiendas de las que desean recibir ofertas y por tanto enviarles su información y además gestionar las ofertas acumuladas.

Esta aplicación está destinada a su uso en dispositivos iOS, y por ello ha sido desarrollada en el entorno de desarrollo Xcode y programa en lenguaje Swift.

El código completo de la aplicación se puede encontrar en el siguiente repositorio: https://bitbucket.org/jessimr/tfg.

4.3.1. Autenticación de usuarios

El primer paso en la implementación de la aplicación es la definición de las funciones mediante las cuales los usuarios podrán registrarse en la aplicación, iniciar sesión y cerrar sesión. Como se cita anteriormente la aplicación habilita tres de los métodos proporcionados por Firebase Authentication para la realización de estas funciones: autenticación mediante correo y contraseña, autenticación mediante cuenta de Google y autenticación mediante cuenta de Facebook.

Cuando un usuario se registre en la aplicación, Firebase le asignará un identificador único (UID) que identifica de manera unívoca a un usuario. El UID

identifica la sección de cada usuario dentro de la base de datos y solo será conocido por el propio usuario, es decir, ningún comerciante podrá conocer los UID de los usuarios.

Cada vez que el usuario inicie sesión en la aplicación, se le proporcionará un pseudónimo nuevo. El pseudónimo se guardará en la base de datos del usuario para su posterior uso tanto en el envío de eventos como en el registro de nuevas posiciones validadas. A continuación, se muestra la parte del código en la que se genera un nuevo pseudónimo y se guarda en la base de datos:

```
//Crear pseudónimo
func crearPseudonimo(){
      //Cadena aleatoria de 10 bytes de longitud
      let pseudonimo = randomStringWithLength(len: 10)
      print("Pseudónimo: \(pseudonimo)")
      //Referencia a la base de datos
      let bbdd = FIRDatabase.database().reference()
      //UID del usuario
      let uid = FIRAuth.auth()?.currentUser?.uid
      //Guardar pseudónimo en la base de datos del usuario
      bbdd.child("users/\(uid!)/pseudonimo").setValue(pseudonimo)
}
```

4.3.2. Selección de tiendas

Uno de los principales propósitos perseguidos en la aplicación es la protección de la privacidad de los usuarios, otorgándoles a estos el poder de decisión de a qué tiendas quieren enviar sus datos de localización. Con este fin, se ha implementado una vista en la aplicación en la que aparece una lista con todas las tiendas del centro comercial, en esta lista los usuarios podrán seleccionar o deseleccionar las tiendas que les interesen en cualquier momento.

Esta vista se asocia a la clase VistaConfiguracion del proyecto Xcode, en esta clase es dónde se realiza la lógica de la selección de tiendas. Para ello se hace uso de Firebase Realtime Database y de una segunda clase llamada ColeccionDeTiendas.

La clase ColeccionDeTiendas define un único atributo y varios métodos. El atributo es un *array* bidimensional en el que cada fila corresponde a la información de una tienda diferente y las columnas guardan la siguiente información: nombre de la tienda, major de los iBeacons asociados a la tienda, estado de la tienda (seleccionada o no por el cliente) y una clave. Los métodos

son utilizados para modificar el estado de las tiendas, añadir la clave asociada a esta y consultar el nombre de la tienda para un major dado. A continuación, se muestra el código de la clase ColeccionDeTiendas:

```
class ColeccionDeTiendas {
	//Conterido Array (nombre_tienda, major, estado, key(para poder
	borrar las tiendas de la base de datos))
	var tiendas: [[String]] = [["Zara","100","0","0"],
	                           ["Mango","101","0","0"],
	                           ["Pull&Bear","102","0","0"],
	                           ["Stradivarius","103","0","0"],
	                           ["Blanco","104","0","0"],
	                           ["Springfield","105","0","0"],
	                           ["Catchalot","106","0","0"],
	                           ["Marypaz","107","0","0"],
	                           ["Tezenis","108","0","0"],
	                           ["Calcedonia","109","0","0"],
	                           ["Oysho","110","0","0"],
	                           ["Parfois","111","0","0"],
	                           ["Misako","112","0","0"]]

	//Método para cambiar el estado de una tienda (seleccionado o no
	por el usuario)
	func switchState (position: Int, state: String){
		tiendas[position][2] = state
	}

	//Método para obtener el nombre de la tienda a partir del major
	asociado a ella
	func obtenerTienda (major: String) -> String{
		var tienda = ""
		for i in 0...(tiendas.count-1){
			if(tiendas[i][1] == major){
				tienda = tiendas[i][0]
			}
		}
		return tienda
	}

	//Método para guardar key
	func guardarKey (position: Int, key: String) {
		tiendas[position][3] = key
	}
}
```

La primera vez que el usuario abra la aplicación la lista aparecerá sin ninguna tienda seleccionada, cuando el usuario seleccione una tienda junto al nombre de esta aparecerá una marca de verificación o *tick*, además el major de la tienda será añadido a la base de datos del usuario. Cada tienda será guardada en la base de datos con una clave aleatoria generada justo en el momento de guardar el major en la base de datos, un ejemplo de estas claves se muestra en la figura 4.16., de esta forma se evita que la próxima vez que el usuario

seleccione una tienda, la información de esta nueva tienda sobrescriba la información ya guardada.

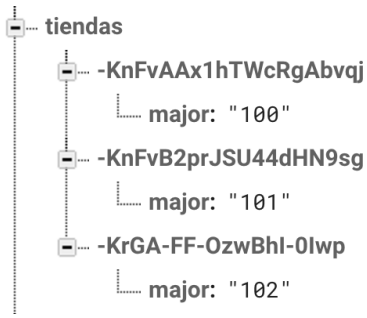


Figura 4.16. Claves aleatorias.

Si el usuario selecciona una tienda de la lista que ya tiene una marca de verificación, se procederá a la eliminación de dicha marca y al borrado de la tienda de la base de datos del usuario. Para este paso es importante saber cuál fue la clave generada cuando se guardó la tienda en la base de datos, esta es la razón de que se incluya la clave como columna del *array* bidireccional definido en la clase ColeccionDeTiendas. A continuación, se muestra la parte del código en la que se realiza la selección o deselección de una tienda y por tanto la escritura o borrado de la misma de la base de datos del usuario:

```
//Cuando el usuario seleccione una celda
func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath) {
      //Si la tienda está seleccionada por el usuario
      if (self.mistiendas.tiendas[indexPath.row][2] == "1"){
      //Cambiar estado a no seleccionada
      self.mistiendas.switchState(position: indexPath.row,
      state: "0")
            //Borrar tienda de la base de datos
            borrarTiendaDeBBDD(position: indexPath.row)
            //Recargar la tabla
            tableView.reloadData()
      }else{ //Tienda no seleccionada por el usuario
            //Cambiar estado a seleccionada
            self.mistiendas.switchState(position: indexPath.row,
            state: "1")
            //Añadir tienda a la la base de datos
            addTiendaABBDD(position: indexPath.row)
      }
}
```

