
Título en español (**definido en Cascaras\cover.tex**)
Title in English (**defined in Cascaras\cover.tex**)



Trabajo de Fin de Grado
Doble grado en Ingeniería Informática y Matemáticas
Curso 2019–2020

Autor
Nombre Apellido1 Apellido2

Director
Director 1
Director 2

Colaborador
Colaborador 1
Colaborador 2

Facultad de Informática
Universidad Complutense de Madrid

Título en español (defined in
Cascaras\cover.tex)
Title in English (defined in
Cascaras\cover.tex)

Trabajo de Fin de Máster en **Ingeniería Informática**
Departamento de **XXXXXXXXXXXXXX**

Autor
Nombre Apellido1 Apellido2

Director
Director 1
Director 2

Colaborador
Colaborador 1
Colaborador 2

Dirigida por el Doctor

Director 1
Director 2

Facultad de Informática
Universidad Complutense de Madrid

16 de mayo de 2020

Dedicatoria

*A Pedro Pablo y Marco Antonio, por crear TeXiS
e iluminar nuestro camino*

Agradecimientos

A Guillermo, por el tiempo empleado en hacer estas plantillas. A Adrián, Enrique y Nacho, por sus comentarios para mejorar lo que hicimos. Y a Narciso, a quien no le ha hecho falta el Anillo Único para coordinarnos a todos.

Resumen

Título en español (definido en Cascaras\cover.tex)

Un resumen en castellano de media página, incluyendo el título en castellano. A continuación, se escribirá una lista de no más de 10 palabras clave.

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Title in English (defined in Cascaras\cover.tex)

An abstract in English, half a page long, including the title in English. Below, a list with no more than 10 keywords.

Keywords

10 keywords max., separated by commas.

Índice

1. Introducción	1
1.1. ¿Por qué la navegación por interiores?	2
1.2. Objetivos	3
1.3. Plan de trabajo	3
1.4. Estructura del documento	3
2. Estado de la Cuestión	5
2.1. Aplicaciones de guía	5
2.1.1. Google Maps	5
2.1.2. BlindSquare	7
2.1.3. Nearby Explorer	9
2.1.4. Lazarillo	10
2.1.5. Wayfindr	11
2.2. Sistemas de posicionamiento	11
2.2.1. GPS	12
2.2.2. Wi-Fi	12
2.2.3. Balizas Bluetooth	13
2.3. Trabajos previos	15
2.3.1. Sistema de guía por voz en interiores 2012-2013	15
2.3.2. Generador interactivo de instrucciones de guía sobre plataformas móviles 2013-2014	15
2.4. Conclusiones	16
3. Mapeo de interiores mediante balizas Bluetooth	17
3.1. Estudio de la precisión del posicionamiento mediante los <i>beacons</i>	17
3.1.1. Aplicación <i>miniapp</i>	18
3.1.2. Aplicación <i>cuadrantes_v1</i>	19
3.1.3. Conclusiones de la precisión de los <i>beacons</i>	21
3.2. Mapeo del edificio de la Facultad de Informática	22
3.3. Mediciones y distribución de los <i>beacons</i> en la Facultad de Informática	24
3.4. Representación del mapeo en los XML	26
4. Diseño e implementación	31
4.1. Servidor	31

4.1.1.	Funcionamiento del servidor	32
4.1.2.	Cálculo de la ruta óptima	35
4.1.3.	Generación de instrucciones	37
4.2.	Cliente	39
4.2.1.	Diseño de la aplicación Blind Bit	39
4.2.2.	Funcionamiento del cliente	42
4.3.	Adaptación para la reutilización de la aplicación sobre otro edificio	45
4.3.1.	Cambios en el servidor	45
4.3.2.	Cambios en el cliente	46
5.	Evaluación	49
5.1.	Adaptación de la aplicación a un nuevo edificio	49
5.1.1.	Cambios en el servidor	49
5.1.2.	Cambios en el cliente	50
5.2.	Objetivos de la evaluación	50
5.3.	Realización y resultados de la evaluación	51
5.3.1.	Mapeo del nuevo espacio	51
5.3.2.	Seguimiento de la ruta	53
5.3.3.	Usuario perdido	55
5.4.	Conclusiones de la evaluación	55
6.	ONCE	57
6.1.	Reunión en el Centro de Tiflotecnología e Innovación de la ONCE	57
6.1.1.	Introducción	57
6.1.2.	Entrevista	58
6.1.3.	Conclusiones	60
7.	Conclusiones y Trabajo Futuro	61
8.	Introduction	63
9.	Conclusions and Future Work	65
Bibliografía		67
A. Título del Apéndice A		69
B. Título del Apéndice B		71

Índice de figuras

2.1.	Plano de un edificio proporcionado por Google Maps.	6
2.2.	Vista del interior del Madison Square Garden.	7
2.3.	Ejemplo de navegación y búsqueda en Google Maps Indoors.	7
2.4.	Método de triangulación GPS.	12
3.1.	Aplicaciones auxiliares	19
3.2.	Gráfico con las distancias medidas al <i>beacon</i> CPne.	20
3.3.	Gráfico con las distancias medidas al <i>beacon</i> eAaw.	20
3.4.	Gráfico con las distancias medidas al <i>beacon</i> 8v2c.	21
3.5.	Gráfico de los <i>beacons</i> 8v2c y CPne superpuestos.	21
3.6.	Primera versión del mapeo de la primera planta de la Facultad de Informática.	22
3.7.	Versión final del mapeo de la primera planta de la Facultad de Informática.	23
3.8.	Mapeo de la planta baja de la Facultad de Informática, incluyendo la numeración de los cuadrantes y las posiciones de los <i>beacons</i>	24
3.9.	Mapa de la primera planta de la Facultad de Informática con la ubicación de los <i>beacons</i> (rojo) y los puntos de medición (verde).	25
3.10.	Mapa de la planta baja de la Facultad de Informática con la ubicación de los <i>beacons</i> (rojo) y los puntos de medición (verde).	25
3.11.	Mapa de la planta baja de la Facultad de Informática con la ubicación definitiva de los <i>beacons</i> (rojo) en el hall.	27
4.1.	Diagrama de secuencia para el arranque del servidor.	32
4.2.	Ejemplo de la información generada por el servidor para una ruta desde el cuadrante 14 al 4.	33
4.3.	Diagrama de secuencia para la generación de la ruta.	35
4.4.	Ejemplo de ruta óptima entre dos puntos.	36
4.5.	Ejemplo de giro en la ruta.	38
4.6.	Interfaz de la aplicación Blind Bit	40
4.7.	Diagrama de secuencia que comprende la interacción con el usuario, la resolución del posicionamiento y la conexión con el servidor en el cliente.	43
4.8.	Diagrama de actividad simplificado para la función <i>onEddystonesUpdated</i>	45
5.1.	Mapeo de la vivienda	52

Índice de tablas

Capítulo 1

Introducción

En la actualidad, los *smartphones* se han convertido en los protagonistas indiscutibles de nuestro día a día. El informe anual de *Ditrendia* (Rivero, 2019) recoge que el 68 % de la población mundial (5.100 millones) cuenta con un *smartphone*, mientras que este porcentaje ascienden al 96 % cuando hablamos de la población española. Es decir, aproximadamente 32,6 millones de españoles navegan por Internet a diario con su teléfono móvil.

Por otro lado, resulta prácticamente imposible imaginar un *smartphone* que hoy en día no tenga instalada una aplicación de guía. Este tipo de aplicaciones se han convertido en herramientas indispensables para un mundo que cada vez está más globalizado, ya que facilitan, por ejemplo, la circulación por ciudades desconocidas aportando la ruta óptima entre dos puntos y sus distintas alternativas: ir a pie, con transporte público, información sobre el mismo como horarios, cambios temporales, etc. Se estima que el 75 % de los usuarios españoles utilizan aplicaciones de navegación mensualmente, siendo la tercera utilidad más empleada después de la mensajería instantánea y la visualización de videos online (Rivero, 2019).

No cabe duda de lo útil que resulta poder consultar la ruta entre dos puntos. Pero, ¿estas aplicaciones son igual de apropiadas para todos los usuarios? ¿Se tienen en cuenta las necesidades de aquellas personas con discapacidad visual? En España, 70.775 personas sufren *ceguera legal* según la ONCE (Gómez Ulla). Este término engloba dos tipos marcados y diferenciados: lo que se conoce como ceguera (ausencia de visión o solo percepción de luz) y la deficiencia visual (mantenimiento de un resto de visión funcional para la vida cotidiana). En ambos casos, las personas que las padecen afrontan numerosos desafíos en su vida diaria, la mayor parte de ellos derivados de la falta de información. Un vistazo a nuestro alrededor es suficiente para darnos cuenta de cuán visuales son la mayor parte de los mensajes útiles que usamos en nuestro entorno, desde leer la etiqueta de un producto en el supermercado, hasta saber si nos encontramos en la parada de autobús correcta. De igual forma esto sucede en el caso concreto del ocio y la tecnología. No abundan los libros adaptados. De hecho, según la *World Blind Union* “más del 90 % del material publicado no es accesible para invidentes o personas con deficiencia visual” (Envision, 2019). E igual ocurre con Internet. El grueso de las páginas web y de las aplicaciones no consideran las necesidades especiales de estos potenciales usuarios, dejándoles completamente al margen. De ahí que los ojos sean considerados los principales órganos sensoriales, pues su pérdida conlleva una reducción considerable de autonomía derivada de la falta de acceso a la información. En ocasiones, esto viene acompañado de un segundo problema con el que muchos están acostumbrados a lidiar, el exceso de protección. A menudo, familiares, amigos o incluso desconocidos asumen que un invidente necesita ayuda sin preguntar o sin esperar a

ser llamados. Este frecuente comportamiento genera impotencia en el individuo en lugar de independencia y le quita espacio para aprender a realizar una tarea por sí mismo.

En resumen, la falta de accesibilidad es el eje central del que nacen numerosos problemas que afectan a la vida de las personas que presentan ceguera legal. Por ello, la respuesta a las dos preguntas lanzadas al comienzo de esta sección es no, actualmente son pocas las aplicaciones que tienen en cuenta a las personas que sufren discapacidad visual y, en particular, son pocas las aplicaciones de navegación que están adaptadas. Es por esto, que en nuestro trabajo de fin de grado hemos querido abordar este problema, estudiando, para ello, tecnologías accesibles que nos permitan desarrollar una aplicación de navegación a través de interiores que facilite una guía adaptada para estos usuarios.

1.1. ¿Por qué la navegación por interiores?

En un contexto urbano, el concepto de la navegación es bien conocido. Con frecuencia nos vemos obligados a buscar la ubicación de una tienda, un hospital, la casa de un amigo o cualquier otro edificio. Para llegar hasta el destino buscamos la ruta más rápida o la más conveniente. Sin embargo, la navegación no finaliza una vez estamos dentro del inmueble, pues, normalmente, necesitamos buscar un lugar dentro de ese edificio (la recepción, los baños, etc). A la determinación de la ubicación de un punto concreto y a la guía hasta el mismo por el interior de un espacio “cerrado” lo denominamos navegación por interiores.

Como es natural, todos nos vemos obligados a desplazarnos en nuestro día a día. Normalmente suele ser a lugares conocidos a los que llegamos de una manera más o menos automática, sin tener que pensar mucho, ya que conocemos y memorizamos todo lo que hay en dichos recorridos. Sin embargo, de manera puntual modificamos dichas rutinas, ya sea por problemas temporales que inhabilitan la ruta en cuestión, como por la necesidad de desplazarnos a un lugar al que no habíamos ido antes. Paralelamente, hay un conjunto de edificios que visitamos con cierta frecuencia y que, por ende, nos resultan familiares y donde nos ubicamos perfectamente, pero en ocasiones nos surge la necesidad de ir a otros por primera vez, véase un hospital, un museo o un centro comercial.

A menudo estas situaciones despiertan desorientación, incomodidad y rechazo en las personas que las viven ya que se encuentran frente a una situación de descontrol e incertidumbre debido a la falta de conocimiento. A nadie le gusta sentirse perdido, pero cuando te falta uno de los cinco sentidos, y uno de los más esenciales (la vista) esto se vuelve mucho más duro, ya que hay un gran vacío informativo. Basta pensar en cuántas personas se te cruzan por la calle, cuántos obstáculos sorteas a diario, tanto en interiores como en exteriores, cuántas veces cruzas la carretera para alcanzar tu destino, cuántas veces te apoyas leyendo el nombre de una calle o un cartel en un edificio, cuántas veces bajas/subes unas escaleras o esperas al ascensor, o miras el número del autobús que está por llegar... Todo esto son ejemplos de situaciones muy cotidianas que para las personas videntes no suponen ningún esfuerzo mientras que para las personas con discapacidad visual suponen un gran reto.

En los últimos años se ha estudiado mucho el sector de la navegación por exteriores, y actualmente son varias las apps que mediante el GPS proporcionan una guía de origen a destino de manera satisfactoria. Este hecho, acompañado de la creciente sensibilización con las personas con discapacidad visual y del *boom* de las tecnologías accesibles, ha favorecido que cada vez más desarrolladores se interesen por la accesibilidad y la promuevan en este tipo de aplicaciones. Sin embargo, en la navegación por interiores aún vemos un claro vacío ya que es un terreno menos explorado en general y, consecuentemente, menos adaptado. Por ello, hemos decidido centrar nuestras investigaciones en este sector, buscando paliar el

malestar al que estos usuarios se enfrentan en su día a día. Para la consecución de este fin, desarrollaremos una aplicación accesible que sirva de guía a invidentes por espacios interiores, y más concretamente por la Facultad de Informática de la Universidad Complutense de Madrid.

1.2. Objetivos

Esta aplicación se enmarca dentro del Proyecto MILES (Models of Interaction centred on Language, space and computational Semantics) del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la UCM y tiene como objetivo tanto mejorar funcionalidades ya existentes en proyectos predecesores como integrar otras nuevas que hagan de ella una aplicación de navegación por interiores completa adecuada para personas con discapacidad visual.

Los requisitos a tener en cuenta para nuestro proyecto se dividen en dos grandes grupos: aquellos relativos a la guía y los que a la adaptabilidad se refieren.

Con respecto a la guía, nos encargaremos en primera instancia de localizar a un individuo en la Facultad de Informática utilizando para ello una tecnología novedosa respecto a trabajos anteriores: los *beacons*. De esta manera aportaremos una diferencia significativa a nuestro proyecto que lo distinguirá de los otros incluidos en MILES. A continuación, adaptaremos el espacio mapeado por trabajos anteriores y lo ampliaremos incluyendo la planta baja y finalmente, conectaremos las plantas entre sí para añadir la funcionalidad de guiar a un usuario hasta un destino que se encuentre en una planta distinta a la suya.

En lo relativo a la adaptabilidad a usuarios que presentan discapacidad visual haremos una segunda distinción. Por un lado adaptaremos la ruta y las instrucciones de modo que el camino sea lo más adecuado posible y las instrucciones sean cuanto más precisas mejor (distancias en metros, ubicación del destino a la derecha o a la izquierda, etc.), incluyendo incluso la posibilidad de proporcionar más información sobre el entorno (qué aulas se encuentra a su paso, si hay baños, etc.) si el usuario lo desea. Por otro lado, adaptaremos la interfaz de manera que sea intuitiva y fácil de utilizar con ayuda del lector de pantalla. Para ello, diseñaremos pantallas sencillas y poco cargadas que incluirán botones grandes y fáciles de ubicar. También añadiremos la posibilidad de controlar la aplicación mediante voz y adaptaremos las instrucciones para que aparezcan tanto por escrito en la pantalla como reproducidas en voz alta.

1.3. Plan de trabajo

Aquí se describe el plan de trabajo a seguir para la consecución de los objetivos descritos en el apartado anterior.

1.4. Estructura del documento

El proyecto que se desarrolla en las páginas que siguen guarda una estructura clara, que viene determinada por capítulos:

En el Capítulo 2 se expone el contexto en el que se enmarca este trabajo de fin de grado. En primer lugar se revisan algunas de las aplicaciones existentes en el ámbito de la navegación (tanto por interiores como por exteriores), prestando atención a aquellas que están adaptadas o cuyos usuarios objetivo son personas con discapacidad visual (ver Sección 2.1). Por su parte, la Sección 2.2 describe distintas tecnologías para resolver el problema

del posicionamiento. En concreto tratamos tres: GPS, Wi-Fi y balizas Bluetooth. De cada una de ellas destacaremos sus ventajas e inconvenientes en cuanto al posicionamiento en interiores se refiere. En la Sección 2.3 se resumen los dos trabajos de fin de grado previos sobre los que se apoya este. Por último, en la Sección 2.4 se detallan las conclusiones que se han tomado sobre este capítulo. Entre ellas algunas de las características básicas con las que debe contar nuestra aplicación o la elección de la tecnología empleada en este proyecto para resolver el problema del posicionamiento.

Es en el Capítulo 3 donde comienza el trabajo específico de nuestra aplicación. La primera sección, Sección 3.1, aborda la primera investigación que se realizó con las balizas Bluetooth o *beacons*. En ella se detalla el estudio de la precisión de los *beacons* en cuanto a distancia se refiere. Para ello se implementaron dos aplicaciones muy sencillas, *miniapp* y *cuadrantes_v1*, cuya funcionalidad puede verse en detalle en las Secciones 3.1.1 y 3.1.2, respectivamente. Así mismo, en la Sección 3.1.2.1 se hace un estudio exhaustivo de las distintas pruebas que se realizaron con la aplicación *cuadrantes_v1* y cuyas conclusiones se recogen en la Sección 3.1.3. Una vez analizada la tecnología y estudiado su comportamiento, comienza el mapeo de la Facultad de Informática de la UCM en la Sección 3.2. Es en esta sección donde se establece la primera aproximación de la estructura e implementación de los archivos xml en los cuales se recoge la información referente al edificio. La investigación detallada sobre las mediciones y distintas pruebas que se llevaron a cabo para establecer la distribución de los cuadrantes y la ubicación final de los *beacons* se recoge en la Sección 3.3. Por último, la Sección 3.4 expone la estructura de los mencionados archivos xml.

En el Capítulo 4 se abordan los detalles técnicos sobre el diseño e implementación de la aplicación. Al igual que la aplicación se divide en dos partes, el cliente y el servidor, este capítulo también. La primera parte, Sección 4.1, expone el funcionamiento general del servidor (Sección 4.1.1) así como la implementación de sus dos funcionalidades principales: el cálculo de la ruta óptima (Sección 4.1.2) y la generación de instrucciones (Sección 4.1.3). La segunda, Sección 4.2, se centra en la implementación y el diseño de la aplicación móvil. En la Sección 4.2.1 revisamos la interfaz de la aplicación y la justificación de su diseño, mientras que en la Sección 4.2.2 nos adentramos en su funcionamiento desde el punto de vista técnico.

Capítulo 2

Estado de la Cuestión

Este capítulo lo dedicaremos a estudiar el contexto sobre el que se sustenta nuestro trabajo. Antes de comenzar debemos conocer los trabajos previos que se han realizado en la navegación y cómo se han abordado desde el punto de vista tecnológico. Nos interesaremos, especialmente, en aquellos enfocados a la navegación por interiores y, en concreto, aquellos cuyo usuario final es una persona con discapacidad visual.

En la Sección 2.1 se exponen distintas aplicaciones de navegación. De ellas extraemos sus puntos más fuertes y, sobre todo, sus carencias, pues serán esos puntos donde nuestro trabajo tratará de incidir en mayor medida. En la Sección 2.2 veremos la tecnología necesaria para el posicionamiento, comparando las tres principales vertientes: GPS, Wi-Fi y Bluetooth.

2.1. Aplicaciones de guía

En los últimos años ha aumentado la sensibilización tecnológica en áreas como la inclusión de usuarios con discapacidad visual. Es por ello que las tecnologías accesibles tienen, cada vez más, un papel central en el desarrollo de aplicaciones, logrando que estas se abran a un público más amplio.

Al igual que las personas videntes, las personas con ceguera son usuarios de aplicaciones de muy variada índole. Debido a esto, encontramos apps ya adaptadas en categorías como: redes sociales, entretenimiento, lectura, identificación de colores y objetos, etc.

En esta sección, haremos un pequeño estudio sobre las aplicaciones accesibles existentes en el campo de la navegación, bien sea por interiores o exteriores, y su funcionamiento.

2.1.1. Google Maps

El pasado 10 de Octubre de 2019, en el “World Sight Day”, Google dió a conocer la última actualización de la famosa aplicación *Google Maps*¹. Esta incluiría una nueva característica desarrollada desde cero por y para personas con discapacidad visual que convertiría a la misma en una app accesible.

El proyecto consiste en la implementación de una nueva funcionalidad que facilita la posibilidad de recibir instrucciones de voz más detalladas y nuevos tipos de anuncios verbales muy útiles para las rutas de a pie para personas con visibilidad reducida. Algunas de las nuevas instrucciones incluidas son: informar de manera proactiva que estás en la ruta

¹<https://blog.google/products/maps/better-maps-for-people-with-vision-impairments/>



Figura 2.1: Plano de un edificio proporcionado por Google Maps.

correcta, la distancia hasta el próximo giro, la dirección en la que estás caminando, avisos para cruzar con precaución si te aproximas a una gran intersección, notificaciones en caso de ser redirigido por causa de haber abandonado accidentalmente la ruta correcta, etc. De esta manera, la aplicación pretende hacer más independientes a las personas con discapacidad visual tratando de que se sientan cómodas y seguras a la hora de explorar lugares nuevos y desconocidos. La guía de voz detallada para la navegación está actualmente en desarrollo para diversos idiomas, estando ya disponible en inglés en los Estados Unidos y en japonés en Japón. Su soporte para otros idiomas y países sigue en camino.

En cuanto a la navegación por interiores, *Google Maps*² con su actualización 6,0 incorporó los primeros planos de ciertos edificios públicos, entre los cuales destacan aeropuertos, centros comerciales, estadios y puntos de transporte público. Gracias a esta nueva versión denominada *Google Maps Indoors*, *Google Maps* ayuda a determinar dónde estás, en qué planta y hacia dónde ir. Para ello, basta con hacer *zoom* sobre un edificio cuyo plano esté disponible en la app, y este aparecerá automáticamente y completamente detallado. En la Figuras 2.1 y 2.2 vemos un ejemplo del famoso Madison Garden de Nueva York.

Con estos nuevos planos podrás localizar dónde están los baños, escaleras, ascensores, entradas y salidas, etc., los cuales aparecen representados mediante iconos globalmente aceptados (ver Figura 2.1). También aparecen detallados los distintos establecimientos que se localizan en el edificio e incluye la posibilidad de hacer ciertas búsquedas, tanto generales (de cafeterías, librerías, tiendas, restaurantes...) como concretas (Starbucks, McDonald's...) (ver Figura 2.3). Otra funcionalidad que no falta en la versión de interiores es la posibilidad de señalar un destino y recibir indicaciones sobre cómo llegar a él. Para ello, aparece el habitual punto azul que te acompaña e indica tu posición, actualizando el plano con cada movimiento que lleves a cabo (incluidos cambios de una planta a otra) (ver Figura 2.3). Esta aplicación es un proyecto colaborativo y por ende, desde la web es posible actualizar y subir nuevos planos. Está disponible tanto para ordenador como plataformas Android e

²<https://www.google.es/intl/es/maps/about/partners/indoormaps/>

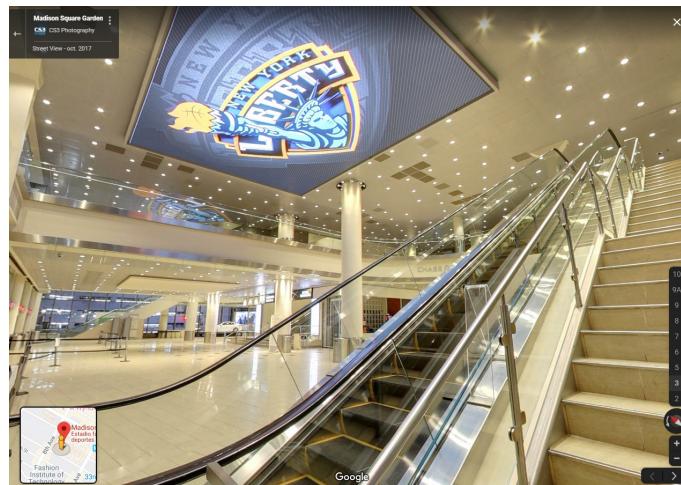


Figura 2.2: Vista del interior del Madison Square Garden.

