UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

ASISTENTE VIRTUAL DE NAVEGACIÓN Y GENERACIÓN DE RUTAS EN ESPACIOS CERRADOS

Trabajo de Integración Curricular Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y Redes
LÍNEA DE INVESTIGACIÓN: Programación y Redes
AUTOR: Jonathan Daniel Núñez Bonilla
TUTOR: Ing. Víctor Santiago Manzano Villafuerte, Mg.

Ambato - Ecuador
marzo – 2023

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Integración Curricular con el tema: ASISTENTE VIRTUAL DE NAVEGACIÓN Y GENERACIÓN DE RUTAS EN ESPACIOS CERRADOS, desarrollado bajo la modalidad Proyecto de Investigación por el señor Jonathan Daniel Núñez Bonilla, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 de las segundas reformas al reglamento para la ejecución de la Unidad de Integración Curricular y la obtención del título de tercer nivel, de grado en la Universidad Técnica de Ambato y el numeral 7.4 del respectivo instructivo del reglamento.

Ambato, marzo 2023.

Firmado electrónicamente por:
VICTOR SANTIAGO MANZANO VILLAFUERTE

Ing. Víctor Santiago Manzano Villafuerte, Mg.

TUTOR

AUTORÍA

El presente trabajo de Integración Curricular titulado: ASISTENTE VIRTUAL DE NAVEGACIÓN Y GENERACIÓN DE RUTAS EN ESPACIOS CERRADOS es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2023.

Jonathan Daniel Núñez Bonilla

C.C. 1804371159

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Integración Curricular como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Integración Curricular en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, marzo 2023.

Jonathan Daniel Núñez Bonilla

C.C. 1804371159

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Integración Curricular presentado por el señor Jonathan Daniel Núñez Bonilla, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado ASISTENTE VIRTUAL DE NAVEGACIÓN Y GENERACIÓN DE RUTAS EN ESPACIOS CERRADOS, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 de las segundas reformas al reglamento para la ejecución de la Unidad de Integración Curricular y la obtención del título de tercer nivel, de grado de la Universidad Técnica de Ambato y sus reformas y al numeral 7.6 del respectivo instructivo del reglamento. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, marzo 2023.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Pilar Urrutia, Mg.

PRESIDENTE DEL TRIBUNAL

ELIZABETH PAULINA AYALA BANO

Firmado digitalmente por ELIZABETH PAULINA AYALA BANO
DN: cn=ELIZABETH PAULINA AYALA BANO, c=EC, l=QUITO, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Motivo: He revisado este documento
Ubicación:
Fecha: 2023-03-15 12:17:05:00

Ing. Paulina Ayala, Mg.

PROFESOR CALIFICADOR

Firmado electrónicamente por:
ANDREA PATRICIA SANCHEZ ZUMBA

Ing. Andrea Sánchez, Mg.

PROFESOR CALIFICADOR

DEDICATORIA

Al niño, quien, con su curiosidad y su espíritu de aventura, comenzó a descubrir el maravilloso mundo de la educación y el conocimiento. A ti, quien, con tu ingenuidad y tu entusiasmo, te enfrentaste a las primeras dificultades y aprendiste a superarlas. A ti, quien, con tu perseverancia y tu dedicación, has llegado hasta aquí. Este trabajo es también tuyo. Te prometo seguir aprendiendo y descubriendo, como tú lo hiciste.

Jonathan Daniel Núñez Bonilla

AGRADECIMIENTO

A mi familia, quienes han sido mi roca sólida y mi guía en todo momento. Su amor incondicional y apoyo incansable han sido fundamentales para lograr esta meta.

A mi tía Carmen quien ha sido una figura materna en mi vida y una constante fuente de amor y apoyo. Quien me ha impulsado siempre a perseguir mis metas y a no abandonarlas frente a las adversidades.

A mis amigos Sarita, Naythan, Nancy y Shirley quienes han sido una parte esencial en mi vida académica y personal. Sus consejos, risas y apoyo han sido insustituibles para superar los obstáculos y alcanzar este logro.

A mi tutor, el Ing. Santiago Manzano quien, con su dedicación, paciencia y experiencia, me guió en este proceso de investigación.

Jonathan Daniel Núñez Bonilla

RESUMEN EJECUTIVO

El avance tecnológico y el internet de las cosas han potenciado la creación de edificios inteligentes que se acoplen a los requerimientos de los propietarios como administradores, dando paso a la recolección de información relevante para la gestión del lugar y mejorar la experiencia de los visitantes.

Una de las necesidades más atractivas y a la vez desafiante de los edificios o espacios cerrados es la navegación en sus interiores. Para solventar esta demanda es necesario el uso de tecnologías. En la actualidad existen diversas tecnologías para navegación de las cuales destacan: UWB, Wifi, Bluetooth de baja energía (BLE), RFID y GPS. Los sistemas GPS en entornos exteriores combinados con celulares independientes puede lograr una alta precisión. Sin embargo, en interiores y entornos con efectos de sombra profunda, las señales satelitales y celulares a menudo se interrumpen, y el posicionamiento es problemático.

En el presente proyecto se expone una solución de sistemas de navegación en interiores basada en dispositivos móviles utilizando Bluetooth de baja energía (Beacons) como tecnología de apoyo e implementado en una aplicación móvil que traza la ruta para llegar a un determinado lugar con la ayuda de un asistente virtual.

La aplicación móvil denominada Rutas app está desarrolla en el framework Flutter a partir de lenguaje de programación Dart y se encuentra disponible tanto para visitantes como administradores, en la cual se presenta un mapa dinámico del espacio con información de los lugares de interés para los usuarios y un apartado de rutas en el cual dependiendo al lugar al que se desee llegar presenta la ruta más óptima a seguir, esto con ayuda del asistente virtual incorporado, mismo que es activado por voz. Abordando así problemas de desubicación y mejorando la experiencia de visitar el lugar, además muestra a los administradores reportes de los lugares con más recurrencia y los usuarios que más frecuentan el espacio, esta información es de relevancia ya que a partir de ella se puede tomar acciones y aplicar estrategias para gestionar de mejor manera el lugar.

Palabras clave: Sistema de posicionamiento interior, Flutter framework, beacons, aplicación móvil.

ABSTRACT

Technological progress and the internet of things have boosted the creation of intelligent buildings that meet the requirements of owners and managers, giving way to the collection of relevant information for the management of the place and improve the experience of visitors.

One of the most attractive and at the same time challenging needs of buildings or enclosed spaces is the navigation in their interiors. To solve this demand, the use of technologies is necessary. At present there are several technologies for navigation of which stand out: UWB, Wi-Fi, Bluetooth low energy (BLE), RFID and GPS. GPS systems in outdoor environments combined with stand-alone cell phones can achieve high accuracy. However, indoors and in environments with deep shadow effects, satellite and cellular signals are often disrupted, and positioning is problematic.

This project presents an indoor navigation system solution based on mobile devices using Bluetooth Low Energy (Beacons) as an enabling technology and implemented in a mobile application that suggests the most optimal route to reach a certain location with the help of a virtual assistant.

The mobile application called Rutas app is developed in the Flutter framework based on Dart programming language and is available for both visitors and administrators, which presents a dynamic map of the space with information on places of interest to users and a section of routes in which depending on the place you want to reach presents the optimal route to follow, this with the help of the built-in virtual assistant, which is activated by voice. Thus, addressing problems of misplacement and improving the experience of visiting the place, it also shows the administrators reports of the places with more recurrence and users who frequent the space, this information is relevant because from it you can act and implement strategies to better manage the place.

Keywords: Indoor positioning system, Flutter framework, beacons, mobile application.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de Investigación

Asistente virtual de navegación y generación de rutas en espacios cerrados.

1.1.1. Planteamiento de Problema

Según la ONU el 54% de la población mundial actual reside en áreas urbanas y se prevé que para 2050 llegará al 66% [1]. Hoy en día gracias al avance tecnológico y el internet de las cosas han posibilitado la creación de edificios inteligentes que proporcionan información útil que ayuda a mejorar la gestión del espacio y brindar una experiencia elevada a los visitantes. En el Ecuador un edificio de estas características debe disponer de sistemas de seguridad modernos (tarjetas magnéticas de identificación para entradas y ascensores, monitoreo con cámaras de video), servicios de cisterna, gas centralizado, plantas de electricidad de emergencia y equipos de lucha contra incendios [2], sin embargo, una de las atracciones y desafíos que aún no se encuentran aplicados es la navegación dentro de sus interiores. Ya sea por ser edificaciones extensas o por recibir usuarios que no conocen las instalaciones surge el problema constante de la desubicación dentro de las instalaciones.

La generación de sistemas de navegación en espacios cerrados con modelos interactivos como los asistentes virtuales pueden resultar de gran utilidad dentro del desarrollo de ciudades inteligentes, dando como resultado una mejor calidad de servicio y experiencia al momento de ubicarse dentro de un edificio o cualquier espacio cerrado en el cual no sea posible el uso de sistemas de posicionamiento globales (GPS).

1.2. Antecedentes Investigativos

Tras realizar la investigación bibliográfica en repositorios de universidades, publicaciones en revistas, así como en diversos bases de datos de artículos científicos se pueden encontrar los siguientes antecedentes.

J. Garcés y F. Rubio (2018) de la Universidad De Las Fuerzas Armadas Campus Latacunga-Ecuador con el tema "DESARROLLO DE UN ASISTENTE VIRTUAL MÓVIL PARA POTENCIALIZAR LA EXPERIENCIA TURÍSTICA ARQUITECTÓNICA PATRIMONIAL DE LA CIUDAD DE LATACUNGA" exponen una solución para incrementar el turismo y mejorar la experiencia a través de un asistente virtual el cual muestra información relevante al usuario así como la rutas que debe seguir para llegar a un determinado sitio, el desarrollo de este asistente virtual en aplicaciones móviles se realizó ocupando una metodología de desarrollo ágil Mobile-D la cual tiene como objetivo conseguir ciclos de desarrollo muy rápidos en equipos muy pequeños y se basa en metodologías para el desarrollo de aplicaciones móviles. Teniendo como resultado un asistente virtual amigable con el usuario y con interfaz sencilla para su manejo logrando solventar dudas al momento de realizar visitas turísticas, obteniendo así un producto completamente funcional y de calidad [3].

Morales Andrés (2018), del Instituto Tecnológico de Chihuahua, desarrolló el tema "SISTEMA DE NAVEGACIÓN EN INTERIORES POR MEDIO DE DISPOSITIVOS MÓVILES Y EMISORES DE SEÑAL BLUETOOTH" en el cual propone un sistema de navegación en interiores (SNI) para dispositivos móviles utilizando beacons que difunden señales bluetooth de baja energía usando la emisión del RSSI, para esto basa su sistema en tres etapas: la primera es la localización del dispositivo móvil en interiores, la segunda consta del desarrollo del mapa en donde se ubicara el usuario y la tercera presenta la generación de la ruta de navegación entre la ubicación actual y la de destino. En el cual expone que el uso de la emisión del RSSI para ubicación de dispositivos móviles es imprecisa y requiere mucha inversión en tiempo en calibración y ajuste [4].

López Luis (2019), del Tecnológico Nacional de México plantea el tema "GENERACIÓN AUTOMÁTICA DE RUTAS DE NAVEGACIÓN EN INTERIORES UTILIZANDO LA TECNOLOGÍA DE BEACONS" en el mismo se propone el desarrollo de una aplicación móvil para la generación de rutas en ambientes interiores en el cual usa el algoritmo Dijkstra como opción para generación de rutas ya que ayuda a la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. Los resultados de aplicar

algoritmo Dijkstra en pruebas de funcionamiento fueron de 100% de efectividad en el proceso de encontrar el camino dándole un punto inicial y un destino [5].

Garibay Fabricio (2020) de INFOTEC Centro De Investigación e Innovación en Tecnologías de la información y Comunicación de México, en su proyecto de investigación "DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE VIRTUAL (CHATBOT) PARA OFRECER ATENCIÓN A LOS CLIENTES DE UNA AEROLÍNEA MEXICANA POR MEDIO DE SUS CANALES CONVERSACIONALES" plantea la creación de un chatbot como asistente virtual usando canales conversacionales en WhatsApp y Web, haciendo uso de la tecnología Aivo, específicamente, el producto CP-Bot, el cual es un asistente virtual que posee inteligencia artificial, posibilitando así el uso de esta herramienta como auxiliar de la atención al cliente. Dando como resultado una mejor calidad de servicio a los clientes de la aerolínea y optimizando el tiempo de atención a los mismos ocupando recursos del teléfono móvil [6].

Jover Héctor (2020) de la Universidad Politécnica de Valencia en su trabajo de fin de grado con el tema "DESARROLLO DE UN CHATBOT PARA LA RECOMENDACIÓN DE EVENTOS O LUGARES DE INTERÉS" realiza un chatbot para la ciudad de Valencia, España el cual funciona por medio de la aplicación Telegram el cual implementa un servicio de geolocalización permitiendo encontrar diferentes puntos de interés dependiendo la ubicación del usuario. Para esto utiliza el lenguaje de programación JavaScript el cual fue de gran utilidad para crear un servicio de localización totalmente a mano y fácil de usar, para su funcionamiento se accede desde la terminal de comandos a la carpeta donde se ha desarrollado el proyecto y ejecutando el comando node index.js. una vez ejecutado el Bot se inicializa y puede iniciar la comunicación vía Telegram [7].

1.3. Fundamentación Teórica

1.3.1. Edificios Inteligentes

Un edificio inteligente es aquel que crea un entorno que maximiza la eficiencia de los ocupantes del edificio y al mismo tiempo permite una gestión eficaz de los recursos con costes mínimos de por vida [8].

Los edificios inteligentes tienen buen sentido comercial. A medida que la era de la información permite escalar a nuevas tecnologías, los sistemas de construcción inteligente (IBS) tienen la flexibilidad y el modularidad para adaptarse a cada cambio. Un sistema inteligente de distribución de premisas permite al propietario, administradores y ocupantes aprovechar la nueva tecnología a medida que esté disponible, a un costo mínimo y sin una interrupción importante de la productividad del espacio de trabajo de la oficina [8].

Son edificios que integran y dan cuenta de la inteligencia, la empresa, el control y los materiales y la construcción como un sistema de construcción completo, con adaptabilidad y no reactividad, en el centro, para cumplir con los impulsores de la progresión del edificio: energía y eficiencia, longevidad, y comodidad y satisfacción. La mayor cantidad de información disponible de esta gama más amplia de fuentes permitirá que estos sistemas se vuelvan adaptables y permitirán que un edificio inteligente se prepare para el contexto y el cambio en todas las escalas de tiempo [9].

1.3.2. Sistemas de posicionamiento exterior (EPS)

En la actualidad existen tres sistemas de posicionamiento exterior (EPS):

- ***GALILEO***

Es un sistema global de navegación por satélite que está siendo desarrollado por Europa. El sistema proporcionará un servicio de posicionamiento global garantizado de alta precisión bajo control civil. El sistema completo tendrá 30 satélites (27 operativos más 3 de repuesto activos) posicionados en tres planos circulares de órbita terrestre media a 23616 km de altitud sobre la Tierra, y a una pulgada de los planos orbitales de 56 grados con referencia IO al plano ecuatorial. Una vez completado el sistema, las señales de navegación de Galileo proporcionarán una buena cobertura incluso en latitudes de hasta 75 grados norte, que corresponde al Cabo Norte, y más allá. El gran número de satélites junto con la optimización de la constelación y la disponibilidad de los tres satélites activos de repuesto garantizarán que la pérdida de un satélite no tenga un efecto perceptible en el usuario [10].

- ***GLONASS***

El Sistema Satelital de Navegación Global de Rusia (GLONASS) se desarrolló durante la guerra fría con fines militares como el GPS. En 1995, el sistema alcanzó la constelación completa con 24 satélites, el número de satélites más antiguos se redujo rápidamente porque la vida útil promedio de diseño de los satélites era de solo tres años [10]. Están colocados en dos órbitas (tres órbitas son nominales) y el rendimiento del sistema no es óptimo [11].

Al contrario del GPS, cada satélite GLONASS transmite en su propia frecuencia. Sin embargo, algunos satélites tienen las mismas frecuencias. pero esos satélites se colocan en ranuras antípodas de planos de órbita y no aparecen al mismo tiempo en la vista de los usuarios [10].

- ***GPS***

El Sistema de Posicionamiento Global (GPS), llamado también NAVSTAR (Sistema de Navegación usando Tiempo y Rango) tiene 30 satélites activos en 6 órbitas (el número nominal de satélites es 24) [11].

El sistema de posicionamiento global (GPS) consta de tres segmentos: el segmento espacial, el segmento de control y el segmento de usuario [12].

- El segmento de control rastrea cada satélite y carga periódicamente al satélite la predicción de las posiciones futuras del satélite y las correcciones de la hora del reloj del satélite. Estas predicciones son transmitidas continuamente por el satélite al usuario como parte del mensaje de navegación.
- El segmento espacial consta de 24 satélites, cada uno de los cuales transmite continuamente una señal de alcance que incluye el mensaje de navegación que indica la posición actual y la corrección de la hora.
- El receptor del usuario rastrea las señales de alcance de los satélites seleccionados y calcula la posición tridimensional y la hora local.

1.3.2.1. Limitaciones de los EPS

En caso de visibilidad clara, hay varias señales de satélite fuertes presentes y la tarea de seguimiento de adquisición y el cálculo de la posición son relativamente fáciles.

La otra situación es para el caso de la ubicación del receptor del usuario en un entorno difícil, es decir, un cañón urbano o en interiores. Las señales provenientes de los satélites están obstruidas por follaje, paredes y otras estructuras. La señal que puede utilizar el receptor del usuario en dicho entorno consiste principalmente en una señal directa fuertemente atenuada y señales reflejadas (y/o dispersas) que llegan normalmente por la ruta de menor resistencia. Por lo que los dos fenómenos principales con los que debe contar el receptor interior son la atenuación de la señal y el efecto multicamino [13].

1.3.3. Sistema de Posicionamiento Interior (IPS)

La navegación y el posicionamiento exterior se basan en tecnologías como GPS con la ayuda de una red de satélites. Esta tecnología se utiliza para navegar en grandes áreas y espacios abiertos, pero no son útiles en entornos interiores debido a la falta de línea de visión y discrepancias en la propagación de la señal [14]. Por lo tanto, se han desarrollado Sistemas de Posicionamiento Interior (IPS) los cuales son capaces de localizar personas u objetos dentro de un espacio cerrado.

Dado el número significativo de diferentes sistemas de posicionamiento en interiores existentes, es difícil tener una definición más estrecha o solo una métrica [15]. Aunque la precisión es de suma importancia, no es el único criterio que se tiene en cuenta para evaluar un IPS. La cobertura, la complejidad, la solidez, la escalabilidad, el costo, la privacidad y el consumo de energía son métricas que generalmente se usan para la evaluación y comparación de IPS [16].

- La cobertura se refiere al rango de las señales de la tecnología que soporta un IPS. Una alta cobertura puede traducirse en la aplicabilidad del IPS a grandes áreas utilizando un bajo número de emisores.
- La complejidad se refiere a los esfuerzos requeridos para la construcción, implementación o configuración del hardware y software del IPS.
- La robustez es la resistencia del sistema a condiciones más allá de las que se consideran nominales.
- El costo se refiere a cualquier tipo de costo relacionado con los dispositivos de posicionamiento o la infraestructura requerida.

- La privacidad está relacionada con las restricciones del sistema que evitan la recopilación de información que puede usarse para identificar o rastrear a los usuarios.
- Finalmente, mientras menores sean los requisitos de energía, mejor. En los dispositivos de usuario, un bajo requisito de energía se traduce en un bajo consumo de batería. Para la infraestructura IPS, un bajo consumo de energía puede traducirse en que solo se requieren pequeños esfuerzos para el mantenimiento [17].

1.3.3.1. Técnicas empleadas en IPS

La siguiente lista presenta brevemente las técnicas más comunes empleadas con las tecnologías utilizadas en IPS.

Tabla 1 Técnicas más comunes utilizadas en IPS

Hora de llegada (TOA)	Diferencia horaria llegada (TDOA)
Mide el tiempo de llegada de la señal de un emisor, según lo registrado por el receptor. Se utiliza para estimar la distancia a cada emisor, ya que la velocidad de propagación de la señal (sonido, radiofrecuencias) se conoce para el medio de transmisión (aire).	Es similar a TOA. Mide las diferencias en el tiempo de llegada de las señales de diferentes emisores. Se utiliza para estimar las diferencias en las distancias a cada emisor.
Ángulo de llegada (AOA)	**Intensidad de señal recibida (RSS)**
Se refiere al ángulo en el que la señal llega al sensor. Los ángulos se utilizan para obtener una fijación de posición.	Es la intensidad a la que se mide la señal de un emisor. La intensidad de la señal disminuye a medida que aumenta la distancia al emisor, aunque su relación puede verse afectada por la atenuación y la interferencia.

Fuente: Investigador basado en [16].

1.3.4. Métodos para generación de rutas

La técnica empleada para una solución determina cómo se estima la posición. Básicamente existen dos métodos de IPS que emplean las técnicas de posicionamiento anteriormente expuestas.

Métodos basados en rango

Tanto la lateración como la angulación se clasifican comúnmente como métodos basados en rangos, y requieren el conocimiento previo de las posiciones de los emisores.