La próxima vez que el usuario acceda a la lista, se realizará una lectura de las tiendas guardadas en la base de datos del usuario y se mostrarán como seleccionadas las tiendas incluidas en dicha base de datos. De igual forma se

creará un objeto de la clase ColeccionDeTiendas donde se guardará el estado de cada tienda y las claves (para permitir el borrado de las tiendas de la base de datos). El siguiente trozo de código es el que realiza las acciones anteriormente descritas:

```
//Leer las tiendas que el usuario tiene guardadas en su base de datos
func leerKeysDeLasTiendasDeLaBD(){
      var arrayKey: [String] = []
      //Leer keys asociado a las tiendas guardadas en la base de datos
      <- necesario para poder borrarlas
      self.ref2.observeSingleEvent(of: .value, with: { snapshot in
            for child in snapshot.children {
                  let key = (child as AnyObject).key!
                  arrayKey.append(key)
          }
            //Leer los major de las tiendas
            self.ref2.observe(FIRDataEventType.childAdded, with: {
            (snapshot: FIRDataSnapshot) in
                  let major = (snapshot.value as
                  AnyObject).object(forKey: "major") as! String
                  //Actualizar tiendas activas (solo al inicio)
                  for i in 0 ..< self.mistiendas.tiendas.count{
                        //Si la tienda está en la base de datos del
                        usuario y es la primera vez que se realiza la
                        actualización
                        if ((major ==
                        self.mistiendas.tiendas[i][1])&&(arrayKey.count
                        > 0)){
                              //Cambio el estado de la tienda a
                              seleccionado
                              self.mistiendas.switchState(position: i,
                              state: "1")
                              //Guardar key asociada a la tienda
                              self.mistiendas.guardarKey(position: i,
                              key: arrayKey[0])
                              //Eliminar la key ya guardada
                        arrayKey.remove(at: 0)
                        }
                  }
            self.tabla.reloadData() //Recargar tabla
            })
      })
}
```

Hay que tener en cuenta que la lectura de datos de la Realtime Database se realiza de forma asíncrona, por ello las funciones que requieran de esos datos para su correcto funcionamiento deben hacerse dentro de la propia función de lectura.

4.3.3. Localización de usuarios

La correcta localización de los usuarios es uno de los pilares fundamentales de la aplicación, dependiendo de dónde se encuentren los usuarios estos podrán

recibir ofertas (tras validar su posición) y enviar eventos a Firebase Analytics para que los comercios puedan realizar sus estadísticas.

La aplicación, gracias al uso de la librería Core Location de Apple, es capaz de detectar la cercanía de un iBeacon y la exactitud de dicha medida. Para ello el terminal mide la fuerza de la señal o RSSI, cuanto más fuerte es la señal recibida más exacta será la estimación de la proximidad realizada por el teléfono. Hay que tener en cuenta que existen numerosos factores, además de la distancia, que pueden afectar a la fuerza de la señal recibida (p. ej. obstáculos entre el iBeacon y el terminal receptor de la señal). Además, la aplicación podrá detectar cuando un usuario entra o sale de una región, esta acción será realizada aun cuando la aplicación no se está ejecutando en primer plano.

Medidas de la proximidad (Ranging)

Gracias al *ranging* la aplicación puede determinar la cercanía de los iBeacos, definiendo cuatro estados de la proximidad: *immedite* (muy cerca), *near* (cerca), *far* (lejos) y *unknown* (no se puede determinar la distancia a la que se encuentra el iBeacon).

Esta función sólo se realiza cuando la aplicación se está ejecutando en primer plano.

Cuando la aplicación detecta varios iBeacon al mismo tiempo, estos serán ordenados según su proximidad, de tal forma que en el *array* de iBeacons detectados el que se encuentra en la posición 0 siempre será el que está más cerca del usuario en un determinado momento.

En la clase VistaSesionIniciada del proyecto es donde se realiza la lógica de la localización de los usuarios cuando la aplicación trabaja en primer plano. El objetivo es detectar qué iBeacon es el que está más cerca del usuario y comprobar si este pertenece a la lista de tiendas seleccionadas por el usuario, en caso afirmativo se enviará un evento a Firebase.

La detección de iBeacons se ha realizado siguiendo el tutorial [34], aunque se han hecho algunas modificaciones para conseguir el propósito buscado:

- Ya que se espera que la detección de iBeacons en la aplicación se realice aun cuando la aplicación no se está ejecutando, se ha realizado la configuración mostrada en la figura 4.17. en el Info.plist, esta configuración produce que se muestre una alerta como la capturada en la figura 4.18. cuando el usuario instala la aplicación.

Figura 4.17. Permitir que la aplicación tenga acceso a la ubicación, aunque esta no se esté ejecutando.

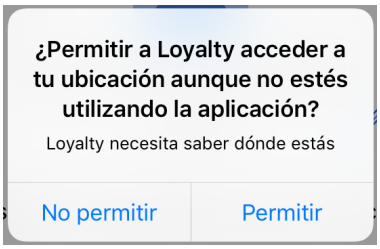


Figura 4.18. Preguntar al usuario por permiso para acceder a la ubicación.

- En el código siguiente es donde se comprueba si el usuario da su permiso para que la aplicación utilice su localización. En caso de que el usuario haya aceptado, además de comenzar el escáner o búsqueda de iBeacons, se realizará una lectura de la Realtime Database para saber cuál es la lista de tiendas del usuario.

```
//Cuando se detecte un cambio en los permisos de detección de iBeacon
en la aplicación
func locationManager(_ manager: CLLocationManager,
didChangeAuthorization status: CLAuthorizationStatus){
      if status == .authorizedAlways {
            if CLLocationManager.isMonitoringAvailable(for:
            CLBeaconRegion.self) {
                  if CLLocationManager.isRangingAvailable() {
                        //Leer las tiendas guardadas en la base de
                        datos del usuario
                        self.ref.observe(FIRDataEventType.childAdded,
                        with: { (snapshot: FIRDataSnapshot) in
                              let major = (snapshot.value as
                              AnyObject).object(forKey: "major") as!
                              String
                              print ("Tienda de la lista del usuario
                              \(major)")
                              //Actualizar tiendas activas
                              self.tiendas.append(Int(major)!)
                              //Empezar escaner
                              self.startScanning()
                        })
                  }
            }
      }
}
```