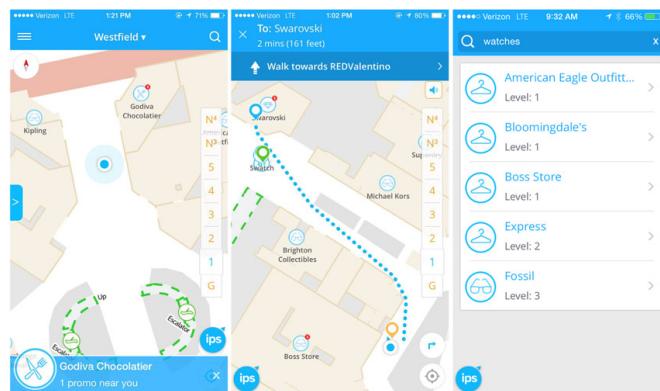


Figura 2.3: Ejemplo de navegación y búsqueda en Google Maps Indoors.

iOS.

Esta aplicación pone a nuestro servicio la utilidad de *Maps* pero en interiores. Además, nos permite colaborar, pudiendo subir nosotros mismos el plano de un edificio. Sin dudar del gran avance que esta aplicación supone en la navegación por interiores, no debemos olvidar algunas de sus desventajas: el posicionamiento, al contrario que en exteriores, no es muy preciso (en la web hablan de varios metros), y las búsquedas que puedes realizar son limitadas, no pudiendo, por ejemplo, preguntar por la ubicación de los baños; esto es, puedes ver dónde están pero no puedes seleccionarlos como destino para que te vaya indicando la ruta a seguir. Pero sobre todo, tiene el inconveniente de que no es una tecnología accesible: *Google Maps Indoors*³ es una aplicación completamente visual que no cuenta con soporte auditivo por lo que descarta completamente a usuarios invidentes.

2.1.2. BlindSquare

*BlindSquare*⁴ es una de las aplicaciones de navegación más populares. Su uso se extiende a más de 130 países y está habilitada en 25 idiomas, entre los cuales se incluye el español. Esta aplicación, desarrollada para iOS y diseñada para personas con discapacidad visual, proporciona una guía completa, de origen a destino, tanto en exteriores como en interiores.

³https://www.youtube.com/watch?v=cPsTWj_03Qs

⁴<https://www.blindsightsquare.com>

Además, describe el entorno y anuncia posibles puntos de interés para el usuario (como pueden ser los lugares considerados populares o aquellos visitados frecuentemente). Su principal característica es que permite interactuar mediante voz gracias al controlador de música de Apple.

Esta aplicación determina tu posición mediante localización *GPS* y, a partir de ahí, puede darte información sobre las proximidades utilizando *Foursquare* y *OpenStreetMap*. De este modo, es capaz tanto de guiarte a un cierto destino como de notificarte qué establecimientos hay en tu radio: restaurantes a 200m, parques más cercanos, farmacias...

Con el fin de agilizar el uso de la app, y que por tanto esta sea cómoda y rentable para los usuarios finales, incluye: accesos directos a funciones mediante gestos (como sacudir el móvil para que nos diga la ubicación actual y puntos cercanos) y la posibilidad de establecer filtros para recibir únicamente el tipo de información deseada. Por ejemplo, permite filtrar por restaurantes para no tener notificaciones sobre estaciones de tren o librerías.

En cuanto a la navegación por interiores, *BlindSquare*⁵ emplea un sistema de balizas Bluetooth, también llamadas *beacons*, que colocan en sitios estratégicos de los edificios, para solventar el problema del posicionamiento. Por lo demás, incluye las mismas posibilidades y funcionalidades que la navegación por exteriores, con la única limitación de que el edificio debe estar provisto de dichos sistemas de posicionamiento.

En su web encontramos un ejemplo de la utilización de los *beacons* en un campus⁶: una vez que entras en el edificio, uno de los *beacons* se dará cuenta de tu aplicación *BlindSquare* y te hará saber dónde te encuentras y cómo llegar a tu destino, indicándote los ascensores, escaleras e intersecciones más cercanas. Integrar en el campus servicios como estos favorece tanto a visitantes como a estudiantes con discapacidad visual moverse por el entorno con total autonomía y seguridad.

A modo de resumen, entre los puntos fuertes de esta aplicación destacamos los siguientes:

- Da información sobre los metros que quedan hasta llegar a un determinado objetivo. Resulta útil porque si van disminuyendo sabes que vas por el camino adecuado.
- Utiliza indicaciones de tipo reloj (a las 10, a las 3,...) muy usadas por las personas con discapacidad visual⁷.
- Avisa de las intersecciones.
- Cuando se supera una indicación dada por la aplicación, esta emite el sonido asociado a correto o *check*. En el caso de que esta indicación no se complete, reproduce otro sonido en consecuencia.
- Se pueden añadir ubicaciones en una lista de lugares marcados.
- Se puede ir girando con el móvil mientras la aplicación indica los puntos de interés que se encuentre delante.
- También tiene opción de simulación, que permite prepararse un camino antes de ir.
- Permite ser más autónomo y descubrir nuevos sitios.

⁵<https://www.youtube.com/watch?v=9jH-Bdjmgb4>

⁶<https://www.blindsightsquare.com/2019/11/01/blindsightsquares-getting-straight-as-on-campus/>

⁷http://www.riate.org/version/v1/materiales_en_prueba/e_inclusiva_discapacidad/pdf/m6_dv.pdf

- A la hora de desplazarte te indica las distintas alternativas por adelantado. Esto es, mientras que para espacios exteriores te señala la posible ruta utilizando transporte público, privado, a pie, etc, para espacios interiores te especifica, siempre que la haya, la opción de utilizar escaleras, ascensor, escaleras mecánicas, etc., de esta manera te proporciona una idea global del espacio y de las distintas vías que puedes seguir para llegar al destino.
- Permite llevar las manos libres.
- Incluye un lector de códigos QR, que es cómodo porque puede dar más información que la línea braille.

Su principal punto negativo es el precio, ya que cuesta 40 libras.

Al contrario que la aplicación *Google Maps Indoors*, esta sí es una aplicación diseñada para personas con discapacidad visual. Las diferencias saltan a la vista: el modo de dar las indicaciones, avisos constantes para indicarte si vas por el camino correcto, permite más autonomía; gracias a la comunicación constante que ofrece permite llevar las manos libres, entre otras cosas. Parece imposible pensar que el interior de un edificio pueda resultar menos seguro que una gran avenida, pero lo cierto es que, para las personas con discapacidad visual, muchas veces es así. El interior de un gran centro comercial o una biblioteca resultan un laberinto cuando se va por primera vez, más aún si tenemos algún tipo de dificultad para leer las indicaciones que, normalmente, suelen estar en lugares altos y no adaptadas para personas con discapacidad visual. Lo que se pretende con esta aplicación es mantener la autonomía del usuario tanto dentro como fuera de un edificio⁸.

2.1.3. **Nearby Explorer**

*Nearby Explorer*⁹ es otra de las aplicaciones que encuadramos en el campo de la navegación accesible por interiores y exteriores. Está habilitada tanto para Android como para iOS y su descarga se encuentra disponible de manera gratuita.

La guía por exteriores se basa en la misma idea que *BlindSquare*, y por ende funciona de manera similar. Entre sus características destacan: la posibilidad de ejecutar ciertas acciones poniendo el móvil en distintas posiciones, como por ejemplo, inclinarlo verticalmente para que funcione como una brújula; y, la capacidad de filtrar la información de modo que ésta se adapte completamente a las necesidades del usuario. Entre la información que *Nearby Explorer* puede proporcionar a sus usuarios encontramos los lugares cercanos a la ubicación actual, los nombres de las calles por las que pasa, los números de los bloques de las calles por las que pasa, la distancia que hay al destino desde un punto de referencia (como casa, trabajo...), etc. Además de la posibilidad de filtrar la información deseada, las indicaciones por audio pueden ser pausadas en cualquier momento de modo que no interfieran con otras señales auditivas (como las paradas en un autobús, por ejemplo). Otra gran funcionalidad con la que cuenta *Nearby Explorer* es la de explorar una ruta por adelantado, sin tener que estar físicamente en el sitio, pudiendo incrementar o decrementar el radio de exploración.

Por otro lado, la navegación por interiores se basa en un sistema de *beacons* que sustituye a las señales GPS y se encarga de solventar el problema del posicionamiento en interiores. Pueden configurarse de dos maneras: *ad hoc* y *mapeo completo*.

⁸<https://www.blindsight.com/2019/10/24/independence-on-both-sides-of-the-door/>

⁹<https://play.google.com/store/apps/details?id=org.aph.nearbyonline&hl=es>

En el caso de la configuración *ad hoc*, cuenta con la ventaja de que tiene una instalación muy sencilla (basta con posicionar los *beacons* sobre la pared, en la ubicación que queramos)¹⁰, pero aparecen los siguientes problemas:

- No se puede determinar la ubicación exacta de un *beacon*.
- No se puede obtener información del entorno a menos que te encuentres dentro del radio de detección de un *beacon*.
- Tienes que habilitar cierto soporte para detectar los *beacons* (no se detectan de manera automática).

Por el contrario, el *mapeo completo* es más robusto por lo que su instalación es más compleja pero a cambio nos proporciona una localización precisa del dispositivo por lo que tiene un comportamiento similar al de otras aplicaciones.

Desde el punto de vista de la navegación por exteriores es una aplicación completa y fácil de utilizar, que incluye una interfaz sencilla y trata de adaptarse siempre a las distintas necesidades o situaciones del usuario mediante opciones configurables. Además, cuenta con una versión gratuita (algo poco común en aplicaciones de tal categoría) que aunque no incluye todas las funcionalidades de la versión de pago te permite probarla y familiarizarte con ella antes de tomar una decisión final. Por otro lado, la funcionalidad de navegación por interiores no está tan desarrollada y su uso está supeditado exclusivamente a aquellos lugares que cuenten con la instalación necesaria y hayan incluido sus datos en OpenStreetMap, configurando el espacio en nodos, aristas y relaciones. Esta tarea es tediosa y a menudo parte de cero por lo que son pocos los edificios que actualmente están mapeados y pueden aprovechar al máximo la app.

2.1.4. Lazarillo

Lazarillo¹¹ es una aplicación de guía para personas con discapacidad visual que actualmente solo proporciona guía para exteriores. Inicialmente la idea era cubrir también la navegación por interiores pero su desarrollo no fue posible por problemas de financiación.

La navegación por exteriores cuenta con las funcionalidades básicas que ya hemos mencionado en las apps anteriores:

- Buscar lugares de interés, cercanos a la ubicación actual. Esta búsqueda se puede acotar filtrando por categorías que vienen predefinidas (transporte, bancos y cajeros, salud, comida, tiendas, etc.).
- Buscar una dirección específica a partir de la cual se desplegarán todas las posibles rutas (a pie, en transporte público, privado, etc.) y una vez seleccionada la ruta deseada, comenzarán las indicaciones mediante audio con la información pertinente (metros, giros a derecha e izquierda, etc.).
- Guardar una lista de lugares favoritos.
- Posibilidad de rastrear una dirección, previamente marcada con la opción “Seguir este lugar”, de modo que con independencia de a dónde nos estemos dirigiendo se activará una alerta a medida que nos acerquemos a dicha ubicación.
- Ajustar la configuración de las indicaciones, velocidad, tipo de voz...

¹⁰<https://tech.aph.org/neios/#Beacon>

¹¹<https://www.lazarillo.app/es/>

En resumidas cuentas, *Lazarillo* es una aplicación que, como otras, busca mejorar la calidad de vida de las personas con discapacidad visual indicándoles para ello qué les rodea y proporcionándoles una mayor independencia. Ésta, sin embargo, cubre únicamente los aspectos más básicos y elementales sin reparar en otras posibles funcionalidades o indicaciones (obstáculos, peligros...), por lo que es una aplicación incompleta.

La app es completamente gratuita y cuenta con versión para Android y iOs.

2.1.5. Wayfindr

*Wayfindr*¹² nació en 2015 en Londres, con la misión de capacitar a las personas con discapacidad visual para que viajen de manera independiente a través de una navegación de audio inclusiva y accesible. Con este fin, han desarrollado el primer estándar del mundo aprobado internacionalmente para la navegación de audio accesible y ya cuentan con las primeras demostraciones de un sistema de navegación en red. Este sistema, basado en audio, pretende dar soporte para que las personas con discapacidad visual puedan adentrarse por esos lugares que están repletos de señales escritas, por los que las personas que ven pasan sin pensar, pero que son precisamente los que más temen y evitan aquellos que tienen discapacidad visual (estaciones de metro, tren, aeropuertos, centros comerciales, hospitalares, etc.).

Este proyecto de código abierto ha realizado ya numerosas pruebas en distintos escenarios, como por ejemplo en el metro de Londres (o, más recientemente, en el metro de Los Ángeles) donde el funcionamiento¹³ de la aplicación es tan práctico como sencillo: se basa en una serie de *beacons*, colocados en puntos estratégicos a lo largo de las distintas estaciones de metro, que emiten unas señales que son captadas por el móvil a su paso por un cierto radio de detección. Estas señales permiten ubicar al usuario y darle la siguiente indicación con el propósito de alcanzar su objetivo (coger un tren o salir de la estación).

Los desarrolladores recomiendan el uso de auriculares de conducción ósea, de manera que puedan escuchar otros sonidos del exterior.

La idea de este proyecto supone un gran avance para las personas que tienen algún tipo de discapacidad visual, ya que pretende empoderarlas para que no se sientan retenidas por su pérdida de visión ni tengan que vivir supeditadas a una persona vidente que las ayude, logrando así que se rompan de una vez por todas las barreras a las que están sometidas. Para la consecución de este fin, esta organización sin ánimo de lucro proporciona a los fabricantes de navegación digital y propietarios de espacios públicos las habilidades y técnicas para proporcionar a las personas con discapacidad visual servicios de navegación digital consistentes y de alta calidad. *Wayfindr* utiliza varias tecnologías para rastrear la ubicación de una persona y activar el audio de las instrucciones en su teléfono móvil en el momento adecuado para llevarlos a su destino. De esta manera, busca permitir que las personas con discapacidad visual naveguen por el mundo utilizando las instrucciones de audio de sus teléfonos inteligentes.

2.2. Sistemas de posicionamiento

Para la consecución de nuestro objetivo, el desarrollo de una aplicación de navegación por interiores, uno de los primeros problemas que se nos plantea es el del posicionamiento en un mapa ya que es de vital importancia poder determinar donde estamos para después indicar la ruta pertinente hacia el destino indicado. En esta sección haremos un pequeño

¹²<https://www.wayfindr.net/>

¹³<https://www.youtube.com/watch?v=mc3KmbfxuUQ>

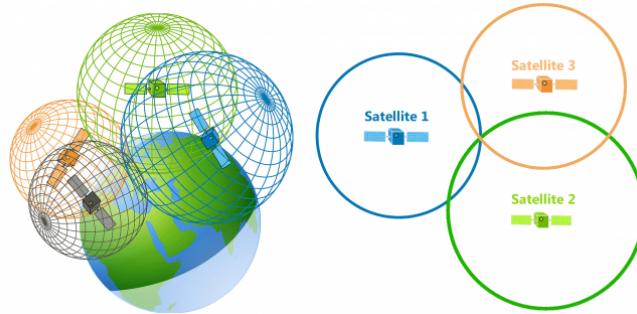


Figura 2.4: Método de triangulación GPS.

estudio sobre las distintas tecnologías existentes que nos permiten solventar nuestro problema y determinar la posición exacta de un cierto dispositivo, y discutiremos su validez para su aplicación a este trabajo de fin de grado.

2.2.1. GPS

El Sistema de Posicionamiento Global (GPS por sus siglas en inglés) es un sistema de localización diseñado por el Departamento de Defensa de los Estados Unidos con fines militares para proporcionar estimaciones precisas de posición, velocidad y tiempo. Este sistema se encuentra operativo desde enero de 1994 y se desarrolló a partir de los 24 satélites que componen la constelación NAVSTAR, cada uno de los cuales cuenta con una órbita de 26.560 Km de radio y un periodo de 12h (Pozo-Ruz et al., 2000).

El método mediante el cual el GPS determina la altitud, longitud y latitud de cualquier objeto que se encuentre en la superficie terrestre se conoce como triangulación. Este requiere la distancia desde el dispositivo en cuestión (receptor) a tres satélites como mínimo cuya localización es conocida de antemano. Entonces, cuando el receptor detecta el primer satélite, se genera una esfera a su alrededor cuyo radio será la distancia desde el receptor hasta dicho satélite. De este modo, el receptor se encontrará en un punto de la superficie de esa esfera, aún por determinar. Repetimos el proceso con otro satélite. Al crearse esa segunda esfera, el dispositivo receptor se encontrará en alguno de los puntos de corte de ambas esferas, por lo que el resto de puntos se descartan. De nuevo, se utiliza un tercer satélite de modo que se crea una nueva esfera que cortará a las anteriores. De este modo, con el corte de las tres esferas, y teniendo en cuenta que el dispositivo se encuentra en la superficie terrestre, tendremos el punto concreto buscado. En caso de querer conocer la altitud, bastará con usar un cuarto satélite como referencia y repetir el proceso. En la Figura 2.4 vemos un esquema del proceso que acabamos de explicar empleando 3 y 4 satélites.

El problema del Sistema de Posicionamiento Global es que pierde mucha precisión cuando nos encontramos bajo superficies como túneles, tejados, etc. ya que la señal se debilita enormemente y el dispositivo no es capaz de llevar a cabo la triangulación de manera exacta. Es por esto que descartamos este sistema para nuestro trabajo, que se basa en la guía por interiores.

2.2.2. Wi-Fi

La técnica de posicionamiento mediante señales Wi-Fi fue una de las primeras que surgió para solventar el problema de la localización en interiores. Tal y como hemos mencionado

do en el apartado anterior, la señal GPS no es suficiente pues las barreras arquitectónicas como las paredes o tejados la debilitan enormemente. Por ello apareció esta técnica que se basa en leer con un dispositivo la intensidad de señal que recibe desde distintos puntos de acceso Wi-Fi del edificio. Una vez leída esta señal existen varios métodos para establecer la posición exacta del dispositivo. El más utilizado se basa en medir la intensidad con la que se recibe la señal en un cierto punto (*Received Signal Strength Indicator, RSSI*) desde varios puntos de acceso y, en base a eso, establecer la localización en la que se encuentra el dispositivo. Este método es fácil y barato de implementar. Su mayor inconveniente es que no proporciona una buena exactitud (de 2 a 5 metros) (Shah, 2012), ya que la señal es muy dependiente del ambiente: personas, paredes, etc.

Otro método que también se utiliza es el denominado *fingerprint*. Este consiste en guardar la intensidad de la señal desde distintos puntos de acceso en una base de datos de acuerdo a coordenadas conocidas del cliente durante una fase sin conexión. Durante la fase de conexión, las medidas actuales RSSI de una posición que no se conoce se comparan con las guardadas en la base de datos y, aquellas que más se le parezcan, se devuelven como estimación de la posición del usuario. Esta técnica tiene mejores resultados en cuanto a la exactitud, cuyo error baja a menos de 3m (Shah, 2012).

Una de las grandes ventajas de la tecnología Wi-Fi es que prácticamente la totalidad de las casas, colegios y edificios en general están equipados con una red Wi-Fi.

2.2.3. Balizas Bluetooth

Los *beacons* o balizas Bluetooth son pequeños dispositivos que emiten señales de radio. Estas señales los identifican de manera única y pueden ser captadas por otros dispositivos receptores, estableciéndose así un canal de comunicación que permanece vivo siempre que los receptores permanezcan en un radio de alcance de entre 10 y 30 metros como máximo, según el dispositivo. Es importante remarcar que generalmente los *beacons* no aceptan conexiones de otros dispositivos, lo que significa que no pueden registrar qué aparatos están cerca. Por tanto, esta simplicidad conlleva la necesidad de una aplicación capaz de interpretar la señal de la baliza. Otra característica de los *beacons* es que son de bajo consumo, es decir, sus baterías tienen una duración muy prolongada (aproximadamente 2 años) con una simple pila de botón, y su coste es reducido.

Esta tecnología se hizo muy popular en 2013 cuando Apple introdujo el estándar *iBeacon*¹⁴ y comenzó a utilizarlos para la navegación, más concretamente, para el posicionamiento en interiores. En 2015 Google, que no quiso quedarse atrás, lanzó el protocolo Eddystone, un protocolo que, a diferencia del de Apple, es de código abierto y ofrece soporte oficial tanto para iOS como para Android. Otra ventaja que incluye la versión de Google es que proporciona dos APIs que facilitan mucho el manejo de los *beacons*, y que emite cuatro paquetes distintos de información¹⁵, en lugar de uno como en el caso de los *iBeacons*. Estos paquetes son:

- **Eddystone-UID:** transmite un identificador de baliza único compuesto por 16 bytes, 10 de ellos referidos al espacio de nombres, que identifican a un grupo de *beacons*, y 6 que se refieren e identifican a la instancia particular dentro del grupo. Esta distinción entre espacio de nombres e instancia se pensó para optimizar el escaneo de *beacons*. Este paquete es idéntico al que ofrecen los *iBeacons*.

¹⁴<https://developer.apple.com/ibeacon/>

¹⁵<https://developers.google.com/beacons/eddystone>

- **Eddystone-URL:** transmite una URL, utilizando un formato de codificación, que se muestra como notificación silenciosa en el dispositivo.
- **Eddystone-TLM:** transmite información sobre la baliza: el nivel de la batería, los datos del sensor u otra información relevante para los administradores de balizas. Para poder usarse también como baliza necesita ir acompañado de otro tipo de marco (Eddystone-URL o Eddystone-UID).
- **Eddystone-EID:** emite un identificador encriptado que cambia periódicamente, de modo que su uso está restringido a aplicaciones y dispositivos autorizados.

Por todo esto, consideramos que utilizar el protocolo Eddystone es más ventajoso para nuestra aplicación, debido a la comodidad y facilidad que nos proporcionan las APIs

Por otro lado, de cara a establecer la posición exacta de un dispositivo receptor hay dos alternativas, una de ellas es triangular la posición a partir de la distancia obtenida por la señal de los tres *beacons* más cercanos. El inconveniente de este método es que se requiere un número muy alto de balizas para poder cubrir por completo todo el espacio, por lo que los costes de la instalación se elevan. Además, cualquier cambio en la estructura del edificio puede afectar a la posición de los *beacons* y, por ende, al algoritmo de triangulación. El otro método consiste en establecer los *beacons* en puntos de decisión (*landmarks*) donde los usuarios esperan recibir instrucciones. Algunos de estos puntos son las puertas, escaleras, intersecciones, etc. El número de balizas necesario es mucho más reducido por lo que los costes derivados de la instalación también. No obstante, hay que tener en cuenta que es muy importante decidir con cuidado la disposición exacta de los *beacons* para que estén lo menos expuestos posible a interferencias.