Lateración

TOA, TDOA y RSS se utilizan para estimar distancias a emisores de señal. Las distancias estimadas a un conjunto de emisores se utilizan en lo que se denomina lateración para encontrar la estimación de posición que mejor se ajuste al conjunto de distancias. La lateración se llama trilateración si se utilizan tres distancias, mientras que se llama multilateración si se utilizan más de tres [18].

Angulación

Tanto la lateración como la angulación se clasifican comúnmente como métodos basados en rangos o rangos, y requieren el conocimiento previo de las posiciones de los emisores [18].

Método libre de rango

Huellas dactilares

La técnica RSS también se emplea para un método libre de rango, muy popular en IPS, llamado huellas dactilares o, a veces, análisis de escena. La toma de huellas dactilares abarca dos etapas. En la primera etapa, también conocida como etapa fuera de línea, la cantidad de señal de cada emisor detectado en un momento y posición dados (una huella digital) se mide en varios lugares del escenario objetivo y se almacena para crear una caracterización de las señales en ese escenario lo más completa posible.

La técnica RSS también se emplea para un método libre de rango, muy popular en IPS, llamado huellas dactilares o, a veces, análisis de escena. La toma de huellas dactilares abarca dos etapas. En la primera etapa, también conocida como etapa fuera de línea, la cantidad de señal de cada emisor detectado en un momento y posición dados (una huella digital) se mide en varios lugares del escenario objetivo y se almacena para crear una caracterización de las señales en ese escenario lo más completa posible [18].

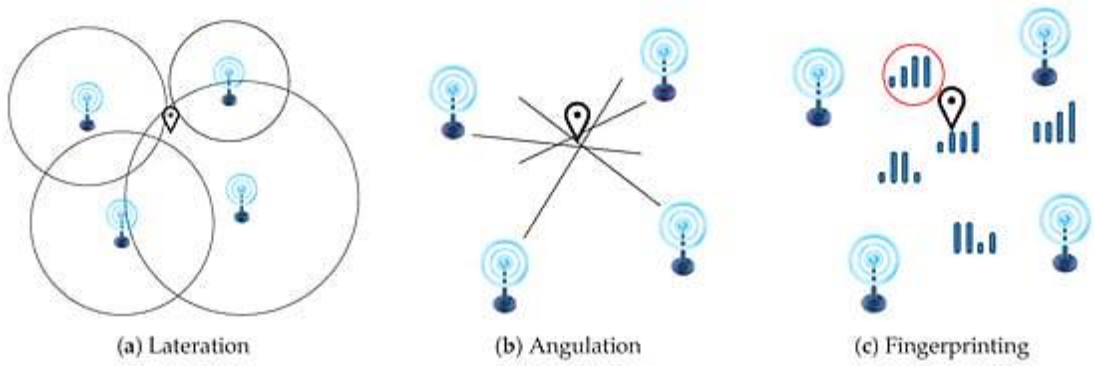


Figura 1 Métodos más comunes utilizados en IPS [18]

Conteo de saltos

Un salto se produce cuando un paquete pasa de un segmento de red al siguiente. El número de saltos de nodos conocidos se utiliza para estimaciones de distancia (gruesas) [19] o para inferir posiciones a través de un problema de incrustación de gráficos [20].

Análisis de visión

Se refiere a la aplicación de enfoques de visión por computadora a las imágenes recopiladas utilizando alguna técnica de imagen como, por ejemplo, cámaras. Los análisis detectan características relevantes en la escena que permiten estimar las posiciones de las entidades en la escena o la posición del dispositivo de grabación de imágenes [21].

Basado en dispositivos

Las soluciones basadas en dispositivos móviles son las más comúnmente abordadas por su versatilidad y compatibilidad con diferentes tecnologías para IPS. Estas soluciones por lo general se tratan de aplicaciones móviles aplicables a teléfonos inteligentes diseñadas para un espacio en específico las cuales dependiendo el enfoque usaran o no tecnologías IPS dado que los teléfonos inteligentes modernos inteligentes modernos tienen un número limitado de capacidades sensoriales.

1.3.5. Soluciones IPS

La tecnología subyacente sí importa, y sus particularidades deben tenerse en cuenta al crear un IPS[18].

La siguiente lista presenta las técnicas más comunes empleadas con las tecnologías utilizadas en IPS. La alta demanda de soluciones para sistemas de posicionamiento en interiores ha impulsado un número creciente de trabajos de investigación en los últimos años. Sin embargo, algunas presentan mejores resultados que otras.

Tabla 2 Tecnologías utilizadas en IPS

Tecnología		Base	Principio	Proceso	Resultados
Luz	**Luz visible (VLC)**	Basadas en dispositivos	Iluminación y su cambio de nivel de intensidad imperceptible al ojo humano	El receptor captura las propiedades de la señal (RSS, TDOA, AOA)	Los resultados varían según el escenario y las condiciones de prueba
	Infrarrojos	Basadas en dispositivos o sin dispositivos			
Visión artificial		Basado en cámaras	Localización y mapeo` simultáneos (SLAM) se basa principalmente en cámaras para la entrada sensorial	Colección de imágenes se utiliza para identificar los objetivos junto con los detalles del entorno registrados para el escenario objetivo	Requiere de indicadores en el entorno para que sean decodificados
Sonido		Señales acústicas	distinción entre sistemas que operan con ultrasonido y frecuencias audibles	Generalmente usa TOA o TDOA para realizar posteriorización	La velocidad del sonido se ve afectada por la temperatura y la humedad
Campos magnéticos		Basadas en magnetismo	Usan el campo magnético natural de la tierra o uno artificial	Las variaciones de fuerza en el campo magnético medido infieren en la posición	Requieren transmisores y receptores que influyen en el consumo de energía.
PDR		Basado en dispositivos	Estimación de la posición actual y deducción del movimiento	Comúnmente se usan acelerómetros y giroscopios	Mientras más uso de sensores mejor resultado
UWB		Señales RF	Funciona emitiendo pulsos muy cortos	Detección de pulsos para generar rutas	Falta de soporte en casi todos los teléfonos inteligentes

Wi-Fi	Basado en Wifi	Funcionan con anchos de canal típicos 20, 40, 80.	Detectan la intensidad de la señal para localizar el dispositivo	Sufren de interferencias con las señales del entorno
BLE	Basado en Bluetooth	Uso del salto de frecuencia para comunicarse	Emisión de señal para ser detectado por dispositivos	Presentan mejor precisión y menor interferencia dependiendo el uso
RFID	Señales RF	Etiquetas electrónicas que almacenan datos	Los receptores obtienen los datos a través de RF	Se requiere el despliegue de barias etiquetas para mejorar la precisión

Fuente: Investigador basado en [18, 22, 23, 24, 25].

1.3.6. Bluetooth de baja energía (BLE)

Las aplicaciones impulsadas por la tecnología basada en Bluetooth han demostrado su valía en los últimos tiempos. Desde los conocimientos de la Industria 4.0 hasta la búsqueda y el seguimiento sencillos, estas soluciones dependen de una pieza de hardware simple pero versátil que forma la columna vertebral de las redes que ofrecen una lista muy larga de beneficios comerciales y para el consumidor: balizas [38].

Una baliza (o una etiqueta) es un pequeño dispositivo inalámbrico que funciona basado en Bluetooth Low Energy. Es como un faro: transmite repetidamente una señal constante que otros dispositivos pueden ver. Sin embargo, en lugar de emitir luz visible, transmite una señal de radio que se compone de una combinación de letras y números transmitidos en intervalos cortos y regulares. Un dispositivo equipado con Bluetooth, como un teléfono inteligente, una puerta de enlace o un punto de acceso, puede "ver" una baliza una vez que está dentro del alcance, al igual que los marineros que buscan un faro para saber dónde están.

La información que emiten las balizas se denomina paquete publicitario. El paquete publicitario contiene información básica sobre el dispositivo (su ID, el firmware con el que se ejecuta y el nivel de la batería), así como su potencia de transmisión, RSSI (Indicador de intensidad de señal recibida) calibrado a 1 metro y, según el proveedor y el dispositivo, otros bits de datos, como lecturas de sensores [39].

En algunos casos, como en la búsqueda de rutas, las balizas pueden transmitir los datos de ubicación, pero esto no se usa con frecuencia. Los datos de ubicación de la baliza se calculan en base a tres factores combinados:

Tabla 3 Factores que influyen en los datos de ubicación de las balizas

Potencia de transmisión	RSSI de referencia	RSSI real
la potencia a la que emite la señal	la fuerza que alcanza la señal a esta potencia de transmisión establecida en el rango de un metro de la baliza	la intensidad de la señal en el lugar donde la señal fue captada por un receptor

Fuente: Investigador basado en [38].

Los datos de la baliza son bastante inútiles si no hay un dispositivo que pueda capturarlos y ningún software que pueda interpretarlos. Por lo que cualquier proyecto basado en balizas requiere al menos dos componentes más: un receptor y una aplicación.

Un receptor es un dispositivo habilitado para Bluetooth que "escucha" las señales de baliza en su rango y pasa toda la información que capta a la aplicación. La aplicación calcula los datos y los traduce a lo que fue programado para mostrar [35].

Balizas vs etiquetas

Las balizas son estacionarias. Las etiquetas están destinadas a estar en movimiento.

Independientemente de si se trata de una baliza o una etiqueta, un dispositivo Bluetooth Low Energy funcionará igual. Esto significa que técnicamente se puede comprar un producto de baliza y, siempre que cumpla con sus requisitos, usarlo como una etiqueta o viceversa. Los dos términos no se utilizan para describir dos tecnologías diferentes, sino dos aplicaciones de una tecnología [38].

Las implementaciones estacionarias (basadas en balizas) y en movimiento (basadas en etiquetas) funcionan de manera un poco diferente, por lo que distinguir los nombres de los dispositivos ayuda a evitar confusiones cuando se habla de diferentes casos de uso.

Si desea pegar los dispositivos Bluetooth a paredes u objetos fijos, se usará balizas.

Los casos de uso tradicionales para esta configuración son: notificaciones de proximidad, orientación en interiores, programas de fidelización y otras aplicaciones que se puede poner bajo un término general, "experiencia del cliente". En este caso, el receptor de la señal de la baliza suele ser el teléfono móvil del cliente. Cuando esté dentro del alcance de una baliza, captará la señal y la transmitirá a la aplicación instalada en ella. La aplicación leerá los datos de la baliza, verá qué acción se asigna a estos datos y realizará la acción. La acción puede ser simple: mostrar una notificación sobre las ventas del producto al lado del usuario o mostrar contenido en la aplicación con una descripción detallada de una exhibición cercana en un museo. Pero también puede ser más complejo, como calcular la posición del usuario en función de su proximidad a un par de balizas dentro del alcance y mostrar su ubicación como un punto azul en un mapa interactivo [37].

Si se desea que personas o activos lleven los dispositivos para monitorear sus movimientos, se usará etiquetas.

Cuando los activos están en movimiento, el receptor debe estar quieto. Entonces, mientras que en el escenario anterior el receptor y la aplicación podrían funcionar en el mismo dispositivo, aquí se necesita un dispositivo adicional. Por lo general, este dispositivo es una puerta de enlace o un punto de acceso y su trabajo es transmitir los datos recopilados a la nube o al servidor local. La aplicación, por ejemplo, un RTLS (Sistema de ubicación en tiempo real), toma los datos y los traduce en cosas como mapas en tiempo real que muestran las ubicaciones de los empleados, herramientas de búsqueda y búsqueda para ubicar activos o personas, paneles de productividad, análisis de temperatura y otras funciones. fue diseñado para proporcionar [38].

Estándares para balizas

Las balizas (y etiquetas) transmiten información alfanumérica simple. La forma en que se estructura esta información depende del estándar (a veces llamado protocolo) que utiliza la baliza. Hay un par de estándares de comunicación para dispositivos Bluetooth LE, pero los dos más comunes son iBeacon (de Apple) y Eddystone (de Google) [40].

- **iBeacon**

El perfil iBeacon es el primer protocolo de comunicación, y actualmente el más discutido. No es un hardware físico, sino el lenguaje utilizado para impulsar la

tecnología física de "baliza". Desarrollado por Apple, es compatible de forma nativa con iOS y tiene integraciones profundas con el sistema operativo móvil. Aunque el perfil iBeacon funciona en otros sistemas operativos móviles, funciona mejor en el entorno para el que fue diseñado: iPhone y iPads [40].

La señal de iBeacon contiene tres piezas principales de información.

Tabla 4 Piezas principales para las señales de iBeacon

Identificador Universal Único (UUID)	La información más general de la baliza. Por ejemplo: La baliza A pertenece a la persona #1.
Major	La información más general de la baliza. Por ejemplo: La baliza A se encuentra en la tienda #8.
Minor	Información más minuciosa. Por ejemplo: Esta es la baliza en el pasillo 2.

Fuente: Investigador basado en [38].

- **Eddystone**

El formato Eddystone es un protocolo de comunicación nuevo y abierto desarrollado por Google pensado para usuarios de Android [38].

A menudo se compara las balizas con un faro; son solo objetos simples que envían constantemente una señal. El protocolo ahora es conocido por ser “abierto”, creado con el aporte y la colaboración de varias empresas. En lugar de crearse para potenciar aplicaciones orientadas al usuario altamente específicas, sus cualidades clave son la interoperabilidad y la solidez a largo plazo [40].

Eddystone envía 4 paquetes:

Tabla 5 Lista de paquetes que envía el protocolo Eddystone

UID	Es un ID estático único (similar al UUID, Majors y Minors) con dos partes: espacio de nombres e instancia.
URL	Incluye una URL comprimida que el usuario final puede usar directamente.
TLM	Es otro paquete que no se encuentra en iBeacon. Contiene datos de telemetría que son excelentes para fines de gestión de flotas.
EID	Es una medida de seguridad adicional.

Fuente: Investigador basado en [38].

1.3.7. Teléfonos inteligentes

Un teléfono inteligente es un teléfono móvil basado en un sistema operativo móvil, con una capacidad informática y una conectividad más avanzadas que un teléfono básico. Los primeros teléfonos inteligentes eran una combinación funcional de un asistente digital personal (PDA) y un teléfono móvil. Algunas funciones se agregaron en modelos posteriores, como reproductores multimedia portátiles, cámaras digitales compactas de gama baja, cámaras de video de bolsillo y unidades de navegación GPS para formar un dispositivo multiuso, pantallas táctiles de alta resolución y navegadores web para mostrar sitios web estándar y dispositivos móviles. páginas optimizadas. Además, Wi-Fi proporcionó acceso a datos de alta velocidad y banda ancha móvil [41].

Los sistemas operativos móviles (SO) más habituales utilizados por los teléfonos inteligentes son Android de Google y iOS de Apple. Dichos sistemas operativos pueden adaptarse a muchos modelos de teléfonos diferentes [41].

1.3.7.1. Principales Sistemas Operativos

Tabla 6 Principales Sistemas Operativos

Android	iOS
Es un sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con pantalla táctil, desarrollado por Google en conjunto con Open Handset Alliance. Google lanza el código de Android como código abierto, bajo la Licencia Apache. Las aplicaciones de Android se programan en Java utilizando las propias bibliotecas de Android	iOS (anteriormente iPhone OS) es un sistema operativo móvil desarrollado y distribuido por Apple Inc. Lanzado originalmente en 2007 para iPhone y iPod Touch, se ha ampliado para admitir otros dispositivos Apple como iPad y Apple TV. A diferencia de Windows CE (Windows Phone) de Microsoft y Android de Google, Apple no otorga licencias de iOS para la instalación en hardware que no sea de Apple.

Fuente: Investigador basado en [42].

1.3.8. Asistentes Virtuales

Un asistente virtual es un agente personal basado en software que ayuda a realizar sencillas tareas diarias. Sus funciones son, en muchos sentidos, similares a lo que

puede hacer un asistente humano personal: verificar reservas, configurar alarmas, hacer llamadas, escribir mensajes, automatizar flujos de trabajo, programar citas, dar recordatorios y dar direcciones. Ofrecen asistencia personalizada y resolución de problemas de una manera que parece humana. Los asistentes populares incluyen Amazon Alexa, Cortana de Microsoft, Apple Siri y Google Assistant. De hecho, Alexa está equipada con funciones que ayudan en el desarrollo de chatbots basados en voz [43].

1.3.8.1. ChatBots

Un chatbot (asistente virtual) describe un software de computadora capaz de interactuar con humanos. En las organizaciones modernas, algunas tareas realmente se pueden asignar a las computadoras. Dentro de la Ingeniería de Sistemas, algunos de los procesos o actividades pueden concebirse como automatizables, como por ejemplo consultar información en el proceso de gestión de la Información, o ayudar en el análisis de compensación, apoyando decisiones en el proceso de gestión de decisiones. La automatización también se puede aplicar a procesos más orientados a la ingeniería, como seleccionar los mejores requisitos desde un enfoque de ingeniería de línea de productos, etc.

Los chatbots son programas inteligentes que interactúan con los usuarios en conversaciones similares a las humanas a través de medios textuales o auditivos. Algunas funciones realizadas por los chatbots incluyen ofrecer instrucciones a los clientes para completar una tarea, recopilar y compartir información como productos o detalles de contacto de los clientes, programar citas desde un calendario e integrarse con una base de conocimientos o un portal de autoservicio para brindar soporte en tiempo real [43].

1.3.8.2. Asistentes de voz (VA)

Los asistentes de voz (VA) son una combinación de tecnologías que incluyen procesamiento de lenguaje natural (NLP), reconocimiento de voz y síntesis de voz para comunicarse con sus usuarios usando lenguaje natural.

Las principales diferencias que tienen estos agentes radican en la forma en que interactuamos con ellos. Por ejemplo, los chatbots son un asistente virtual basado en

texto que simula conversaciones similares a las de los humanos con los usuarios. Por otro lado, los asistentes de voz son asistentes virtuales que utilizan el habla natural para resolver consultas e interactuar con los usuarios [44].

1.3.9. Aplicaciones móviles

Una aplicación móvil es esencialmente una pieza de software. Hoy en día, hay una gran cantidad de aplicaciones en el mercado, que vienen en todas las formas, tamaños y colores imaginables.

Las herramientas de desarrollo de aplicaciones también están disponibles para diseñadores y programadores, lo que hace que sea más fácil producir y lanzar una aplicación ahora que nunca [45].

1.3.9.1. Tipos de aplicaciones

A nivel de programación existen varias formas de desarrollar una app. Cada uno tiene diferentes características y limitaciones, especialmente desde el punto de vista tecnológico.

Tabla 7 Tipos de Aplicaciones Móviles

Nativas	Son aquellas que han sido desarrolladas con el software que ofrece cada sistema operativo, genéricamente denominadas Software Development Kits (SDK). Android, iOS y Windows Phone cuentan con diferentes SDK y aplicaciones nativas diseñadas y programadas específicamente para cada plataforma en sus respectivos lenguajes de SDK.
Web	Se basa en HTML, JavaScript y CSS, herramientas que ya son familiares para los programadores web. En tales casos, no se utilizan SDK, lo que permite programar independientemente del sistema operativo.
Híbridas	Es una combinación de las anteriores. El desarrollo es similar al de una aplicación web (con HTML, CSS y JavaScript), y una vez que la aplicación está terminada, se compila y empaqueta de tal manera que el resultado final se parece a las aplicaciones nativas.

Fuente: Investigador basado en [45].

Aplicaciones multiplataforma

Las aplicaciones multiplataforma combinan lo mejor de dos mundos: estas son aplicaciones reales que se instalan en un teléfono inteligente y en la mayoría de los casos funcionan como si se tratase de una aplicación nativa. Al mismo tiempo, no es

necesario contratar dos equipos de desarrollo separado, ya que las aplicaciones de IOS Y Android se construyen exactamente con el mismo conjunto de herramientas uno de los varios frameworks de JavaScript o el de Flutter desarrollado por Google [46].

Las aplicaciones multiplataforma tienen como ventajas principales las siguientes:

- **Alto rendimiento:** muestran mayor rendimiento que las aplicaciones basadas en la web y son capaces de cumplir con casi cualquier requisito.
- **Desarrollo rentable y rápido:** En comparación con una aplicación nativa, las aplicaciones multiplataforma tienen un tiempo de desarrollo significativamente más corto y menor costo.
- **Aspecto y sensación nativos:** En la mayoría de los casos, los usuarios no pueden reconocer una aplicación multiplataforma, ya que utiliza el hardware del teléfono inteligente y los controladores nativos, al igual que las aplicaciones nativas.

Las aplicaciones multiplataforma siguen siendo de gran interés para muchas empresas, ya que las aplicaciones nativas para iOS y Android son más caras de desarrollar y mantener. Además, el desarrollo multiplataforma no es tan complejo como el nativo y también es un punto importante para los negocios.

1.3.9.2.Lenguaje de programación Dart

Dart es un lenguaje de programación dinámico, basado en clases, orientado a objetos con cierre y alcance léxico. Sintácticamente, es bastante similar a Java, C y JavaScript.

Dart es un lenguaje de programación de código abierto que se usa ampliamente para desarrollar la aplicación móvil, las aplicaciones web modernas, la aplicación de escritorio y el Internet de las cosas (IoT) utilizando el marco Flutter. También admite algunos conceptos avanzados, como interfaces, mixins, clases abstractas, genéricos de campo e interfaz de tipo [47].

Características principales del lenguaje de programación Dart:

- Dart es un lenguaje independiente de la plataforma y es compatible con todos los sistemas operativos como Windows, Mac, Linux, etc.