- Para poder detectar iBeacons, es necesario definir la región en la que se encuentran estos, en el caso de esta aplicación, cada iBeacon representará una región en sí mismo, inicialmente la única región monitorizada será la definida por el iBeacon que se encuentra a la entrada del centro comercial, a medida que el usuario va avanzando por el centro comercial, las regiones a monitorizar se irán actualizando dinámicamente (para más información sobre cómo se realiza esta actualización dinámica de regiones véase al capítulo 4.3.3.). Una vez definida la región inicial a monitorizar se realiza la activación de las funciones de *ranging* y *monitoring* como se muestra en el siguiente código, a estas se ha añadido la activación de la actualización de la ubicación del usuario, startUpdatingLocation() [35], de esta forma se conseguirá tener un aproximación más precisa de la localización del usuario.

```
//Empezar el escaneo
func startScanning() {
      //Inicialmente la única región que se va a monitorizar es la
      correspondiente al acceso al centro comercial
      let beaconRegion = CLBeaconRegion(proximityUUID: uuid, major:
      200, minor: 0, identifier: "MyBeacon2000") //Definir la región
      //Empezar monitoring
      locationManager.startMonitoring(for: beaconRegion)
      //Empezar ranging
      locationManager.startRangingBeacons(in: beaconRegion)
      //Empezar updatingLocation -> mejora la precisión de la
      ubicación de usuario
      locationManager.startUpdatingLocation()
      //Se inicializa el conjunto initialRegion
      initialRegion.insert(beaconRegion)
    }
```

- Aproximadamente cada segundo la función locationManager:didRangeBeacon:inRegion: es activada, en esta función es donde se lee la información de los iBeacons cercanos. Se ejecutará una función locationManager:didRangeBeacon:inRegion: por cada región que esté siendo monitorizada en un determinado momento. En la figura 4.19. se muestra el flujo de ejecución de la función locationManager:didRangeBeacon:inRegion:.

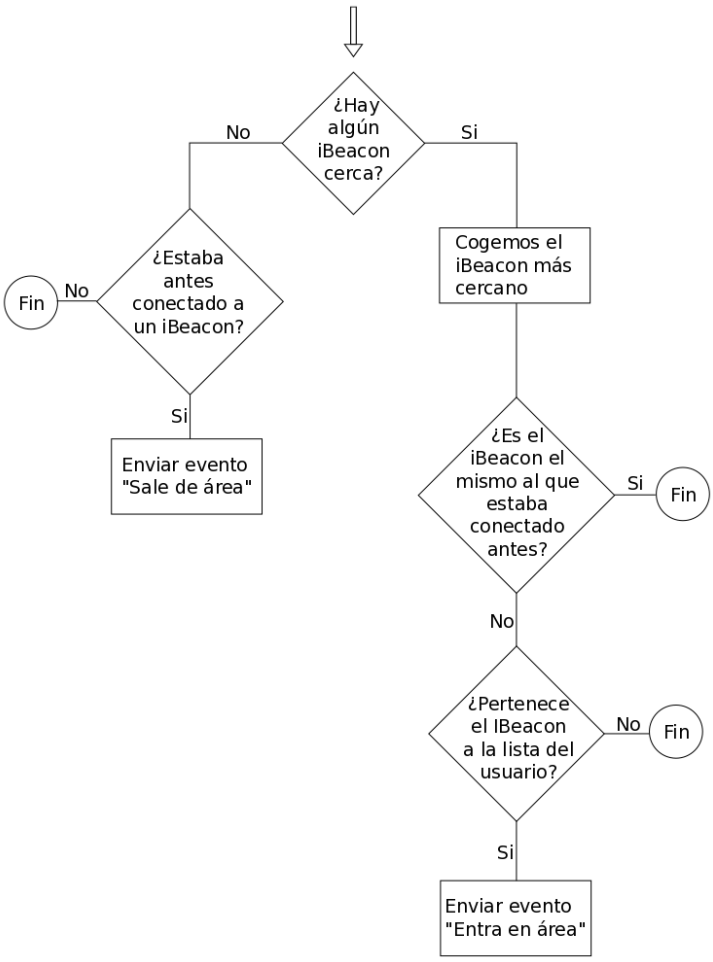


Figura 4.19. Diagrama de ejecución de la función locationManager:didRangeBeacon:inRegion:.

- El siguiente código, se aprovecha el conocimiento de la cercanía del usuario a un iBeacon para mostrar un mensaje en la aplicación del usuario dónde le indica dónde se encuentra:

```
//Función que determina la cercanía del usuario al iBeacon
func update(distance: CLProximity, major: NSNumber, minor: NSNumber) {
      UIView.animate(withDuration: 0.8) { [unowned self] in
            switch distance {
                  case .unknown:
                        self.distancia.text = "Está fuera del área del
                        Centro Comercial"
                  case .far:
                        self.mostrarTexto(major: major, minor: minor)
                  case .near:
                        self.mostrarTexto(major: major, minor: minor)
                  case .immediate:
                        self.mostrarTexto(major: major, minor: minor)
            }
      }
}
```

Monitoreo de regiones (Region Monitoring)

La aplicación es capaz de detectar cuando el usuario entra o sale de la zona de cobertura de una región. Las regiones pueden estar definidas simplemente por el UUID (la región estaría formada por todos los iBeacon pertenecientes al centro comercial), por el UUID y el major (regiones formadas por los iBeacon pertenecientes a una tienda) o por el UUID, el major y el minor (cada iBeacon define una región).

A diferencia del *ranging*, el monitoreo de regiones sigue funcionando aun cuando la aplicación no se está ejecutando, si un terminal entra o sale del área de una determinada región la aplicación será notificada y comenzará a trabajar en segundo plano.

En el contexto de la aplicación se considera importante el registro de eventos (Firebase Analytics) cuando un usuario entra o sale del área de cobertura de un iBeacon, ya que estos eventos servirán de utilidad a la hora de crear futuros estudios sobre el comportamiento de los usuarios, es por ello que se han definido las regiones teniendo en cuenta el UUID, el major y minor. De esta forma, cuando un usuario entre o salga del área de cobertura de una región, aun cuando la aplicación no se está ejecutando en primer plano, está será lanzada y los eventos quedarán registrados.