2.2.3.1. Interferencias en la señal Bluetooth

Los *beacons* son dispositivos que se pueden colocar tanto en interiores como en exteriores pero hay que tener en cuenta qué lugares son más aptos para su establecimiento de manera que la señal no sufra interferencias. Algunas recomendaciones son:

- Lugares altos (alrededor de 2,5m) para evitar las interferencias provocadas por los cuerpos de las personas ya que estos absorben parte de la señal.
- Lugares alejados a un metro aproximadamente de elementos que pueden alterar la intensidad de la señal como elementos metálicos, conductos de electricidad, iluminación, otras balizas, etc.
- En pasillos de menos de 4m de ancho, colocar la baliza en el centro para que cubra el espacio por igual. Si por el contrario el ancho es mayor, se deberán usar varias balizas.
- Colocar las balizas aproximadamente 1m antes de los lugares de interés (*landmarks*). Ejemplo: puertas, ascensores, escaleras, esquinas, etc.
- Considerar la orientación adecuada de la antena direccional del *beacon*, aunque esta depende por completo del fabricante. En ocasiones la baliza no emite una señal simétrica por completo sino que emite una señal en forma elíptica.

Otros factores difíciles de controlar que también afectan a la señal son las condiciones meteorológicas, la señal Wi-Fi, las señales Bluetooth de otros dispositivos (Wayfindr, 2018)...

2.3. Trabajos previos

Además del estudio de las aplicaciones ya existentes y de las distintas opciones para resolver el problema del posicionamiento, es importante conocer algunos de los trabajos previos que se han hecho en el campo de la navegación por interiores. En concreto nos centramos en dos trabajos de fin de grado realizados por alumnos de la Facultad de Informática de la UCM. Estos son especialmente interesantes puesto que, al igual que el nuestro, su caso de estudio se centra en la misma Facultad. Así mismo, se pone de manifiesto el relevante trabajo de otros compañeros.

2.3.1. Sistema de guía por voz en interiores 2012-2013

Proyecto realizado por Mariana Martín- Calderín de la Villa como trabajo de fin de grado durante los años 2012 y 2013 (de la Villa, 2014).

Este proyecto establece las bases para la creación de un sistema de guía en interiores. Es uno de los trabajos sobre el que se basó el proyecto de la sección siguiente y, por ende, también el nuestro.

La finalidad de este trabajo es guiar a un usuario en un escenario provisto de red Wi-Fi. En este caso, se utilizó como escenario, la primera planta de la Facultad de Informática de la Universidad Complutense de Madrid. Mediante el posicionamiento Wi-Fi, el sistema es capaz de captar la posición en la que se encuentra el usuario. Gracias al sistema de voz integrado en la aplicación, el usuario debe indicar a través de él el destino al que desea ir. Una vez queda localizado en el escenario el origen y el destino del usuario, el sistema genera una serie de instrucciones que serán transmitidas por voz al usuario e indicarán la ruta que debe seguir para alcanzar el destino solicitado.

2.3.2. Generador interactivo de instrucciones de guía sobre plataformas móviles 2013-2014

Proyecto realizado por Víctor Manuel Pose Murga, Víctor Gutiérrez Rodríguez y Juan Diego Lozano Martín como trabajo de fin de grado durante los años 2013 y 2014 (Víctor Gutiérrez Rodríguez y Murga, 2014).

Como el propio título del trabajo ya avanza, este proyecto tenía como finalidad el desarrollo de una aplicación de guía sobre Android. Esta aplicación permitía al usuario introducir el destino al que quería dirigirse mediante voz para que el sistema de generación de instrucciones le guiara paso a paso, mediante instrucciones sencillas. El caso de estudio utilizado en su caso fue la Facultad de Informática y el sistema de posicionamiento utilizado fue la triangulación mediante señales Wi-Fi.

Este proyecto estaba claramente dividido en dos partes, el cliente y el servidor. El cliente estaba constituido por la propia aplicación, mientras que el servidor constituía la parte lógica y con más carga computacional, pues era el encargado de la generación de instrucciones. Este servidor estaba implementado de manera independiente al edificio en el que se fuera a utilizar. Esto fue posible gracias a que la estructura, dividida en distintas secciones (ver Sección 3.2), del edificio se guardaba en archivos xml independientes del código. En su caso solo se disponía de la estructura correspondiente a la primera planta de la Facultad.

En los capítulos que siguen veremos con detalle los aspectos de este trabajo que han sido utilizados total o parcialmente.

2.4. Conclusiones

Tras este breve recorrido por algunas de las aplicaciones de navegación adaptadas para personas ciegas o con visibilidad reducida podemos decir que cada vez son más las opciones disponibles. Hemos visto desde aplicaciones de navegación por exteriores, como también por interiores, llegando hasta algunas tan específicas como *Wayfindr* que está dirigida al metro de Londres concretamente. Todas ellas se rigen por un patrón común: el de la simpleza, sin conllevar por ello una reducción de la funcionalidad. Estas aplicaciones nos permiten filtrar la información que se quiere recibir, guardar nuestros lugares más visitados, manejarlas mediante voz o con sacudidas del teléfono..., es decir, nos proporcionan un gran abanico de posibilidades que el usuario puede ejecutar de manera sencilla.

Por otro lado, si comparamos las apps, encontramos que aquellas de navegación por interiores están aún por desarrollar ya que el mapeo del interior de los edificios debe realizarse de manera particular e individual, convirtiéndose en una tarea mucho más tediosa que la que lleva a cabo el famoso coche de *Google Maps*. Además, el posicionamiento también es más complejo ya que no es posible utilizar el sistema GPS y hay que recurrir a la triangulación de señales Wi-Fi o a las balizas Bluetooth, teniendo que estudiar de nuevo cada caso concreto.

En cuanto al sistema de posicionamiento, para este trabajo descartamos desde un primer momento la tecnología GPS. En interiores su rendimiento no es el deseado, ya que la señal pierde intensidad cuando el dispositivo se encuentra en un lugar cerrado. Así, aunque la alternativa tecnológica Wi-Fi es también adecuada para solventar el problema del posicionamiento en interiores y cuenta con ventajas como la de aprovechar la infraestructura del edificio sin necesitar ningún dispositivo extra, también conlleva ciertos inconvenientes. Entre ellos destacamos que la intensidad de las señales Wi-Fi dependen mucho del entorno y, en ocasiones, puede ser complicado diferenciar la posición entre plantas, si estas no se encuentran a suficiente distancia. Además, los *beacons* cuentan con ventajas como su bajo coste y flexibilidad: podemos colocarlos donde queramos (son pequeños y ligeros) mientras que los puntos de acceso Wi-Fi vienen predeterminados, y tienen una precisión de 1 a 3 metros, algo más alta que con la señal Wi-Fi¹⁶. Por ello, para este proyecto nos hemos decantado por las balizas Bluetooth acompañadas del protocolo Eddystone y más concretamente por el método de los *landmarks*, ya que debido a las limitaciones de presupuesto y a las características de nuestros usuarios lo consideramos más adecuado pues así podemos estudiar mejor los puntos de interés y asegurarnos una buena señal y más precisión en ciertos puntos.

¹⁶<https://www.infsoft.com/technology/positioning-technologies/bluetooth-low-energy-beacons>

Capítulo 3

Mapeo de interiores mediante balizas Bluetooth

En este capítulo explicamos el estudio realizado sobre el funcionamiento de los *beacons* y su efectividad en el posicionamiento de interiores, concretamente en la Facultad de Informática de la UCM. Este trabajo de investigación ha sido muy satisfactorio ya que nos ha facilitado la toma de numerosas decisiones con las que elaborar una propuesta en resolución al mapeo del edificio y a la disposición concreta de las balizas que optimice el posicionamiento.

La Sección 3.1, aborda la primera investigación que se realizó con las balizas Bluetooth o *beacons*. En ella se detalla el estudio de la precisión de los *beacons* en cuanto a distancia se refiere. Para ello se implementaron dos aplicaciones muy sencillas, *miniapp* y *cuadrantes_v1*, cuya funcionalidad puede verse en detalle en las Secciones 3.1.1 y 3.1.2, respectivamente. Así mismo, en la Sección 3.1.2.1 se hace un estudio exhaustivo de las distintas pruebas que se realizaron con la aplicación *cuadrantes_v1* y cuyas conclusiones se recogen en la Sección 3.1.3. Una vez analizada la tecnología y estudiado su comportamiento, comienza el mapeo de la Facultad de Informática de la UCM en la Sección 3.2. Es en esta sección donde se establece la primera aproximación de la estructura e implementación de los archivos xml en los cuales se recoge la información referente al edificio. La investigación detallada sobre las mediciones y distintas pruebas que se llevaron a cabo para establecer la distribución de los cuadrantes y la ubicación final de los *beacons* se recoge en la Sección 3.3. Por último, la Sección 3.4 expone la estructura de los mencionados archivos xml.

3.1. Estudio de la precisión del posicionamiento mediante los *beacons*

Una vez que hemos concluido, por las razones expuestas en la Sección 2.4, escoger los *beacons* como sistema de posicionamiento, vamos a realizar un estudio profundizando en los aspectos tecnológicos de estas balizas. El objetivo es comprender por completo su funcionamiento y comportamiento a la hora de medir distancias, es decir, cuánto rango tiene la señal Bluetooth, con ayuda de qué función podemos recibir esta señal e interpretarla para determinar la distancia, qué margen de error presenta, en qué lugares es más aconsejable establecer las balizas para recibir mejor la señal y, por tanto, reducir el error, cómo de precisa es la distancia medida cuando el dispositivo móvil que recibe la señal Bluetooth

está en movimiento, etc.

Los *beacons* utilizados para este proyecto cuentan con una SDK (Kit de Desarrollo Software) propia de su marca (*Kontakt*) en la que incluyen funciones ya implementadas que nos permiten conocer qué *beacons* están en nuestro rango en un momento determinado y a cuántos metros están. Esta información se va actualizando cada cierto tiempo. También incluye un sistema de categorías en función de cómo de cerca o lejos esté un cierto dispositivo. Las categorías son las siguientes:

- *IMMEDIATE*: Si el dispositivo se encuentra a menos de 0,5 metros.
- *NEAR*: Si el dispositivo se encuentra entre 0,5 y 3 metros.
- *FAR*: Si el dispositivo se encuentra a más de 3 metros.
- *UNKNOWN*: Si se ha perdido la señal del dispositivo.

Una vez descubiertas estas funciones y comprendido el código decidimos desarrollar dos pequeñas aplicaciones que sirviesen para un doble propósito. Por un lado, para tener una primera toma de contacto con el entorno de programación escogido, Android Studio, y con la incorporación de las funciones ofrecidas por la librería de *Kontakt*. Y por otro, para analizar la precisión de la señal Bluetooth (con y sin movimiento), concretar la posición óptima de los *beacons* y desarrollar una primera idea sobre la estructuración de la información relevante del edificio en el mapeo y la implementación del código relativo al posicionamiento en el edificio. En las siguientes secciones presentamos las dos aplicaciones desarrolladas para estos fines.

3.1.1. Aplicación *miniapp*

Esta aplicación constituye la primera toma de contacto con los *beacons* y con el código de la SDK. Presenta una interfaz sencilla en la que aparece una tabla con las distintas categorías de proximidad para 3 balizas y, debajo, el espacio donde se indica el identificador de la baliza correspondiente. En el cuadro inferior aparecen dos botones muy intuitivos con los que el usuario puede interactuar con la app: Stop Scanning y Start Scanning. De esta manera, una vez que empieza el escaneo y las balizas se encuentran en el radio de detección, se incluyen los identificadores en cuestión y se colorea en verde la categoría de proximidad estimada en cada caso según corresponda. La lectura de los *beacons* se actualiza cada 2 segundos, tiempo establecido de antemano. En la Figura 3.1a podemos ver la interfaz descrita.

La idea que subyace al desarrollo de esta aplicación es introducirnos en Android Studio y manejar las funciones ofrecidas por la SDK de *Kontakt*, de ahí que hayamos creado una primera interfaz con botones, cambios de color, cuadros de texto, etc. Una vez hecho esto, comprobamos que las funciones preestablecidas por la SDK funcionaban y realizamos distintas pruebas en una posición fija para establecer el grado de confianza que podíamos tener en las categorías de proximidad definidas por *Kontakt*. El resultado fue muy positivo puesto que, en su mayoría, la categoría asignada se ajustaba a la realidad. Sin embargo, nos faltaba saber la precisión estricta de los *beacons* al recibir la distancia exacta en metros. Por ello creamos una segunda aplicación que, aunque fuese menos visual desde el punto de vista de la interfaz, nos diese más información sobre los *beacons* detectados.

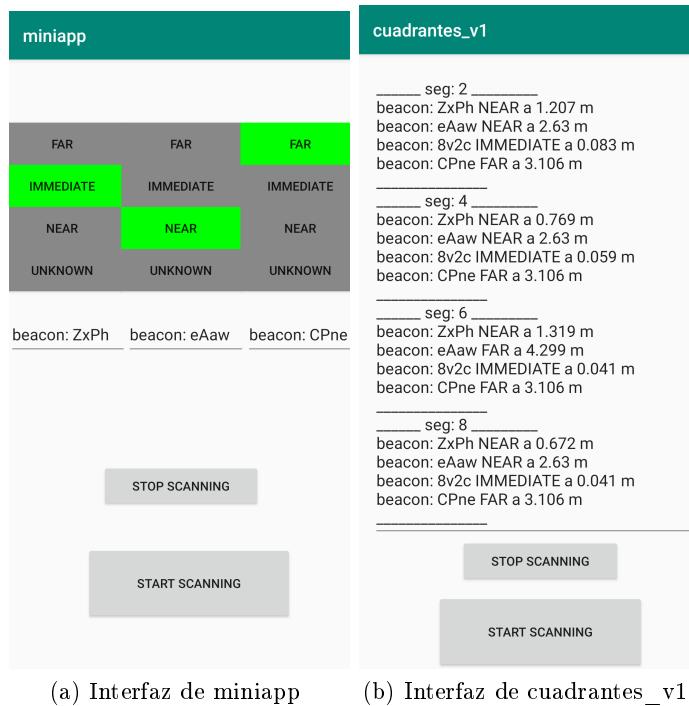


Figura 3.1: Aplicaciones auxiliares

3.1.2. Aplicación *cuadrantes_v1*

Cuadrantes_v1 es una aplicación austera. En ella aparece un gran cuadro de texto que ocupa prácticamente la totalidad de la pantalla y dos botones que se sitúan en el cuadro inferior, Stop Scanning y Start Scanning, a través de los cuales el usuario puede interactuar con la app. Su funcionamiento es sencillo e intuitivo: cuando el usuario pulsa Start Scanning comienza el escaneo de balizas y cuando una o varias entran en el radio de detección, la app plasma su información (identificador, distancia en metros y categoría de proximidad) en el cuadro de texto. Esta información se actualiza cada 2 segundos (tiempo configurable) hasta que el botón Stop Scanning es pulsado, momento en el cual cesa el escaneo. La Figura 3.1b muestra la interfaz de la aplicación descrita.

Como mencionamos anteriormente, hemos creado esta aplicación con el fin de obtener más información sobre el error cometido a la hora de determinar la distancia en metros a la que se encuentran los *beacons* (tanto si el dispositivo se encuentra parado como en movimiento), así como para estudiar cuales son los factores que alteran su señal. De esta manera podremos determinar los puntos claves donde colocar las balizas y abordar, tras ello, el mapeo del edificio. De ahí que el nombre de la aplicación sea *cuadrantes_v1* ya que, como veremos, los cuadrantes serán una pieza fundamental en el mapeo. A continuación exponemos las pruebas realizadas con ayuda de esta aplicación y algunas de las consecuencias derivadas de los resultados.

3.1.2.1. Pruebas con *cuadrantes_v1*

Las primeras pruebas realizadas con la aplicación *cuadrantes_v1* han consistido en colocar tres *beacons*, concretamente aquellos con identificadores CPne, 8v2c y eAaw, en distintos puntos de la Facultad de Informática y observar la distancia registrada desde un dispositivo móvil ubicado en un punto fijo. De esta manera, hemos podido medir el error



Figura 3.2: Gráfico con las distancias medidas al *beacon* CPne.



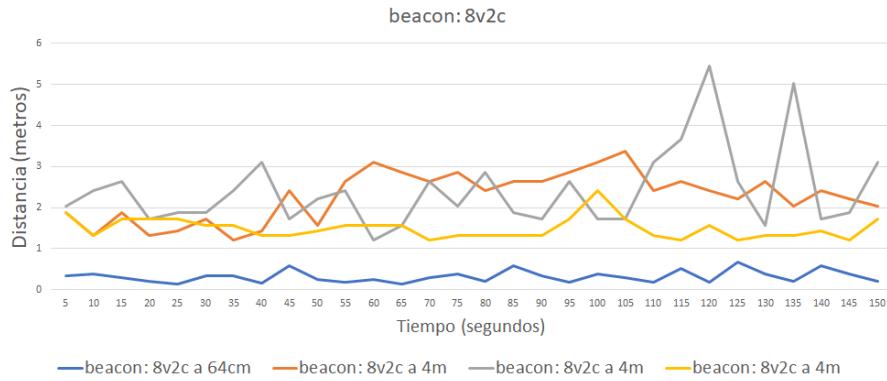
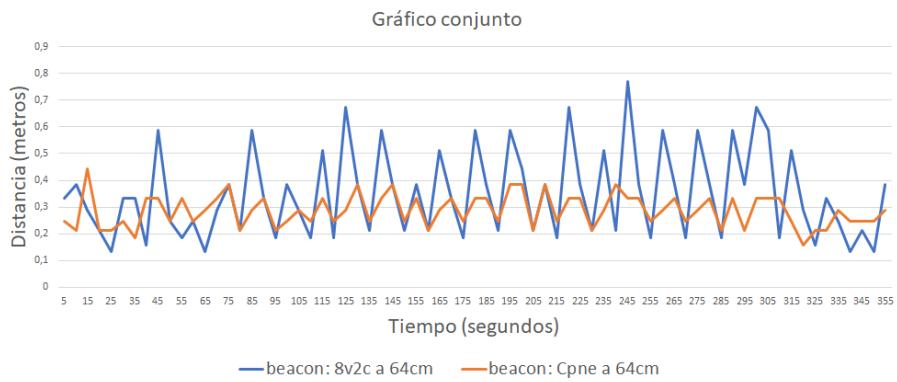
Figura 3.3: Gráfico con las distancias medidas al *beacon* eAaw.

cometido en cada caso y estimar las posibles interferencias. Para mostrar los resultados hemos creado tres gráficas que muestran la distancia a la que se detectan las balizas a lo largo del tiempo y la medida real a la que estaban situadas.

En la Figura 3.2 vemos las lecturas que nos ha proporcionado la aplicación *cuadrantes v1* del *beacon* con identificador CPne. En este caso, hemos realizado 4 estudios independientes marcados con distintos colores: azul -*beacon* a 64cm, naranja -*beacon* a 2m, gris -*beacon* a 2,5m y amarillo -*beacon* a 5 metros-. De esta gráfica concluimos que cuanto mayor es la distancia a la que se encuentra la baliza, véase la función amarilla (5m), más fluctúa la medida estimada convirtiéndose en un dato poco fiable. Correlativamente, a menor distancia menor error. Podemos ver que la función naranja (2m) es bastante fiel a la realidad, fluctúa en un intervalo de un metro a lo largo de toda la medición. Por último, el resultado de la gráfica azul (64cm) es bastante preciso, presenta un error despreciable durante toda la medición.

En el caso de la Figura 3.3, en la que se estima la distancia para el *beacon* eAaw, pero usando distancias más pequeñas, el comportamiento es similar. En líneas generales, el valor medio estimado se corresponde con la distancia real a la que se encuentran las balizas. Sin embargo, en esta medición hemos contemplado una novedad que se ha repetido en numerosas ocasiones convirtiéndose en un patrón de comportamiento y es que durante los primeros segundos en los que arranca la aplicación, las estimaciones son menos precisas, dando lugar a picos importantes que más tarde se suavizan.

La Figura 3.4 recoge cuatro mediciones para el *beacon* 8v2c, tres de ellas tomadas para una misma distancia (4m) y una a 64 cm. Observamos de nuevo que el error es despreciable cuando la baliza se encuentra muy próxima. Por el contrario, los valores que recogen las gráficas en las que el dispositivo se encuentra a 4m están muy por debajo de lo esperado,

Figura 3.4: Gráfico con las distancias medidas al *beacon* 8v2c.Figura 3.5: Gráfico de los *beacons* 8v2c y CPne superpuestos.

y solo una de ellas alcanza el valor real y en todas las direcciones. Este factor común nos reveló una alteración considerable de la intensidad de la señal por lo que empezamos a entrever que había factores en el entorno que la debilitan. Es por esto que la última gráfica, plasmada en la Figura 3.5, hace referencia a una situación particular. Se trata de dos *beacons* diferentes situados a la misma distancia uno encima del otro. El objetivo de este estudio era ver si uno de los factores que podía alterar la intensidad de señal era la presencia de otras balizas, y efectivamente como podemos observar, la intensidad de ambas señales es bastante más baja de lo esperado, prácticamente en ningún momento alcanzan el valor real. En este caso particular, el error es tolerable ya que los *beacons* están situados a muy poca distancia pero sí nos advierte de que la señal Bluetooth es sensible a interferencias provocadas por otros *beacons* del entorno, lo cual hemos de tener muy presente a la hora de determinar la ubicación de los *beacons* a lo largo de la facultad para no colocarlos demasiado próximos. En la Sección 2.2.3.1 del Capítulo 2 se han expuesto algunas de las causas de las interferencias.

3.1.3. Conclusiones de la precisión de los *beacons*

Con todos estos resultados hemos decidido que para el posicionamiento utilizaremos el dato del *beacon* más cercano ya que es la medida más fiable (por estar a menos metros del dispositivo) y consideraremos que el usuario se encuentra en las inmediaciones de dicho *beacon*. En la Sección 3.2 especificaremos la asociación realizada entre “las inmediaciones de un *beacon*” y un punto concreto de la Facultad de Informática.

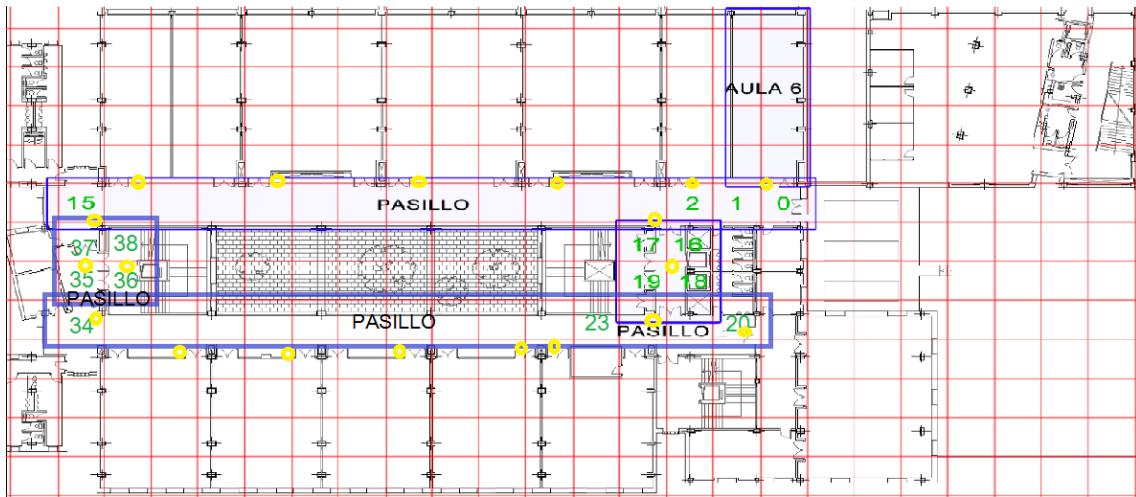


Figura 3.6: Primera versión del mapeo de la primera planta de la Facultad de Informática.