- Es un lenguaje de código abierto, lo que significa que está disponible de forma gratuita para todos. Viene con una licencia BSD y reconocida por el estándar ECMA.
- Es un lenguaje de programación orientado a objetos y admite todas las características de oops, como herencia, interfaces y características de tipo opcional.
- Dart es muy útil en la construcción de aplicaciones en tiempo real debido a su estabilidad.
- Dart viene con el compilador dar2js que transmite el código Dart al código JavaScript que se ejecuta en todos los navegadores web modernos.
- La máquina virtual Dart independiente permite que el código Dart se ejecute en un entorno de interfaz de línea de comandos.

1.3.9.3. Backend

El backend es el cerebro de una aplicación móvil. Entre otras cosas, el backend se encarga del procesamiento, el almacenamiento y la seguridad de los datos. Opera en el servidor, y es esa parte de la aplicación se ve, pero la aplicación móvil depende de ella para su funcionalidad [35].

El backend en una aplicación móvil se encarga de:

- Procesamiento y almacenamiento de datos independiente de las capacidades de un teléfono inteligente
- Sincronización y uso compartido de datos entre múltiples dispositivos y plataformas
- Actualizaciones de contenido dentro de la aplicación móvil
- Gestión de la lógica de negocio de la aplicación
- Autorización y autenticación que controlan el acceso a los datos

Node.js, Java y Python son opciones populares para construir backend, cada uno con sus propias fortalezas y debilidades.

Node.js es conocido por su rápido rendimiento y capacidad para manejar sitios web de alto tráfico. Utiliza JavaScript, que es un lenguaje ampliamente utilizado, y tiene una gran comunidad de apoyo [36].

Java es conocido por su estabilidad y escalabilidad. Las grandes empresas lo utilizan para crear aplicaciones de misión crítica y tiene un rico ecosistema de bibliotecas y herramientas [37].

Python es conocido por su simplicidad y facilidad de uso. Es una excelente opción para la creación rápida de prototipos y tiene una gran cantidad de bibliotecas para el aprendizaje automático, la computación científica y el análisis de datos [38].

Tabla 8 Comparación de las opciones más utilizadas para construir backends

Parámetros	**Java** 	**Python** 	**Node.js** 
Velocidad	Muy rápida	Rápida	Mas rápida
Actuación	Alta	Alta	Baja
Escalabilidad	Alta	Media	Mas alta
Simplicidad	Simple	Muy simple	Media
Comunidad	Fuerte	Fuerte	Fuerte
Librerías	Buena	Buena	Excelente
Costo	Pagado	Gratis	Gratis
Cross-function ality	Alto	Alto	Alto

Fuente: Investigador basado en [36].

1.4. Objetivos

1.4.1. Objetivo General

Implementar un prototipo de asistente virtual de navegación y generación de rutas en espacios cerrados.

1.4.2. Objetivos Específicos

- Analizar las tecnologías de navegación y métodos de generación de rutas en interiores.
- Implementar un sistema inalámbrico de localización en espacios cerrados.
- Diseñar un asistente virtual a través de una aplicación móvil para trazar rutas de navegación en interiores.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

Para el desarrollo del presente proyecto se usó tres dispositivos bluetooth de baja energía (BLE) de la marca accent systems en su modelo iBKS105 los cuales esta destinados a cumplir la función de faros, para reconocer estas señales se hace uso de teléfonos inteligentes con sistema operativo Android 5+ que tengan bluetooth y la aplicación móvil integrada.

Para el desarrollo de la aplicación móvil se utilizó el IDE de Android Studio como entorno de desarrollo y Flutter como tecnología de desarrollo el cual es un SDK multiplataforma de código abierto que utiliza lenguaje de programación Dart que está destinado para el desarrollo de aplicaciones móviles o en la web. Para la parte del backend de la aplicación móvil se utilizó Node.js así como SQL server para el alojamiento y gestión de las bases de datos, además se hace uso de Firebase para facilitar la carga de archivos en la aplicación llamándolos desde la nube.

Posteriormente para la configuración he integración del asistente virtual que va integrado en la aplicación móvil se hace uso de la plataforma web de Alan IA la cual es un complemento de Flutter para la creación de asistentes virtuales comandados por voz, la configuración se la realiza en la web y es enviada a la aplicación a través de un toque único generado por la misma.

2.2. Métodos

2.2.1. Modalidad de Investigación

El presente trabajo investigativo tiene la característica de ser un proyecto de investigación aplicada, porque se buscó dar solución al problema planteado a través de la puesta en práctica de conocimientos teóricos, para llegar al diseño del prototipo.

Se realizó una investigación bibliográfica con el propósito de obtener conocimientos de las tecnologías de navegación en interiores y los métodos de generación de rutas. Con este fin se recolectó fuentes documentadas como revistas, artículos o trabajos de investigación dentro de repositorios universitarios.

El proyecto es una investigación de campo, debido a que se recopiló información y se implementó el asistente virtual dentro de un espacio cerrado en donde radica el problema de navegación.

2.2.2. Procesamiento y Análisis de Datos

Para el procesamiento y análisis de datos se realizó teniendo en cuenta los siguientes pasos:

- Indagar en fuentes de información.
- Organizar y estructurar la información encontrada.
- Analizar posibles soluciones que permitan desarrollar la mejor estrategia para la problemática.
- Planteamiento de la propuesta de solución.
- Verificar que el prototipo de aplicación móvil este completamente funcional y brinde una solución a la navegación en espacios cerrados.

2.2.3. Propuesta de Solución

El crecimiento poblacional y de infraestructura supone nuevos retos al momento de generar una mejor experiencia y optimizar tiempo de las personas al visitar un edificio o espacio cerrado. Se plantea desarrollar un sistema de navegación en interiores utilizando tecnologías de localización indoor, las cuales por medio de una red inalámbrica permite localizar y gestionar datos de activos o personas dentro de un espacio delimitado, esto acoplado a un asistente virtual a través de un dispositivo móvil, el cual haciendo uso de algoritmos de búsqueda, genera y sugiere rutas óptimas para llegar a un punto de destino con mayor facilidad de manera interactiva y mostrando información del lugar, considerando la ubicación actual del usuario dentro del mapa del edificio, optimizando su tiempo y generando una mejor experiencia al momento de visitar las instalaciones que cuenten con dicho sistema.

2.2.4. Desarrollo del Proyecto

Para el desarrollo de proyecto de asistente virtual de navegación y generación de rutas en espacios cerrados, se realizó las siguientes actividades:

Tabla 9 Actividades para el desarrollo del proyecto

Tema: Asistente virtual de navegación y generación de rutas en espacios cerrados.	
Objetivo General	
Implementar un prototipo de asistente virtual de navegación y generación de rutas en espacios cerrados.	
Objetivo Específico 1	
Analizar las tecnologías de navegación y métodos de generación de rutas en interiores.	
1	Análisis de las tecnologías de navegación actuales dentro de espacios cerrados.
2	Descripción de los métodos para la generación de rutas.
3	Análisis de los tipos de asistentes virtuales existentes.
4	Análisis del software que permita la creación de un asistente virtual.
Objetivo Específico 2	
Implementar un sistema inalámbrico de localización en espacios cerrados.	
1	Selección del hardware y software necesario para la navegación en espacios cerrados.
2	Diseño de planos del espacio donde se realizará el prototipo.
3	Programación de los dispositivos electrónicos necesarios.
4	Diseño de un sistema inalámbrico de localización.
Objetivo Específico 3	
Diseñar un asistente virtual a través de una aplicación móvil para trazar rutas de navegación en interiores.	
1	Diseño de la aplicación móvil con asistente virtual.
2	Pruebas de funcionamiento del asistente virtual para navegación dentro de un espacio cerrado mediante la generación de rutas.
3	Análisis de resultados obtenidos tras las pruebas de funcionamiento del asistente.
4	Elaboración del informe final.

Fuente: Investigador

2.2.1. Recolección de Información

La técnica aplicada para llevar a cabo el desarrollo del proyecto será la lectura de libros, revistas, fuentes online y proyectos relacionados al tema de investigación, además, la guía del profesor tutor, por lo que se tomará en cuenta bases de datos confiables que permitan el desarrollo del proyecto.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de los resultados

El desarrollo e implementación de la aplicación con asistente virtual de navegación y generación de rutas en espacios cerrados es una solución de sistema de posicionamiento interior (IPS) que se basa en teléfonos inteligentes apoyándose en la tecnología Bluetooth de baja energía (BLE). Esta aplicación móvil permite al usuario la opción de navegar de manera interactiva acompañado por un asistente virtual activado por voz, dentro de un edificio o espacio en el cual la tecnología de posicionamiento global GPS no tiene acceso, permitiendo resolver problemas de desubicación y falta de información del sitio en el que se encuentra, a través del mapa que muestra la ruta desde el lugar de inicio hasta el lugar de destino ingresado por el usuario, mejorando la experiencia de visitar el edificio y generando buenas referencias al mismo. De igual manera el usuario administrador tiene acceso a información relevante de los lugares más visitados, dando la opción de hacer uso de esta para tomar acciones o aplicar estrategias dependiendo el caso.

3.2. Desarrollo de la propuesta

El desarrollo del proyecto se llevó a cabo mediante la revisión de información y datos obtenidos por diferentes autores, consultados en las principales bases de datos nacionales e internacionales, así como, repositorios, bibliotecas virtuales y revistas de prestigio para la comunidad. El análisis de investigaciones que abordan temas de solución para sistemas de posicionamiento interior permite fundamentar el desarrollo del prototipo de asistente virtual para navegación y generación de rutas en espacios cerrados cumpliendo con el objetivo y por medio de los datos obtenidos se puede comprobar la eficiencia del programa.

3.2.1. Requerimientos del sistema

El sistema está desarrollado en el primer piso de una casa cuyas dimensiones son 28m de largo y 9m de ancho, que cuenta con: la entrada principal, jardines, estudio, sala, comedor, cocina, bodega, taller, garaje y baños, estos han sido tomados como sitios de

interés para el usuario y de los cuales se mostrará la información de las rutas para desplazarse de un lugar a otro, véase en el Anexo 2.

El proyecto de investigación tiene como finalidad la implementación de una solución de sistema de posicionamiento interior la cual usa una aplicación móvil y un asistente virtual para mostrar la ruta que debe seguir el usuario según lo requiera.

3.2.1. Descripción del método para la generación de rutas.

El trabajo investigativo tiene como finalidad la implementación de una solución de sistema de posicionamiento interior la cual a través de una aplicación móvil y un asistente virtual sea capaz de presentar al usuario rutas dentro de un espacio cerrado, para llegar a dicho objetivo fue importante conocer las técnicas y los métodos de IPS con la finalidad de compararlos y optar por la mejor solución que se adapte al proyecto, mismas que se encuentran detallas en el Capítulo 1, de las cuales se seleccionó el método libre de rango basado en dispositivos como mejor opción ya que este presenta mayor independencia y versatilidad al momento de implementar un sistema de localización además se tratan de aplicaciones móviles compatibles con teléfonos inteligentes que permiten el uso de nuevas tecnologías y la aplicabilidad de opciones como las de un asistente virtual primordial para el desarrollo de este proyecto.

Como se muestra en la Tabla 1 las técnicas de posicionamiento comunes se basan en la hora de llegada (ToA), la diferencia horaria de llegada (TDoA), la intensidad de la señal recibida (RSS) y el ángulo de llegada (AoA). Al plantearse una solución basado en dispositivos la técnica RSS resulta la más atractiva debido a su respuesta positiva al usarse con aplicaciones móviles.

3.2.2. Análisis de las tecnologías de navegación actuales dentro de espacios cerrados.

En la actualidad, existen variedad de tecnologías de navegación propuestas las cuales se detallan en la Tabla 2, sin embargo, no todas se ajustan a los requerimientos del sistema, por esta razón se realizó un análisis de estas tecnologías considerando parámetros como: accesibilidad, precisión típica, costo y escalabilidad. Se utilizó esta comparación como base para elegir la tecnología más adecuada para satisfacer las necesidades y requisitos del sistema, a fin de lograr un funcionamiento óptimo.

Tabla 10 Comparación de las tecnologías de IPS con los parámetros a tomar en cuenta para el proyecto

Tecnología	Accesibilidad	Precisión típica	Costo	Escalabilidad
Luz	Requiere de actuadores para interactuar con dispositivos móviles	Depende de la técnica y la configuración (1-2 metros)	Costo de implementación elevado	Requiere reestructuración
Visión artificial	Todos los teléfonos inteligentes cuentan con cámara	Para la odometría, del 0.25% al 8.5% de la longitud del camino	Bajo costo	Fácil de escalar
Sonido	Requiere de sistemas externos para generar señales acústicas	Para ecografías y sonido audible (1-10 centímetros)	Costo de infraestructura elevado	Requiere reestructuración de todo el sistema
Campos magnéticos	Se necesita de generadores de campos magnéticos y calibración de los mismo	Para campos artificiales y campos naturales (1-5 metros)	Costo de infraestructura elevado	Requiere reestructuración de todo el sistema
PDR	La mayor parte de teléfonos inteligentes cuentan con sensores para determinar la posición	Para entornos específicos (2-5 metros)	Bajo costo	Fácil de escalar
UWB	Falta de compatibilidad con teléfonos inteligentes	Comúnmente oscila en 50 centímetros	Costo elevado al tratarse de espacios amplios	Fácil de escalar
Wi-Fi	Los teléfonos inteligentes traen implementada esta tecnología	Dependiendo el método (2-5 metros)	Bajo costo	Fácil de escalar
BLE	Fácil de instalar y compatibles con dispositivos móviles	Oscila en 2 metros	Bajo costo	Fácil de escalar
RFID	Compatibles con teléfonos inteligentes	Oscila entre 2 metros dependiendo la cantidad de etiquetas	Bajo costo, pero rendimiento limitado	Fácil de escalar

Fuente: Investigador

A partir de esta comparación, se seleccionó la tecnología BLE para el desarrollo del sistema debido a que es la solución más atractiva para IPS dado su costo, privacidad y una baja huella en la batería del teléfono inteligente y el tráfico de red. BLE se considera la tecnología de posicionamiento más adecuada para la navegación y el seguimiento en interiores actualmente. Su idoneidad está respaldada por el costo relativamente bajo de los emisores BLE, su muy bajo consumo de energía que les permite funcionar con baterías durante meses y una capacidad generalizada de los teléfonos inteligentes modernos para leer sus anuncios.

3.2.3. Análisis de tipos de aplicación móvil a desarrollar

Como se detalla en el marco teórico, existen 4 tipos de aplicaciones móviles: nativas, web, hibridas y multiplataforma, de las cuales las aplicaciones multiplataforma destacan por características como:

- Menor costo y tiempo de desarrollo: Se desarrollan en una sola base de código que se puede utilizar en múltiples plataformas, lo que reduce el costo y el tiempo de desarrollo en comparación con el desarrollo de aplicaciones nativas para cada plataforma individualmente.
- Mayor alcance de audiencia: Se pueden ejecutar en múltiples plataformas, lo que significa que tienen un alcance de audiencia más amplio en comparación con las aplicaciones nativas. Los usuarios de diferentes plataformas pueden utilizar la misma aplicación multiplataforma, lo que aumenta su base de usuarios.
- Mantenimiento fácil: Se desarrolla en una sola base de código, es más fácil de mantener y actualizar en comparación con las aplicaciones nativas para cada plataforma individual.
- Experiencia del usuario consistente: Ofrecen una experiencia de usuario consistente en múltiples plataformas, lo que significa que los usuarios no necesitan aprender una nueva interfaz de usuario cada vez que utilizan la aplicación en una plataforma diferente.
- Acceso a las funciones del dispositivo: Tienen acceso a las funciones del dispositivo como la cámara, la ubicación, el almacenamiento y otras funciones nativas a través de plugins, lo que permite a los desarrolladores crear aplicaciones con características avanzadas.

Estas características aportan escalabilidad y compatibilidad a la aplicación con funciones del dispositivo necesarias para el correcto funcionamiento del sistema como son el acceso al bluetooth, notificaciones, cámara, micrófono y almacenamiento.

Las aplicaciones multiplataforma se desarrollan utilizando frameworks o plataformas que permiten a los desarrolladores crear una única base de código que se puede utilizar en múltiples plataformas, por estas razones se comparan los 4 frameworks multiplataforma más usado y completos del mercado.

Tabla 11 Comparación de los frameworks multiplataforma más usados

Parámetros	**FLUTTER**	**REACT NATIVE**	**IONIC**	**XAMARIN**
Creadores	Google, 2017	Facebook, 2015	Drifty Co. 2011	Microsoft
Lenguaje	Dart	JavaScript	HTML5, CSS, JavaScript +TypeScript	C# con entorno .net
Interfaz de Usuario	Utiliza widgets patentados para una interfaz de usuario impresionante	Usa controladores UI nativos	CSS, HTML	Usa controladores UI nativos
Rendimiento	Excelente	Cercano al nativo	Moderado	iOS/Android: cercano al nativo Formas: Moderado
Hot Reload	Si	Si	No	Si
Documentación	Extensa y detallada, poco soporte aún	Mucha documentación, pero dependientes de terceros	Mucha documentación y soporte	Mucha documentación
Plataformas	Android, iOS 8+, Mac, Windows, Linux, Web	Android 4.1+, iOS 8+	Android 4.4+, iOS 8+, Web, PWA	Android 4.0.3+, iOS 8+, Windows 10

Reutilización de código	50-90% código reutilizable	90% código reutilizable	98% código reutilizable	90% código reutilizable
Aplicación conocida	GoogleAds, The New York Times, eBay	Instagram, Facebook, Airbnb, UberEats	JustWatch, Pacifica and Nationwide	Storyo, Olo, The World Bank
Costo	Código abierto	Código abierto	Planes de código abierto + pagos	Planes de código abierto + pagos

Fuente: Investigador

Se seleccionó el framework de Flutter para el desarrollo de la aplicación móvil debido a su alto rendimiento, amplia personalización, compatibilidad con dispositivos BLE, documentación y soporte extenso además de ser una multiplataforma de código abierto.

3.2.4. Descripción general del sistema

Para llevar a cabo este proyecto se definió un esquema general el cual describe las etapas que intervienen en la estructura del sistema.

- **Usuarios:** La aplicación cuenta con dos tipos de usuarios: el administrador y los visitantes, el usuario administrador se diferencia al tener un apartado de reportes que muestra información relevante para la gestión del lugar.
- **Interfaz:** hace referencia al software con el que interactúan los usuarios, construidos a través del backend y frontend.
- **Datos:** Representa el almacenamiento de los datos en la base de datos local, usando métodos get y post para el envío y recepción de estos.
- **Posicionamiento:** Hace referencia al hardware que usa el sistema para el posicionamiento en espacios cerrados
- **Multimedia:** En esta sección se encuentran alojados los archivos multimedia necesarios para la aplicación móvil.
- **Api:** Conforman las funciones y procesamientos del asistente virtual.

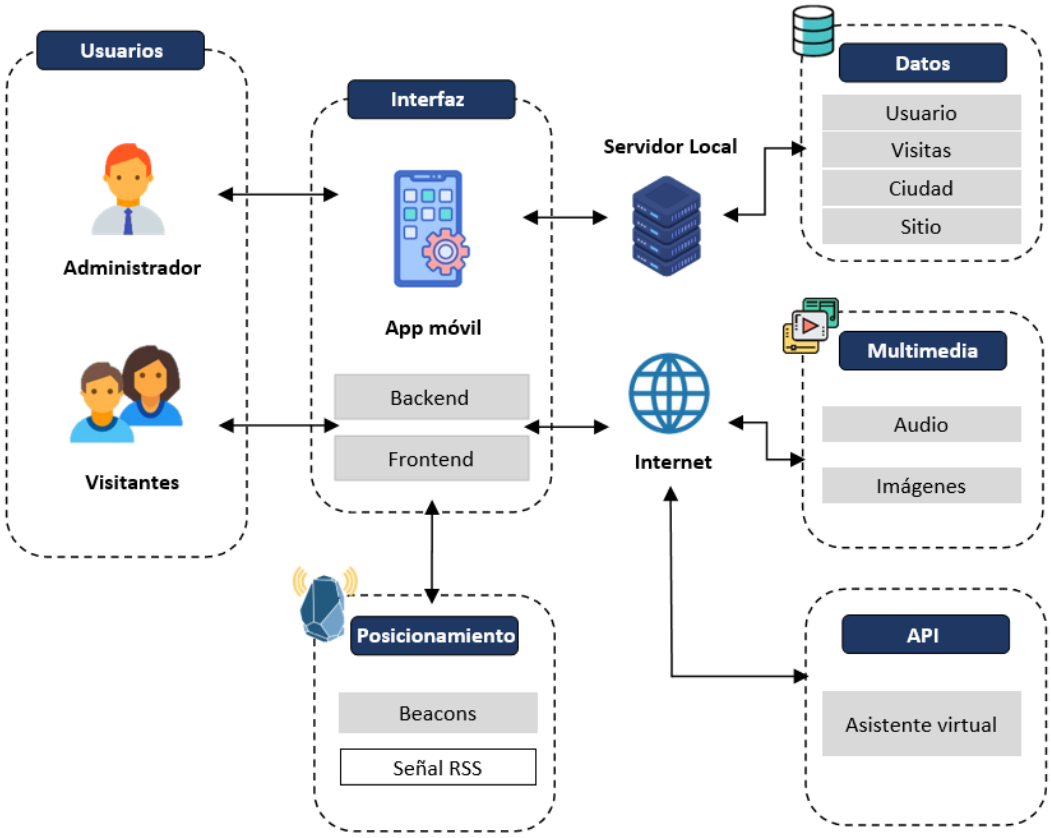


Figura 2 Esquema general del sistema

Fuente: Investigador

3.2.1. Selección del modelo de Beacons

Para la selección del dispositivo Bluetooth de baja energía se realizó un estudio de consideraciones a tomar en cuenta sobre el factor y forma basado en la aplicación que se le va a dar, el espacio al que va a estar expuesto y el lugar en donde van a estar ubicados

Tabla 12 Factor de forma del dispositivo Bluetooth de baja energía

Factor de Forma	
Tamaño y Forma	El tamaño del dispositivo debe ser armónico con el lugar, por lo general, son dispositivos pequeños que se camuflan en el ambiente, al necesitarse de dispositivos que se mantengan estáticos su forma no es tan relevante.
Método de montaje/aplicación	Al tratarse de dispositivos livianos y su implementación es una superficie plana, basta con unirlos con cinta adhesiva para un fácil montaje.
Seguridad e Integridad Física	Al tratarse de espacios cerrados y los dispositivos se encontrarán fijos no están propensos a golpes ni caídas sin

	embargo requieren un case que los proteja del polvo y posibles salpicaduras
Baterías reemplazables	Se requiere de un dispositivo que use baterías reemplazables para mayor duración del sistema.
Diseño y personalización	Equipos pequeños que no irrumpan en la estética del ambiente.