Para conseguir que la aplicación pueda detectar cuando un usuario entra o sale del área de cobertura de la región definida por un iBeacon se ha implementado en la clase AppDelegate las funciones de *monitoring*, de esta forma la aplicación no solo será capaz de detectar este comportamiento cuando la aplicación se está ejecutando en primer plano, sino que también lo detectará cuando la aplicación no se esté ejecutando, haciendo que esta pase a un primer plano. Para ello se ha seguido el tutorial [36] adaptándolo a las necesidades propias de la presente aplicación:

- Para detectar cuando un usuario entra en la región de cobertura de un iBeacon, se utiliza la función locationManager:didEnterRegion:, tal como se muestra a continuación:

```
//Cuando el usuario entre en la región definida por un iBeacon
func locationManager(_ manager: CLLocationManager, didEnterRegion
region: CLRegion) {
      if (region as? CLBeaconRegion) != nil {
      //Obtener major y minor de la región en la que entra el
      usuario
            let beaconRegion = region as! CLBeaconRegion
            let majorRegion = beaconRegion.major!
            let minorRegion = beaconRegion.minor!
            print ("Usuario entra en el area del iBeacon con major:
            \(majorRegion) y minor: \(minorRegion)")
            //Activa el ranging (actualización dinámica de áreas)
            locationManager.requestState(for: region)
      }
}
```

- Para detectar cuando un usuario sale de la región de cobertura de un iBeacon, se utiliza la función locationManager:didExitRegion: de la siguiente forma:

```
//Cuando el usuario salga de la región definida por un iBeacon
func locationManager(_ manager: CLLocationManager, didExitRegion
region: CLRegion) {
      if (region as? CLBeaconRegion) != nil {
            //Obtener major y minor de la región de la que sale el
            usuario
            let beaconRegion = region as! CLBeaconRegion
            let majorRegion = beaconRegion.major!
            let minorRegion = beaconRegion.minor!
            print ("Usuario sale del area del iBeacon con major:
            \(majorRegion) y minor: \(minorRegion)")
            //Activa el ranging (actualización dinámica de áreas)
            locationManager.requestState(for: region)
      }
}
```

Por simplicidad y para evitar duplicar código, estas funciones realizan una llamada a la función de *ranging* (locationManager:didRangeBeacon:inRegion:), que tras hacer las comprobaciones explicadas anteriormente enviará, en su caso, el evento correspondiente a Firebase Analytics.

Actualización dinámica de regiones a monitorizar

Como se especifica en el apartado 4.3.2. las regiones de balizas han sido definidas por su UUID, major y minor, es decir, cada baliza en sí misma representa una región, esto acarrea un importante problema, ya que las aplicaciones sólo pueden monitorizar un máximo de veinte regiones al mismo tiempo y suponemos que en un centro comercial habrá más de veinte balizas, por lo que no se podrán monitorizar todas las regiones al mismo tiempo.

La solución propuesta se basa en la actualización dinámica de las regiones que se deben monitorizar en cada momento, de esta forma dependiendo de dónde se encuentre el usuario su aplicación móvil monitorizará unas regiones u otras.

Para este propósito se guardan en la Realtime Database de Firebase información sobre las regiones vecinas a una determinada región, de tal forma que cuando la aplicación detecta que el usuario entra en una región, realiza una consulta a la base de datos para saber qué regiones debe empezar a motorizar y en cuales debe parar el monitoreo (también se activa o desactiva el *ranging* de dichas regiones).

Si tomamos como ejemplo la figura 4.20., el Cliente 1, que se encuentra fuera del centro comercial, sólo necesitaría monitorizar la región creada por el iBeacon1; el Cliente 2, que está entrando en el centro comercial, monitorizará las regiones definidas por los iBeacons 1 y 2; el Cliente 3, que está en el escaparate de la tienda, monitorizará las regiones de los iBeacons 1, 2 y 3; por último, el Cliente 4, que está entrando en la tienda, parará de monitorizar la región del iBeacon 1 y activará el monitoreo de las regiones formadas por los iBeacons 4 y 5, además seguirá monitorizando las regiones de los iBeacons 2 y 3.

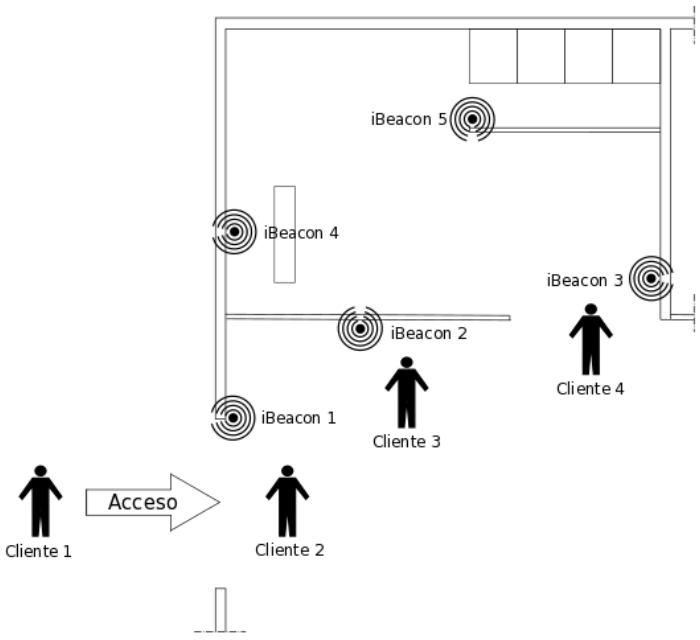


Figura 4.20. Diagrama ejemplo de actualización dinámica.

En el código de la aplicación, la actualización dinámica de las regiones a monitorizar se lleva a cabo de manera conjunta con la localización del usuario,

de manera que cuando obtenemos el iBeacon más cercano al usuario dentro de la función locationManager:didRangeBeacon:inRegion: es cuando se produce la actualización de áreas. Para ello, como se menciona con anterioridad, la aplicación hará una consulta a la base de datos de la que obtendrá las regiones vecinas a la actual, a continuación se creará un conjunto formado por todas las nuevas regiones vecinas, se comparará este conjunto con las regiones que está siendo actualmente monitorizadas y se parará el monitoreo de aquellas regiones que están siendo monitorizadas pero no están entre las nuevas regiones vecinas, así mismo se activará el monitoreo de aquellas regiones que estando en el conjunto de las nuevas regiones vecinas no están siendo monitorizadas todavía.

En la figura 4.21. un esquema de cómo funcionaría finalmente la función locationManager:didRangeBeacon:inRegion: para que se lleve a cabo de manera correcta la actualización dinámica de regiones.