Con esta idea hemos desarrollado una primera aplicación de prueba que además de captar el *beacon* más próximo al usuario emite un pitido a mayor o menor frecuencia según su categoría de proximidad. Esta app se llama *pruebaSonido* y presenta la misma estética que *cuadrantes_v1*. Su funcionamiento es sencillo: al iniciar el escaneo capta el *beacon* más cercano y lo fija. Tras ello, actualiza su información (concretamente, la distancia registrada y la categoría de proximidad) y pita a una frecuencia u otra según la distancia a la que se encuentre el *beacon* (cuanto más cerca, el pitido es más rápido y se ralentiza a medida que la baliza está a mayor distancia). La actividad continúa hasta que el botón Stop Scanning es pulsado. Al igual que en *cuadrantes v1* la información sobre las balizas detectadas queda plasmada en el cuadro de texto.

El objetivo de *pruebaSonido* no es solo generar un primer código que seleccione el *beacon* más cercano sino también llevar a cabo las primeras pruebas de estimación de la distancia a la que se encuentran las balizas en movimiento. Estos resultados han sido muy similares a los obtenidos estéticamente: la estimación sigue siendo más precisa cuanto más próxima está la baliza del dispositivo que la rastrea. Sin embargo, hay que ajustar el tiempo de actualización del escaneo si se quiere caminar a mayor velocidad (para que la aplicación reaccione en tiempo real y no sufra retrasos). Tras diversas pruebas hemos concluido que 2s es una medida óptima en lo que a nuestra aplicación concierne ya que nuestros usuarios, al presentar discapacidad visual, no caminan especialmente rápido. Por otro lado, esta aplicación nos ha servido también para incluir sonidos. Esto puede parecer insignificante, pero si nos paramos a pensarlo hay muchísimos sonidos que tenemos asociados con determinadas acciones, situaciones y/o aplicaciones que sin darnos cuenta nos guían en su uso y nos proporcionan la seguridad de que estamos utilizándola bien o la certeza de que tenemos que rectificar. Este recurso es indispensable en nuestra app ya que nuestros principales usuarios no podrán ver la aplicación.

3.2. Mapeo del edificio de la Facultad de Informática

Para el mapeo de la Facultad de Informática nos hemos apoyado fundamentalmente en el proyecto de TFG *Generador interactivo de instrucciones de guía sobre plataformas móviles* (Víctor Gutiérrez Rodríguez y Murga, 2014). De este, hemos reutilizado el sistema de estructuración basado en plantas que a su vez se dividen en cuadrantes con identificador



Figura 3.7: Versión final del mapeo de la primera planta de la Facultad de Informática.

único, lo que facilita determinar la planta en la que nos encontramos. Originariamente, cada cuadrante se correspondía con nueve baldosas aproximadamente, tanto en el largo como en el ancho, y los cuadrantes se juntaban formando estancias (pasillos, aulas, etc.). De esta manera quedaban definidas cada una de las plantas del edificio y así empezamos definiendo también las nuestras. Sin embargo, sus cuadrantes contaban con dos números asociados que hacían referencia a las coordenadas sureste y noroeste que empleaban en el posicionamiento mediante triangulación Wi-Fi. Este es el primer cambio que hemos implementado ya que carecen de sentido en un posicionamiento como el nuestro, que se basa en puntos de decisión. En su lugar, debemos asociar los *beacons* a los cuadrantes correspondientes. Para ello, empezamos determinando que no todos los cuadrantes tendrían una baliza asociada, sino que serían solo aquellos que tuviesen un punto de decisión. En la Figura 3.6 mostramos el primer acercamiento al mapeo de la planta 1 con los cuadrantes originales (en rojo), su identificador (en verde), algunas estancias (en azul) y la posición final de los *beacons* (en amarillo). La disposición de los *beacons* y su evolución hasta alcanzar la posición final se explicará y justificará adecuadamente en la Sección 3.3.

Como nuestro posicionamiento se fundamenta en el *beacon* más cercano, hemos considerado que el usuario se encuentra en el cuadrante asociado a dicho *beacon*, de esta manera es como hemos relacionado la frase “encontrarse en las inmediaciones de un *beacon*” con un punto del mapa. Sin embargo, por este mismo motivo, aquellos cuadrantes que no tienen asociada ninguna baliza Bluetooth carecen de interés ya que no hay manera de detectarlos y saber si el usuario se encuentra en ellos. Es por esto que el siguiente cambio ha sido juntar ciertos cuadrantes formando unos nuevos más grandes. En la Figura 3.7 encontramos la versión final del mapeo, en la que solo permanecen los cuadrantes con baliza. Además podemos observar que también hemos suprimido los cuadrantes de las aulas, sala de juntas, etc. esto se debe a que nuestra aplicación no ofrece una guía dentro de estas estancias sino que funciona como guía de puerta a puerta. Por ello, en lugar de tener cuadrantes que se unen formando estancias hemos realizado el cambio a cuadrantes que se agrupan en plantas.

La novedad más importante que incluye nuestro proyecto con respecto a los trabajos

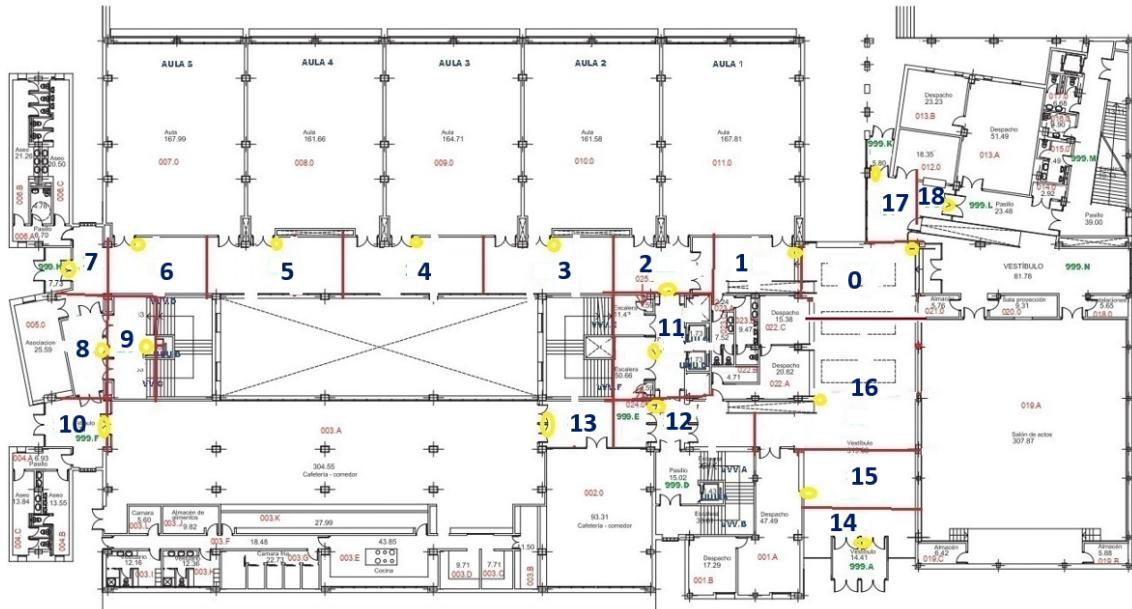


Figura 3.8: Mapeo de la planta baja de la Facultad de Informática, incluyendo la numeración de los cuadrantes y las posiciones de los *beacons*.

predecesores en este aspecto es el mapeo de la planta baja (ver Figura 3.8) y la conexión de unas plantas con otras a través de los cuadrantes en los que se encuentran los ascensores y escaleras de manera que sea viable incluir rutas de una a otra.

3.3. Mediciones y distribución de los *beacons* en la Facultad de Informática

Ahora que hemos superado la primera barrera tecnológica y tenemos una idea más clara del mapeo, nos disponemos a dar el siguiente paso hacia la resolución del problema del posicionamiento. En este apartado proponemos una posible disposición de los *beacons* en la Facultad de Informática de la UCM, para ello explicamos qué puntos hemos empezado considerando como puntos de decisión y su evolución hasta formar la disposición final que aparece en las Figuras 3.7 y 3.8 como consecuencia de una serie de mediciones.

Antes de nada vamos a definir qué es un punto de decisión: Los puntos de decisión son aquellos lugares en los que el usuario requerirá la siguiente instrucción bien porque se ha creado incertidumbre (intersección de caminos), porque ha pasado mucho tiempo de la última instrucción (recta muy larga) o bien porque ha llegado al destino y requiere confirmación. Los puntos de decisión que hemos considerado en la Facultad de Informática son: los destinos (aulas, cafetería, biblioteca, conserjería, secretaría, salón de actos, sala de juntas, sala de grados, etc.), las intersecciones de caminos, los ascensores y escaleras, y las puertas de entrada o salida del edificio.

En la Figura 3.9 mostramos visualmente los puntos de decisión que inicialmente hemos considerado (círculos rojos) en uno de los pasillos principales de la primera planta y los puntos desde los cuales hemos llevado a cabo las mediciones (cruces verdes). Esta estructura se repite tanto en la misma planta como en la planta baja por lo que las decisiones tomadas han sido las mismas y la evolución de la ubicación de los *beacons* hasta ocupar su posición final ha sido copiada sin realizar pruebas distintas.

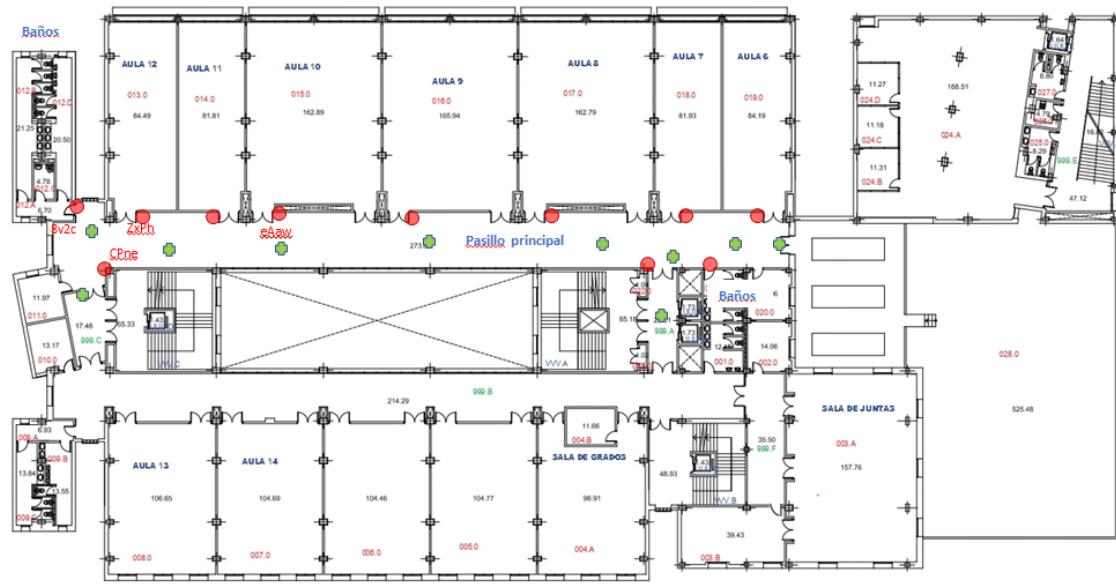


Figura 3.9: Mapa de la primera planta de la Facultad de Informática con la ubicación de los *beacons* (rojo) y los puntos de medición (verde).

De manera análoga, en la Figura 3.10 mostramos la ubicación inicial de los *beacons* en la zona del *hall* de la planta baja y los puntos desde los cuales se han realizado las mediciones. El objetivo de todas estas mediciones es conocer la ubicación óptima de los *beacons* para que a nuestro paso cerca de ellos, la distancia registrada sea lo más precisa posible y no sufra muchas interferencias. Este estudio debe ser especialmente exhaustivo en las zonas más delicadas, como las intersecciones o los lugares en los que se acumulan varios puntos de interés y que por ende, son más susceptibles a sufrir interferencias procedentes de otros *beacons*. En la Sección X se pueden ver los resultados de estas mediciones.

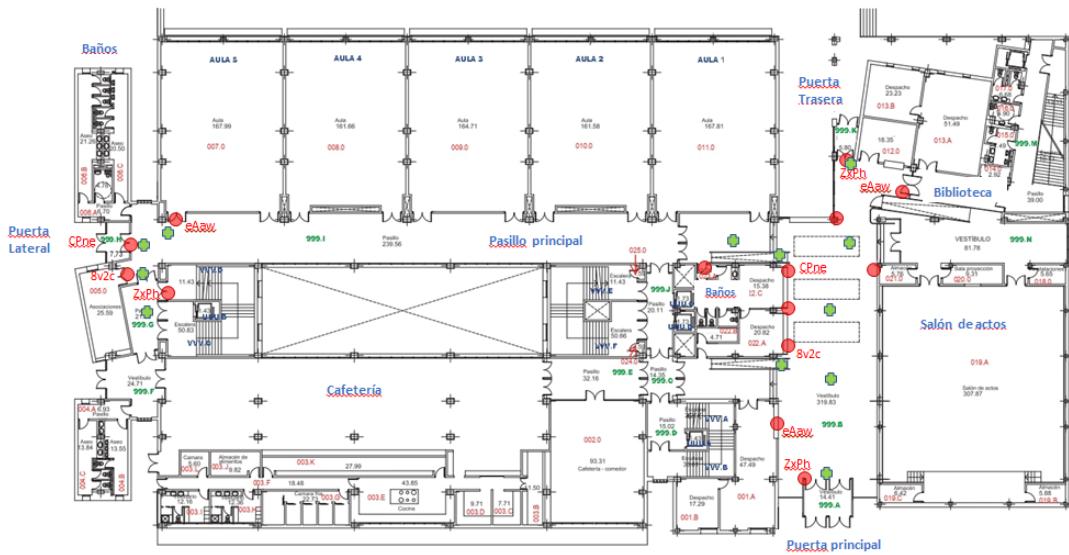


Figura 3.10: Mapa de la planta baja de la Facultad de Informática con la ubicación de los *beacons* (rojo) y los puntos de medición (verde).

A continuación exponemos las conclusiones obtenidas tras el estudio de los resultados:

- Los *beacons* no deben situarse demasiado cerca ya que las señales interfieren entre sí y alteran las distancias no pudiendo distinguir cuál es el *beacon* más cercano. Por este motivo, hemos acordado no poner, por el momento, balizas en los baños ya que, tanto en la primera planta como en la planta baja, se encuentran en zonas repletas de puntos de interés. Para informar de la presencia de los baños y otras zonas de interés se desarrollará una funcionalidad que al pasar cerca de ellos te avisará de su presencia.
- En lugares diáfanos, como el *hall*, la señal de los *beacons* fluye con mayor libertad ya que no se encuentra con obstáculos. Es por esto, que deben situarse a mayor distancia entre sí. Uno de los problemas que se ha derivado de este hecho es cómo cubrir dicho espacio. Inicialmente colocamos una baliza en conserjería, otra en la intersección superior con el pasillo principal, otra en la intersección inferior con el pasillo que conduce a cafetería y otra enfrente para indicar la entrada al salón de actos (ver Figura 3.10). Tras numerosas mediciones, la solución que proponemos es colocar las balizas de modo que una pueda servir para cubrir dos o más puntos de interés. Consecuentemente, hemos suprimido el *beacon* de conserjería y hemos dejado exclusivamente el de las intersecciones ya que la inferior está muy próxima a la ventanilla de conserjería y puede reutilizarse para los dos puntos clave, y hemos desplazado un poco hacia arriba el *beacon* del salón de actos para que sirva tanto para marcar dicho destino como para marcar la esquina superior que conduce hacia la biblioteca. En la Figura 3.11 se puede ver el resultado final.
- Hemos hecho pruebas con los *beacons* sobre distintas superficies y hemos comprobado que efectivamente el material puede alterar la señal. Particularmente, en el caso de las mediciones de la puerta lateral del edificio (lado izquierdo del mapa) hemos colocado el *beacon* CPne justo encima de la puerta, en un bisel metálico que sobresale. El resultado ha sido que la señal de la baliza se proyecta con mayor intensidad, dando lugar a que la distancia estimada sea menor. Es decir, la aplicación sugiere que CPne está más cerca de lo que en realidad está. Tras esto, hemos concluido que lo óptimo es poner los *beacons* sobre las mismas superficies, a ser posible no metálicas, y a la misma altura para que estén en igualdad de condiciones.
- La última puntualización que hemos hecho tras las mediciones es que los *beacons* de las intersecciones deben situarse en un punto lo más neutro posible ya que, al menos, se puede llegar desde dos puntos distintos y la señal se debe recibir de manera simétrica.

Este estudio no ha sido todo lo extenso que nos hubiese gustado debido al cierre de la Facultad de Informática a causa del COVID-19.

En el próximo apartado veremos cómo queda reflejado todo el mapeo visto en la Sección 3.2.

3.4. Representación del mapeo en los XML

Todo nuestro mapeo se representa en ficheros XML que hemos sacado fuera de la aplicación (se encuentran en una carpeta llamada `xml_modif`). En ellos hemos seguido el sistema de divisiones y estructuración del proyecto de TFG *Generador interactivo de instrucciones de guía sobre plataformas móviles* (Víctor Gutiérrez Rodríguez y Murga,

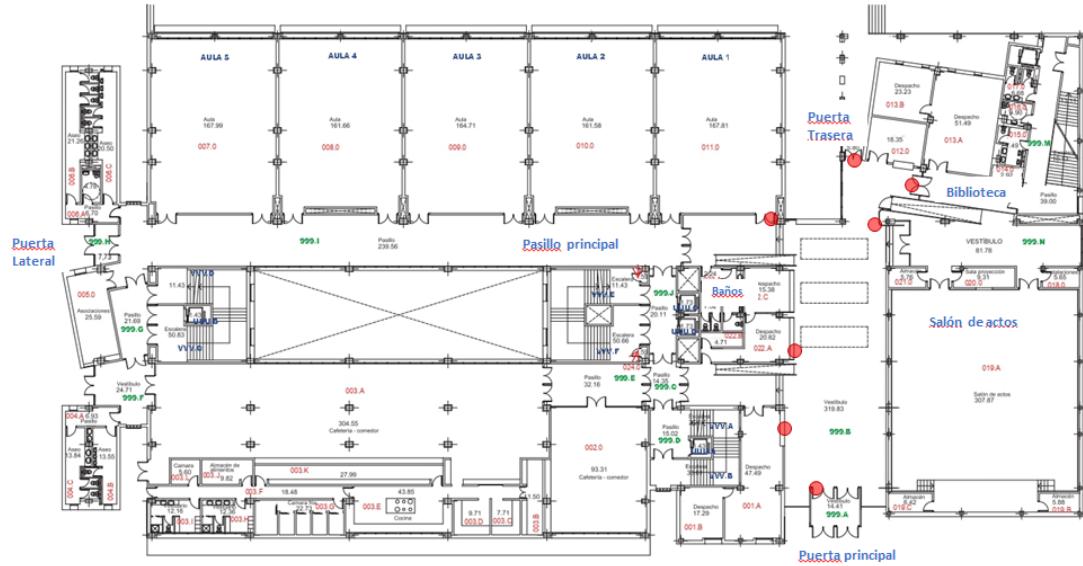


Figura 3.11: Mapa de la planta baja de la Facultad de Informática con la ubicación definitiva de los beacons (rojo) en el hall.

2014), este permite que se puedan incluir de manera sencilla las diferentes plantas del edificio, es decir, se indican las distintas plantas y el archivo xml asociado a cada una de ellas. A continuación detallamos el significado de cada campo y vemos un ejemplo de la estructura del archivo.

- *nombre*: Nombre representativo de la planta que estemos implementando. En el ejemplo se ha utilizado el valor *plantabaja* y *planta1*.
- *archivo*: Nombre del fichero xml de la planta en cuestión. No es necesario añadir la extensión .xml. En el ejemplo lo hemos llamado *plantabajaarchivo* y *planta1archivo*.

```
<?xml version="1.0" encoding="UTF-8"?>
<edificio>
    <planta nombre="plantabaja">
        <archivo>plantabajaarchivo</archivo>
    </planta>
    <planta nombre="planta1">
        <archivo>planta1archivo</archivo>
    </planta>
</edificio>
```

NombreFichero.xml: Se corresponde con el archivo propio de cada planta. Cada planta incluye un valor que indica el número de planta en el que nos encontramos y un identificador de estancia que da nombre a la planta y que agrupa al conjunto de cuadrantes que la forman. A su vez, cada cuadrante contiene un identificador único, el nombre del beacon asociado, información sobre los cuadrantes colindantes, la posición en el cuadrante donde se encuentra el punto de interés (norte, sur, este u oeste), información sobre los pesos que tiene cada una de sus conexiones (cuya utilidad se presenta en la Sección 4.1.2),

información relevante del cuadrante y la medida en metros del mismo. A continuación detallamos el significado de cada campo y vemos un ejemplo de la estructura del archivo.

- *Z*: Indica el número de planta. En el ejemplo le hemos dado el valor 0.
- *id*: Se corresponde con el nombre de la planta en la que nos encontramos. En el ejemplo la hemos llamado *plantabaja*.
- *idc*: Hace referencia al identificador único de cada cuadrante. En el ejemplo le hemos dado el valor 20.
- *beacon*: Hace referencia al identificador de la baliza asociada a dicho cuadrante. En el ejemplo le hemos dado el valor *beacon0*.
- *conectado*: Indica los identificadores de los cuadrantes colindantes por los cuatro puntos cardinales. El valor -1 representa la pared. Este campo ha sido reutilizado de trabajos anteriores ya que resulta clave para establecer una red de cuadrantes que nos permita generar una ruta válida pasando a través de ellos.
- *posdestino*: Nos indica en qué posición del cuadrante está situado el punto de interés. En nuestro caso es el salón de actos, correspondiente al cuadrante 0, como se puede ver en la Figura 3.8 el salón de actos se sitúa al lado derecho del cuadrante (oeste)¹.
- *pesos*: Para cada una de las conexiones del cuadrante se establece el peso de dicha conexión. Esto será de gran utilidad pues el algoritmo utilizado para el cálculo de la ruta es el algoritmo de *Dijkstra*. Los detalles pueden verse en la Sección 4.1.2. Los pesos negativos se asignan a conexiones inexistentes.
- *info*: Este campo contiene la información relevante del cuadrante. La utilidad e importancia de este campo radica en informar al usuario, si lo desea, de qué hay a su paso por la ruta hasta el destino seleccionado. Esta idea nació tras la reunión en la ONCE en la que nos acercamos mucho a las necesidades de nuestros usuarios. En el ejemplo le hemos dado el valor *salon de actos*.
- *metros*: Este campo contiene la medida en metros del cuadrante lo que aporta mucha precisión a las instrucciones. En el ejemplo le hemos dado el valor 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<planta>
    <Z>0</Z>
    <estancia id="plantabaja">
        <cuadrantes>
            <cuadrante idc="0">
                <beacon>beacon0</beacon>
                <conectado>
                    <norte>16</norte>
                    <sur>17</sur>
                    <este>1</este>
                    <oeste>-1</oeste>
                </conectado>
                <posdestino>oeste</posdestino>
                <pesos>
```

¹La nomenclatura se establece con el sur en la parte de arriba, el oeste a la derecha, el norte abajo y el este a la izquierda del cuadrante.