Fuente: Investigador

Después de descartar dispositivos Bluetooth que no cumplen con las consideraciones de factor de forma se cuenta con 3 candidatos idóneos para la aplicación en navegación en espacios cerrados. Por esta razón se realizó una comparación de dichos dispositivos con la finalidad de escoger el que mejor se adapte al objetivo del proyecto.

Tabla 13 Comparación de parámetros de los Beacons

Parámetros	**Estimote Beacons** 	**iBKS105** 	**Anchor Beacon 2** 
Dimensiones	Longitud: 62,5 mm Ancho: 46,4 mm Altura: 24,7 mm	Ø52,6 x 11,3 mm	Altura: 49 mm Ancho: 49 mm Profundidad: 15 mm
Peso	86 g	24 g	35 g
Núcleo	ARM Cortex M4	Nordic nRF51822	Nrf52832
Protocolo de radio	BLE 4.2	Bluetooth Low Energy	BLE 5.0
Rango de Distancia	Hasta 100 m	Hasta 50 m	Hasta 100 m
Batería	LR6 (tipo AA)	Coin Cell CR2477 3V – 1000mAh	ER14250 (1.2 Ah) 2400 mAh
Consumos de corriente en reposo	2.0 µA	2.4 µA	2.4 µA
Material de la caja	Silicona	ABS	P-550k
Acabado de la caja	Disponible en varios colores	Blanco Mate	Off White, RAL 9003
Método de fijación	cinta adhesiva de doble cara	cinta adhesiva de doble cara	cinta adhesiva de doble cara

Temperatura de funcionamiento	0°C hasta +60°C	-25°C hasta +60°C	-25°C hasta +85°C
Temperatura de almacenamiento	+15°C hasta +30°C	0°C hasta +35°C	-60°C hasta +85°C
Protocolos de baliza	iBeacon	iBeacon Eddystone: UID, URL, TLM & EID	iBeacon Eddystone Kontakt.io Telemetry
Actualización de firmware	OTA (Over The Air)	OTA (Over The Air)	OTA (Over The Air)
Certificaciones	Apple-certified	CE/FCC/IC/Anatel	CE/FCC/RoHS/ REACH/IC/UKCA
Costo	$33	$24	$39

Fuente: Investigador basado en [50], [51], [52].

Para el desarrollo del prototipo se planteó una solución basada en aplicaciones móviles la cual se apoya en la tecnología Bluetooth de baja energía, la función principal de los Beacons en este proyecto es la de actuar como "faros" los cuales al ser reconocidos por el dispositivo móvil enviarán notificaciones para permitir al usuario ubicarse de mejor manera dentro del mapa. Al ser una tecnología de apoyo los Beacons iBKS105 se convierten en la mejor elección debido a su bajo costo y su compatibilidad con los diferentes protocolos de balizas, a diferencia de los otros modelos que cuentan con características adicionales que no serán aprovechadas en este proyecto.

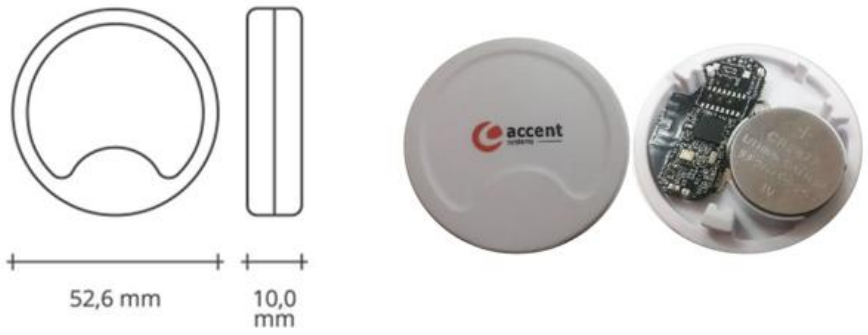


Figura 3 Beacon iBKS 105 [52]

3.2.2. Diseño de un sistema inalámbrico de localización.

El diseño e implementación de un sistema inalámbrico abarca dos aspectos fundamentares los cuales son la capacidad de la red y el área de cobertura, tras

considerar estos aspectos se determina el número de dispositivos necesarios así como la ubicación de estos.

En la implementación de una red de beacons, la capacidad de la red no es un factor crítico para considerar, ya que los beacons solo emiten señales Bluetooth de baja energía (BLE) a intervalos regulares trabajando en modo broadcast. A diferencia de una red Wi-Fi, por ejemplo, en la que muchos dispositivos se conectan simultáneamente y requieren una cantidad significativa de ancho de banda, los beacons solo transmiten pequeñas cantidades de datos de manera intermitente.

En lugar de centrarse en la capacidad de la red, el diseño de una red de beacons se enfoca en la cobertura y la ubicación. Se busca ubicar los beacons en lugares estratégicos que permitan la detección de la posición de los usuarios que tengan dispositivos móviles con capacidad para recibir señales BLE, lo que les permitirá recibir información de contexto relevante a su ubicación.

Para la ubicación y cobertura de los beacons se tomó en cuenta varios factores importantes:

- Función de los Beacons en el proyecto: Como se mencionó anteriormente la función de los Beacons en este caso son la de "faros" que en conjunto con la aplicación móvil permite al usuario ubicar el punto en el mapa del faro más cercano. Con esta función se requiere menor cantidad de beacons que al tratarse de soluciones de triangulación que demandan mayor número de beacons.
- Geometría del espacio: La presencia de obstáculos, como paredes y pilares, puede afectar la propagación de la señal y, por lo tanto, influir en la ubicación de los beacons.
- Distancia de transmisión: La distancia de transmisión de los beacons es un factor importante para garantizar una cobertura efectiva. La distancia de transmisión depende de la potencia de transmisión del beacon, la frecuencia y el entorno físico.
- Interferencias: Las interferencias pueden afectar la calidad de la señal y la precisión de la ubicación. Las interferencias pueden ser causadas por otros dispositivos electrónicos en el área, como sistemas Wi-Fi y Bluetooth, así como por interferencias físicas

Tras considerar los puntos antes expuestos se determinó el uso 3 dispositivos distribuidos en la parte inferior, central y superior del lugar. Como se mencionó anteriormente la función de los Beacons en este caso son la de “faros” y para esto se han colocado 3 faros en lugares estratégicos de la casa. Estos lugares han sido elegidos debido a que son los puntos donde más sitios de interés tienen alrededor y sacar provecho para describir los lugares al activar el asistente virtual, además al estar ubicados en partes altas no se encuentran cerca de otros dispositivos electrónicos que puedan generar interferencias.

Tabla 14 Ubicación de los faros dentro del plano del lugar

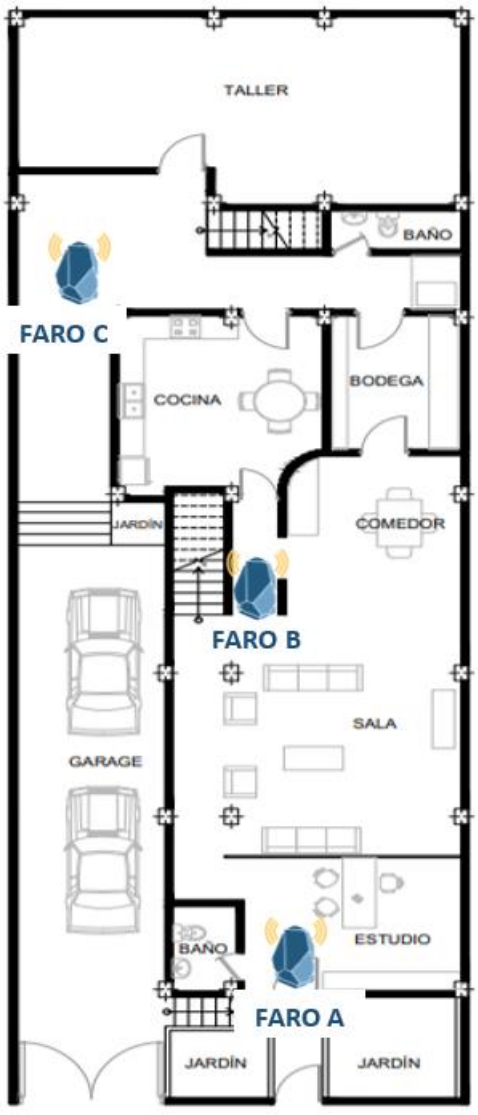


Figura 4 Plano general con la ubicación de los faros

FARO A

Figura 5 Ubicación del Faro A

FARO B

Figura 6 Ubicación del Faro B

FARO C

Figura 7 Ubicación del Faro C

Fuente: Investigador

Cálculo de la distancia en relación con la señal RSS

Se utilizó el indicador de potencia de señal recibida RSSI (Received Signal Strength Indicator) para calcular las distancias entre el transmisor y el receptor. Los valores de RSSI varían según la distancia entre ambos dispositivos, y es importante considerar las atenuaciones que la señal puede experimentar debido a que no se aplicará el modelo de propagación en un espacio libre.

El RSSI es influenciado por ciertos factores ambientales típicos de espacios interiores, debido a la naturaleza de las señales de radiofrecuencia. Estos factores pueden causar fluctuaciones en las mediciones de potencia.

La relación de la distancia con el parámetro RSSI se trata de un modelo semi empírico determinado en diversos estudios usando tecnologías WiFi, ZigBee y Bluetooth. Para este cálculo se usa la ecuación variante para RSSI:

$$RSSI = A - 10n \log d$$

Donde:

n : Exponente de atenuación calculado

d: Distancia entre el nodo transmisor y receptor (m)

A: RSSI entre el receptor a 1 metro de distancia del transmisor (dBm)

El exponente de atenuación para espacio libre es $n = 2$, sin embargo, en ambientes interiores copados de obstáculos la señal se ve afectada por reflexiones, difracciones y dispersiones, en consecuencia, se debe calcular este factor tomando como referencia un conjunto de mediciones realizadas dentro del entorno.

$$n = -\left(\frac{RSSI - A}{10 \log d}\right)$$

Los dispositivos BLE trabajan con una potencia de transmisión de +4 dBm, tras la medición de RSSI a una distancia de un metro se tienen un promedio de -65 dBm este será el valor de A.

Tabla 15 Cálculo del coeficiente de atenuación

Distancia (m)	RSSI promedio (dBm)	Coeficiente de atenuación (n)
2	-72.6	2.5246
3	-77.3	2.5779
4	-79.8	2.4582
5	-82.4	2.4893
6	-84.9	2.5573
7	-86.6	2.5559
8	-87.4	2.4803
9	-88.6	2.4731
10	-89.9	2.49
11	-92.4	2.6311
12	-94.5	2.7335
13	-96	2.7829

Fuente: Investigador

El promedio de los coeficientes de atenuación parciales da como resultado el coeficiente de atenuación general.

$$n_g = 2.5628$$

Y a través del despeje de esta fórmula se tiene que la distancia estimada entre nodos es:

$$d = 10^{\frac{-RSSI-A}{10n}}$$

$$d = 10^{\frac{-RSSI+65}{25.628}}$$

En donde tomando en cuenta que el máximo RSSI detectado es -96 se obtiene una distancia máxima de detección de hasta 16.20 metros, por lo cual para el desarrollo de este proyecto no presenta problemas al momento de la implementación.

3.2.1. Detección de las direcciones MAC de los módulos Beacons.

Los dispositivos Bluetooth de baja energía (Beacons) de la marca accent systems cuentan con un aplicativo móvil desarrollado por el fabricante el cual permite la gestión de estos. iBKS Config Tool se encuentra disponible en la play store y app store de donde se puede realizar la descarga gratuita.

Para el desarrollo del proyecto se usa la técnica RSS que se basa en la intensidad de señal recibida, esta intensidad disminuye a medida que aumenta la distancia al emisor, para que la aplicación móvil detecte el beacon se usó su dirección MAC como distintivo. Mediante la aplicación proporcionada por el fabricante se realizó el reconocimiento de la MAC de cada dispositivo y las pruebas de distancia para la ubicación de los Beacons dentro del espacio.

Figura 8 Detección de los Beacons a través de iBKS Config Tool
Fuente: Investigador

En la siguiente tabla se muestra el nombre del Faro con su respectiva dirección MAC

Tabla 16 Direcciones MAC de los faros a implementar

Nombre	MAC
FARO A	FD:15:BF:0C:99:87
FARO B	DB:F5:F5:FC:FC:CE
RARO C	E9:62:4B:5A:06:46

Fuente: Investigador

3.2.2. Diseño de la aplicación móvil con asistente virtual.

El desarrollo de la aplicación se realizó en base a la metodología scrum. La metodología Scrum es un enfoque ágil para el desarrollo de proyectos que se utiliza ampliamente en el desarrollo de software, incluyendo el de aplicaciones móviles. Scrum se basa en un enfoque iterativo e incremental en el que se divide el proyecto en pequeñas etapas llamadas sprints. Cada sprint es un ciclo de planificación, ejecución y revisión. Este método ayuda a mantener el enfoque en cumplir con los objetivos propuestos, las principales ventajas que brinda es la de realizar cambios, agregar nuevas características, realizar pruebas en cada etapa, usar la retroalimentación para mejorar aspectos o corregir problemas que se presentan a lo largo de la ejecución del proyecto, logrando así la creación de la aplicación móvil que satisface los requerimientos del usuario.

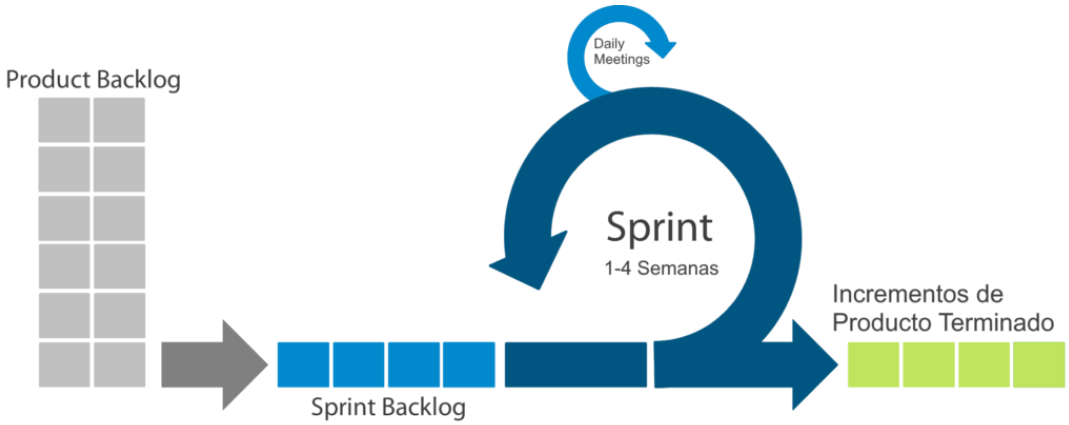


Figura 9 Metodología Scrum [53]

Es importante destacar que con esta metodología se busca la flexibilidad y la capacidad de adaptarse a los cambios, permitiendo trabajar de manera ágil. Es importante también tener en cuenta el trabajo de manera constante y transparente, con el objetivo de asegurar el éxito del proyecto.

Normativa de usabilidad

Para el desarrollo de la aplicación móvil se tomó en base la norma ISO 9241-11 que estables los principios ergonómicos para el diseño de la interacción persona-ordenador (IPO). Esta norma se aplica a todo tipo de sistemas interactivos, incluyendo aplicaciones móviles.

Al desarrollar la aplicación móvil, la norma ISO 9241-11 es útil para garantizar que la interfaz de usuario sea intuitiva y fácil de usar. La norma proporciona una guía para el diseño de la IPO y se enfoca en la usabilidad, la eficacia y la satisfacción del usuario.

Algunos de los principios de la norma ISO 9241-11 que se tomaron en cuenta para la aplicación móvil son:

- Adecuación para la tarea: La aplicación móvil debe estar diseñada para satisfacer las necesidades del usuario y ser adecuada para la tarea que se realiza.
- Aprendizaje: La aplicación móvil debe ser fácil de aprender y utilizar, incluso para los nuevos usuarios.
- Flexibilidad y eficiencia de uso: La aplicación móvil debe ser flexible y permitir que los usuarios realicen las tareas de diferentes maneras. Además, debe ser eficiente para que los usuarios puedan realizar las tareas de forma rápida.
- Compatibilidad: La aplicación móvil debe ser compatible con diferentes dispositivos y plataformas, para que los usuarios puedan utilizarla en cualquier dispositivo que prefieran.
- Estética y diseño minimalista: La aplicación móvil debe tener un diseño minimalista y estético, para evitar que los usuarios se distraigan con elementos innecesarios.

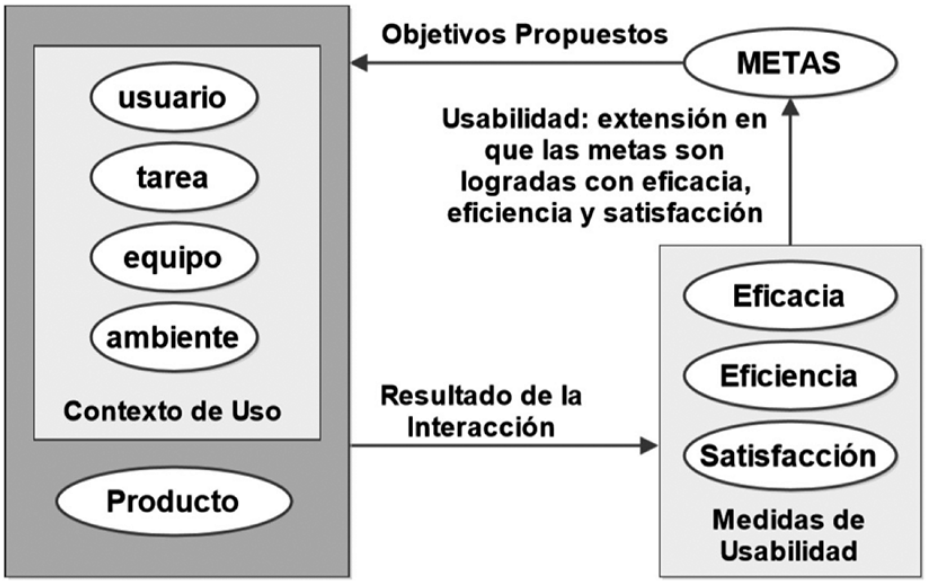


Figura 10 Marco de definición de usabilidad en la ISO/IEC 9241-11 [46]

3.2.2.1.Arquitectura de la aplicación móvil

En la arquitectura propuesta, tanto los usuarios normales como el usuario administrador tendrán acceso a la aplicación móvil, misma que estará conectada al servidor local para extraer los datos relevantes para el inicio de sesión y carga del plano general, de igual forma mediante internet se extrae la información alojada en la base de datos Firebase como audios e imágenes importantes para el funcionamiento del aplicativo móvil.