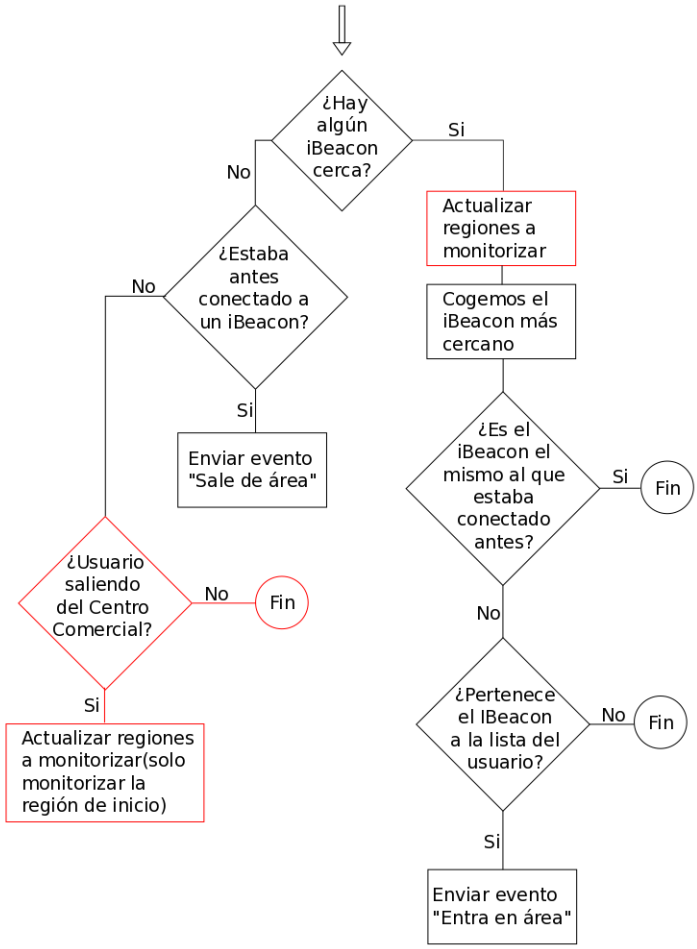


Figura 4.21. Diagrama de ejecución de la función locationManager:didRangeBeacon:inRegion: con actualización dinámica de regiones.

Actualización de la localización del usuario

Para el correcto funcionamiento de la aplicación, es necesario que el usuario tenga activo el Bluetooth de su teléfono móvil, en el caso de que el usuario active el Bluetooth cuando ya se encuentra dentro del centro comercial se producirá un error en la ubicación del usuario, esto se debe a que inicialmente la aplicación sólo está monitorizando los accesos al centro comercial.

Con el objetivo de solventar este problema se ha añadido un botón en la aplicación, que al ser pulsado por el usuario lo localiza con respecto al centro comercial. Para ello, cuando el usuario pulse el botón se realizará el *ranging* y *monitoring* de una región definida por el UUID de centro comercial, tal como se muestra en el siguiente trozo de código, englobando de esta forma a todas las balizas que se encuentran en él. Una vez detectada la baliza más cercana se pasará a activar el *ranging* y *monitoring* de las regiones vecinas, y la aplicación seguirá funcionando con normalidad.

```
//Pulsar actualizar posición
@IBAction func pulsaActualizarPosicion(_ sender: Any) {
      //Región definida por el UID del centro comercial
      let beaconRegion = CLBeaconRegion(proximityUUID: uuid,
      identifier: "Actualizar")
      //Activar monitoreo de la región
      locationManager.startMonitoring(for: beaconRegion)
      //Activar ranging de la región
      locationManager.startRangingBeacons(in: beaconRegion)
      //Activar la actualización de la localización de la región
      locationManager.startUpdatingLocation()
}
```

4.3.4. Validación de las posiciones de los usuarios

Otro de los principales objetivos perseguidos en el desarrollo del sistema es la seguridad, como se describe en el capítulo 3.1.2. la seguridad se basa en el uso de balizas Eddystone. Para poder trabajar con este tipo de dispositivos en la aplicación de iOS se ha usado la librería Core Bluetooth de Apple, la cual permite la comunicación entre una aplicación iOS y cualquier otro dispositivo que utilice tecnología BLE.

Monitoreo de regiones (Region Monitoring)

Al igual que ocurría con los iBeacons, las regiones definidas por las balizas del tipo Eddystone pueden ser monitorizadas por una aplicación tanto si esta está

ejecutando en primero plano como si la aplicación no está siendo ejecutada. Cada baliza Eddystone representa una región en sí misma [37].

Lo primero a tener en cuenta es la activación de la comunicación usando CoreBluetooth cuando la aplicación trabaje en modo *background*, para ello se añade lo mostrado en la figura 4.22. en el Info.plist.

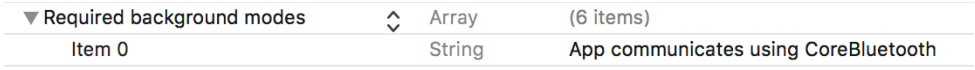


Figura 4.22. Activar la comunicación de la aplicación mediante el uso de CoreBluetooth.

En la clase AppDelegate de la aplicación se han añadido dos nuevos métodos para poder realizar esta función de monitoreo, el primero de ellos es el mostrado en el trozo de código siguiente, centralManagerDidUpdateState(_:), es el encargado de activar el escaneo de balizas Eddystone, para ello es necesario indicar el UUID (Universal Unique Identifier) del servicio ofrecido por estos dispositivos BLE, este servicio se corresponde con el valor 0xFEAA.

```
//Escanear por nuevos BLE (CoreBluetooth) -> En busca de beacons de
seguridad
func centralManagerDidUpdateState(_ central: CBCentralManager) {
      if (central.state == .poweredOn){
            //Para que detecte los beacons Eddystone-UID en el
            background hay que definir los servicios que queremos que
            sean escaneados
            //CBUUID es el identificador del tipo de servicio en el
            caso de los beacons Eddystone-UID este tipo de servicio es
            FEAA
            let arrayOfServices: [CBUUID] = [CBUUID(string: "FEAA")]
            self.centralManager?.scanForPeripherals(withServices:
            arrayOfServices, options: nil)
      }
}
```

El segundo método es centralManager(_:didDiscover:advertisementData:rssi:), este será llamado cada vez que el usuario entre en el área de una baliza Eddystone. Este método recibe como parámetro de entrada la información transmitida por la beacon de seguridad (valor HMAC). En caso de que el iBeacon asociado pertenezca a la lista del usuario, se escribirá el valor HMAC recibido en la base de datos del usuario, de esta forma se activa la Cloud Function encargada de realizar la verificación de la localización. A continuación se muestra el código del método centralManager(_:didDiscover:advertisementData:rssi:):

```
//Cuando entre en la región de un beacon de seguridad
func centralManager(_ central: CBCentralManager, didDiscover
peripheral: CBPeripheral, advertisementData: [String : Any], rssi
RSSI: NSNumber) {
      print ("Detectado beacon de seguridad")
      //Leer datos enviados por el beacon
      let datos = advertisementData["kCBAdvDataServiceData"] as?
      NSDictionary
      let ns = String(describing: datos!.allValues)
      //Obtener el HMAC
      let start = ns.index(ns.startIndex, offsetBy: 2)
      let end = ns.index(ns.endIndex, offsetBy: -2)
      let range = start..<end
      let hmac = ns.substring(with: range)
      //Obtener UID del usuario
      let uid = FIRAuth.auth()?.currentUser?.uid
      //Iniciar referencia a la base de datos
      let refseg = FIRDatabase.database().reference()
      //Si el valor de beacon es distinto de nil, es decir, si estamos
      en el área de un iBeacon asociado a alguna tienda de la lista
      del usuario
      if(self.beacon != nil){
            print ("Se guarda el token del usuario:
            \(tokenNotifications) y el valor HMAC recibido: \(hmac) en
            la base de datos")
            //Registrar el token
            refseg.child("users/\(uid!)/tokens").setValue(tokenNotific
            ations)
            //Registrar el valor HMAC en la base de datos -> Este
            registro lanza la Cloud Function
            refseg.child("users/\(uid!)/seguridad").setValue(hmac)
      }
}
```

Para saber si el iBeacon detectado pertenece o no a la lista de tiendas del usuario, cada vez que el método locationManager:didEnterRegion: detecte un iBeacon realizará la comprobación, consultado la base de datos del usuario, si el major se corresponde con alguna de las tiendas de la lista su valor será guardado en una variable global que será más tarde consultada por el método centralManager(_:didDiscover:advertisementData:rssi:), y además se guardará su valor en la base de datos del usuario. En caso contrario la variable global se igualará a nil. El código correspondiente a las acciones anteriormente citadas se muestra en la figura 4.23.