```
<p_norte>1</p_norte>
<p_sur>1</p_sur>
<p_este>1</p_este>
<p_oeste>-1</p_oeste>
</pesos>
<info>salon de actos</info>
<metros>5</metros>
</cuadrante>
</cuadrantes>
</estancia>
</planta>
```


Capítulo 4

Diseño e implementación

En este capítulo abordaremos los detalles técnicos de nuestra aplicación Blind Bit. Esta aplicación está diseñada como un modelo cliente-servidor, en la que el cliente (dispositivo móvil) se encarga de solicitar los datos necesarios (origen, destino, posición actual, etc.) al usuario para enviárselos al servidor, quien calculará la ruta y responde con la guía correspondiente. De esta manera, nuestra aplicación está mejor organizada, es más eficiente y evita que los dispositivos móviles se quedan sin batería rápidamente a causa de una pesada carga computacional.

Al igual que la aplicación este capítulo se divide en dos partes. La primera de ellas, Sección 4.1, expone el funcionamiento general del servidor (Sección 4.1.1) así como la implementación de sus dos funcionalidades principales: el cálculo de la ruta óptima (Sección 4.1.2) y la generación de instrucciones (Sección 4.1.3). La segunda, Sección 4.2, se centra en la implementación y el diseño de la aplicación móvil. En la Sección 4.2.1 revisamos la interfaz de la aplicación y la justificación de su diseño, mientras que en la Sección 4.2.2 nos adentramos en su funcionamiento desde el punto de vista técnico.

4.1. Servidor

El servidor constituye una parte indispensable del proyecto ya que se encarga de realizar los cálculos más pesados para no sobrecargar al dispositivo móvil. Sus principales funcionalidades son:

- Almacenar la estructura e información referente al edificio mapeado.
- Permanecer a la escucha de cualquier cliente que solicite conexión.
- Solventar el posicionamiento del cliente conectado.
- Generar la ruta óptima desde la posición actual hasta el destino indicado por el cliente.

La aplicación servidor está diferenciada en dos partes: el código escrito en lenguaje Java y los archivos correspondientes al edificio que se mapea, en nuestro caso la Facultad de Informática de la UCM. Estos archivos son los xml mencionados en la Sección 3.2 y un archivo json que contiene la información referente a los distintos cuadrantes asociados a los destinos considerados por nuestra aplicación (por ejemplo, secretaría, conserjería, cafetería, las aulas, etc.). Buena parte del código que conforma el servidor ha sido reutilizado de

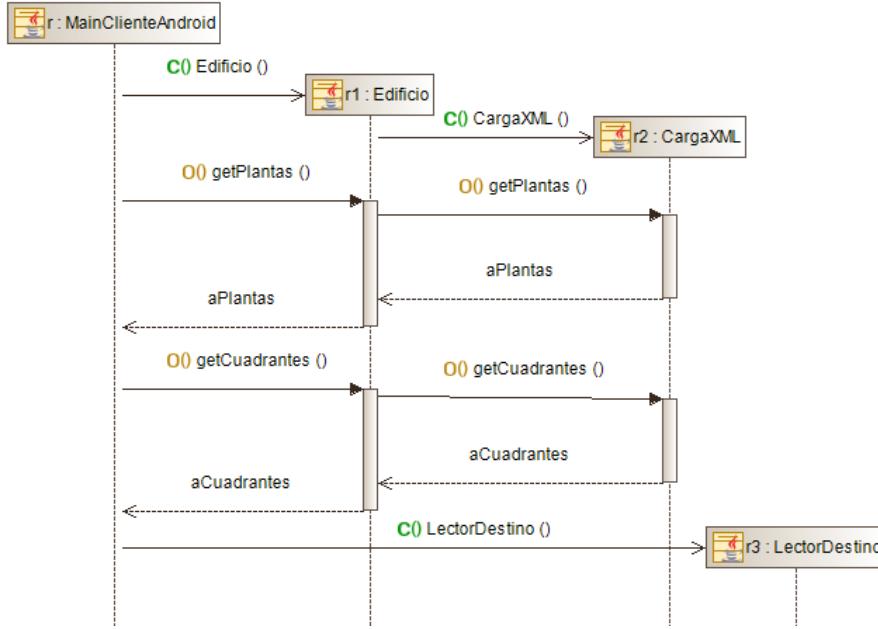


Figura 4.1: Diagrama de secuencia para el arranque del servidor.

trabajos anteriores, concretamente de los proyectos de TFG *Generador interactivo de instrucciones de guía sobre plataformas móviles* (Víctor Gutiérrez Rodríguez y Murga, 2014) y del proyecto *Sistema de guía por voz en interiores* (de la Villa, 2014). La estructura de nuestro servidor se corresponde con la de Víctor Gutiérrez Rodríguez y Murga (2014). Sin embargo, hemos introducido cambios notorios para el desarrollo de esta aplicación que comentaremos a continuación.

4.1.1. Funcionamiento del servidor

Para entender mejor la estructura del servidor, haremos un recorrido desde la carga de la información procedente de los archivos xml que representan el mapeo del edificio hasta la conexión con el cliente y el cálculo de la ruta:

- *Arranque del servidor*: Cuando el servidor arranca es necesario que guarde la información relativa al edificio para que pueda almacenar su estructura y generar una ruta válida para proporcionársela al cliente que la solicita. Al igual que los xml, el código relacionado con la carga de estos archivos, que conforma las clases *CargaXML* y *Edificio*, también se apoya en el del proyecto de Víctor Gutiérrez Rodríguez y Murga (2014) ya mencionado en otras ocasiones. Estas clases permiten cargar los archivos y almacenar la información referente a los cuadrantes estructurada en plantas. En la Figura 4.1 puede verse el diagrama de secuencia correspondiente a las explicaciones que siguen.

Para nuestro trabajo se han añadido los atributos de clases necesarios para guardar la información nueva incluida en los xml (como los *beacons*, los metros que ocupa el cuadrante, los pesos asociados a las conexiones, la ubicación del punto de interés o la información relevante de este) y se han eliminado aquellas que ya no se utilizan (como las coordenadas sureste y noroeste que no están presentes en nuestro proyecto). Una vez que está cargada esta información y ya tenemos la lista de cuadrantes existentes, pasamos a generar la matriz de adyacencia de la clase *ListaCuadrantes*