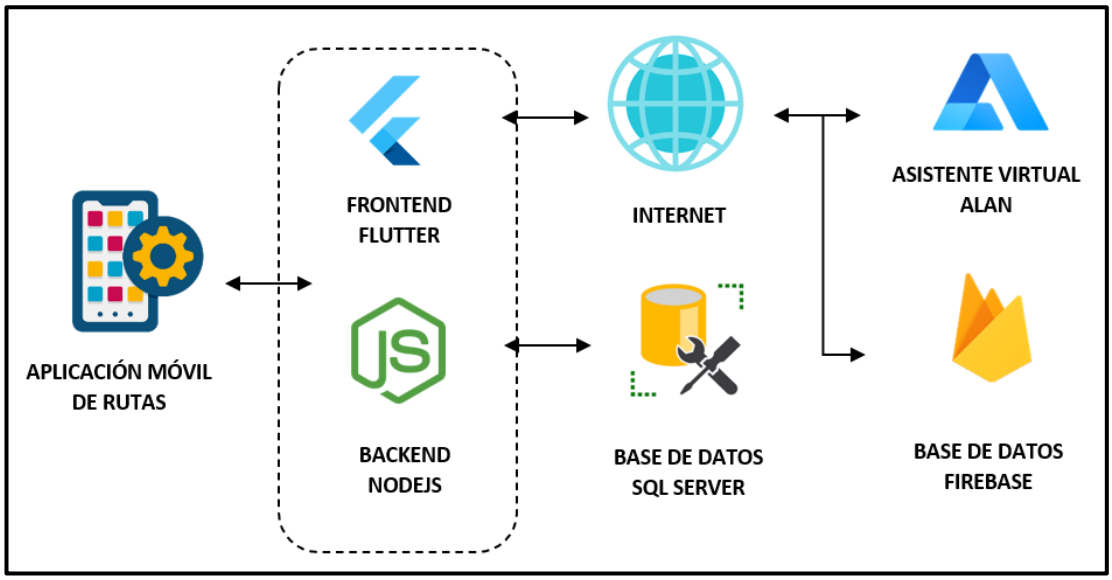


Figura 11 Arquitectura de la aplicación móvil de rutas
Fuente: Investigador

3.2.2.2.Desarrollo del backend de la aplicación móvil

Para el desarrollo del backend de la aplicación móvil se utilizó Node.js el cual es una plataforma de desarrollo de código abierto basada en JavaScript. Una vez que el servidor está en ejecución, puede manejar solicitudes HTTP de clientes y realizar operaciones en la base de datos, como insertar, actualizar, eliminar y recuperar datos. El backend en Node.js permite a los desarrolladores crear aplicaciones de servidor escalables y de alto rendimiento que pueden manejar un gran número de solicitudes simultáneas.

- **Node modules**

En el apartado de ‘node_modules’ se encuentran todos los módulos descargados de npm ocupados en el proyecto incluyendo los que vienen por defecto, Los módulos npm

son paquetes de código que se utilizan en el backend construido en Node.js para añadir funcionalidades adicionales.

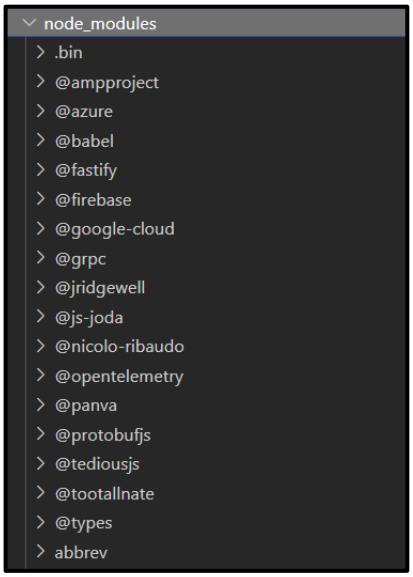


Figura 12 Lista de módulos del backend
Fuente: Investigador

- **Src**

Dentro de la carpeta 'src' se encuentra toda la programación que constituye el backend para la aplicación, dentro de esta carpeta se organizan los archivos por funcionalidad y por su tipo, esta se divide en los apartados de controller, database, routes y contiene los ficheros de app.js, config.js y el index.js. A continuación, se explica el funcionamiento de cada uno de ellos.

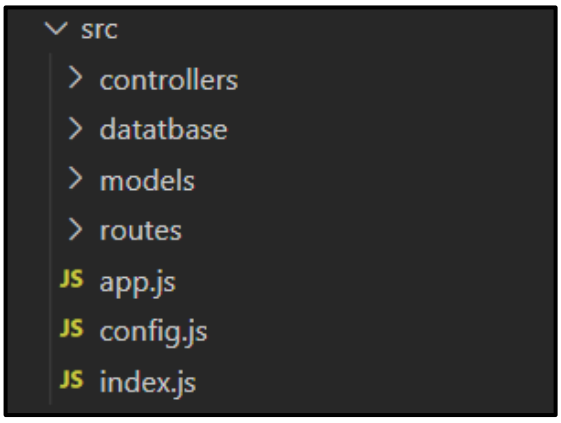


Figura 13 Lista de carpetas y archivos que conforman el src en el backend
Fuente: Investigador

Controllers

La carpeta controllers o controladores son una capa intermedia entre la vista (interfaz de usuario) y el modelo (capa de datos) en la arquitectura de aplicación de Node.js.

Los controladores se encargan de procesar las solicitudes del usuario, comunicarse con el modelo para obtener o guardar información de la base de datos y devolver una respuesta al usuario.

- **CiudadContoller:** llama a la lista de ciudades almacenadas dentro de la base de datos
- **ClienteControlles:** se maneja todos los datos del usuario ingresados en el registro cómo nombre, apellido, teléfono, nombre de usuario, contraseña, foto de perfil, genero, ciudad.
- **GeneroController:** Datos del genero
- **Sitio.Controller:** Se gestiona todos los datos referentes a los sitios del lugar como llamar a los lugares de destino, buscador de rutas, número de visitas realizadas, sitios más visitados, usuarios con más visitas y el registro de las visitas realizadas.

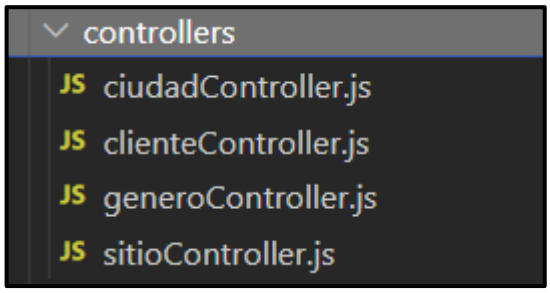


Figura 14 Archivos que conforman la carpeta 'controllers'
Fuente: Investigador

A continuación, se muestra como está constituido el controlador de ciudades denominado *ciudadController*.

```
import { getConnection, sql } from "../datatbase/connection.js";

export const getCiudades = async (req, res) => {
    try{
        const pool = await getConnection();
        const result = await pool.request().query("SELECT ID_CIU,
NOM_CIU FROM CIUDAD");
        console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
```

Básicamente, los siguientes controladores siguen la misma lógica y su código se encuentra plasmado en la sección de anexos.

Database

Dentro de este archivo se realiza la conexión a la base de datos sql para eso se ingresan las credenciales.

```
import sql from 'mssql/msnodesqlv8.js';

const dbSettings = {
    user: "sa",
    password: "1234",
    server: "DESKTOP-LBEN4K8\\SQLEXPRESS",
    database: "BD_RUTAS",
    driver: "msnodesqlv8",
    options: {
        encrypt: false,
        trustServerCertificate: true,
        enableArithAbort: true,
        trustedConnection: true
    },
    port: 1433
}
```

Se creó una función para poder exportar desde este módulo hacia los otros módulos usando el comando *export async function.* El comando *export* indica que la función es accesible desde otros módulos, mientras que el modificador *async* indica que la función es una función asíncrona.

```
export async function getConnection(){
    const pool = await sql.connect(dbSettings);
    return pool;
}
```

Routes

El archivo *routes* o enrutador es una clase o módulo que se utilizado para manejar las solicitudes en la aplicación. El enrutador se encarga de tomar una solicitud y enviarla al controlador adecuado para su procesamiento.

Dentro de Node.js se usó la librería de enrutamiento Express.js la cual es una de las librerías más populares y ampliamente utilizada.

```
import {Router} from 'express';
```

En esta se define un enrutador utilizando el método *router* de la librería, luego se definen las rutas para diferentes métodos de solicitudes. Los métodos utilizados son GET y POST.

El método GET se utilizó para obtener información del servidor. Por ejemplo, se usa para obtener una lista de elementos de una base de datos o para obtener un archivo específico.

El método POST se utilizó para enviar información al servidor. Por ejemplo, se usa para crear un nuevo elemento en una base de datos o para subir un archivo.

A continuación, se muestran todas las ***routes*** creadas para el desarrollo del backend.

```
//CLIENTE
router.get('/usuario', getUsuario);
router.post('/ingresarUsuario', registrarUsuario);
router.post('/usuario/login', login);
router.post('/usuario/loginImage', loginImage);
router.post('/usuarioId', getUsuarioPorID);
router.post('/usuario/:id', updateUsuarioById);

//CIUDAD
router.get('/ciudades', getCiudades);

//GENERO
router.get('/genero', getGenero);

//SITIOS
router.post('/sitiosDesOri', getDestinos);
router.post('/sitioAsistente', getDestinosAsistente);
router.get('/sitios', getSitios);
router.post('/registrarVisita', registrarVisita);
router.get('/getVisitas', getVisitas);
router.get('/getSitiosVisitados', getSitiosMasVisitados);
router.get('/getUsuariosMasVisitas', getUsuariosConMasVisitas);
router.post('/buscador', Buscador);
```

app.js

Dentro de este archivo se realizó la inicialización del backend para exportar la app. Dentro de los posesos que realiza se encuentran los siguientes:

- Inicializar Firebase

```
admin.initializeApp({
    credential: admin.credential.cert(serviceAccount)
})
```

- Asignación del puerto

En el servidor local, el puerto 3000 se utiliza para escuchar las solicitudes entrantes de la aplicación. La aplicación procesa la solicitud y envía una respuesta al navegador del usuario.

```
let port = 3000;
app.set('port', port || 3000);
```

- Limitar la imagen de usuario

```
app.use(express. json({limit: '50mb'}));
app.use(express.urlencoded({ extended: true, limit: '50mb',
parameterLimit: 50000 }))
```

config.js

Dentro de este archivo se encuentra especificado el puerto 3000 a usar para que funcione el servidor local.

```
export default {
    port: 3000
}
```

index.js

Dentro del archivo index, se inicializó el servidor dentro de la red para esto se colocó la dirección ip asignada a la máquina.

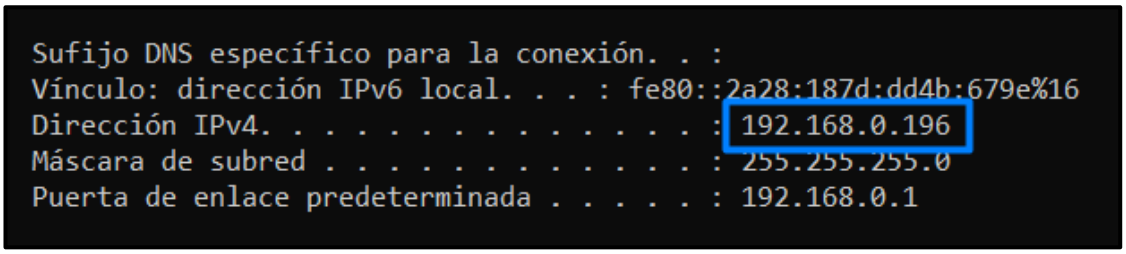


Figura 15 Dirección IP del servidor
Fuente: Investigador

```
import app from './app.js';
import './datatbase/connection.js';

app.listen(3000, '192.168.0.196' || localhost, function(){
    console.log('Aplicación de NodeJS' + ' Iniciado....');
});
```

Para poner en funcionamiento el backend se lo realiza a través del siguiente comando:

```
npm run dev
```

3.2.2.3.Base de datos en sql server

Se usó Microsoft SQL Server como gestor de datos debido a que es un sistema relacional completo desarrollado por Microsoft, que ofrece varias ventajas para su uso

en entornos empresariales, una de estas es su alta escalabilidad, ya que es capaz de manejar grandes cantidades de datos y un gran número de usuarios simultáneos.

Al abrir SQL Server, se presenta la pantalla de inicio de sesión la cual permite conectar con el servidor, mismos datos que son ingresados dentro del backend de aplicación para que compartan la información almacenada en las tablas.

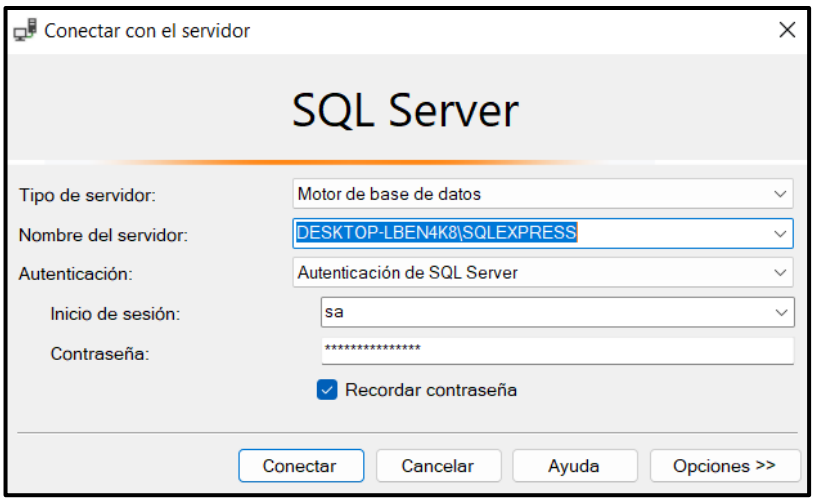


Figura 16 Inicio de sesión de SQL Server
Fuente: Investigador

Modelo entidad relación

El diagrama entidad-relación (ER) es especialmente útil para comunicar de manera clara y precisa la lógica de la base de datos. Se muestra la gráfica utilizada para modelar y diseñar, en donde las entidades se encuentran representadas mediante rectángulos y las relaciones entre ellas mediante líneas.

Las entidades son objetos y conceptos importantes en el desarrollo de la aplicación móvil, en donde se muestra entidades como el rol, genero, visitas y ciudad relacionadas con el usuario, así como la entidad del detalle de la ruta relacionada con el sitio.

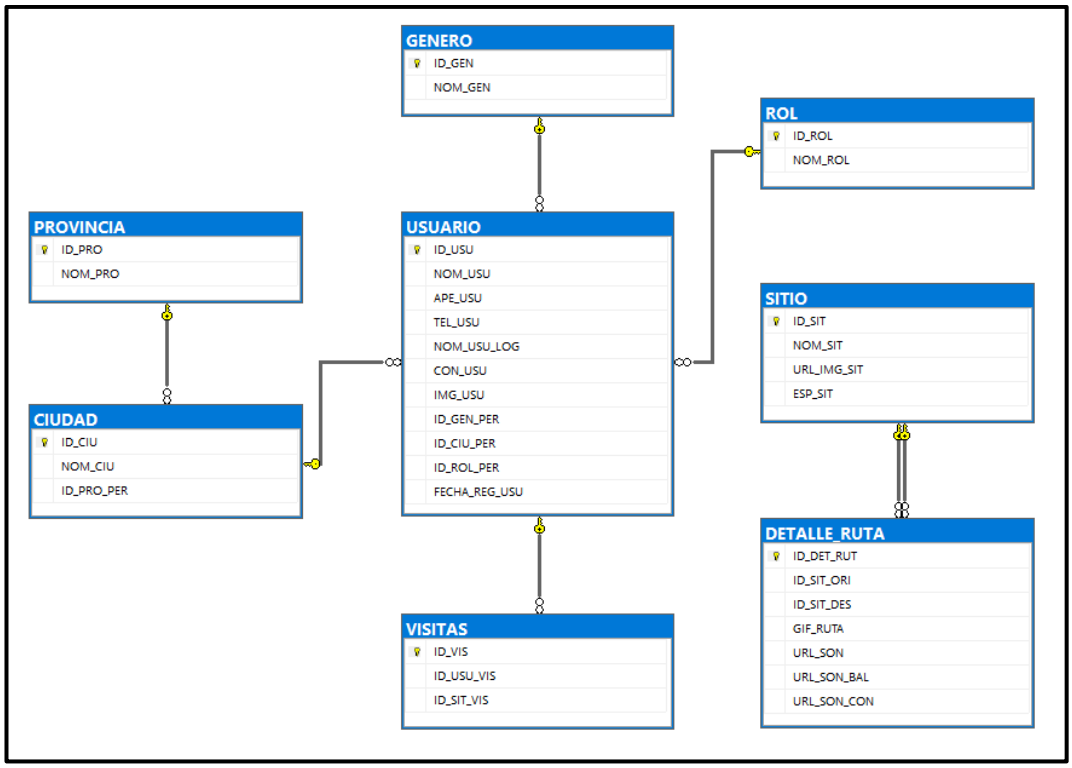


Figura 17 Diagrama entidad relación de la base de datos
Fuente: Investigador

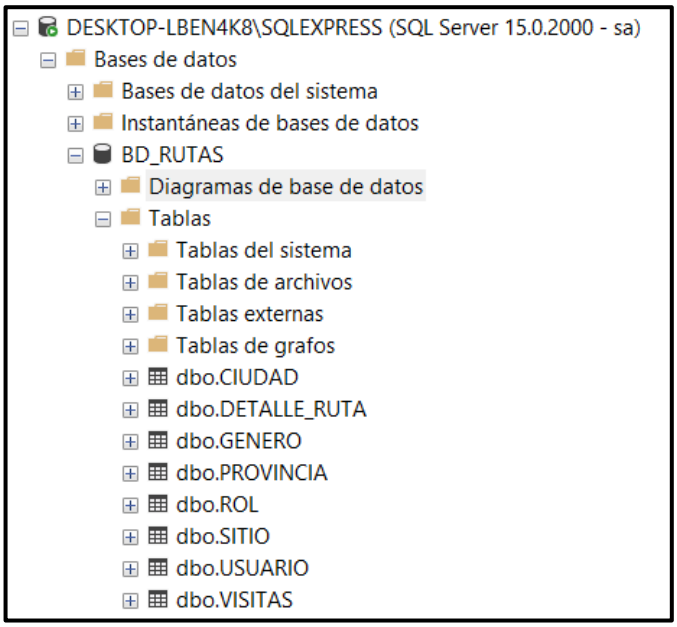


Figura 18 Lista de tablas generadas dentro de SQL server
Fuente: Investigador

A continuación, se muestra la tabla de usuario creada con los datos almacenados en la misma, la aplicación móvil basa muchos de sus procesos a partir de estos datos.

ID_USU	NOM_USU	APE_USU	TEL_USU	NOM_USU_LOG	CON_USU	IMG_USU	ID_GEN_PER	ID_CIU_PER	ID_ROL_PER	FECHA_RE...
63	Carolina	Guzmán	0987654321	cguzman1234	8d969eef6e...		2	1803	2	2022-12-08
65	Luis	Perez	0987654321	lperez4526	8d969eef6e...		1	1801	2	2022-12-14
66	Lucas	Tañeda	0987654321	ltaneda1234	8d969eef6e...		1	1802	2	2022-12-14
69	Amuro	Toru	0987654321	atoru1234	8d969eef6e...		1	1801	2	2022-12-14
70	Jonathan	Nunez	0995567333	jnunez1159	8d969eef6e...		1	1801	1	2022-12-14
75	Carmen	Bonilla	0999999999	cbonilla1234	8d969eef6e...		2	1701	2	2023-01-15
77	Melanie	Nunez	0999999999	mnunez1234	8d969eef6e...		2	1802	2	2023-01-17
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 19 Tabla de los datos del Usuario
Fuente: Investigador

3.2.2.4.Desarrollo de la aplicación móvil en Flutter

En base a la comparación realizada en la Tabla 9, se usó el Framework multiplataforma Flutter para el desarrollo de este proyecto ya que es una de las mejores soluciones para desarrollar aplicaciones para Android y iOS. Las versiones para teléfonos inteligentes de estas aplicaciones funcionan como aplicaciones verdaderas y nativas en dispositivos Apple y Android y se compilan para la plataforma respectiva antes de su publicación.

El SDK de Flutter se basa en el lenguaje de programación Dart, también desarrollado por Google. Su intención es suplantar el JavaScript clásico.

Flutter es una tecnología basada en widgets. Esto significa que puede aplicar programación orientada a objetos a cualquier elemento. Uno de los beneficios de usar Flutter es que puede modificar o personalizar widgets con facilidad. Además, proporciona widgets de interfaz de usuario que cumplen con los requisitos clave de diseño de la aplicación.

En la página oficial de Flutter, se muestran los pasos para la instalación dependiendo el sistema operativo en el que se va a trabajar, en este caso se trabajó en Windows 11, es importante tener una máquina que trabaje con 64-bit y al menos 10 GB de almacenamiento libres.

Como primer requerimiento para comenzar a usar la tecnología de desarrollo Flutter es obtener el SDK el cual, se encuentra disponible en la página oficial en formato .zip. Una vez descargado se agrega Flutter al PATH para ejecutar comandos en la consola normal de Windows.

Para verificar que Flutter se encuentre instalado correctamente, desde la consola se ejecuta el comando ***flutter doctor***. Este comando verifica su entorno y muestra un informe del estado de la instalación de Flutter.