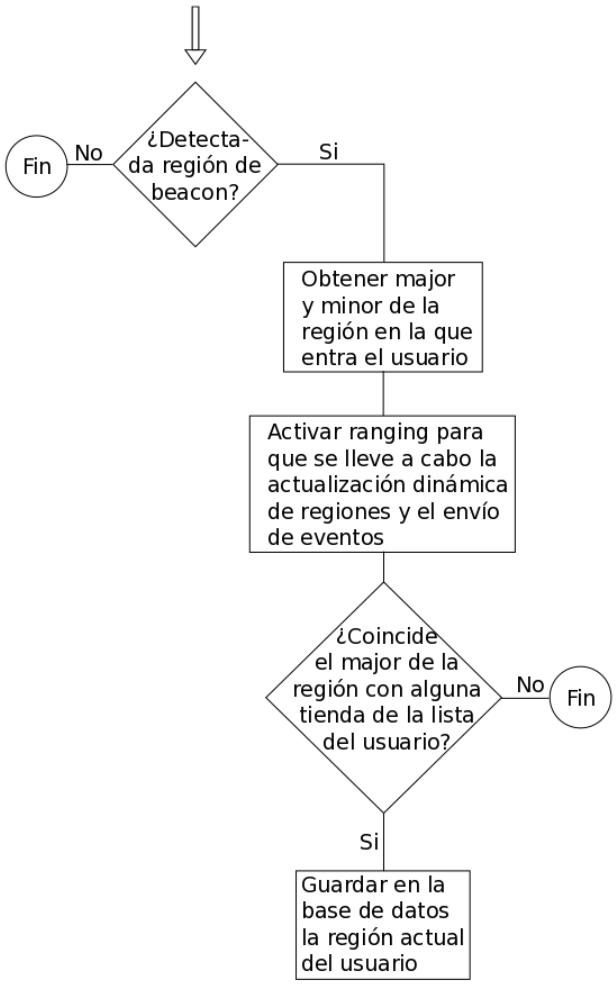


Figura 4.23. Diagrama de ejecución de la función locationManager:didEnterRegion:.

Por tanto, cuando un usuario se acerque al escaparate de una tienda su teléfono móvil recibirá por un lado el valor HMAC procedente de la baliza de seguridad y por otro lado el major del iBeacon (baliza de posición), si el major pertenece a alguna de las tiendas de su lista estos valores serán guardados en la Realtime Database, lanzando a su vez una Firebase Cloud Function, descrita en el apartador 4.1.2.

4.3.5. Ofertas recibidas

Hasta ahora se ha explicado cómo se ha realizado la autenticación y localización de clientes, así como el desarrollo de las medidas de seguridad y privacidad de los usuarios, sin embargo, ninguna de estas propiedades por si solas hacen que la aplicación sea atractiva de cara a los usuarios. Por ello, es necesario proporcionar algún incentivo a los clientes para que estén dispuestos a usar la aplicación y a compartir sus datos de localización (con aquellas

tiendas que ellos decidan), este incentivo son las ofertas o cupones de descuento proporcionados por las tiendas que los usuarios elijan.

Debido a que la aplicación, la mayor parte del tiempo, no estará trabajando en primer plano, las ofertas les llegarán a los usuarios, tras verificar su posición, en forma de notificaciones *Push* cuando estos entren en el área de una tienda de su lista que tenga alguna promoción vigente en ese momento. Además, las ofertas acumuladas por los usuarios se irán guardando en su base de datos para que estos puedan consultarlas y hacer uso de ellas en cualquier otro momento.

Notificaciones

Como desarrollador Apple, es necesario tener un certificado APNs que permita el envío de notificaciones a las aplicaciones iOS, en el apartado 4.2.3. se explica cómo obtener dicho certificado.

Además del certificado APNs, habrá que añadir algunas funciones en la clase AppDelegate del proyecto para establecer la conexión entre la aplicación y Firebase Cloud Messaging (FCM) [38]. Una vez establecida la conexión la aplicación recibirá un *token*; este *token* es el que necesita el servidor de la aplicación para poder enviar notificaciones al usuario.

En esta aplicación las notificaciones se envían al usuario como resultado de una función Cloud Function, con motivo de poder obtener durante la ejecución de esta función el *token* del usuario, este será guardado en la Realtime Database.

Lita de ofertas

En la aplicación se proporciona una vista al usuario en la que podrá consultar sus ofertas acumuladas, además de eliminar las que no le interese. Para ello, además del envío de la correspondiente notificación, a la salida de la Cloud Function también tendremos una entrada en la base de datos del usuario con el descuento obtenido.

La aplicación en la clase Ofertas, que es la asociada a la vista anteriormente descrita, realiza una lectura de la base de datos de un modo similar al descrito en el apartado 4.2.

Capítulo 5. Conclusiones y líneas de trabajo futuro

Tras el diseño y la posterior implementación y desarrollo del sistema para la mejora de la privacidad y la seguridad en la fidelización de clientes en el sector *retail* propuesto en el presente TFG, podemos llegar a las siguientes conclusiones con respecto a los objetivos inicialmente marcados:

- Para la obtención de un sistema seguro, se ha utilizado una segunda baliza BLE, que corrobora la posición de los usuarios enviando el resultado de calcular la función HMAC al major de la baliza de posición a la que se asocia y la fecha y hora actual. De esta forma, aunque un atacante pueda conocer el major de la baliza de posición o incluso el valor MAC enviado por la baliza de seguridad, no será capaz de saber cuál es la clave utilizada en la función HMAC y por tanto copiar las balizas de seguridad. Aunque las balizas de posición sí podrían ser copiadas, la copia de estas no serviría de nada ya que para recibir las promociones de las diferentes tiendas es necesario que la posición del usuario sea validada.
- La privacidad de los datos de los usuarios se garantiza con el uso de dos mecanismos: por un lado, en la aplicación los usuarios pueden decidir a qué tiendas quieren enviar sus datos de localización, de tal forma que si una tienda no se encuentra en la lista de un usuario concreto no recibirá ninguna información sobre dicho usuario; por otro lado, gracias al uso de pseudónimos las tiendas no tendrán forma de saber a qué usuario pertenecen los datos recolectados.