```

beacon34 beacon35 beacon36 beacon20 beacon21 beacon22 beacon23
beacon24 FINAL|Continua recto 20.0 metros. Luego gira a la
izquierda.|Continua recto 10.0 metros. Luego gira a la
izquierda.|Continua recto 10.0 metros. Luego gira a la izquierda.|Gira a
la izquierda. Luego continua recto 20.0 metros.|Continua recto 15.0
metros. Luego espera a la siguiente indicación.|Continua recto 10.0
metros. Luego espera a la siguiente indicación.|Continua recto 5.0 metros.
Luego espera a la siguiente indicación.|Su destino está a la derecha. El
recorrido ha finalizado.|no@no@no@si@no@no@no@no|Información
adicional: secretaria (medida ancho)|Información adicional: intersección y
conserjería (necesitará medida de ancho y de largo)|Información
adicional: salón de actos (necesitará medida de ancho y
largo)|Información adicional: intersección, aula 1 y hay baños (medida
largo). Hay dos pequeños escalones|Información adicional: intersección
(medida de largo y ancho)|Información adicional: aula 2|Información
adicional: aula 3@no

```

Figura 4.2: Ejemplo de la información generada por el servidor para una ruta desde el cuadrante 14 al 4.

con la que representamos el mapa en forma de grafo y establecemos las conexiones entre cuadrantes. Esta tarea es relativamente sencilla, pues los xml nos proporcionan la información sobre los cuadrantes colindantes. El cambio más notorio que se ha introducido en este punto es dotar a las conexiones de la matriz de adyacencia de determinados pesos según la adaptabilidad de la ruta a nuestros usuarios objetivo. Estos cambios aparecen explicados con detalle en la Sección 4.1.2.

Es en este momento inicial cuando el servidor también carga el archivo *destinos.json* con la lista de destinos y los cuadrantes asociados correspondientes. Una de sus entradas es, por ejemplo: {"lugar": "aula 5", "cuadrante": "6"}. En la clase *LectorDestino* (reutilizada del proyecto de TFG (Víctor Gutiérrez Rodríguez y Murga, 2014)) se almacenan cada una de las entradas en una tabla hash que cuenta con un campo clave de tipo String y un campo valor de tipo Integer que relaciona el lugar con su cuadrante asociado.

- *Conexión con el cliente:* Una vez que el servidor ha almacenado toda la información correspondiente al edificio está preparado para recibir peticiones de los clientes. El servidor queda entonces a la espera de los clientes en la clase *MainClienteAndroid*, escuchando a través de un *webSocket* en un puerto determinado. Esta conexión cliente-servidor constituye otro de los cambios principales realizados con respecto a Víctor Gutiérrez Rodríguez y Murga (2014), pues se ha reestructurado por completo. En primer lugar, la conexión ya no se hace por medio de *sockets* sino por medio de *webSockets*. Esto implica una mayor seguridad en el intercambio de mensajes (se emplea el protocolo http) y permite que el código del servidor pueda ser utilizado como servidor externo. En nuestro caso hemos montado el servidor sobre una máquina virtual con una IP pública de la Facultad de Informática utilizando Tom-

Cat¹. Esto permite que los clientes puedan acceder al servidor desde cualquier red, lo cual es primordial para poder establecer conexión con el servidor desde la propia Facultad, en particular. En segundo lugar, el número de conexiones que tiene que establecer el cliente con el servidor para obtener la información necesaria de la ruta se ha optimizado al máximo. En un solo mensaje el servidor envía al cliente toda la información necesaria. En el proyecto que hemos tomado como modelo, el cliente, en cambio, debía solicitar al servidor una nueva instrucción cada vez que actualizaba su posición. Ahora el cliente hace una única petición indicando su *beacon* más cercano (que se tomará como origen) y el destino al que quiere ir, en un mensaje del tipo *IDdelBeaconOrigen|destino*, por ejemplo “CPne|aula 8”. Cuando el servidor recibe este mensaje, se encarga de generar la ruta desde el origen hasta el destino y enviársela al cliente. La información que se envía está compuesta por la lista de *beacons* asociados a los cuadrantes que conforman la ruta desde el origen hasta el destino, las instrucciones necesarias, una lista de *booleanos* que indican cuándo hay que hacer un giro en la ruta y la información adicional de los cuadrantes que conforman la ruta. Por ejemplo, para la petición del cliente *beacon14|aula 3*, donde suponemos que *beacon14* se encuentra en el cuadrante 14 y que el aula 3 es el cuadrante 4 (ver Figura 4.4, línea verde), obtendríamos el mensaje que vemos en la Figura 4.2. La lista de *beacons* se representa en verde con un centinela FINAL que indica que en el *beacon4* se termina la ruta, la lista de instrucciones a seguir en azul, separadas por el carácter @ atendiendo al cuadrante al que pertenecen. Debido a la imposibilidad de ir a la Facultad de Informática a causa del COVID-19 se ha supuesto que todos los cuadrantes miden cinco metros, aunque esta información no se corresponde con la realidad. En morado se representa el cuadrante en el que hay que girar, como vemos el “si” corresponde al cuadrante 20, que es en el que hay que hacer el giro. Por último, en gris se presenta la información adicional de cada cuadrante, donde un “no” indica que no hay información asociada a dicho cuadrante.

- *Generación de la ruta*: Esta funcionalidad es la más importante del servidor. El objetivo es obtener toda la información referente a la ruta desde el origen al destino seleccionado (ver Figura 4.2). Lo primero que hay que hacer es obtener los cuadrantes origen y destino, recordemos que lo que tenemos es el *beacon* más cercano al cliente y el destino como cadena de caracteres (String), pero se desconoce a qué cuadrantes corresponden. Para ello, se busca el cuadrante asociado al *beacon* y al destino recorriendo la lista de cuadrantes del edificio (*aCuadrantes*) y solicitando el valor asociado al destino (clave) en la tabla hash (*lectorDest*). Con esta asociación entre *beacons*/destinos y cuadrantes se resuelve el problema del posicionamiento.

Una vez se tienen los puntos que determinan la ruta se genera la lista de cuadrantes y de *beacons* correspondientes mediante la función *calculaRuta*. Esta está basada en el algoritmo de *Dijkstra* (que reemplaza a la búsqueda en anchura utilizada en el proyecto de Víctor Gutiérrez Rodríguez y Murga (2014)) que tiene como entrada la matriz de adyacencia de los cuadrantes (en la Sección 4.1.2 pueden verse algunos detalles). Una vez que tenemos la lista de cuadrantes por los que el usuario debe pasar hasta llegar al destino se generan las instrucciones necesarias para guiar al usuario. Este es el cometido principal de la función *generar*. Esta función tiene como entradas el cuadrante actual (en el que se encuentra el usuario), que se va simulando mediante un bucle, y el destino, y en función de los cuadrantes de la ruta, más concretamente de los inmediatos al cuadrante actual, construye la siguiente instrucción (ver detalles

¹<http://tomcat.apache.org/>

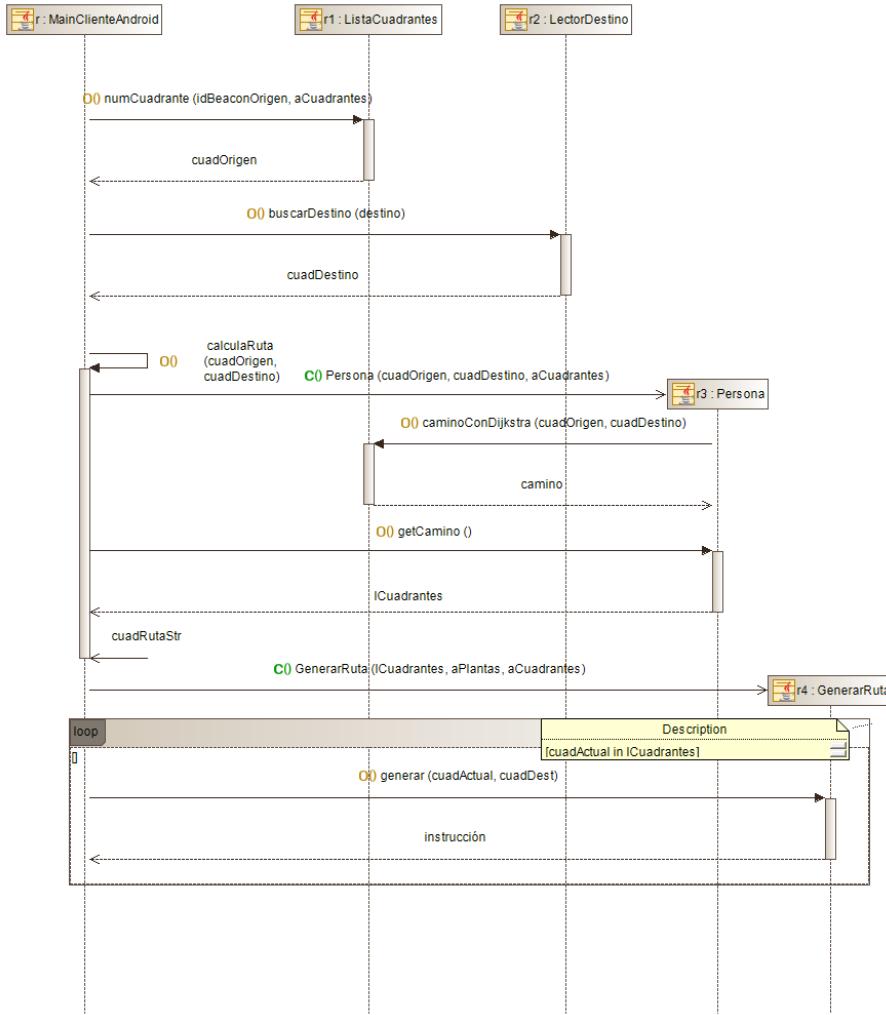


Figura 4.3: Diagrama de secuencia para la generación de la ruta.

en la Sección 4.1.3). En la Figura 4.3 puede verse el diagrama de secuencia. Cuando se ha llamado a *generar* con todos los cuadrantes que conforman la ruta, ya se tiene la lista de instrucciones completa y se contesta al cliente.

4.1.2. Cálculo de la ruta óptima

En esta sección veremos las modificaciones que se han hecho para lograr guiar al usuario por la ruta más conveniente. Como ya hemos mencionado anteriormente, el mapeo nos proporciona un grafo en el que los cuadrantes son los nodos y las conexiones entre ellos, las aristas. Para su representación hemos empleado una matriz de adyacencia y de esta manera el cálculo de la ruta más corta entre dos cuadrantes se reduce al algoritmo de *Dijkstra*.

Sin embargo, no debemos olvidar que nuestra aplicación tiene un usuario final muy concreto: personas con discapacidad visual. Es por ello que la ruta debe ser lo más sencilla posible, libre de obstáculos y otros elementos que puedan entorpecerlos, por lo que en ocasiones la ruta óptima no coincide con la más corta sino con la que esté mejor adaptada. Para representar esta adaptabilidad, a aquellas conexiones que presenten una mayor dificultad para las personas invidentes se les ha asignado un peso mayor en la matriz de adyacencia

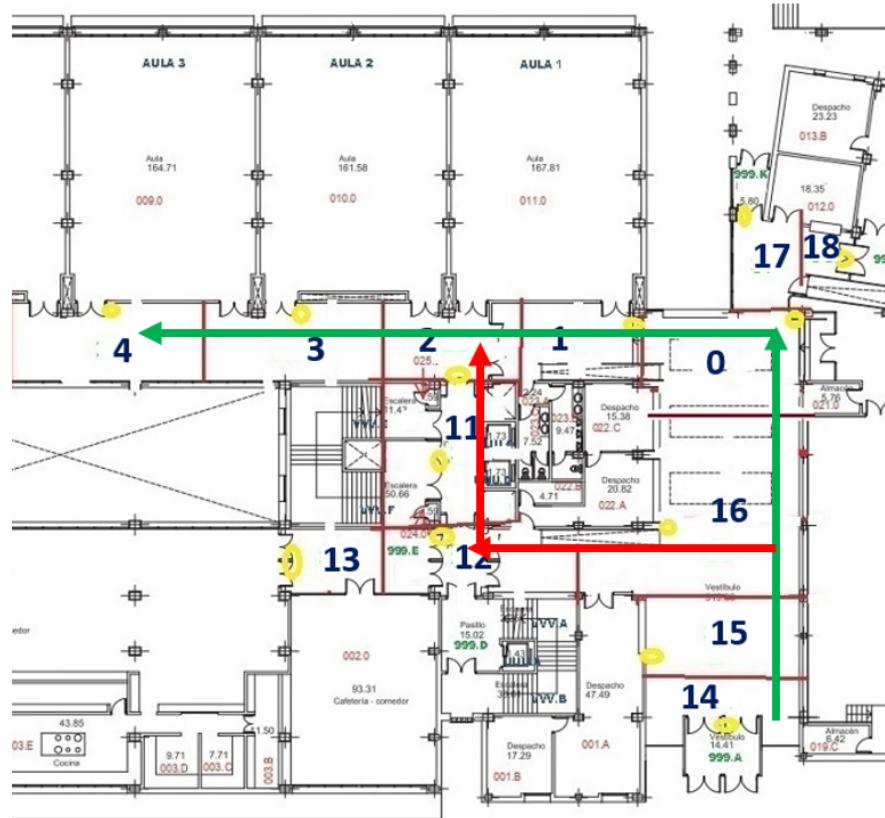


Figura 4.4: Ejemplo de ruta óptima entre dos puntos.

(ver Sección 3.4), a fin de buscar este equilibrio entre la ruta más corta y la más adecuada. En la Figura 4.4 aparece un ejemplo de esta situación: si quisiéramos ir desde la puerta principal (cuadrante 14) al aula 3 (cuadrante 4), el camino más corto implicaría pasar por delante de los ascensores (ruta roja). Sin embargo, la conexión entre los cuadrantes 11 y 12, y 2 y 11 puede resultar tediosa para una persona invidente. Las razones que hemos considerado son:

- El tramo compuesto por los cuadrantes 2, 11 y 12 es más estrecho que su camino paralelo por el *hall*.
- Presenta más giros e intersecciones de caminos.
- Normalmente acumula más gente ya que es zona de paso para coger los ascensores o subir/bajar por las escaleras o entrar/salir de la cafetería.

Todo esto es potencialmente problemático en ausencia de la vista y considerando además que nuestros usuarios es probable que vayan acompañados de un perro guía o un bastón. Por ello, hemos concluido ajustar los pesos en la matriz de adyacencia y la ruta generada pasa a ser la señalada en verde. De esta manera limitamos el paso por el pasillo de los ascensores a aquellas rutas en las que es estrictamente necesario, es decir, cuando se requiere un cambio de planta. Este tipo de consideraciones se han tenido en cuenta en las distintas zonas del edificio que presentan este tipo de características.

4.1.3. Generación de instrucciones

La función que contiene toda la lógica relativa a la generación de las instrucciones es *generar*. Esta función es una de las que más modificaciones ha sufrido con respecto a los trabajos predecesores, pues no solo la hemos adaptado para personas con discapacidad visual sino que también hemos incluido mucha complejidad procedente de los cambios de planta y demás casuística que no estaba previamente incluida. Algunas de las modificaciones son:

- Se ha añadido más precisión e información a las instrucciones. De esta manera, cuando el usuario llega al destino correspondiente la instrucción específica dónde se encuentra este, véase “Su destino está a la derecha (o a la izquierda o delante...).” dependiendo de por dónde haya llegado el usuario (en la Sección 4.1.3.1 se detalla la implementación de esta funcionalidad). Otro caso en el que se ha añadido mayor precisión es al puntualizar los metros que el usuario debe continuar en una dirección dada. En la Figura 4.2 podemos ver un ejemplo de ruta en el que se especifica el número de metros que el usuario debe seguir recto hasta ejecutar el siguiente giro o la siguiente acción, y como a medida que el usuario va avanzando la distancia va disminuyendo. Esto favorece que el usuario reconozca que va en la dirección correcta y pueda calcular mejor cuando ha de girar o ejecutar cualquier otra acción. A diferencia de los trabajos predecesores, nuestra app proporciona una instrucción nueva en cada cuadrante, es decir, no se salta ninguno como sucedía en el caso de las rectas (pasillos) en el proyecto precedente. Esto se ha hecho así ya que por un lado hemos considerado cuadrantes más grandes y por tanto la distancia recorrida de un cuadrante a otro es mayor, y porque creemos muy conveniente que el usuario reciba de manera continuada *feedback* por parte de la aplicación para cerciorarse de que va por el camino correcto y que la aplicación no ha dejado de funcionar. Finalmente, también se ha ajustado el orden de las instrucciones para proporcionarlas de manera correcta. De modo que si la instrucción siguiente es un giro se invierte el orden para indicar primero el giro y después los metros que debe continuar en la nueva dirección.
- Se han incluido cambios de planta, lo que supone una gran novedad respecto al proyecto citado Víctor Gutiérrez Rodríguez y Murga (2014). Ahora los cuadrantes de distintas plantas correspondientes a los ascensores están unidos² permitiendo así que la lista de cuadrantes de la ruta esté formada por cuadrantes de distinta planta. La función *generar* detecta cuando el siguiente cuadrante al que queremos ir no está en la misma planta que el de nuestra posición actual y genera la instrucción en consonancia “Los ascensores están a tu izquierda. Sube a la primera planta” o “El ascensor está delante. Sube a la primera planta”, por ejemplo. También se tiene en cuenta en las instrucciones el caso particular de las zonas de ascensores, pues hay dos zonas por planta pero no son simétricas por lo que las instrucciones son distintas. Por ello, para salir de la zona de los ascensores correspondientes al cuadrante 11 (los que se sitúan detrás de conserjería) debemos simplemente girar, mientras que para salir de los de la zona de la puerta trasera de la cafetería (cuadrante 29) debemos caminar escasos metros hacia adelante y después girar. A pesar de que en la Facultad de Informática contamos con ascensores y escaleras, se ha supuesto que la ruta se seguirá por medio de los ascensores ya que de esta manera es más fácil el cambio de planta y, además, estos se encuentran adaptados con botones en *braille*.

²El cuadrante 11 (correspondiente a los ascensores situados detrás de conserjería) está conectado con el 37 (ascensor correspondiente al anterior en la planta 1) y el 9 (ascensor más cercano a la puerta trasera de la cafetería) con el 29 (ascensor correspondiente al anterior en la planta 1).

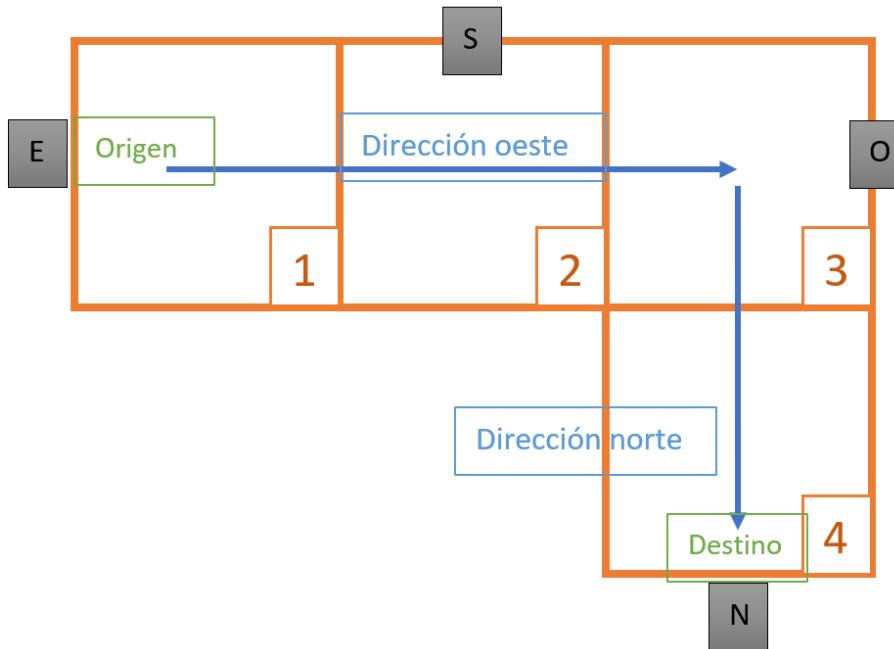


Figura 4.5: Ejemplo de giro en la ruta.

- Se ha añadido una funcionalidad que permite informar al usuario (si lo desea) sobre lo que se va encontrando a su paso por la ruta, véase los baños, la cafetería, el despacho de Delegación de Alumnos, unos escalones, etc. Esto es fundamental pues además de proporcionar seguridad al usuario en su primera visita a un edificio, le da la opción de tener una mejor idea global del espacio en el que se encuentra. Para ello lo que se hace es incluir en la ruta la información contenida en el cuadrante siguiente al que se encuentra el usuario, de esta manera conseguimos adelantarnos y avisar al usuario con antelación. Así mismo, la función *generar* enviará información sobre cuándo el usuario deberá hacer un giro para permitir avisar a este de manera especial (con una vibración) desde la aplicación del cliente.

4.1.3.1. Implementación de las instrucciones de giro

Como ya avanzábamos al comienzo de la sección anterior, la función que genera las instrucciones completas es *generar*. Sin embargo, esta se apoya en la función *indicaDirFinal*, cuya finalidad es indicar si se debe hacer un giro y, en ese caso, hacia qué lado. Esto resulta de gran utilidad tanto para saber hacia dónde debe dirigirse el usuario en caso de que tenga que hacer un cambio de dirección en su camino como para poder indicarle dónde se encuentran ciertos puntos de interés durante la ruta, (como los ascensores o el destino) si a su izquierda, a su derecha o delante.

La lógica de la función *indicaDirFinal* es bastante sencilla. Esta toma como parámetro de entrada dos direcciones, la primera de ellas indica la dirección en la que vamos y la segunda la dirección a la que queremos llegar. En la Figura 4.5 se ilustra el ejemplo de una ruta con un giro. En este caso, la dirección en la que vamos sería el oeste (en gris se muestra la posición de los puntos cardinales de referencia). Al llegar al cuadrante 3, la dirección que debemos tomar es norte, la cual no coincide con la dirección que llevábamos. Es entonces cuando la función *generar* sabe que el usuario debe hacer un giro y llama a la función

indicaDirFinal con los parámetros oeste y norte, en ese orden. La lógica de *indicaDirFinal* se basa en una serie de *if-elses* que determinan las direcciones de giro correspondientes. En este caso en el cuadrante 3 nos devolvería el giro a la derecha. Una vez que nos encontramos ya en el cuadrante 4, *generar* reconoce que es el cuadrante destino (pues este uno de sus parámetros de entrada) y, en función de dónde se sitúe el punto de interés de este cuadrante destino, indica al usuario dónde está con ayuda de la función *indicaDirFinal*. En este caso sería *su destino está delante*, pues no hay que realizar ningún cambio de dirección.

4.2. Cliente

En nuestro proyecto el cliente constituye la aplicación en sí misma. Esta la hemos bautizado como Blind Bit y ha sido desarrollada para Android. A través de ella el usuario solicita la ruta a un destino determinado, la aplicación conecta entonces con el servidor, que es quien la calcula, y se la reenvía al cliente. Finalmente, la aplicación se encarga de proporcionar, en el momento adecuado, las instrucciones necesarias para llegar al destino y utiliza sonidos y vibraciones para advertir de diferentes situaciones, como giros o aprobación de que seguimos en el camino correcto. A continuación presentamos tanto los detalles de diseño de la aplicación como los detalles técnicos de su implementación.

4.2.1. Diseño de la aplicación Blind Bit

A la hora de abordar el diseño de la aplicación hemos tenido muy presente el hecho de que nuestros potenciales usuarios finales son personas con discapacidad visual. Por ello hemos plateado interfaces sencillas y poco aglomeradas, en las que los botones sean lo más grandes posibles y estén bien organizados para que sea fácil de utilizar y memorizar. De esta manera, la mayor parte de los botones que hemos incluido tienen forma rectangular y ocupan todo el ancho de la pantalla, lo que favorece el uso de la app con ayuda del lector de pantalla. No solo hemos tenido en cuenta el lector de pantalla en la forma y el tamaño de los botones sino también a la hora de cambiar el nombre de cada pantalla especificando en cual se encuentra el usuario (por defecto aparecía el mismo nombre en todas) y en el hecho de evitar que este reproduzca todas las figuras que aparecen en la pantalla (modo por defecto) ya que sobrecarga al usuario proporcionándole información que en algunos casos carece de interés, como sucede con el logotipo de la app o ciertos cuadros de texto.

Por otro lado, hemos incluido algunas vibraciones y sonidos característicos con el fin de advertir al usuario de ciertas situaciones y acciones de manera no verbal. De esta manera, no saturamos al usuario con demasiada información extra pero le ayudamos en el uso de la app. Estos son: sonido de *acuerdo* cuando el usuario completa correctamente la instrucción dada, una vibración larga en el momento en el que pasa por una intersección de caminos y ha de girar, y dos vibraciones más cortas cuando ha llegado a su destino.

A continuación describimos las pantallas de nuestra aplicación, que pueden verse en la Figura 4.6, y los detalles de su diseño:

- *Pantalla principal*: En ella aparece el logo de la aplicación que ocupa el cuadro superior, y tres botones alargados que se sitúan en el cuadro inferior ocupando prácticamente todo el ancho de la pantalla. Los botones son, en orden descendente, *Iniciar Ruta*, *Ajustes* e *Instrucciones*. La finalidad de estos botones es muy intuitiva: el botón de *Iniciar Ruta* nos conduce a una pantalla de destinos en la que podremos seleccionar uno para después comenzar la ruta hasta él; el botón de *Ajustes* sirve para modificar algunos aspectos de la configuración como el volumen, el idioma de

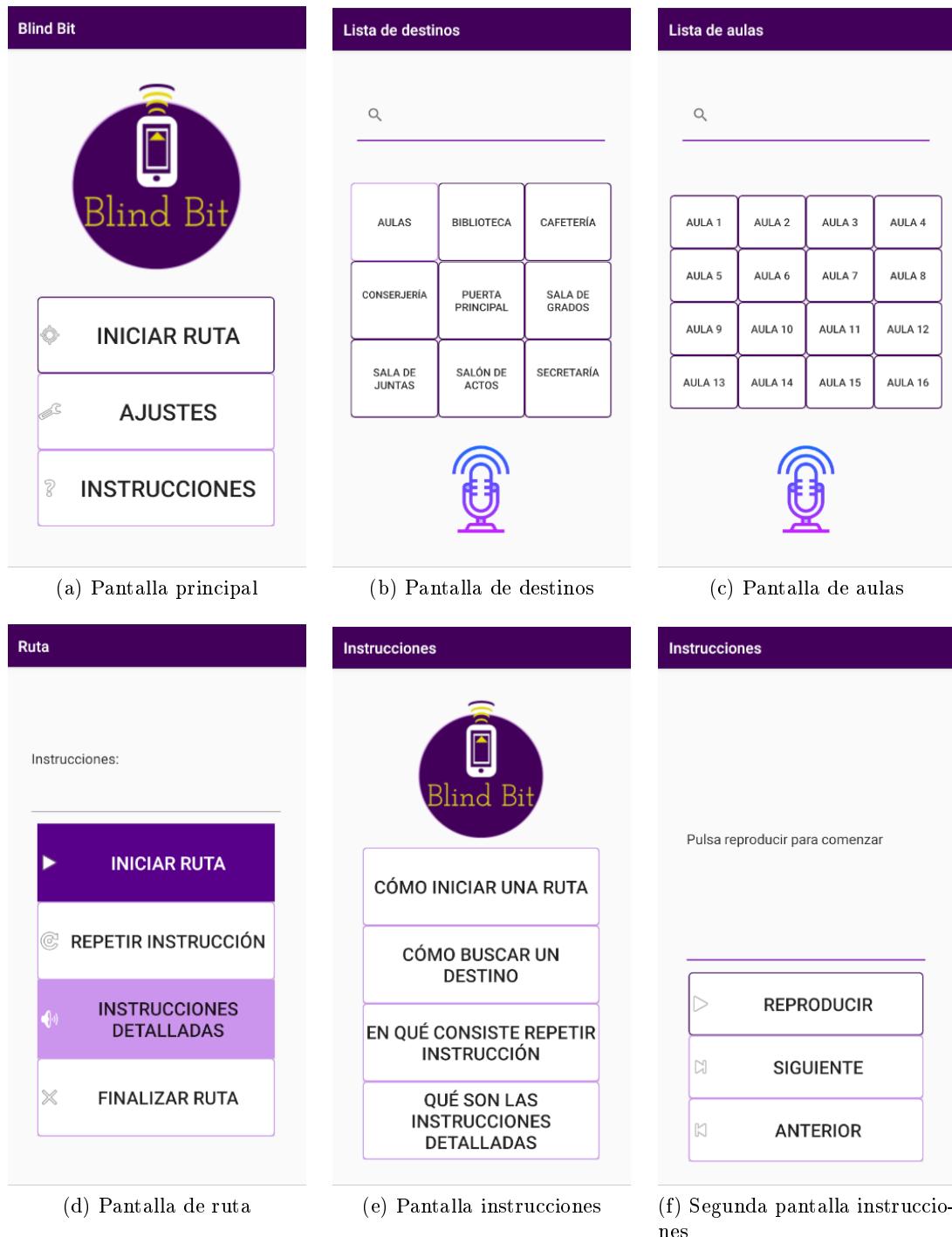


Figura 4.6: Interfaz de la aplicación Blind Bit

la aplicación, el tipo de voz que da las instrucciones, el modo de uso de la aplicación (que de más información en las instrucciones o que sean más escuetas), etc; y por último, el botón de *Instrucciones* conduce a una pantalla en la que se expone el modo de uso de la aplicación, con todas sus posibilidades.

- *Pantalla de destinos:* Esta pantalla tiene como finalidad que el usuario seleccione el destino de la Facultad de Informática hasta el que quiere dirigirse para, tras ello, comenzar con la ruta. Debido al carácter especial de nuestros usuarios, hemos empleado un diseño que aporte distintas alternativas a esta búsqueda del destino pues nuestro objetivo no es solo encontrar una vía que les resulte sencilla (como podría ser utilizar la app mediante voz) sino que también se adapte a distintas situaciones de la vida cotidiana (siguiendo con el mismo ejemplo, en determinadas circunstancias utilizar la app mediante voz puede ser molesto o causar vergüenza al usuario). Por ello, en el margen superior de la pantalla hemos situado una barra de escritura, que ocupa todo el ancho, en la que el usuario puede escribir directamente el nombre del destino al que desea ir. Si por el contrario prefiere utilizar la app mediante voz hemos incluido un micrófono en la parte central del margen inferior a través del cual el usuario puede indicar el destino en voz alta. El motivo por el cual hemos decidido colocar el micrófono en la parte inferior de la pantalla es que pese a que el lector de pantalla va barriendo de arriba a abajo y, por tanto, el micrófono queda el último, ocupa una posición cercana al dedo pulgar tanto de la mano derecha como de la izquierda (ya que se ha colocado en medio) y por ende, aunque al principio haya que esperar un poco más hasta que el lector lo encuentre, una vez que el usuario sepa donde está, el acceso será más sencillo y rápido. Además no hay ningún otro botón próximo, lo que facilita aún más el acceso. Finalmente, siguiendo el modelo de diseño de la aplicación *Lazarillo*, vista en la Sección 2.1 del Capítulo 2, hemos incluido una cuadrícula con 9 botones (3 filas de 3 botones) en la zona central, en la que aparecen todos los posibles destinos de la Facultad (*Aulas, Cafetería, Biblioteca, Salón de Actos, Sala de Juntas, Sala de Grados, Consejería, Puerta Principal, Secretaría*). Si se selecciona el botón *Aulas* aparece una pantalla con el mismo diseño, con la única variante de que en la cuadrícula (esta vez de 4 filas con 4 botones cada una) aparecen las distintas aulas que hay en la Facultad. Si por el contrario se selecciona cualquier otro botón de destino (un aula concreta, la cafetería, la sala de juntas, etc.) o se indica el destino con alguna de las otras dos alternativas (barra de escritura o micrófono) aparece la pantalla de ruta.