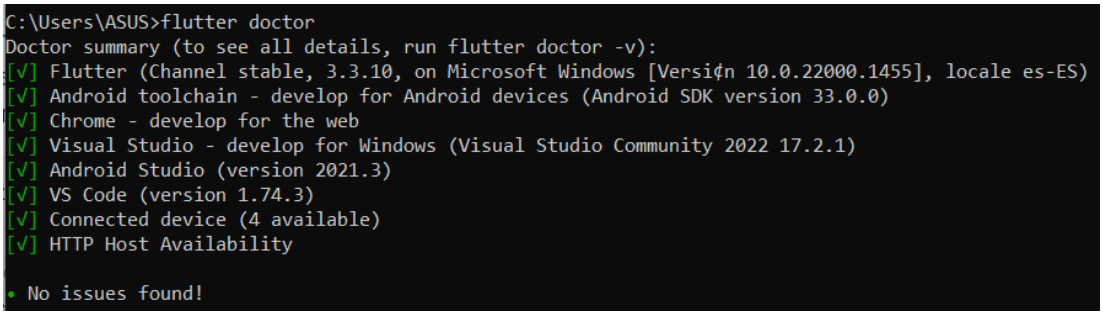


Figura 20 Estado de la instalación de Flutter
Fuente: Investigador

Para construir aplicaciones utilizando la tecnología de desarrollo de Flutter es necesario un editor de texto IDE. Par ejecutar este proyecto se utilizó Android Studio. Es necesario tener instalado Android Studio en su versión 3.0 o superior. Al iniciar se instala los plugins de Flutter y en consola se corre el comando *flutter config --android-studio-dir* para establecer el directorio en el que está instalado.

Configuración del dispositivo móvil Android

Para ejecutar y realizar pruebas de la aplicación desarrollada en Flutter se usa un dispositivo Android en su versión 12, sin embargo, se lo puede realizar con dispositivos Android 5 o superiores.

Dentro del dispositivo se habilita las opciones de desarrollador y depuración USB. Mediante cable USB se conecta el dispositivo a la computadora y en la terminal se ejecuta el comando *flutter devices* para verificar que Flutter reconoce el dispositivo.

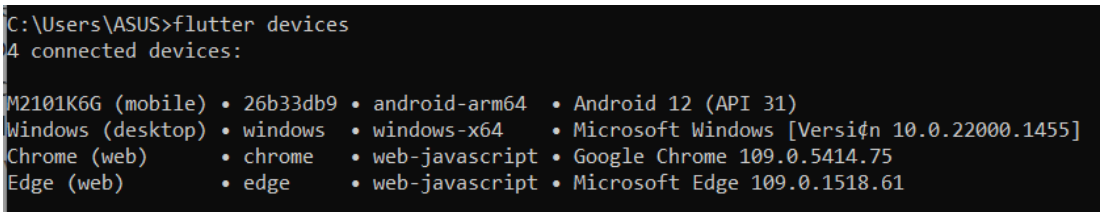


Figura 21 Reconocimiento del dispositivo móvil por medio de Flutter
Fuente: Investigador

Permisos necesarios

En el proyecto de Flutter dentro de la carpeta *android/app/src/main/* se encuentra el archivo *AndoridManifest.xml* el cual es esencial en cualquier proyecto de Android, y también es necesario en un proyecto de Flutter que tiene aplicación Android. Este archivo contiene información importante sobre la aplicación, como los permisos necesarios, la actividad principal, y los metadatos de la aplicación.

Para especificar los permisos necesarios para el funcionamiento de la aplicación se usó la etiqueta <*uses-permission*>. Las funciones específicas del sistema a las que se accede son: internet, bluetooth, administración del bluetooth y localización.

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

Versión del sistema operativo Android

El *buil.gradle* en la carpeta *android* es el archivo en donde se especifica la versión mínima del sistema operativo Android que es compatible con la aplicación. Esto significa que la aplicación solo se podrá ejecutar en dispositivos con una versión del sistema operativo igual o superior a la versión especificada.

Para el desarrollo de este proyecto se usó la *minSdkVersion 21* debido a su compatibilidad con Bluetooth, lo que significa que la aplicación se ejecuta en dispositivos Android con una versión 5.0 (Lollipop) o superior a esta.

```
    defaultConfig {
        applicationId "com.proyecto.rutas_app"
        minSdkVersion 21
        targetSdkVersion flutter.targetSdkVersion
        versionCode flutterVersionCode.toInteger()
        versionName flutterVersionName
    }
```

Almacenamiento de recursos

Dentro del proyecto de Flutter se usó la carpeta ‘assets’ para almacenar los recursos estáticos de la aplicación, como imágenes, audio, fuentes y archivos JSON. Estos recursos son accesibles a través del paquete *‘AssetBundle’* y se utilizan dentro de la

aplicación para mostrar imágenes, reproducir el audio de las notificaciones, definir la fuente Titania y llamar al icono de la pantalla de Login.

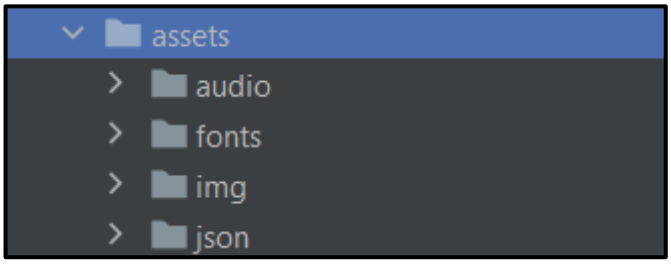


Figura 22 Recursos almacenados en la carpeta 'assets'
Fuente: Investigador

Para usar los recursos de la carpeta *assets*, primero se especificó dentro del archivo *pubspec.yaml* del proyecto.

```
assets:
  - assets/img/
  - assets/json/
  - assets/audio/
fonts:
  - family: Titania
    fonts:
      - asset: assets/fonts/Titania/Titania.ttf
```

Carpeta principal

En el proyecto de Flutter, la carpeta 'lib' es la carpeta principal donde se almacena el código fuente de la aplicación. Dentro de esta carpeta se encuentra la carpeta 'src' en donde se almacenó todos los archivos Dart que conforma la aplicación.

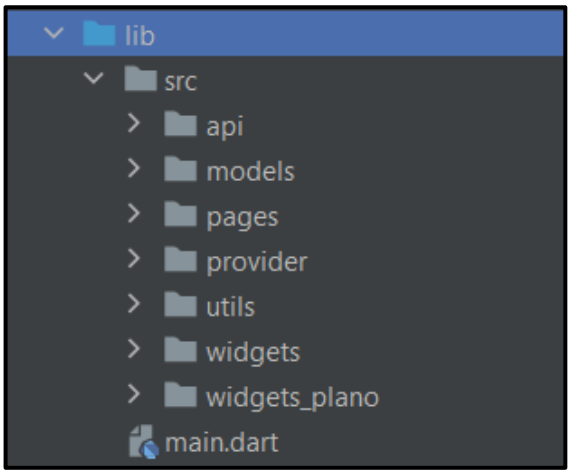


Figura 23 Estructura del src de Flutter
Fuente: Investigador

La carpeta ‘src’ contiene varias subcarpetas con el fin de mantener una mejor estructuración y mantenimiento del código de la aplicación. Dentro de esta se encuentran las subcarpetas de api, models, pages, provider, útil, widgets y widgets_plano.

- Api

Dentro de la subcarpeta ‘api’ se encuentra el archivo Enviroment.dart el cual contiene la clase ‘Enviroment’ donde se define una variable estática API_DELIVERY que contiene la URL de la API relacionada con el manejo de entregas o envíos. Esta variable es utilizada para realizar las peticiones de la aplicación a los servicios de entrega.

```
class Enviroment {
  static const String API_DELIVERY = "192.168.0.196:3000";
}
```

La clase ‘Enviroment’ permite tener una mejor organización y mantenimiento de la configuración de la aplicación, ya que se puede cambiar fácilmente la configuración de un entorno a otro simplemente cambiando la ip en esta clase.

- Models

La subcarpeta ‘models’ se usa para almacenar las clases que representan los modelos de datos de la aplicación móvil. Estos modelos son utilizados para almacenar y manipular los datos.

La ventaja de esta carpeta específica para los modelos es que permite una mejor organización y mantenimiento del código, ya que todas las clases relacionadas están en este lugar. Dentro de esta se encuentran definidas las siguientes clases, mismas que se encargan de enviar o recibir datos:

- **act_user**: datos que serán modificados dentro de la página para editar el perfil, dentro de estos datos se encuentran el teléfono, el id de la ciudad y la foto del usuario.
- **ciudad:** datos de id de la ciudad, así como el nombre de la ciudad que elija el usuario.

- **destinos:** datos del id de la ruta, sitio de destino, sitio de origen, url de las imágenes, especificaciones de los sitios, gif de la ruta seleccionada, url de los audios tanto de las balizas como los audios de confirmación.
- **género:** dato del id del género.
- **light:** información para construir el mapa, como localización, nombre del lugar, estado del punto, posición del mapa, y el título de cada sitio.
- **sitio:** datos del id del sitio, así como el nombre de este**.**
- **sitios_visitados:** nombre de los sitios más visitados, número de visitas y el url de las imágenes para posicionar cada sitio.
- **user:** todos los datos del usuario ingresados para para el registro de este, dentro de estos se encuentra el id del usuario, nombre, apellido, teléfono, nombre de usuario, contraseña, foto de perfil, id del género, id de la ciudad y el rol que cumple ya sea usuario normal o usuario administrador.
- **usuarios_visitas:** datos del usuario que se incluyen en los reportes, como nombre, número de visitas y foto de perfil.
- **visita:** datos del id de la visita, id del usuario y el id del sitio al que se dirige.

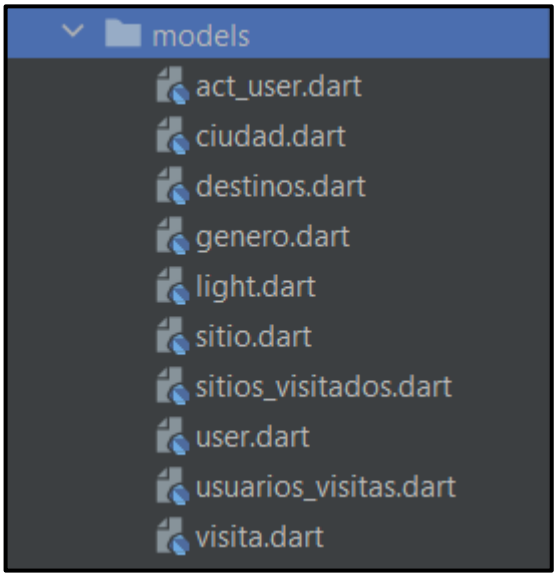


Figura 24 Lista de objetos que conforman la carpeta ‘models’
Fuente: Investigador

Los métodos que se usaron en las clases de los ‘models’ son ‘fromJson’ y ‘toJson’. Estos métodos permiten convertir un objeto de un modelo en una representación JSON y viceversa.

El método ‘toJson’ convierte el objeto de un modelo en un mapa de pares clave-valor, donde las claves son cadenas de texto y los valores son los valores de las propiedades del objeto. Este método se utiliza para enviar datos al servidor o para almacenar datos localmente.

Por otro lado, el método ‘fromJson’ convierte un mapa de pares clave-valor en un objeto de un modelo. Este método se utiliza para recibir datos del servidor o para recuperar datos almacenados localmente.

A continuación, se muestra la construcción de la clase sitios_visitados. Básicamente las demás clases siguen el mismo formato.

```
class SitiosMasVisitados {
  String nomSit;
  int visitas;
  String urlImgSit;
  SitiosMasVisitados({
    this.nomSit,
    this.visitas,
    this.urlImgSit,
  });
  factory SitiosMasVisitados.fromJson(Map<String, dynamic> json) =>
SitiosMasVisitados(
    nomSit: json["NOM_SIT"],
    visitas: json["VISITAS"],
    urlImgSit: json["URL_IMG_SIT"],
  );
  Map<String, dynamic> toJson() => {
    "NOM_SIT": nomSit,
    "VISITAS": visitas,
    "URL_IMG_SIT": urlImgSit,
  };
}
```

- Provider

Dentro de la subcarpeta ‘provider’ se encuentran almacenados los proveedores de datos, que son clases simples que realizan el consumo del api. Estos proveedores manejan y comparten los datos recibidos del API a través de la aplicación según se requiera. Esta subcarpeta contiene los proveedores de ciudad, genero, sitios y usuarios.

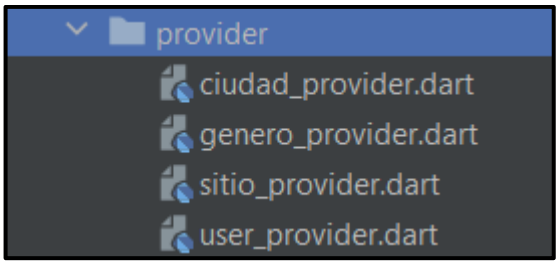


Figura 25 Lista de los proveedores de datos.
Fuente: Investigador

A continuación, se muestra como está constituida la clase del proveedor ciudad. Básicamente los demás 'providers' siguen el mismo contexto.

```
class CiudadProvider {
  String _url = Enviroment.API_DELIVERY;
  String _apigetCiudad = "/ciudades";
  BuildContext context;
  Future init(BuildContext context) {
    this.context = context;
  }
  Future<List<Ciudad>> getCiudad() async {
    Uri url = Uri.http(_url, '$_apigetCiudad');
    final response = await http.get(url);
    final data = json.decode(response.body);
    List<Ciudad> listaCiudades = [];
    for (var item in data) {
      Ciudad ciudad = Ciudad(idCiu: item["ID_CIU"], nomCiu:
item["NOM_CIU"]);
      listaCiudades.add(ciudad);
    }
    return listaCiudades;
  }
}
```

- Utils

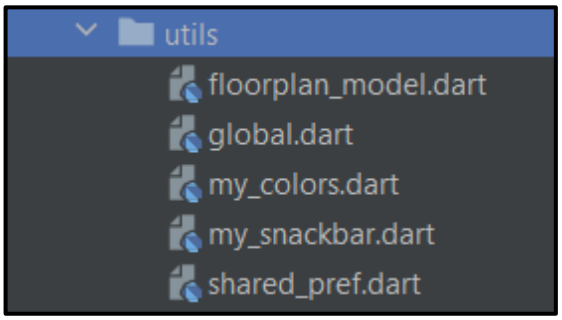


Figura 26 Lista de utilidades usadas en el proyecto de Flutter
Fuente: Investigador

La subcarpeta **utils** se usa para almacenar clases y funciones que son utilizadas en varios puntos de la aplicación, pero no pertenecen a ninguna clase o widget específico.

Dentro de estas utilidades se encuentran las de:

- **Floorplan_model:** Dentro de esta utilidad se establecen clases que ayudan a la posición del mapa dentro de la pantalla, así como las funciones para hacer zoom, escala del mapa y la restauración del mismo a su posición inicial.
  Posición inicial del mapa

```
class Pos {
  double x = 0.0;
  double y = 0.0;
  Pos(x, y) {
    this.x = x;
    this.y = y;
  }
```

  Reestablecer el mapa

```
void reset() {
    _scale = 1.0;
    _previousScale = 1.0;
    _pos = Pos(0.0, 0.0);
    _previousPos = Pos(0.0, 0.0);
    _endPos = Pos(0.0, 0.0);
    _isScaled = false;
    notifyListeners();
  }
```

- **Global:** Dentro de esta utilidad se definen los puntos que se dibujan al mapa y se asigna el nombre del sitio correspondiente, usa condiciones para designar el color en el cual true hace referencia a un sitio y false hace referencia a un faro.

```
static const List lights = [
    {
      'location': 'Taller',
      'name': 'Taller',
      'status': true,
      'position': [0.0, -0.22],
      'tile': 1,
    },
    {
      'location': '3° Faro',
      'name': '3° Faro',
      'status': false,
      'position': [0.0, 0.07],
```

```
      'tile': 1,
    },
```

- **My_colors:** En esta utilidad se encuentran establecidos los colores principales usados para el diseño de la interfaz del usuario.

```
class MyColors{
  static Color primary = const Color(0xFF1e88e5);
  static Color primaryOpacityColor = const Color.fromRGBO(30,
136, 229, 0.09);
  static Color primaryColorDark  = const Color(0xFF0d47a1);
}
```

- **Shared_pref:** Esta utilidad se usa para guardar y leer el dato de la contraseña del local storage el cual se refiere al almacenamiento de datos en el dispositivo del usuario.

```
class SharedPref{
  void save(String key, value) async {
    final prefs = await SharedPreferences.getInstance();
    prefs.setString(key, json.encode(value));
  }
```

  - Widgets

Dentro de la subcarpeta '**widgets**' se almacenan los widgets personalizados creados en este proyecto. Los cuales constituyen bloques básicos de construcción de la interfaz de usuario.

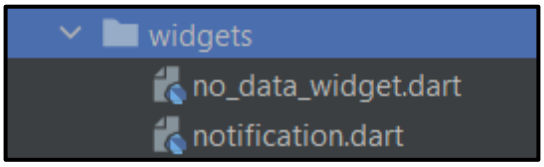


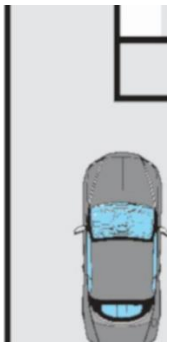
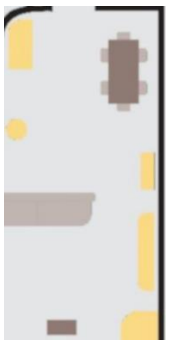
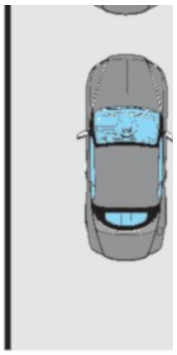
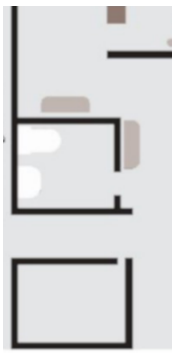
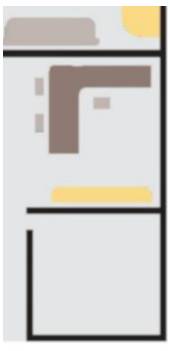
Figura 27 Lista de los widgets generales
Fuente: Investigador

- **No_data_widget:** dentro de este **widget** se encuentra la clase que muestra el mensaje de alerta al no encontrar una ruta disponible. Dentro de la programación del witget se incluye la posición, el margen, las medidas, la imagen y el mensaje de alerta.
- **Notification:** Dentro de este widget se construye la notificación del usuario que se muestra en el dispositivo móvil al realizar la detección de los faros.

  - Widgets_plano

Dentro de esta carpeta se encuentran los archivos que ayudan a la construcción del plano. El mapa presente en la aplicación móvil no se trata de una sola imagen, sino que es una imagen construida a través de una matriz 3x3, con la finalidad de construir un mapa dinámico basado en los mapas de Google maps los cuales se encuentran creados a partir de matrices de imágenes para no perder calidad y con el fin que al acercar y alejar el mapa se muestre la información dentro de este.

Tabla 17 Matriz 3x3 del mapa dinámico

MATRIZ 3X3		
		
		
		

Fuente: Investigador

Para esto se construyeron 3 archivos que ayudan con el proceso de construcción del plano:

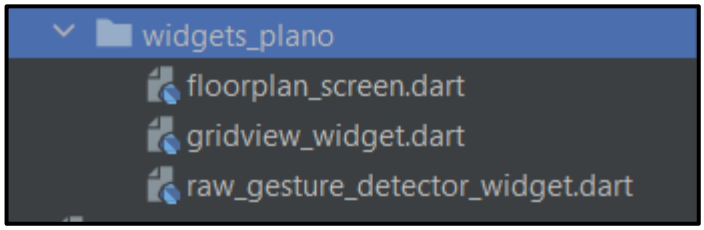


Figura 28 Lista de widgets para la creación del mapa dentro de la aplicación móvil
Fuente: Investigador

- **Floorplan_screen:** Dentro de este archivo se establece el estilo del mapa, tanto como los bordes, color y estilo de la caja que contiene el mismo.
- **Gridview_widget:** Dentro de este widget se muestra la colección de elementos en una cuadrícula. Los elementos de la cuadrícula son dispuestos en filas y columnas, y se pueden desplazar horizontal o verticalmente. Básicamente dentro de este widget se realiza la construcción del plano a través del comando GridView.builder().
- **Raw_gesture_detector:** Este widget se creó con el fin de detectar si el mapa se encuentra ampliado o no, en caso de si estarlo se mostrara la información del lugar, así como los puntos en donde se encuentran ubicados cada uno de los faros.

  - Pages

Dentro de la carpeta '**pages**' se almacena las diferentes páginas o pantallas de la aplicación. Cada página representa una sección dentro de la aplicación en la cual se encuentra contenidos widgets y lógica específica.

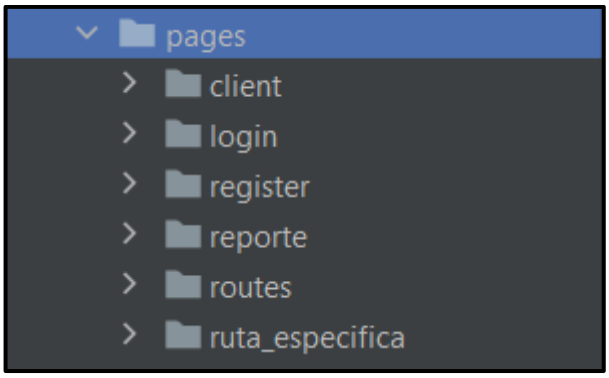


Figura 29 Lista de páginas que conforman la aplicación móvil
Fuente: Investigador

En esta subcarpeta se encuentran las diferentes pantallas de la aplicación, como:

- **Client:** esta carpeta hace referencia a la página de inicio de la aplicación en donde se muestra el plano en conjunto con el botón del asistente virtual y el botón de opciones. Dentro de esta carpeta también se encuentra la página de edición del perfil.
- **Login**: se trata de la pantalla que aparece al momento de abrir la aplicación recién instalada en donde se realiza en inicio de sesión.
- **Register:** Forma la página de registro de usuario.
- **Reporte:** dentro de esta página se acomodan los datos a presentarse en cada reporte al usuario administrador.
- **Routes:** construcción de la página con la lista de las rutas disponibles.
- **Ruta_especifica:** En esta página se muestra la gráfica de la ruta a seguir dentro del mapa según la ruta seleccionada, así como la activación del asistente para mostrar información acerca de esa ruta.