- Los pseudónimos utilizados por los usuarios son generados por la propia aplicación, creándose por primera vez cuando el usuario realiza el registro en la misma. Para mejorar la privacidad del usuario hay diferentes opciones, en el presente TFG en hemos centrado en dos aspectos: elección por parte del usuario las tiendas que recibirán sus datos de localización, pudiendo modificar la lista de tiendas en cualquier momento y cambio de pseudónimo siempre que el usuario desee, en la aplicación se generará un nuevo pseudónimo cada vez que el usuario reinicie sesión, cuando un usuario cambia de pseudónimo este se convierte en un nuevo cliente de cara a los comerciantes, ya que estos no tienen forma de asociarlo con su anterior pseudónimo.
- La aplicación desarrollada es capaz de detectar las balizas BLE aun cuando la aplicación no se está ejecutando, esto hace que la aplicación salte a un primer plano y que se ejecuten acciones como el registro de posiciones, de manera que para el correcto funcionamiento de la aplicación solo se precisa que la comunicación Bluetooth y WiFi del dispositivo móvil del cliente estén activadas. El usuario será alertado cuando reciba una oferta de alguna de las tiendas de su lista.

En el presente TFG, nos hemos centrado en la producción de datos sobre el comportamiento del usuario y su almacenamiento para permitir su procesado. A modo de ejemplo se ha trabajo con Firebase Analytics para mostrar la utilidad que podría tener estos datos en casos reales.

Como línea de trabajo futuro se propone el desarrollo de una aplicación móvil o web destinada a los comerciantes donde pudiesen actualizar sus ofertas de forma más sencilla y consultar sus estudios del comportamiento de los clientes en sus negocios, para ello se debería implementar una plataforma adaptada a la aplicación desarrollada, capaz de generar de manera automática estadísticas detalladas basadas en los eventos recibidos de los diferentes usuarios, dando a los comerciantes la posibilidad de visualizar los datos de manera clara y poder realizar análisis a partir de los mismos. Hoy en día, existen diferentes plataformas tanto comerciales como en código abierto que realizan este tipo de estudios y que servirían como referencia en el desarrollo

del trabajo. Empresas como Cisco y SensePost han desarrollado plataformas de este tipo.

- Meraki [39] es la solución proporcionada por Cisco. Esta es una plataforma comercial que permiten a sus usuarios obtener estadísticas exhaustivas sobre cómo se comportan los clientes en sus tiendas y cómo afectan las diferentes promociones a los clientes.
- Snoopy [40] es la plataforma creada por SensePost, que al contrario que Meraki es *open source* [41].

Otra línea interesante de trabajo futuro serían las balizas de posición, por un lado se podría integrar en un único dispositivo de las funciones realizadas por la baliza de seguridad y la de posición a la que está asociada, es decir, tener un único dispositivo que emita una señal siguiendo el protocolo iBeacon que sirva para localizar al usuario, y una señal siguiendo el protocolo Eddystone que sirva para validar la posición del usuario; también se podría implementar una aplicación que simplifique el despliegue de estos dispositivos y su configuración.

Referencias

[1] U. Ramanathan, N. Subramanian, W. Yu y R. Vijaygopal, “Impact of customer loyalty and service operations on customer behaviour and firm performance: empirical evidence from UK retail sector”, *Production Planning & Control*, Vol. 28, Nº 6-8, Mayo, 2017, pp. 478-488.

[2] J. Turow, *The aisles have eyes: How retailers track your shopping, strip your privacy, and define your power.* New Haven: Yale University Press, 2017.

[3] “Get with the program: Card-carrying consumer perpectives on retail loyalty-program participation and perks”, The Nielsen Company, Noviembre, 2016.

[4] C. Wright y L. Sparks, “Loyalty saturation in retailing: exploring the end of retail loyalty cards?”, *Internal Journal of Retail & Distribution Management*, Vol. 27, Nº 10, 1999, pp. 429-440.

[5] M. Andrews, J. Goehring, S. Hui, J. Pancras y L. Thronswood, “Mobile Promotions: A Framework and Research Priorities”, *Journal of Interactive Marketing*, Vol. 34, 2016, pp. 15-24.

[6] M. C. Scroggie, M. E. Kacaba, D. A. Rochon y D. M. Diamond, “System and method for providing shopping aids and incentives to customer through a computer network”, U.S. Patent 9519915, Dic. 13, 2016.

[7] P. Kriz, F. Maly y T. Kozel, “Improving Indoor Localization Using Bluetooth Low Energy Beacons”, *Mobile Information Systems*, Vol. 2016, 2016, pp.1-11.

[8] A. Thamm, J. Anke, S. Haugk y D. Radic, “Towards the Omni-Channel: Beacon-Based Service in Retail”, *Business Information Systems*, 2016, pp.181-192.

[9] “Privacy Policy for Euclid Analytics”, *Euclid*, 2017. [En línea]. Disponible en la web: http://euclidanalytics.com/about/privacy-statement/. [Accedido: 03 de Julio de 2017].

[10] “MOCA Platform – Big Data and iBeacons platform”, Mocaplatform.com. [En línea]. Disponible en la web: https://mocaplatform.com/features. [Accedido: 18 de Julio de 2017].

[11] J. Polo, “MOCA – para enviar información, vía app móvil, a los que están cerca”, *WWWhat’s new? – Aplicaciones, marketing y noticias en la web*, 2014. [En línea]. Disponible en la web: https://wwwhatsnew.com/2014/05/08/moca-para-enviar-informacion-via-app-movil-a-los-que-estan-cerca/. [Accedido: 18 de Julio de 2017].

[12] “MOCA, la app para impulsar ventar y mejorar la rentabilidad”, *EFE emprende*, 2017. [En línea]. Disponible en la web: http://www.efeemprende.com/noticia/moca-app-impulsar-ventas-y-rentabilidad/. [Accedido: 18 de Julio de 2017].

[13] “Bienvenido – Implantación de Beacons en España. Incrementa las visitas y ventas en tu tienda. BLE. iBeacon.”, Kappta.com, 2017. [En línea]. Diesponible en la web: https://kappta.com/es/. [Accedido: 18 de Julio 2017].

[14] A. Sánchez-Juárez, “La tecnología beacons, una revolución en alza para la experiencia de usuario y las estrategias de marketing”, *Uoc.edu*, 2016. [En línea]. Disponible en la web: https://www.uoc.edu/portal/es/news/actualitat/2016/099-beacons.html. [Accedido: 25 de Agosto de 2017].