- *Pantalla de ruta:* Esta pantalla es la encargada de proporcionar las instrucciones de la ruta hasta el destino seleccionado. En ella aparece un cuadro de texto en la parte superior, en el que se van escribiendo las instrucciones a medida que se reproducen en voz alta. De nuevo hemos tomado la decisión de no usar exclusivamente la vía oral con el fin de crear una aplicación que se adapte a diversas situaciones y usuarios (personas invidentes, personas con discapacidad pero que mantienen algún resto visual e incluso personas videntes que busquen una guía por la Facultad), y sea así, lo más inclusiva posible. Además del cuadro de texto, aparecen 4 botones alargados que ocupan el resto de la pantalla. Estos son *Iniciar Ruta* que como su nombre indica sirve para comenzar una vez que se ha seleccionado el destino, *Repetir Instrucción* cuya finalidad es volver a dar la última instrucción en caso de que no se haya escuchado bien, *Instrucciones Detalladas* que sirve para activar y desactivar, según el estado previo, la funcionalidad de incluir más información durante la ruta (indicar qué hay alrededor). Cuando se modifica el estado de este botón el lector de pantalla avisa del nuevo *modo*.

instrucciones detalladas activado o *modo instrucciones detalladas desactivado*. Y por último, *Finalizar Ruta* que sirve para forzar la finalización de la ruta. Cuando se pulsa este botón o bien la flecha para ir hacia atrás aparece una ventana emergente que pide confirmación para finalizar. De esta manera nos aseguramos de que el usuario no ha pulsado sin querer.

Atendiendo de nuevo a esta diversidad de usuarios, hemos coloreado los botones *Iniciar Ruta* e *Instrucciones Detalladas* en morado y lila respectivamente, para que aquellos que tienen visibilidad reducida puedan distinguir los botones por sus colores. Por otro lado, el motivo por el cual hay un botón de *Iniciar Ruta* en lugar de empezar directamente con la reproducción de instrucciones una vez que el destino ha sido seleccionado, es que tras hacer varias pruebas con el lector de pantalla advertimos que las instrucciones y el *talkback* se solapaban y se volvían ininteligibles.

- *Pantalla de Instrucciones de uso:* En esta pantalla aparece el logo de la aplicación centrado en el margen superior y a continuación cuatro botones formando una columna en los que se plantean algunas de las dudas más frecuentes sobre el uso de Blind Bit: *Cómo iniciar una ruta*, *Cómo buscar un destino*, *En qué consiste Repetir Instrucción* y *Qué son las Instrucciones Detalladas*. Al pulsar cualquiera de estos botones aparece una pantalla que nos recuerda al diseño de la pantalla de ruta, en ella hay un gran cuadro de texto que ocupa la mitad superior de la pantalla y tres botones colocados en forma de columna que ocupan la otra mitad. Estos botones son, en orden descendente, *Reproducir*, *Siguiente* y *Anterior*. De esta manera, si pulsas el primer botón (*Reproducir*) aparece la instrucción correspondiente al modo de uso seleccionado, escrita en el cuadro de texto y reproducida en voz alta. Para volver a escucharla basta con pulsar de nuevo este mismo botón. Por otro lado, si se quiere saber la instrucción siguiente o anterior siguiendo el orden indicado en la pantalla de Instrucciones de uso, basta con pulsar el botón correspondiente (*Siguiente* o *Anterior*, respectivamente). Con respecto a las decisiones tomadas en el diseño de esta última pantalla, hemos colocado los botones en este orden ya que hemos considerado que *Reproducir* será el más utilizado y por ello parece conveniente que esté arriba. Siguiendo con esta linea de pensamiento, parece que lo más lógico a continuación, es querer ir a la instrucción siguiente por lo que hemos colocado justo debajo dicho botón (*Siguiente*) y por último, el botón *Anterior* en caso de que querer volver a una instrucción previa. Por otro lado, al igual que en la pantalla de ruta las instrucciones no empiezan hasta que no se pulsa el botón *Reproducir*. Esto se debe al mismo motivo que entonces, si se usa la app con ayuda del lector de pantalla y las instrucciones comienzan directamente, se producen solapamientos en las reproducciones y se vuelven ininteligibles.

4.2.2. Funcionamiento del cliente

En esta sección veremos el funcionamiento de la aplicación desde la interacción con el usuario hasta la conexión con el servidor y gestión de la ruta.

- *Interacción con el usuario:* Como ya hemos tratado en la Sección 4.2.1 la aplicación está diseñada para que el usuario pueda lograr el objetivo con la mínima cantidad de pasos. Tanto es así que la interacción con el usuario queda reducida al mínimo. Para iniciar una ruta tan solo se pide al usuario que introduzca el destino al que quiere ir (una vez ha pulsado *Iniciar ruta* en la pantalla *principal*). Esto se hace a través de las pantallas *Lista de destinos* y *Lista de aulas*, que se pueden ver en la

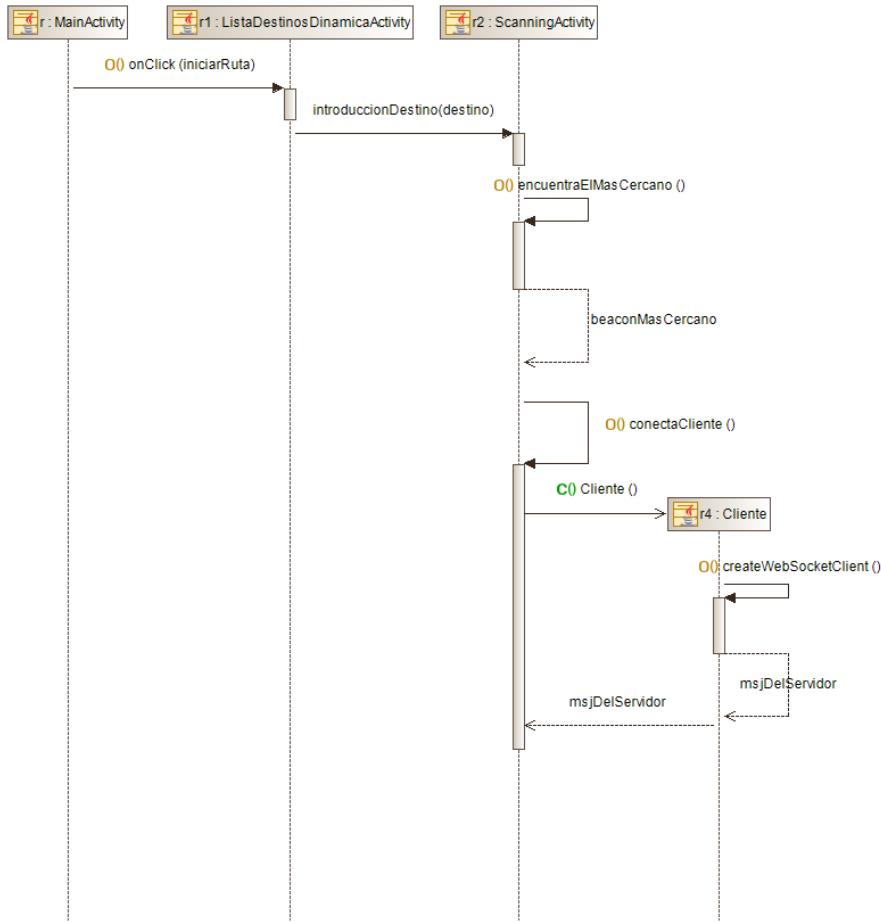


Figura 4.7: Diagrama de secuencia que comprende la interacción con el usuario, la resolución del posicionamiento y la conexión con el servidor en el cliente.

Figura 4.6. Cuando el usuario selecciona el destino, la aplicación abre la pantalla *Ruta*, resuelve el problema de posicionamiento y espera a que el usuario pulse sobre *Iniciar ruta*. En ese momento, la aplicación conecta con el servidor y comienza la ruta con instrucciones guiadas por voz.

- *Posicionamiento*: En la Sección 3.1.3 concluimos que el problema del posicionamiento se resolvería asignando al usuario el cuadrante correspondiente a su *beacon* más cercano. La lógica que lleva a cabo este proceso está en la clase *ScanningActivity*. Para ello comienza el escaneo de los *beacons* con la función *startScanning*. Una vez que se tiene la lista de *beacons* que están en el rango de detección del dispositivo móvil del usuario, se toma el más cercano en función de la distancia estimada por la función *getDistance* de la SDK de *Kontakt* y, cuando el usuario pulsa sobre botón *Iniciar ruta*, se conecta con el servidor para obtener los detalles de la ruta solicitada por el usuario.
- *Conexión con el servidor*: El código correspondiente a esta funcionalidad se recoge en la clase *Cliente*. Cuando se llama a esta clase se pasan como parámetros el *beacon* más cercano y el destino al que quiere ir el usuario. Con esta información se genera un mensaje del tipo *IDdelBeaconOrigen|destino* que se manda al servidor por medio de un *webSocket*. Cuando el servidor genera la ruta manda al cliente un mensaje con

toda la información referente a la ruta (ver Sección 4.1.1). Este mensaje se recibe y desglosa en la lista de *beacons* de la ruta, las instrucciones, la información sobre los giros y la información adicional de los cuadrantes de la ruta. Cuando tiene toda la información avisa a la aplicación (que ha quedado esperando la llegada de este mensaje) y la ejecución continua de nuevo en *ScanningActivity*, donde se guarda la información de la ruta en vectores que se irán recorriendo según avance el usuario. Si no se consigue conectar con el servidor la aplicación genera un mensaje especial que se gestiona en *ScanningActivity* para que la aplicación no quede en espera infinita y acabe bloqueando el dispositivo. En la Figura 4.7 puede verse el diagrama de secuencia que describe las funcionalidades detalladas hasta este momento (interacción con el usuario, posicionamiento y conexión con el servidor).

- *Seguimiento de la ruta*: Una vez que la conexión con el servidor ha tenido éxito y tenemos toda la información de la ruta almacenada en los vectores correspondientes se comienza guiar al usuario. Para ello se inicia el índice *indiceRuta* a cero y se da al usuario la primera instrucción. Este índice indica en qué punto de la ruta estamos, cada vez que cambiamos de cuadrante se va incrementando en uno, siempre que estemos siguiendo la ruta. Una vez que el usuario ha recibido la primera instrucción, se va llamando a la función *onEddystonesUpdated* cada dos segundos (parámetro configurable) a fin de saber en qué momento el usuario tiene como *beacon* más cercano el siguiente *beacon* de la ruta. Imaginemos, por ejemplo, que los dos primeros cuadrantes de la ruta fueran el 0 y el 1. Cuando se inicia la ruta y el usuario recibe la primera instrucción, la función *onEddystonesUpdated* se dará cuenta de en qué momento el usuario tiene el *beacon1* como más cercano y le dará la siguiente instrucción. Esto mismo ocurre con el resto de *beacons* de la ruta, hasta llegar al final. Si la funcionalidad *Instrucciones detalladas* está activada, se va proporcionando al usuario la información adicional correspondiente a su posición inmediatamente después de la instrucción que debe ejecutar.

Durante la ruta el usuario percibe no solo información oral sobre las instrucciones o la información adicional de los cuadrantes (si tiene la funcionalidad *instrucciones detalladas* activada), sino también percibe en el dispositivo sonidos y vibraciones. En el caso de que el usuario complete una instrucción, es decir, llegue al siguiente *beacon* de la ruta, la aplicación emite un sonido *check* para hacerle saber que ha completado esa parte. Además, si la instrucción siguiente es un giro, el dispositivo vibra, durante un segundo, si es un giro a la izquierda, y dos vibraciones de medio segundo, si es un giro a la derecha, para notificar el cambio de dirección. Así mismo, cuando el usuario ha llegado al final de la ruta, el dispositivo emite tres vibraciones cortas avisando de que se ha llegado al destino. Esto provoca en el usuario una sensación de control y seguridad, pues puede comprobar en tiempo real que va en el camino correcto y permite que el usuario tenga menos probabilidades de perderse, pues las notificaciones del cambio de sentido están reforzadas.

Se espera que el usuario pueda llegar al destino con la guía que proporciona la aplicación. Sin embargo, también se ha contemplado el hecho de que el usuario se pierda y salga de la ruta. Este caso se detecta cuando se pasa por la función *onEddystonesUpdated* diez veces (ver *numPasosPerdidos* en la Figura 4.8)³ pero en ninguna de ellas obtenemos que el *beacon* más cercano es el siguiente *beacon* (o un *beacon* más

³Este es un parámetro configurable. Debido a que no se han podido realizar pruebas físicas en la Facultad por la situación sanitaria debida al COVID-19 ajustarlo a lo que podría haber sido una situación real no ha sido posible.

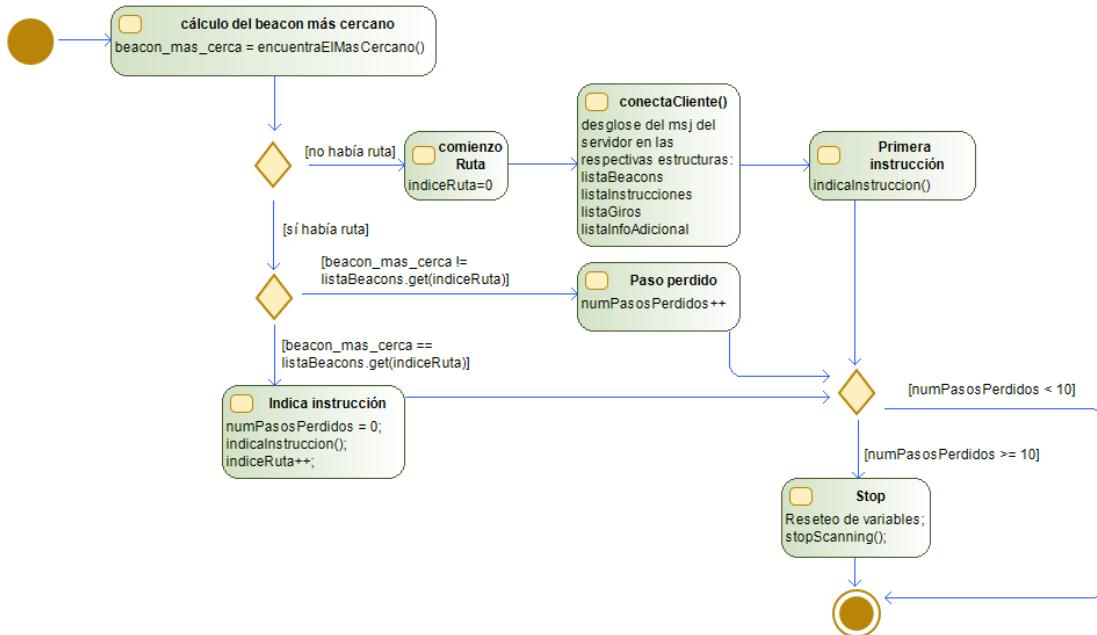


Figura 4.8: Diagrama de actividad simplificado para la función `onEddystonesUpdated`.

avanzado⁴⁾ de la ruta. Si esto ocurre, se considera que el usuario se ha salido de la misma. En este caso, cesa el escaneo de los *beacons* y se notifica al usuario que no se encuentra en el camino correcto y que tras pulsar *Iniciar ruta* de nuevo, la aplicación lo redirigirá al destino al que quería ir. En términos de implementación, se vuelve a resolver el problema del posicionamiento, conectar con el servidor e iniciar de nuevo la ruta correspondiente.

4.3. Adaptación para la reutilización de la aplicación sobre otro edificio

A lo largo de este capítulo se ha visto la implementación orientada a la Facultad de Informática de la UCM. Sin embargo, no se ha podido realizar un estudio exhaustivo del funcionamiento de la aplicación en esta ubicación debido al estado sanitario de emergencia declarado en febrero de este año 2020 y, su posterior prolongación, hasta final de curso. Así las cosas, se ha tratado de hacer la aplicación lo más genérica posible, de tal manera que, si se quisiera utilizar la tecnología *Bluetooth* para mapear otro espacio, nuestro trabajo pueda ser reutilizado haciendo una cantidad mínima de cambios. A continuación veremos los aspectos que deberían ser modificados.

4.3.1. Cambios en el servidor

El código del servidor se basa, principalmente, en la información del edificio que proporcionan los archivos xml que se vieron en la Sección 3.4 y el archivo `destinos.json` que se trató en la Sección 4.1.1. Todos estos archivos deberían ser sustituidos por los correspon-

⁴Podría ocurrir que, en un pasillo, por ejemplo, el usuario haya pasado por el siguiente *beacon* pero la aplicación no lo haya detectado. En ese caso el usuario habría avanzado un cuadrante sin notificar a la aplicación, pero como no se ha salido de la ruta se continúa sin notificar al usuario.

dientes al nuevo edificio que se desea mapear. Prestando especial atención a los pesos que se colocan en las distintas conexiones de los cuadrantes, pues son claves para el algoritmo de generación de la ruta (ver Sección 4.1.2). Así mismo, se debe establecer correctamente el lado del cuadrante en el que se encuentra el punto de interés del mismo (norte, sur, este u oeste), pues de ello dependen las instrucciones que indican al usuario dónde se encuentra el destino o las instrucciones que se dan al usuario tras el cambio de planta, en la zona de los ascensores. De esta manera, si se quisiera utilizar el código del servidor para otro edificio con los mismos requisitos que la Facultad bastaría con sustituir los archivos xml correspondientes al edificio, permitiendo así que no se requieran conocimientos específicos de programación para adaptar el código y facilitando su reutilización.

A pesar de que el código está pensado e implementado para dar servicio a un edificio de características similares a la Facultad de Informática de la UCM también se ha pensado en posibles variantes que podrían surgir a la hora de adaptar el código a otro espacio. A continuación se plantean algunos casos y la solución propuesta:

- En el caso de la Facultad de Informática hemos establecido que cada cuadrante tiene un único punto de interés. Por ejemplo, los cuadrantes de los pasillos tienen un aula como punto de interés, pero podría ocurrir que en otra facultad hubiera aulas a ambos lados del pasillo y fuera, por tanto, necesario que un cuadrante tuviera más de un punto de interés. Esto se resolvería de manera sencilla adaptando la estructura del cuadrante en los archivos xml: incluyendo una etiqueta que estableciera la ubicación de cada punto de interés y adecuando en el código la lectura del mismo. En el archivo *destinos.json* se incluiría una nueva entrada para el destino, de tal manera que tendríamos dos entradas con el mismo cuadrante (esto no supone un problema pues la clave de la tabla hash donde se almacena la información es el nombre del destino y no el cuadrante). Una vez modificados estos archivos bastaría con indicar a la función *genera* cuál es la ubicación del destino dentro de su cuadrante para que indique al usuario la posición del destino correctamente al finalizar la ruta.
- Otra situación que nos podemos encontrar es aquella en la que la estructura del edificio y los puntos que se quieran mapear se encuentren a una distancia próxima, obligando a que el tamaño de los cuadrantes se vea reducido. En este caso, dar instrucciones al usuario cada cuadrante puede resultar molesto, puesto que se darían instrucciones con demasiada frecuencia. Es por ello que se incluye una variable contador en la función *generar* que permite establecer el número de cuadrantes que queremos “saltar” antes de dar una nueva instrucción⁵, siempre que la dirección del usuario se mantenga estable. Es decir, no haya que hacer un giro, pues en ese caso debemos advertir al usuario.

4.3.2. Cambios en el cliente

El cliente es la parte menos dependiente del edificio, puesto que en ningún momento contiene información del mismo, a excepción de la lista de destinos que se incluye tanto en la interfaz como en el archivo *listasStringsApp.xml*. En este archivo se encuentran los destinos con el mismo nombre que el servidor los representa en el archivo *destinos.json*, lo que permite que antes de conectar con el servidor podamos comprobar que el destino es correcto. También habría que modificar esta lista de destinos en las instrucciones de uso

⁵En el propio código se han incluido en comentarios los cambios necesarios para la implementación de esta funcionalidad.

de la aplicación para que el usuario pueda acceder a ella sin necesidad de tener que simular el inicio de una ruta.

Capítulo 5

Evaluación

En este capítulo se describe el proceso de evaluación de la aplicación que se ha llevado a cabo. Como ya se puso de manifiesto en el Plan de trabajo (ver Sección 1.3), la idea inicial era la de realizar una evaluación con usuarios finales y, preferiblemente, en la Facultad de Informática de la UCM, pues ese espacio es nuestro caso de estudio inicial. Debido a la crisis sanitaria y el estado de emergencia declarado en marzo de 2020 a causa del COVID-19, no ha sido posible la ejecución dicha evaluación. Sin embargo, se ha adaptado, en la medida de lo posible, a esta situación. En las secciones que siguen se detalla cómo se ha llevado a cabo la adaptación tanto del plan de evaluación como el despliegue de la aplicación en otro edificio, destacando así su generalidad.

En la Sección 5.1 se describen los pasos a seguir, tanto en el servidor (Sección 5.1.1) como en el cliente (Sección 5.1.2) para poder adaptar la aplicación a otro espacio. Por su parte, la Sección 5.2 detalla los objetivos de la evaluación. Las pruebas que se llevaron a cabo a fin de valorar el cumplimiento de estos objetivos se expone en la Sección 5.3 y las conclusiones finales de la evaluación pueden verse en la última sección (Sección 5.4).

5.1. Adaptación de la aplicación a un nuevo edificio

En esta sección detallaremos los cambios que se deben realizar para poder utilizar la aplicación Blind Bit en un edificio nuevo. Tanto el código del servidor como el del cliente se han implementado de tal manera que la información relativa al edificio sobre el que se despliega quede reducida a la información contenida en archivos adicionales que no forman parte del código. Sin embargo, son algunas las consideraciones que hay que tener en cuenta antes de comenzar con el mapeo de un nuevo espacio.

5.1.1. Cambios en el servidor

Como se puso de manifiesto en la Sección 4.1, la información sobre la estructura del edificio que emplea el servidor para el cálculo de la ruta está contenida en los archivos xml y json correspondientes (ver Sección 3.2). Gracias a los archivos xml, el servidor conoce los cuadrantes que hay en una planta y, por tanto, sabe identificar cuándo hay que hacer un cambio de planta. Además, conoce la dirección en la que se mueve el usuario ya que sabe la dirección de conexión de los cuadrantes. Así mismo, permite indicar al usuario hacia qué dirección se encuentra su destino en función del camino que haya seguido para llegar a él. De esta manera, el código que genera la ruta es totalmente independiente del edificio. Tan

solo se espera que el cambio de planta se haga por medio de un ascensor, lo que parece de esperar cuando se mapean lugares como una facultad o un museo. Además, los ascensores suelen estar adaptados con escritura en braille. A continuación se detallan las claves para realizar el mapeo de un nuevo edificio.

5.1.1.1. Mapeo de un nuevo edificio

TE LA DEJO A TI QUE ERES MÁS EXPERTA

5.1.2. Cambios en el cliente

En las Secciones 4.2.1 y 4.2.2, revisamos el diseño de la interfaz de la aplicación Blind Bit y su funcionamiento, respectivamente. En cuanto a la interfaz se refiere, es sencillo darse cuenta de que la parte dependiente del edificio corresponde a la pantalla de destinos. Sin embargo, esta está implementada de manera dinámica. Es decir, los nombres de los botones, así como la lista de destinos con la etiqueta correspondiente a los destinos tal y como aparecen en el archivo *destinos.json* del servidor¹ se guardan en un archivo xml. El archivo donde se guardan estos y otros strings correspondientes a la aplicación, tales como el texto de las instrucciones o la uri del servidor es *listasStringsApp.xml*. Las estructuras referentes a la lista de destinos son *destinos_array* y *destinosdinamicos_array*. Mientras que en la primera basta con introducir cada destino en un *<item>*, en la segunda hay que indicar si ese destino tiene un segundo nivel. Por ejemplo, en el caso de la Facultad de Informática tenemos un botón aulas, que no corresponde con un destino sino con el acceso a un segundo nivel donde se muestran las aulas destino disponibles. La estructura que debe seguirse es la siguiente:

```
<string-array name="destinos_array">
    <item>aula 1</item>
    <item>aula 2</item>
    <item>secretaria</item>
</string-array>

<string-array name="destinosdinamicos_array">
    <item>Aulas|aula 1,aula 2,aula 3</item>
    <item>Biblioteca|no</item>
    <item>Secretaria|no</item>
</string-array>
```

Donde un “no” tras la barra indica que no hay un segundo nivel y cualquier otra cadena de strings se entiende como los destinos correspondientes a ese nivel, separados por comas. Este constituye el único cambio necesario para adaptar el cliente a un nuevo espacio.

5.2. Objetivos de la evaluación

Debido a la imposibilidad de realizar una evaluación con usuarios. Nos vimos obligadas a reestructurar el *modus operandi* de la evaluación de la aplicación. Concretamente, los objetivos cambiaron, centrándose en el funcionamiento de la misma y dando menos peso a la usabilidad, en la que los usuarios finales tienen un papel decisivo. De esta manera, se decidió establecer cuatro objetivos claros que presentamos a continuación:

¹Los destinos en la interfaz pueden aparecer con tildes, mayúsculas y otros caracteres especiales que puede no ser posible añadir en el archivo *destinos.json*. Es por ello que se guardan los destinos en dos estructuras, una para la interfaz y otra para la conexión con el servidor.

1. *Resolución del problema del posicionamiento:* Una de las funcionalidades principales de la aplicación es, sin duda, la correcta ubicación del usuario. Para ello es necesario que el *beacon* más cercano al usuario sea detectado como el más cercano por la aplicación (ver Sección 4.2.2). De esta tarea depende no solo el inicio de la ruta sino el seguimiento de toda ella, pues en todo momento debemos conocer el cuadrante donde se encuentra el usuario para que la aplicación pueda responder en consecuencia. Una mala ubicación del usuario podría provocar que el usuario se pierda debido a indicaciones que no corresponden con su posición o, en casos más graves, el tropiezo o golpeo del usuario con un obstáculo del que no ha sido advertido.

Cabe destacar que en el posicionamiento influye no solo la correcta implementación del código, sino también la ubicación de los *beacons*, que debe adaptarse a las necesidades específicas de cada edificio.

2. *Generación de instrucciones correctas:* Teniendo en cuenta la ubicación del usuario y el camino que ha seguido, es primordial que la aplicación sea capaz de generar instrucciones correctas. Tanto los giros como la señalización de puntos de interés como los ascensores o el destino debe corresponderse con la ubicación real de estos, tal y como se establece en los archivos xml (ver Sección 3.4).
3. *Precisión de las instrucciones:* Además de que las instrucciones sean correctas, hay que evaluar que el usuario las recibe en el momento adecuado. Esto está claramente relacionado con el posicionamiento, pues el momento de indicación de la ruta se basa en cuándo el usuario llega a un determinado cuadrante. Sin embargo, se debe prestar especial atención a las posibles variaciones que pueden darse (una instrucción puede darse con cierta antelación o, por el contrario, una vez pasado el punto óptimo) y evaluar si esa flexibilidad es asumible para el usuario.
4. *Ejecución correcta en caso de usuario perdido:* Este es un punto importante, pues no debemos asumir que el usuario va a seguir siempre la ruta, puede ocurrir que por diversos motivos (una distracción, un obstáculo o el propio fallo de la aplicación) el usuario se desvíe de la ruta. En ese caso no solo se debe identificar el problema sino también saber re conducir al usuario al destino deseado.

5.3. Realización y resultados de la evaluación

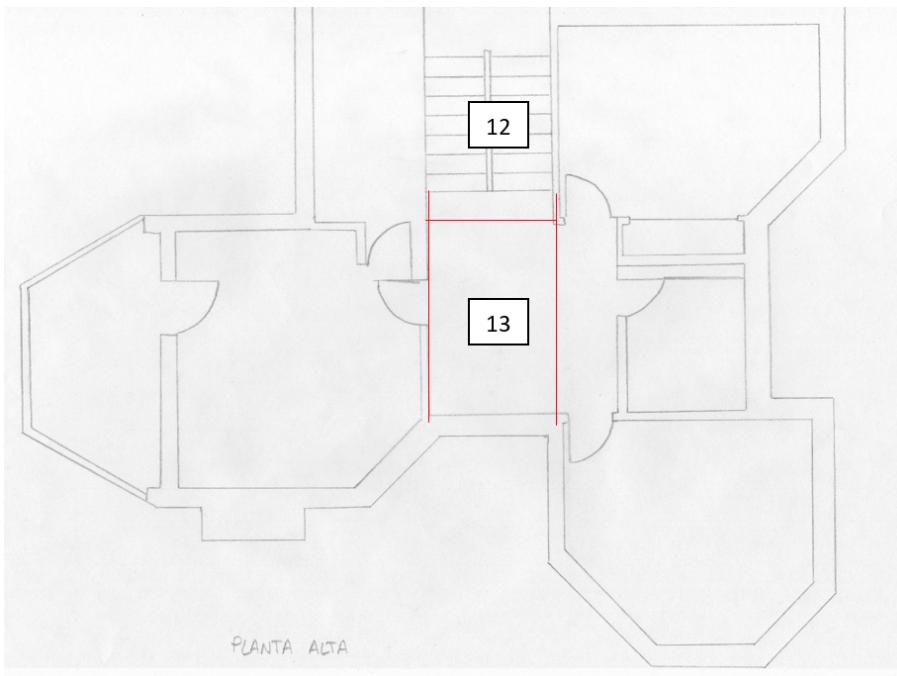
La realización de la evaluación no pudo darse en el edificio objetivo de nuestro estudio (la Facultad de Informática de la UCM) debido a la imposibilidad de acceder a él. Sin embargo, conseguimos sobreponernos a este contratiempo y poner de manifiesto la flexibilidad de la aplicación mapeando otro edificio y realizando diversas pruebas sobre este. Este edificio no pudo ser otro que una vivienda. Cabe destacar que este no es el escenario ideal sobre el que se desplegaría una aplicación como Blind Bit, pues el espacio queda considerablemente reducido en comparación con el de una facultad o museo, por ejemplo. A pesar de ello, permite probar el comportamiento de la aplicación en situaciones donde cierta aglomeración de *beacons* es necesaria y poner a prueba el mapeo de un edificio con características distintas al ya planteado en la Sección 3.2.

5.3.1. Mapeo del nuevo espacio

Como ya se avanzaba en la introducción, el edificio sobre el que se basa la evaluación es una vivienda. En la Figura 5.1 vemos el resultado del mapeo, donde el número de cuadrante



(a) Mapeo de la planta baja de la vivienda



(b) Mapeo de la planta alta de la vivienda

Figura 5.1: Mapeo de la vivienda

está recuadrado en negro, la posición de los beacons en amarillo y los cuadrantes vienen delimitados por las líneas rojas y las paredes de la vivienda. Como muestra la Figura 5.1b, el mapeo de la planta alta se ha reducido a dos cuadrantes, pues son suficientes para *testear* las rutas que incluyen un cambio de planta ya que la estructura es análoga a la de la planta baja.

A continuación se describen las pruebas realizadas y se analizan los resultados obtenidos.

5.3.2. Seguimiento de la ruta

En esta sección se detallan las pruebas realizadas asumiendo que el usuario no va a salir de la ruta. Sin embargo, algunas de ellas están diseñadas para reproducir situaciones extremas como la pérdida de un *beacon* o rutas potencialmente complicadas.

5.3.2.1. Ruta del cuadrante 0 al cuadrante 10

La primera ruta de la evaluación consiste en realizar la ruta desde el cuadrante 0 hasta el cuadrante 10 sin salir de la ruta. En esta se debe prestar especialmente atención a las instrucciones de giro, pues son constantes durante toda la ruta. Además, no debemos olvidar que la aplicación emite distintas vibraciones en función de la dirección de giro. Estas vibraciones deben quedar lo suficientemente claras para el usuario.

Este recorrido se realizó varias veces a fin de encontrar irregularidades en la ruta. A continuación se detallan las conclusiones obtenidas:

- Las instrucciones generadas por la aplicación son correctas y, en la mayor parte de los casos, se indican al usuario en el momento adecuado.
- Las vibraciones asociadas a los giros y a la llegada del destino son lo suficientemente distintas para que el usuario pueda distinguirlas.
- En dos ocasiones las instrucciones correspondientes a los cuadrantes 1 y 2 se dieron demasiado pronto. Esto puede deberse a que el *beacon* del cuadrante 1 está situado en el exterior y que, una vez que se avanza hacia la puerta que separa los cuadrantes 1 y 2, la distancia que hay entre ellos es reducida.
- En una de las ocasiones el *beacon* asociado al cuadrante 4 no fue detectado por la aplicación, provocando la pérdida de esa instrucción (en las pruebas siguientes se detalla el comportamiento de la aplicación en este caso. Ver Secciones 5.3.2.3 y 5.3.3).
- Una vez se ha llegado al destino, la posición del mismo se indica correctamente en función de la dirección desde la que proviene del usuario.

5.3.2.2. Ruta del cuadrante 1 al cuadrante 10

Este caso es prácticamente idéntico al anterior (Sección 5.3.2.1), con la única diferencia que se pone a prueba el posicionamiento inicial del usuario. Tras comprobar que las instrucciones en el cuadrante 1 se daban antes de lo previsto en algunas ocasiones, se decidió empezar la ruta en este cuadrante. De esta manera la aplicación debía detectar el *beacon1* como *beacon* más cercano. Se hicieron cinco pruebas y se pudo comprobar que en dos ocasiones este posicionamiento no se realizó de manera correcta. En uno de ellos el *beacon* más cercano detectado por la aplicación fue el 4 (lo que puede ser provocado porque la separación entre el 4 y el 1 es una ventana) y en otro el 2 (cuya posible explicación está en la Sección 5.3.2.1).

5.3.2.3. Ruta del cuadrante 0 al cuadrante 10, eliminando el *beacon* del cuadrante 4

En este caso se ha vuelto a repetir la ruta del cuadrante 0 al 10 con la particularidad de que el *beacon4*² fue eliminado de la ruta. Sin embargo, no provocamos la situación de

²Asumiremos que *beaconX* es el *beacon* asociado al cuadrante X.

que el usuario se saliera de la ruta (como sería lógico en esta situación, pues la instrucción de giro del cuadrante 4 se pierde), continuamos por el cuadrante 9. De esta manera la aplicación no notifica que el usuario se ha perdido, pues sigue en la ruta. A pesar de que la instrucción indicada en el cuadrante 9 fue la correcta (giro a la izquierda), la aplicación tardó demasiado. Esto se debe a que el umbral de la variable *numPasosPerdidos*, que identifica el momento en el que el usuario puede haberse perdido (ver Sección 4.2.2), es demasiado elevado para este edificio, donde las distancias entre *beacons* son pequeñas.

5.3.2.4. Ruta del cuadrante 1 al cuadrante 5, eliminando el *beacon* del cuadrante 2

Este caso es interesante, pues, a pesar de que se ha eliminado uno de los *beacons* correspondientes a una intersección, la información de giro en el cuadrante 2 no se pierde. Esto se debe a que, como la aplicación es capaz de avisar de los cambios de dirección con antelación, en el cuadrante 1 ya se indica al usuario que, tras avanzar unos metros, debe girar a la izquierda. Bien es cierto que se pierde precisión, pues el usuario no percibe la instrucción de giro en el momento en el que debe realizarse, ni la confirmación sonora y vibración asociada a esta. Sin embargo, el hecho de que se den instrucciones por adelantado favorece que el usuario permanezca en la ruta aún cuando alguna instrucción se pierda.

Particularmente para esta ruta, el giro del cuadrante 2 es el único que hay que realizar y, debido a esto, es más fácil que el usuario no se pierda. Si, por el contrario, el destino hubiera sido el cuadrante 10, es probable que el usuario no hubiera llegado a recibir la instrucción de giro a la derecha del cuadrante 4 tampoco. Esto se debe a que la aplicación necesita un tiempo (llegar a un número de pasos perdidos) para decidir si el usuario se ha perdido. De esta manera, cuando el usuario llega al cuadrante 4 la aplicación sigue a la espera del cuadrante 2 (el umbral de *numPasosPerdidos* es demasiado alto). Esto implica que el usuario se salga de la ruta. Caso que abordamos en la Sección 5.3.3.

5.3.2.5. Ruta del cuadrante 1 al cuadrante 13

En este caso, lo que se quería evaluar eran las instrucciones referentes al cambio de planta. Como ya se comentó en la Sección 5.1.1, el código asume que los cambios de planta se realizan por medio de ascensores. Esta información se ha ignorado en esta ocasión, pues solo se disponía de escaleras.

De la ruta desde el cuadrante 1 al 13 destacamos un punto importante, que es la anticipación de los ascensores en el cuadrante 2 gracias a la información adicional generada. Esto avisa al usuario del cambio de planta. Una vez en el cuadrante 11, la aplicación indica al usuario a qué planta se debe dirigir y en qué dirección se encuentran los ascensores (en este caso, delante del usuario). Una vez en la planta superior, la instrucción del cuadrante 12 continua teniendo en cuenta la nueva orientación del usuario (continúa recto).

5.3.2.6. Ruta del cuadrante 13 al cuadrante 8

La finalidad de esta ruta es, principalmente, comprobar el comportamiento de la aplicación en el *hall* de la planta baja, así como el cambio de planta inverso.

El problema que surge aquí es que los giros que hay que realizar en la planta baja son bastante tediosos. Por un lado, la ubicación del *beacon2* favorece que la instrucción de giro hacia el *beacon3* tarde demasiado en llegar. Una vez que se proporciona esta instrucción, el cuadrante 3 vuelve a hacer girar al usuario hacia el 8, lo que provoca cierta confusión.

De igual manera se probó la ruta inversa, desde el cuadrante 8 al 13, y el resultado fue análogo.

5.3.2.7. Ruta del cuadrante 9 al cuadrante 13

Como la ruta por el *hall* de la planta baja había resultado tediosa en el caso anterior, se hizo una prueba desde el otro extremo. Esta vez la ubicación del *beacon2* favorece el giro del usuario para encontrar el ascensor (escaleras en este caso). De esta manera la ruta resulta mucho más natural y organizada que en el caso anterior.

5.3.3. Usuario perdido

En esta sección se expone el comportamiento de la aplicación cuando el usuario sale de la ruta, basado en ejemplos de ejecución reales.

En la Sección 5.3.2.3 vimos un caso en el cual era probable que un usuario se desviara de la ruta, debido a la pérdida de la instrucción de giro asociada al cuadrante 4. De esta manera el usuario continuaría hasta llegar al cuadrante 5. Es entonces cuando la aplicación detecta que el usuario se ha salido de la ruta e indica al usuario que debe volver por la dirección en la que venía y le permite iniciar de nuevo la ruta al destino desde su posición actual. Como ya se ha comentado, esta indicación se da con cierto retraso debido a que el valor que debe alcanzar la variable *numPasosPerdidos* es demasiado elevada para este edificio.

Además, la aplicación está preparada para el caso en el que no se detecte ningún *beacon*. Si no se produjera ninguna detección transcurrido un tiempo (determinado por una variable umbral), la aplicación advierte al usuario de la situación indicándole que debe posicionarse dentro del edificio³. Esta situación se reprodujo avanzando hacia la izquierda del cuadrante 0 y eliminando los *beacons* más próximos a esa zona para provocar que ninguno de ellos fuera detectado.

A pesar de que no ha sido detallado expresamente durante la realización de las pruebas, hay que mencionar que el resto de funcionalidades como el modo silencio, la repetición de la última instrucción, el modo instrucciones detalladas y la finalización anticipada de la ruta también fueron comprobados, sin destacar ningún comportamiento extraño o inesperado.

5.4. Conclusiones de la evaluación

En las pruebas realizadas y descritas en la Sección 5.3 se ha puesto de manifiesto el comportamiento de la aplicación en situaciones normales y extremas. Abordando también aquellos casos en los que el usuario se pierde. En esta sección se exponen los puntos más relevantes obtenidos tras el análisis de las situaciones vistas.

- *El código de la aplicación funciona de la manera esperada:* A lo largo de las numerosas rutas de prueba se ha podido comprobar que las instrucciones, vibraciones y sonidos se reproducen e indican al usuario de la manera esperada. No se ha apreciado ningún comportamiento inesperado o erróneo.
- *El mapeo del edificio juega un papel primordial:* Como se puede observar en las Secciones 5.3.2.6 y 5.3.2.7 la disposición de los cuadrantes y *beacons* puede facilitar la

³Se asume que si ningún *beacon* es detectado es porque el usuario no se encuentra dentro del edificio o la zona mapeada

ruta al usuario en gran medida. Por ello, cuantos menos cuadrantes más sencilla será la ruta. La ubicación de los *beacons* debe ser lo más neutra posible. Es decir, que no favorezca más una ruta que otra.

- *Se debe ajustar el umbral de la variable numPasosPerdidos al espacio comprendido entre los beacons:* Han sido varias las ocasiones donde se ha puesto en evidencia que las indicaciones sobre el cambio de ruta debían haberse proporcionado con anterioridad. Para solucionarlo basta con reducir este umbral, teniendo en cuenta que el tiempo que tarda en recorrer un espacio una persona con discapacidad visual suele ser ligeramente mayor.
- *La generalidad de la aplicación:* Debido a las circunstancias, la evaluación de la aplicación ha sido realizada en un edificio distinto a la Facultad de Informática de la UCM. Gracias a una implementación general, apenas dependiente del espacio donde se despliega, el esfuerzo para poder adaptarla queda reducido a la realización de los archivos xml y json de la Sección 3.2.
- *Ventajas de las instrucciones e información adicional anticipadas:* Como pudimos ver en la Sección 5.3.2.4, el hecho de que las instrucciones de giro se avisen con un cuadrante de antelación prepara al usuario para el cambio de dirección, favoreciendo que este no se salga de la ruta, y, en caso de que el *beacon* de la intersección no se detecte, facilita que la información de la ruta persista. Lo mismo ocurre con la información adicional, sobre todo en el caso de los ascensores, pues permite al usuario identificar que debe cambiar de planta para llegar a su destino por adelantado.

Capítulo 6

ONCE

6.1. Reunión en el Centro de Tiflotecnología e Innovación de la ONCE

La idea de este trabajo de fin de grado surge de la necesidad de resolver problemas reales para gente real, concretamente para personas con discapacidad visual. De esta manera, empezamos nuestro camino por lo más importante: conocer las necesidades de los usuarios finales. Con este fin y gracias a la oportunidad que nos ha brindado la Universidad Complutense de Madrid con la profesora María Guijarro al frente, hemos podido reunirnos y entrevistar a personas especializadas en el campo de las tecnologías que sufren discapacidad visual.

En este documento recogemos las notas que tomamos durante la reunión el pasado 11 de octubre de 2019 en el CTI (Centro de Tiflotecnología e Innovación) de la ONCE, donde nos dieron una pequeña charla sobre la ceguera y las tecnologías accesibles que han surgido para reducir la brecha, y en la que finalmente conectamos con potenciales usuarios que nos hablaron sobre sus gustos y necesidades.

6.1.1. Introducción

La visita al CTI comenzó con una breve explicación, de la mano de José María Ortiz, director del Departamento de Consultoría e Innovación, sobre las principales tareas que se llevan a cabo en el centro, entre las cuales destacan:

- Ayudar a una persona con discapacidad visual en su **adaptación** al trabajo y a la vida cotidiana, proporcionandole para ello el material necesario (teclados, líneas de braille, bastones, etc.).
- Responder a **consultas** sobre el funcionamiento de dispositivos.

Luego, nos comentó los departamentos en los que se estructura el centro para que pudiésemos hacernos una idea más global de todo lo que abarca. Éstos son:

- **El departamento de Consultoría e Innovación**, donde actualmente están desarrollando el programa EDICO (PONER REFERENCIA) en colaboración con la UCM, que tiene como objetivo hacer las matemáticas accesibles mediante un editor de texto. De manera paralela se encargan del desarrollo de aplicaciones de muy

diversa índole, véase apps para la biblioteca de la ONCE, de películas audio-descritas, etc.

- **Departamento de Evaluaciones y Auditoría**, donde se encargan de evaluar los productos que se van a sacar al mercado.
- **Departamento de Diseño y Producción**, donde se encargan de, tal y como indica su nombre, diseñar y producir elementos de adaptabilidad, como pueden ser unas plantillas con relieve de policarbonato para las vitrocerámicas. Recordemos que estas, aunque no presentan dificultad alguna para los usuarios videntes, son tediosas para aquellos que cuentan con discapacidad visual ya que la pantalla táctil no tienen ningún tipo de relieve que pueda servirles como referencia y guiarles en su uso.
- **Departamento de Asesoría en Tecnología**, especializado en tecnologías accesibles.

Una vez concluida esta sección en la que nos contextualizaron, abrieron paso a la ronda de preguntas en la que pudimos acercarnos a ellos, conociendo sus problemas y necesidades en el día a día.

6.1.2. Entrevista

Durante esta parte, nos dirigimos especialmente a Mónica y José Luis Llorente, ambos ingenieros del CTI, para que, con su experiencia y conocimientos, nos explicaran lo máximo posible sobre tecnologías accesibles y nos dieran su punto de vista en las ideas que proponíamos. Por otro lado, Mónica no solo era experta en la materia sino que además es evidente, por lo que nos pudo contar su perspectiva y necesidades como usuaria.

Las preguntas avanzaron desde temas generales para conocer cómo una persona invidente se desenvuelve con la tecnología, sus gustos y qué sensaciones le despierta, hasta temas concretos dirigidos a conocer los problemas que plantea la navegación por espacios interiores:

- **¿Cómo utiliza una persona con discapacidad visual un dispositivo móvil?**

Para responder a esta pregunta, Mónica nos hace una demostración en directo. Para ello emplea un móvil Xiaomi con sistema operativo Android.

Mónica nos cuenta que para la navegación por su dispositivo utiliza un lector de pantalla, es decir, un software que facilita el uso del sistema operativo. Éste sirve como guía para las personas que, como ella, tienen discapacidad visual, ya que “lee y explíca” mediante voz lo que se ve en la pantalla. Los lectores de pantalla vienen siempre incluidos en el dispositivo y se pueden encontrar en la sección de Accesibilidad, en Ajustes. En el caso de Android, este software se llama *Talkback* y es configurable. Por ejemplo, dice Mónica, se podría usar mediante la línea de braille en vez de la reproducción por voz.

Luego vemos como se desplaza por las aplicaciones utilizando *flicks*, movimientos secos en los que desliza el dedo hacia uno de los lados de la pantalla (izquierda o derecha, según interese). Del mismo modo, para la navegación por la web o dentro de alguna aplicación utiliza estos movimientos hacia arriba y hacia abajo. Por último, nos muestra cómo accede a un elemento mediante doble click.

También nos habla de la posibilidad de la navegación libre, eso sí, solo cuando ya te has familiarizado con el dispositivo lo suficiente como para saber dónde tienes determinadas aplicaciones.

Lo más cansado, según Mónica, es tener que hacer un barrido por toda la pantalla hasta encontrar lo que quieras, en vez de poder ir directamente. Para agilizar un poco este proceso, Mónica, por ejemplo, agrupa las aplicaciones por carpetas, de modo que el barrido es más sencillo que si la pantalla estuviese repleta.

Para las personas con baja visión también existe la posibilidad de hacer más grandes los iconos y ajustar los colores.

■ **Hemos leído que normalmente las aplicaciones se desarrollan para dispositivos iOS, ¿por qué es mejor?**

“Si que es cierto que solía ser así ya que iOS le llevaba la delantera a Android en cuanto a accesibilidad, pero cada vez se usa más Android pues las diferencias están completamente recortadas, están muy igualados y los precios son mucho más asequibles. Yo misma antes tenía un iPhone y ahora me he pasado a Android y no hay nada que eche en falta.”, responde Mónica.

■ **¿Cómo afronta una persona ciega su desplazamiento y orientación por interiores cuando pisa por primera vez dicho espacio u edificio?**

Ante esta pregunta Mónica resopla y nos contesta: “*Buuff..., ¿te vale?*”

Nos puso como ejemplo la llegada a un hospital: “*cuando entras necesitas saber, al menos, dónde está la recepción para pedir ayuda pero los carteles informativos están fuera de mi alcance, entonces entro por la puerta y pienso ¿y ahora qué?. ¿Dónde está el mostrador de recepción? No es tan fácil como echar un primer vistazo, necesitas ayuda mediante voz, algo que te describa el espacio y te vaya diciendo que hay a derecha e izquierda y a cuantos metros.*”

Nos contó que en cuanto a la descripción/guía por espacios interiores ahora mismo no hay disponible ninguna aplicación. Por ello, una vez superada la primera barrera de ubicar y localizar un cierto destino, la única opción que les queda es la de memorizar el camino. Mónica destacó que era increíble la cantidad de rutas que tiene en la cabeza.

Por todo esto, se comentó que una aplicación del tipo que se quiere implementar en este trabajo de fin de grado sería de gran ayuda para ellos. No solo para que les guiase hasta un punto concreto, sino para que también describiese el edificio, facilitándoles una primera idea del mismo que les ayudase a moverse con mas seguridad y les indicase qué posibilidades les ofrece. En este punto se mencionaron otras propuestas e ideas sobre cómo proporcionar esta información estática del edificio. Algunas de ellas fueron:

- Tener subido el plano de las distintas plantas del edificio e implementar un sistema de manera que cuando se deslice el dedo sobre las distintas salas que aparecen la app indique cuales son (aula X, cafetería, secretaría, pasillo, etc.).
- La impresión de un mapa 3D que disponga de un código QR o algo similar que sea capturado por Bluetooth (mejor que por foto) y que tras leerlo cargue el plano del edificio y pueda proporcionar tanto información sobre el espacio en sí mismo (número de plantas, qué hay en cada una...) como información cambiante como la existencia de averías, horarios, disponibilidad de salas, etc.

Como es natural, de la mano de estas ideas surgieron problemas y opiniones a favor y en contra: *¿Dónde estaría dicho mapa?, ¿Cómo encontrarlo?, ¿Todos los edificios*

estarán de acuerdo en facilitar los planos o puede que por motivos de seguridad no sea una idea factible?, ¿Es posible llegar a un standard para que se pueda usar el mismo sistema en cualquier edificio?

- **¿Hay algún tipo de señales que os sirvan como referencia a la hora de desplazaros por un edificio?**

“Hay señales de encaminamiento, que te indican dónde están las escaleras, ascensores, zonas de cruce, etc.”, contesta.

- **¿Cuántos edificios cuentan con estas señales?**

“La verdad que cada vez son más frecuentes y hoy en día se encuentran en casi todos los edificios, especialmente en los nuevos.”, responde.

- **¿Cómo de factible es ir con el dispositivo móvil en la mano, para realizar una foto o cualquier cosa similar?**

“Puedo hacer una foto en un momento puntual, en eso no hay problema alguno pero no es cómodo ir con el móvil en la mano constantemente porque además de que es aparatoso porque ya llevo en la mano el bastón, perro guía, etc. No es práctico, no sería la primera vez que roban un móvil a una persona invidente, es una realidad.”, contesta Mónica. *“Particularmente, con respecto a la foto el problema principal sería saber a dónde enfocar”*, añade.

6.1.3. Conclusiones

Tras el debate, algunas de las conclusiones que sacamos de la visita al CTI son:

- La implementación de una aplicación como la nuestra es muy útil y necesaria.
- Las modalidades más empleadas para interactuar con el móvil cuando tienes algún tipo de deficiencia visual son: flicks, sacudidas, mediante vibración, arrastrando o pulsando la pantalla con un dedo, dos,...
- No resulta cómodo ir barriendo el espacio con la cámara del móvil.
- El uso de dispositivos adicionales como una micro cámara, en principio, no sería un problema, siempre y cuando no la tengan que llevar de manera continuada en la mano.
- En caso de auriculares, se recomienda utilizar auriculares óseos de modo que dejen el canal auditivo libre para captar otros estímulos.
- El objetivo es que el grueso de las aplicaciones sean lo más inclusivas posibles, es decir, que su uso sea apto tanto para personas videntes como invidentes.
- El feedback de la aplicación no debe saturar pero sí se aconseja que sea constante para que no se malinterprete que la aplicación ha dejado de funcionar.

Capítulo 7

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Antes de la entrega de actas de cada convocatoria, en el plazo que se indica en el calendario de los trabajos de fin de máster, el estudiante entregará en el Campus Virtual la versión final de la memoria en PDF. En la portada de la misma deberán figurar, como se ha señalado anteriormente, la convocatoria y la calificación obtenida. Asimismo, el estudiante también entregará todo el material que tenga concedido en préstamo a lo largo del curso.

Chapter 8

Introduction

Introduction to the subject area. This chapter contains the translation of Chapter 1.

Chapter 9

Conclusions and Future Work

Conclusions and future lines of work. This chapter contains the translation of Chapter 7.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.
(modificar en Cascaras\bibliografia.tex)

Miguel de Cervantes Saavedra

ENVISION. Challenges blind people face when living life. Disponible en <https://www.letsenvision.com/blog/challenges-blind-people-face-when-living-life>.

GÓMEZ ULLA, F. Informe sobre la ceguera en España. Disponible en http://www.seeof.es/archivos/articulos/adjunto_20_1.pdf.

LÓPEZ MAÑAS, E., MORENO NACARINO, F. J. y PLÁ HERRERO, J. Proyecto avanti: sistema de asistencia a la evacuación de incendios. 2010.

POZO-RUZ, A., RIBEIRO, A., GARCÍA-ALEGRE, M., GARCÍA, L., GUINEA, D. y SANDOVAL, F. Sistema de posicionamiento global (gps): Descripción, análisis de errores, aplicaciones y futuro. *ETS ingenieros de Telecomunicaciones. Universidad de Málaga*, 2000.

RIVERO, F. Informe di trendia: Mobile en España y en el mundo 2019. Disponible en <https://mktefa.ditrendia.es/hubfs/Ditrendia-Informe%20Mobile%202019.pdf>.

SHAH, D. y SHAH, K. Wi-fi based positioning system. ????

SHAH, K., DARSHAN. Basic of wi-fi based positioning system. 2012.

VÍCTOR GUTIÉRREZ RODRÍGUEZ, J. D. L. M. y MURGA, V. M. P. *Generador interactivo de instrucciones de guía sobre plataformas móviles*. Versión electrónica, 2014.

DE LA VILLA, M. M.-C. *Sistema de guía por voz en interiores*. Versión electrónica, 2014.

WAYFINDR. Open standard for audio-based wayfinding. 2018.

Apéndice A

Título del Apéndice A

Contenido del apéndice

Apéndice **B**

Título del Apéndice B

Este texto se puede encontrar en el fichero Cascaras/fin.tex. Si deseas eliminarlo, basta con comentar la línea correspondiente al final del fichero TFMTeXiS.tex.

*-¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.*

*Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

*-Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.*

*Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