Cada página cuenta con dos archivos el ‘controller.dart’ y el ‘page.dart’ estos se usan para almacenar la lógica y la interfaz de usuario de una página o pantalla específica. Estos archivos funcionan en conjunto para que cada página cumpla con su función establecida.

- El ‘controller.dart’ se usa para almacenar la lógica de negocio de la página
- El ‘page.dart’ se usa para almacenar la interfaz de usuario de la página

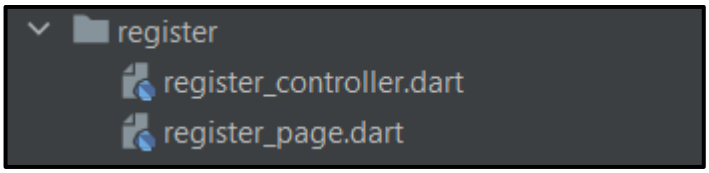


Figura 30 Archivos que conforman cada una de las páginas de la aplicación móvil
Fuente: Investigador

- Main.dart

El main.dart es el archivo principal de la aplicación y es el primer archivo que se ejecuta cuando se lanza la aplicación móvil. Es responsable de inicializar y configurar la navegación entre las diferentes pantallas o página de la aplicación.

Declaración de la clase principal main().

```
void main() {
  runApp(const MyApp());
}
```

Inicialización del widget raíz de la aplicación que contiene la configuración de las rutas que se utilizan para navegar entre las diferentes pantallas de la aplicación.

```
return MaterialApp(
    title: 'Rutas App Flutter',
    debugShowCheckedModeBanner: false,
    initialRoute: 'login',
    routes: {
      'login': (BuildContext context) => LoginPage(),
      'register': (BuildContext context) => RegisterPage(),
      'client/routes/list': (BuildContext context) =>
ClientRoutesListPage(),
      'client/update': (BuildContext context) => UserUpdatePage(),
      'routes': (BuildContext context) => RoutesPage(),
      'widgets_plano/floor': (BuildContext context) =>
FloorPlanScreen(),
      'widgets_plano/gridview': (BuildContext context) =>
GridViewWidget(),
      'widgets_plano/rawgestor': (BuildContext context) =>
          RawGestureDetectorWidget(),
      'reporte': (BuildContext context) => ReportePage()
```

Configuración de la personalización del tema.

```
theme: ThemeData(
          primaryColor: MyColors.primary),);
```

3.2.2.5.Almacenamiento de los archivos de la aplicación móvil

Firebase es una plataforma de desarrollo de aplicaciones móviles y web desarrollada por Google. Ofrece una variedad de herramientas y servicios que ayudan a los desarrolladores a crear y escalar aplicaciones de manera rápida y sencilla.

La función principal que cumple Firebase en este proyecto es la de almacenamiento de datos. Firebase proporciona una base de datos en tiempo real, que permite almacenar y sincronizar datos entre diferentes dispositivos y usuarios.

Para esto dentro de la plataforma se creó un proyecto con el nombre de la aplicación móvil

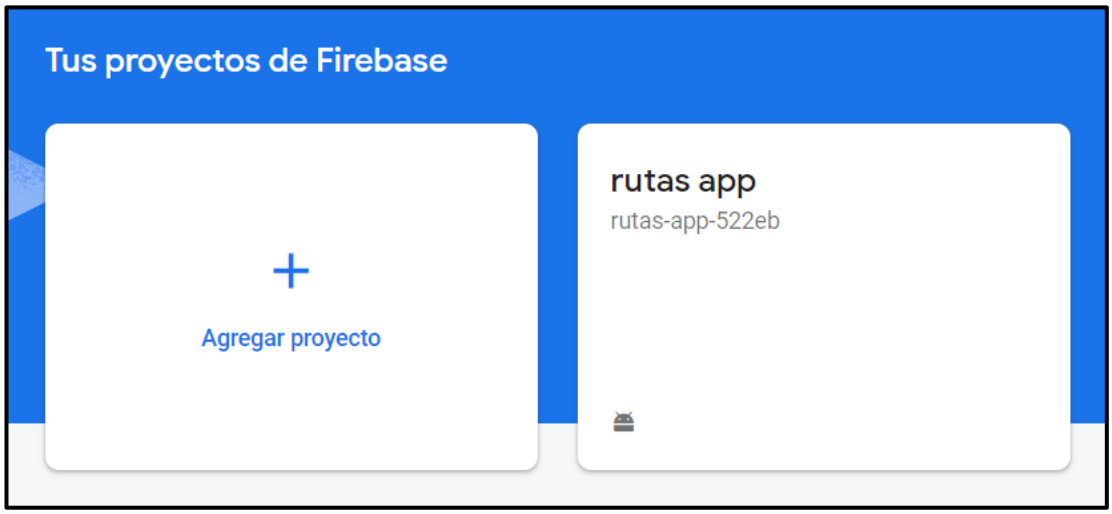


Figura 31 Proyecto creado en Firebase
Fuente: Investigador

Dentro de este se encuentran alojados los archivos de audio personalizados que usa el asistente virtual, así como las imágenes de las rutas generadas en formato gif. La finalidad de mantener estos archivos dentro de la plataforma de Firebase es la optimizar recursos para la aplicación móvil, logrando que no ocupe demasiado espacio de almacenamiento dentro del dispositivo móvil.

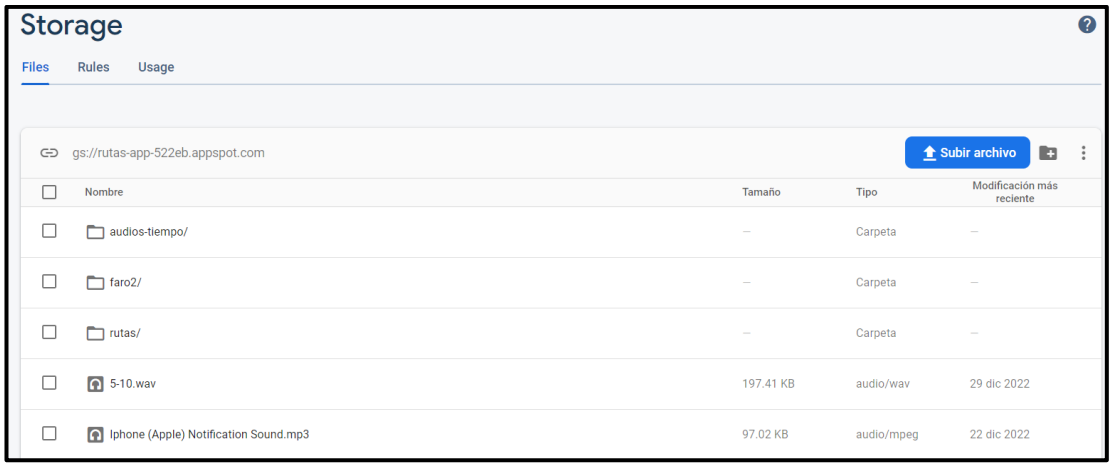


Figura 32 Archivos almacenados dentro de Firebase
Fuente: Investigador

Cada archivo almacenado cuenta con una dirección y ubicación única, esta información es usada para llamar a cada uno de los archivos dentro de la aplicación móvil.

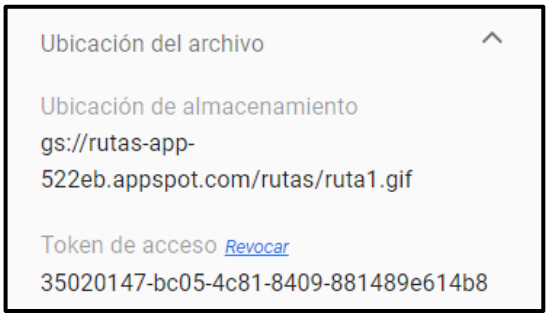


Figura 33 Ejemplo de la ubicación de los archivos y token de acceso
Fuente: Investigador

3.2.2.6.Configuración del asistente virtual

El desarrollo del asistente virtual se lo realizó en la plataforma de IA conversacional de Alan estudio el cual es un potente IDE basado en la web en donde se puede escribir, probar y depurar escenarios de diálogo.

La plataforma Alan automatiza esto con su infraestructura basada en la nube, incorporando una gran cantidad de tecnologías avanzadas de reconocimiento de voz y comprensión del lenguaje hablado. Esto permite a Alan admitir experiencias de voz conversacionales completas, definidas por desarrolladores usando scripts de Alan Studio, escritos en JavaScript. Alan integra la IA de voz en la aplicación con la ayuda del SDK.

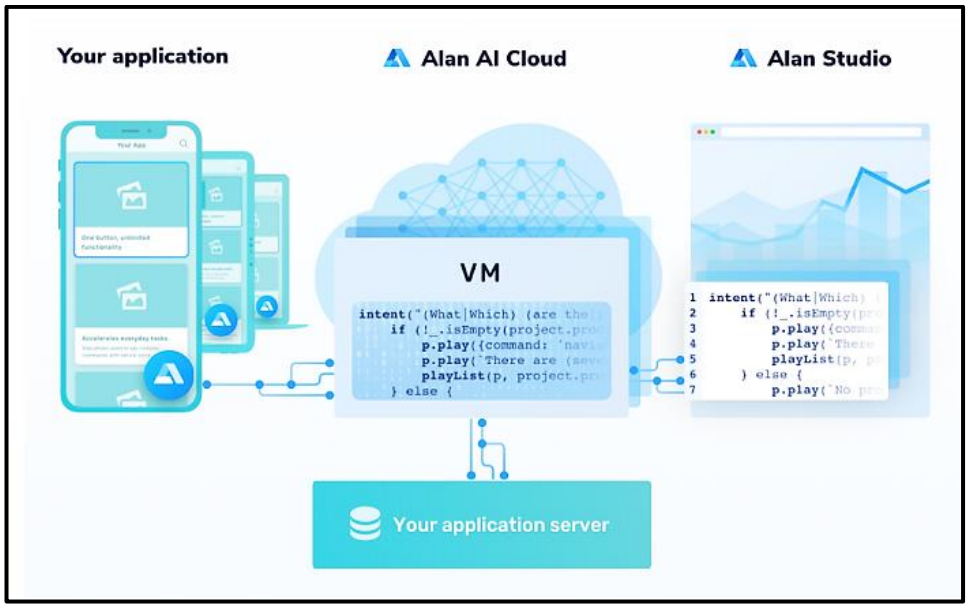


Figura 34 Arquitectura de la plataforma Alan para el desarrollo del asistente virtual [54].

Para la creación del proyecto en Alan, es importante el registro y la creación de una cuenta en su estudio, dicho proceso se lo realiza en la página principal del mismo.

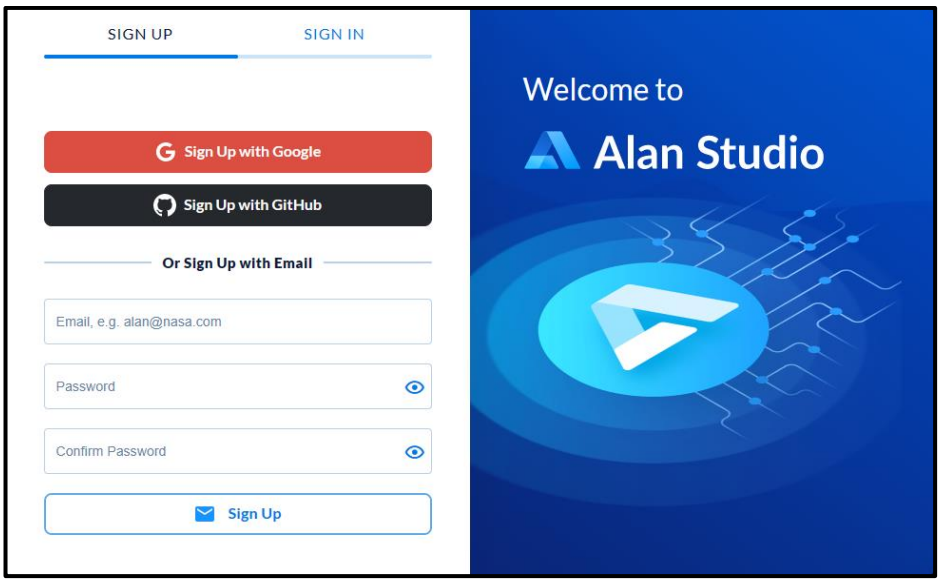


Figura 35 Inicio de sesión dentro de la plataforma Alan Studio
Fuente: Investigador

Al iniciar sesión, se observa el panel del proyecto. En este espacio se ve creado el proyecto para la aplicación de código abierto. Además, muestras gráficas de interacción a través del tiempo, permitiendo tener una visión de los días que más se ocupa el asistente virtual dentro de la aplicación.

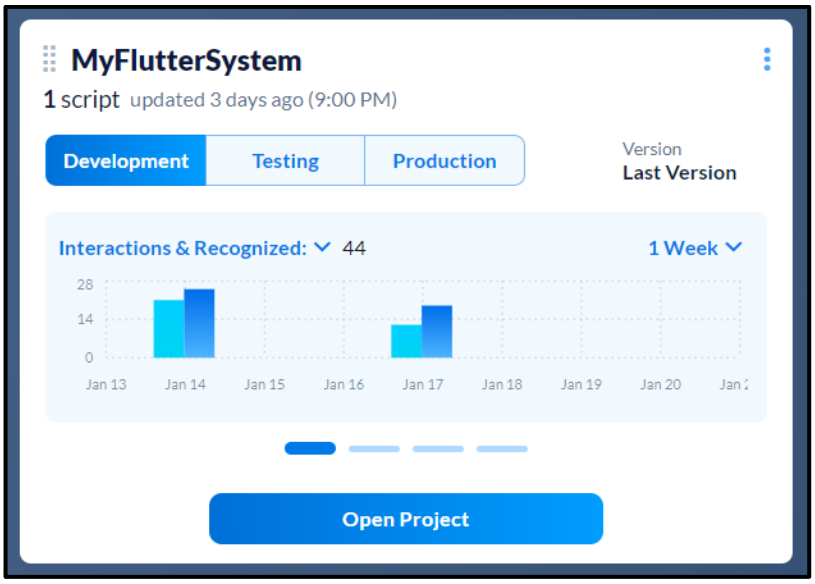


Figura 36 Creación del proyecto dentro de Alan
Fuente: Investigador

Una vez dentro del estudio se configura las acciones para el asistente virtual, cada acción está representada por un *intent* el cual es un objeto de mensajería que se usa para solicitar una acción de otro componente de la aplicación móvil. A continuación, se muestra ejemplos de las acciones programadas dentro del estudio. Todas las acciones del asistente virtual de este proyecto se encuentran en la parte de anexos.

```
// Use this sample to create your own voice commands
intent('Hola', p => {
    p.play('(hola bienvenido)');
});

intent('Como te llamas', p => {
    p.play('(Mi nombre es Alan)');
});

intent('Rutas disponibles', p => {
    p.play({command: 'forward'})
});

intent('informacion de la cocina', p => {
    p.play({command: 'cocina'})
});

intent('ver los Reportes', p => {
    p.play({command: 'reportes'})
});

intent('informacion de las Gradas', p => {
    p.play({command: 'gradas'})
});
```

Figura 37 Programación de las acciones del Asistente Virtual
Fuente: Investigador

Uno de los beneficios del utilizar Alan estudio es que a medida que se van añadiendo acciones, es posible realizar pruebas de voz con el asistente virtual para verificar que cumple con los requerimientos deseados.

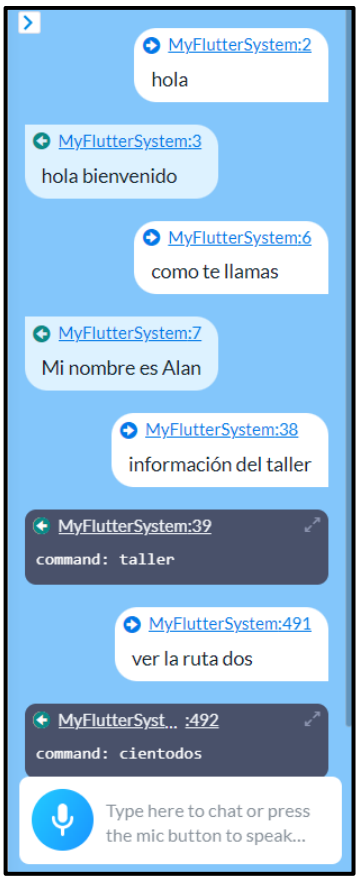


Figura 38 Pruebas de verificación por medio de voz
Fuente: Investigador

En la parte superior derecha en la opción *Integrations* se genera la API única para este proyecto misma que al ser integrada al programa permite llamar todas las acciones creadas en el mismo. Alan funciona de manera independiente es decir que todos los cambios realizados en su estudio no afectan al programa general y no es necesario generar nuevas claves cada que se realice cambios.

Figura 39 Alan SDK Key generado
Fuente: Investigador

Para incluir el proyecto generado en Alan a la aplicación móvil, dentro del archivo ´pubspec´ del proyecto de Flutter, se agregó la dependencia de voz que permite usar Alan.

```
dependencies:
  flutter:
    sdk: flutter
  alan_voice: 2.4.0
```

La integración del Asistente Virtual dentro del proyecto se lo hace en el archivo ´client_routes_list_page.dart´, el cual se encuentra en la página ´client´ en el apartado de ´routes´. Para esto se añade en la parte superior la dependencia del paquete alan_voice.

```
import 'package:alan_voice/alan_voice.dart';
```

Dentro de la clase *_ClientRoutesListPageState* se agrega el método para inicializar el botón Alan, es aquí donde se debe insertar el valor entre comillas arrojado en el campo Alan SDK Key de Alan Studio.

```
_ClientRoutesListPageState() {
  /// Init Alan Button with project key from Alan Studio
  AlanVoice.addButton(

"f779619a4d5a36b48941f225026f409a2e956eca572e1d8b807a3e2338fdd0dc/stage",
      buttonAlign: AlanVoice.BUTTON_ALIGN_RIGHT);
  AlanVoice.onCommand.add((command) =>
_handleCommand(command.data));
}
```

Dentro del mismo archivo se retorna la función de los comandos de voz programados que reproduce el asistente virtual dependiendo el caso, los archivos de audio se encuentran alojados dentro de la base de datos de Firebase, esto con el propósito que la aplicación no sea pesada al momento de la instalación y no ocupe mucho espacio de almacenamiento dentro del dispositivo móvil.

A continuación, se muestra un ejemplo de caso de cómo están constituidos los comandos de voz.

```
case "jardintres":
  player.play(UrlSource(
      'https://firebasestorage.googleapis.com/v0/b/rutas-app-
522eb.appspot.com/o/jardintres.wav?alt=media&token=3fc7e2ac-c489-
40b2-9f07-901b6545cb09'));
  Future.delayed(
      Duration(seconds: 7),
```

```
      () => player.play(UrlSource(
          'https://firebasestorage.googleapis.com/v0/b/rutas-app-
522eb.appspot.com/o/mensaje_info.wav?alt=media&token=81cfabf8-04c1-
4e50-b3cb-0b03b2437de2')));
  break;
```

En este ejemplo el asistente virtual al detectar el comando de voz referente al jardín tres activara la secuencia de voz en donde explica la información de este. Básicamente, este proceso se realiza con los demás casos programados para el asistente virtual.

3.2.2.7.Implementación de los Beacons a la aplicación móvil

Para detectar la señal de los faros y generar una notificación, dentro del caso *escanear* se realiza la comparación de las direcciones MAC de los Beacons y según coincidan con lo registrado, se envía la notificación al dispositivo móvil. Cabe recalcar que esta búsqueda se la realiza solo cuando se activa esta función por medio de voz, de esta forma el programa no estará intentando realizar estos procesos desde que se inicia la aplicación, optimizando así los recursos de este.

A continuación, se muestra el ejemplo de este proceso para el primer faro ubicado en la puerta. Básicamente se sigue el mismo contexto para los dos faros siguientes.

```
case "escanear":
  refresh();
  FlutterBlue.instance.startScan(timeout: Duration(seconds: 4));
  int cont1 = 0, cont2 = 0, cont3 = 0;
  var subscription =
FlutterBlue.instance.scanResults.listen((results) {
    for (var item in results) {
      if (item.device.id.toString() == "FD:15:BF:0C:99:87") {
        if (cont1 < 1) {
          Future.delayed(Duration(seconds: 1), () {
            showNotificacion(
              "Baliza Puerta",
              "La baliza puerta se encuentra activada",
            );
            final player = AudioPlayer();
            player.play(AssetSource('audio/not.mp3'));
          });
        }
        cont1 = cont1 + 1;
      }
    }
```

3.2.2.8.Aplicación móvil de rutas

El prototipo desarrollado como propuesta de navegación y generación de rutas en espacios cerrados, tiene el objetivo de orientar al usuario para llegar a su lugar de destino, apoyándose en las señales bluetooth emitidas por los faros, la generación de rutas automática por parte de la aplicación y la asistencia virtual de Alan IA. Esta aplicación se compone por seis módulos principales, los cuales fueron implementados en una aplicación móvil con el sistema operativo Android.