[15] S. Datoo, “How tracking customers in-store will soon be the norm”, *the Guardian*, 2014. [En línea]. Disponible en la Web: https://www.theguardian.com/technology/datablog/2014/jan/10/how-tracking-customers-in-store-will-soon-be-the-norm. [Accedido: 03 de Julio de 2017].

[16] “About Mobile Location Analytics Technology”, *Smart-places.org*, 2016. [En línea]. Disponible en la web: https://smart-places.org/mobile-location-analytics-opt-out/about-mobile-location-analytics-technology/. [Accedido: 18 de Julio de 2017].

[17] A. Hassidim, Y. Matias, M. Yung y A. Ziv, “Ephemeral Identifiers: Mitigating Tracking and Spoofing Threats to BLE beacons”, 2016. [En línea]. Disponible en la web: https://developers.google.com/beacons/eddystone-eidpreprint.pdf. [Accedido: 18 de Julio de 2017].

[18] “¿Qué es Firebase? La mejorada plataforma de desarrollo de Google”, *El Androide Libre*, 2016. [En línea]. Disponible en la web: https://elandroidelibre.elespanol.com/2016/05/firebase-plataforma-desarrollo-android-ios-web.html. [Accedido: 20 de Julio de 2017].

[19] “Agrega Firebase a tu proyecto de iOS”, *Firebase*. [En línea]. Disponible en la web: https://firebase.google.com/docs/ios/setup?hl=es-419. [Accedido: 20 de Julio de 2017].

[20] “Local and Remote Notification Programming Guide: APNs Overview”, Developer.apple.cpm, 2017. [En línea]. Disponible en la web: https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1apple_ref/doc/uid/TP40008194-CH8-SW1. [Accedido: 19 de Julio de 2017].

[21] “Qué son los beacons y cuál es su potencial”, *The Valley Digital Business School*, 2014. [En línea]. Disponible en la web: https://thevalley.es/blog/que-son-los-beacons-y-cual-es-su-potencial. [Accedido: 25 de Agosto de 2017].

[22] C. Gilchrist, *Learning iBeacon*. Packt Publishing Ltd, 2014.

[23] “Getting Started with iBeacon”, Apple, Junio, 2014.

[24] M. Herrera Vargas, “Indoor Navigation Using Bluetooth Low Energy (BLE) Beacons”, Turku University of Applied Sciences, 2016.

[25] “Developers Doc – What is Eddystone?”, Developer.estimote.com. [En línea]. Disponible en la web: http://developer.estimote.com/eddystone/#eddystone--iBeacon-comparison. [Accedido: 19 de Julio de 2017].

[26] C. Gonzáles García, J. Espada B., C. G-Bustelo y J. Cueva Lovelle, “El futuro de Apple: Swift versus Objective-C”, Redes de Ingeniería, Vol. 6, Nº. 2, 2015, pp.6-16.

[27] “TIOBE – The Software Quality Company”, Tiobe.com, 2017. [En línea]. Disponible en la web: https://www.tiobe.com/tiobe-index/. [Accedido: 19 de Julio de 2017]

[28] H. Krawczyk, M. Bellare, R. Canetti, “HMAC: Keyed-Hashing for Message Authentication”, RFC 2014, 1997.

[29] D. Girish, “Wi-Fi vs RFID vs GPS vs Beacons: Which Location-Based Service Technology should your Brand go ahead with?”, *Blog.beaconstac.com*, 2017. [En línea]. Disponible en la web: https://blog.beaconstac.com/2017/05/wi-fi-vs-rfid-vs-gps-vs-beacon-which-

location-based-service-technology-should-your-brand-go-ahead-with/. [Accedido: 24 de Julio de 2017].

[30] "WiFi vs. Beacons – Which LBS Technology Will Work for your Business?", *July Systems*, 2017. [En línea]. Disponible en la web: http://julysystems.com/wifi-vs-beacons-lbs-technology-will-work-business/. [Accedido: 24 de Julio de 2017].

[31] "Cómo agregar Firebase a tu proyecto de iOS", *Firebase*, 2017. [En línea]. Disponible en la web: https://firebase.google.com/docs/ios/setup?hl=es-419. [Accedido: 25 de Agosto de 2017].

[32] "Cómo configurar una app cliente de Firebase Cloud Messaging en iOS", *Firebase*, 2017. [En línea]. Disponible en la web: https://firebase.google.com/docs/cloud-messaging/ios/client?hl=es. [Accedido: 25 de Agosto de 2017].

[33] "Configuración de APN con FCM", *Firebase*, 2017. [En línea]. Disponible en la web: https://firebase.google.com/docs/cloud-messaging/ios/certs?hl=es. [Accedido: 25 de Agosto de 2017].

[34] "How to detect iBeacons – free Swift 3 example code", *Hackingwithswift.com*, 2017. [En línea]. Disponible en la web: https://www.hackingwithswift.com/example-code/location/how-to-detect-iBeacons. [Accedido: 25 de Agosto de 2017].

[35] "startUpdatingLocation() – CLLocationManager | Apple Developer Documentation", *Developer.Apple.com*, 2017. [En línea]. Disponible en la web:

https://developer.apple.com/documentation/corelocation/cllocationmanager/1423750-startupdatinglocation. [Accedido: 25 de Agosto de 2017].

[36] O. Brown, “iBeacon Tutorial with iOS and Swift”, *Ray Wenderlich*, 2017. [En línea]. Disponible en la web: https://www.raywenderlich.com/152330/iBeacon-tutorial-ios-swift. [Accedido: 25 de Agosto de 2017].

[37] “Core Bluetooth Background Processing for iOS App”, *Developer.Apple.com*, 2017. [En línea]. Disponible en la web: https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothBackgroundProcessingForIOSApps/PerformingTasksWhileYourAppIsInTheBackground.html. [Accedido: 25 de Agosto de 2017].

[38] “Recibe mensajes en una app de iOS”, *Firebase*, 2017. [En línea]. Disponible en la web: https://firebase.google.com/docs/cloud-messaging/ios/receive?hl=es-419. [Accedido: 25 de Agosto de 2017].

[39] “Cisco Meraki | Location Analytics”, *Meraki.cisco.com*, 2017. [En línea]. Disponible en la web: https://meraki.cisco.com/solutions/location-analytics. [Accedido: 25 de Agosto de 2017].

[40] “SensePost | Snoopy: a distributed tracking and profiling framework”, *Sensepost.com*, 2017. [En línea]. Disponible en la web: https://sensepost.com/blog/2012/snoopy-a-distributed-tracking-and-profiling-framework/. [Accedido: 25 de Agosto de 2017].

[41] “SensePost | Snoopy release”, *Sensepost.com*, 2017. [En línea]. Disponible en la web: https://sensepost.com/blog/2012/snoopy-release/. [Accedido: 25 de Agosto de 2017].