- Registro de usuario
- Inicio de sesión
- Pantalla principal
- Editar perfil
- Rutas
- Reportes

La aplicación móvil de rutas cuenta con un interfaz amigable para el usuario y está destinada para alojar 2 tipos de usuarios: clientes y administradores, la diferencia de estos radica en que el usuario administrador cuenta con un módulo de reportes que le permitirá gestionar de mejor manera el lugar.

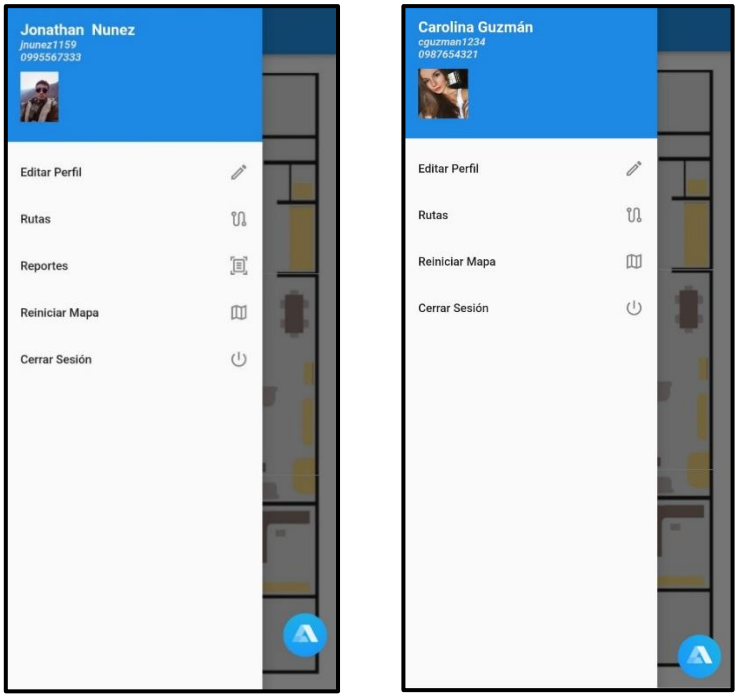


Figura 40 Interfaz de usuario administrador y cliente
Fuente: Investigador

Módulo de Login

Al iniciar la aplicación de rutas se despliega la pantalla de 'login' o inicio de sesión en la cual dependiendo si se trata de un usuario registrado anteriormente se debe ingresar con el nombre de usuario y la contraseña, mismas que deben coincidir con los datos alojados en la base de datos, caso contrario no se permitirá el acceso. Si se trata de un nuevo usuario se tiene la opción de "Regístrate" la cual despliega la pantalla de registro.

Figura 41 Interfaz de la pantalla Login
Fuente: Investigador

Módulo de Registro de Usuario

El apartado de registro es en donde se hace el ingreso de los datos de un nuevo usuario como: nombre, apellido, teléfono, nombre de usuario, contraseña, confirmar contraseña, así como la selección de la ciudad y género, para terminar el proceso de registro se debe colocar una imagen, teniendo la opción de elegir una imagen existente en la galería o hacer una imagen mediante la cámara. Una vez se hayan llenado los campos del registro estos se guardan en la base de datos al momento de presionar la opción "Registrarse" volviendo a la página de Login para realizar el inicio de sesión a la aplicación.

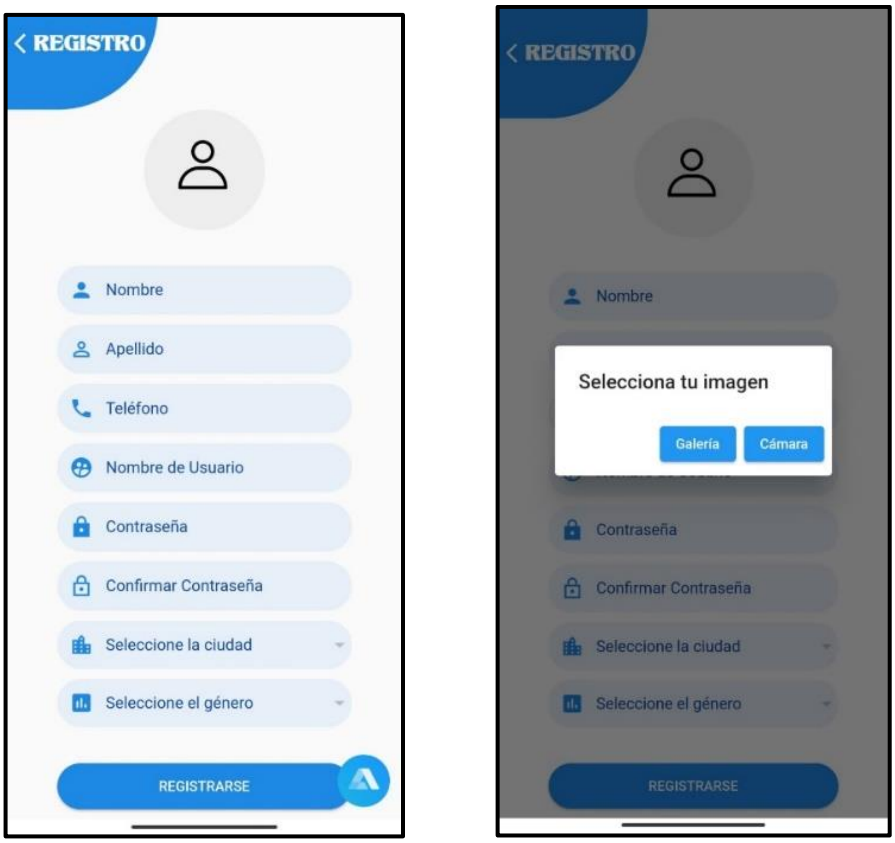


Figura 42 Interfaz de la pantalla de registro y mensaje de selección de la imagen del usuario
Fuente: Investigador

Módulo Pantalla Principal

Al iniciar sesión se presenta la pantalla principal la cual contiene el plano completo del lugar como el botón del asistente virtual Alan en la parte inferior derecha de la pantalla, dicho botón no se mantiene estático ya que puede cambiar de posición según el usuario lo desee, en la parte superior izquierda de la pantalla se encuentra ubicado el botón que da paso a los apartados de editar perfil, rutas, reportes (dependiendo el caso), reiniciar mapa y cerrar sesión, así como mostrar la información del usuario que se encuentra usando la aplicación como: nombre, nombre de usuario, número de teléfono y foto. El mapa presentado en la pantalla principal se puede ampliar o alejar como si se tratase de una foto sin embargo al ampliarlo va mostrando información de los lugares representados con puntos de color azul, así como también la ubicación de los faros representados con puntos de color rojo, convirtiéndolo en un mapa dinámico.

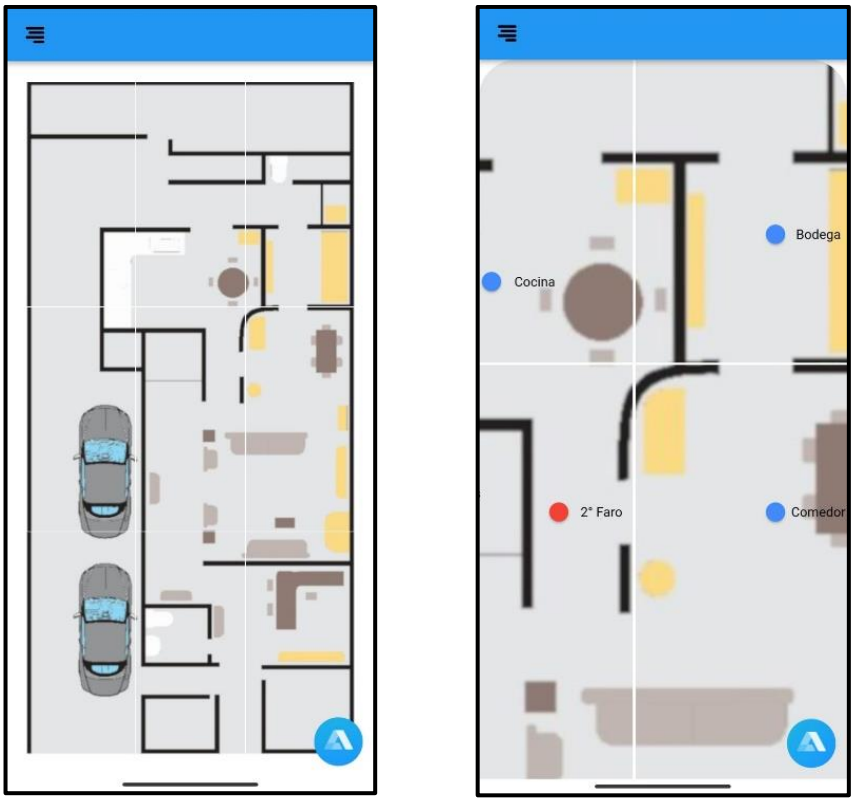


Figura 43 Pantalla principal con el mapa dinámico
Fuente: Investigador

Modulo Editar Perfil

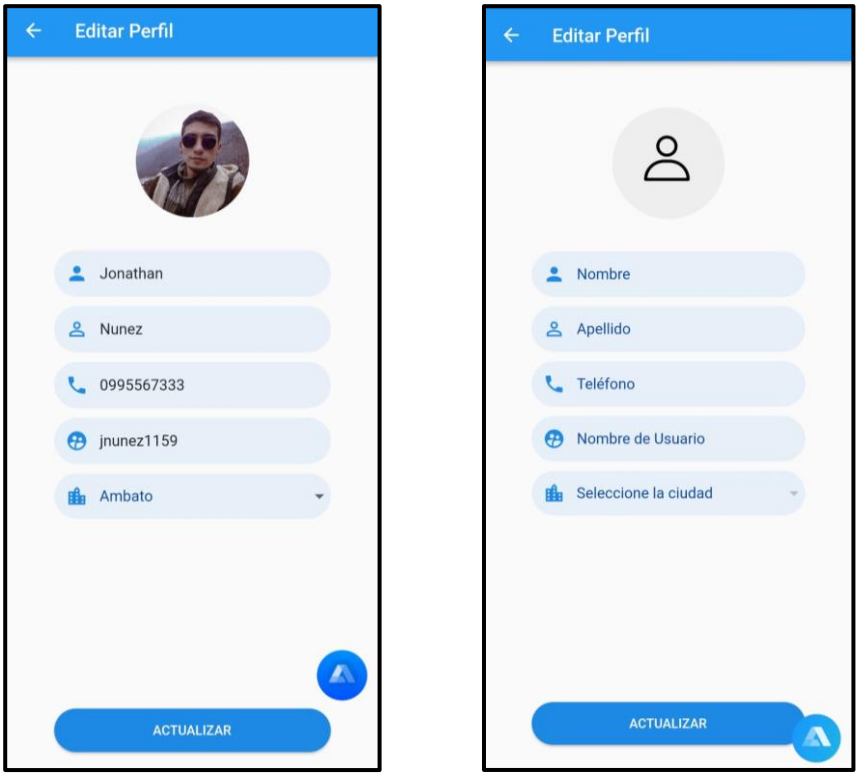


Figura 44 Interfaz de la pantalla editar perfil
Fuente: Investigador

El apartado Editar Perfil está destinado en caso de que el usuario haya ingresado mal alguno de sus datos o desee modificar su información ingresada al inicio de sesión, los datos que pueden ser modificados son los de nombre, apellido, teléfono, nombre de usuario, ciudad y foto de perfil mismos que serán cambiados en la base de datos al momento de presionar la opción actualizar.

Modulo Rutas

La página de rutas muestra apartados con cada sitio de interés para el usuario, dentro de cada uno se encuentran enlistados los lugares a los que puede dirigirse a partir de ahí. Los sitios enlistados cuentan con una pequeña descripción y con el número de ruta correspondiente, cada ruta tiene un numero único, se cuenta con un total de 132 rutas establecidas. Encontrar la ruta puede hacerse de dos formas, la primera en donde el usuario puede explorar por las listas de rutas y la segunda en donde el usuario hace uso del buscador que se encuentra en la parte superior donde se ingresa el dato de donde se encuentra y el lugar a donde se quiere dirigir facilitando así encontrar la ruta correspondiente.

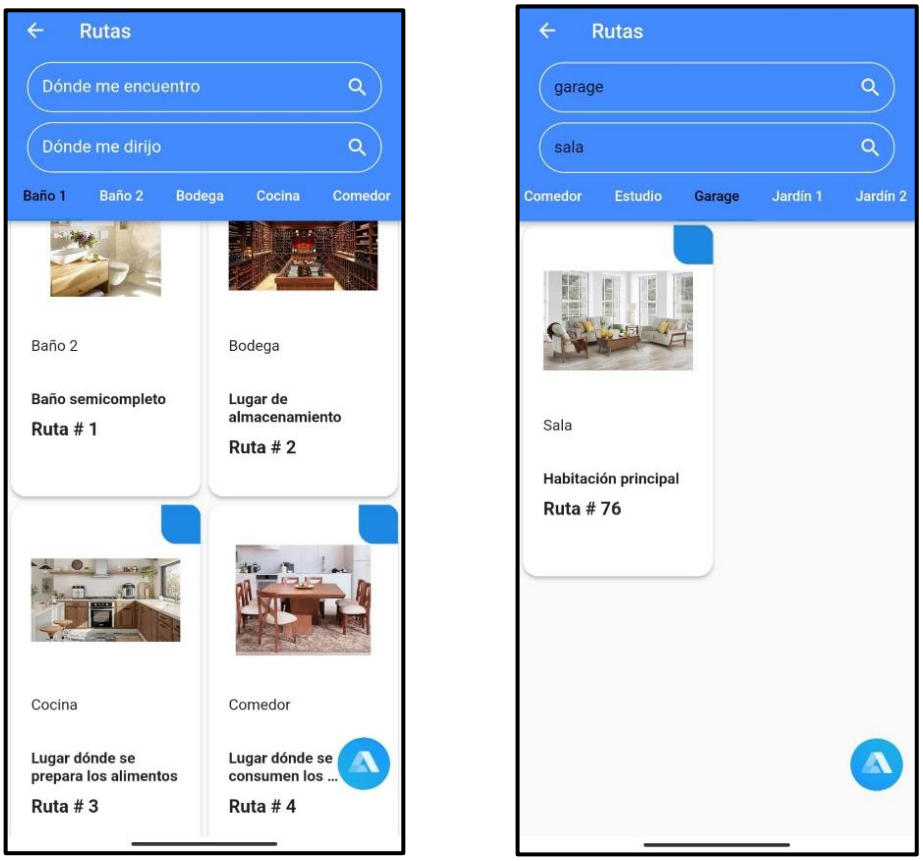


Figura 45 Interfaz de la pantalla de rutas y función del buscador dinámico
Fuente: Investigador

Al tratarse de un buscador dinámico va comparando los caracteres ingresados con las listas de los lugares y en el caso de escribir mal o ingresar el nombre de un lugar que no existe en la lista, esta muestra una imagen al usuario avisando que la ruta ingresada no existe.

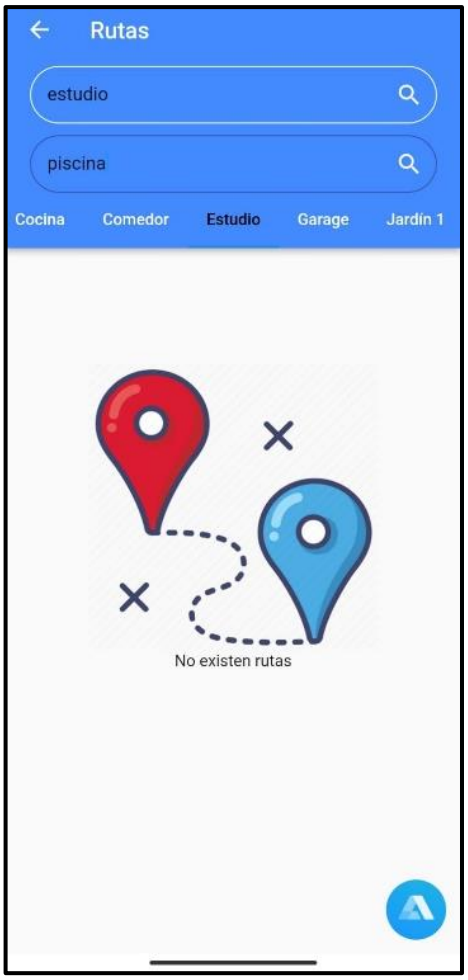


Figura 46 Mensaje de error al no encontrar rutas
Fuente: Investigador

Una vez se haya encontrado la ruta existen dos formas de activar el mapa dinámico el cual va dibujando la ruta a seguir desde el lugar donde se encuentra el usuario hasta el lugar de destino ingresado por el mismo, la primera formar es presionando sobre la imagen de la ruta y la segunda es pedírselo al asistente virtual con el numero arrojado en la búsqueda.

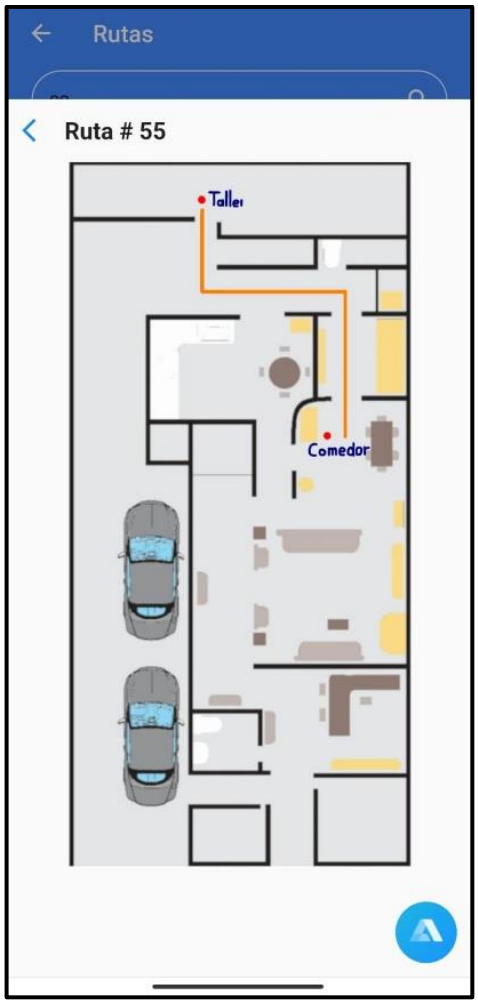


Figura 47 Interfaz de la pantalla que traza la ruta establecida
Fuente: Investigador

Modulo Reportes

La página de reportes solo se encuentra disponible para el usuario administrador y en esta se muestra información de los tres sitios más visitados y los tres usuarios que más han realizado consultas en la aplicación, esta información se muestra de mayor a menor con el número correspondiente de visitas a cada uno, esta información es obtenida de la base de datos en donde se registra las visitas realizadas a cada sitio así como los usuarios que realizan consultas con la aplicación, realizando una comparativa y arrojando los datos necesarios para construir el reporte. El objetivo principal de la página reportes es la de mostrar una perspectiva del lugar para tomar acción o gestionar de mejor manera el espacio, ya sea aplicando estrategias de marketing, mejorando la infraestructura de los sitios a los que el usuario muestra un mayor interés o mejorar los sitios a los que desee que el usuario visite más, todo esto depende del enfoque que le dé el administrador.

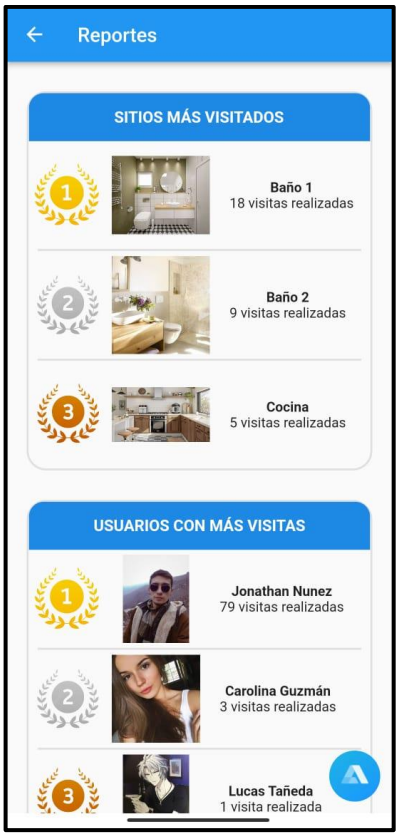


Figura 48 Interfaz de la página de reportes
Fuente: Investigador

Botón Reiniciar mapa

Figura 49 Función del botón para reiniciar el mapa
Fuente: Investigador

La función del botón reiniciar mapa permite al usuario reestablecer la posición del mapa a la posición inicial, esto con el fin de evitar confusión al momento de navegar o en caso de presentar algún problema con la carga del mapa.

Botón Cerrar sesión

Como su nombre lo indica la función de este botón es la de finalizar la sesión del usuario, una vez activada se redirige a la página de login dando la opción de ingresar nuevamente o con otro usuario.

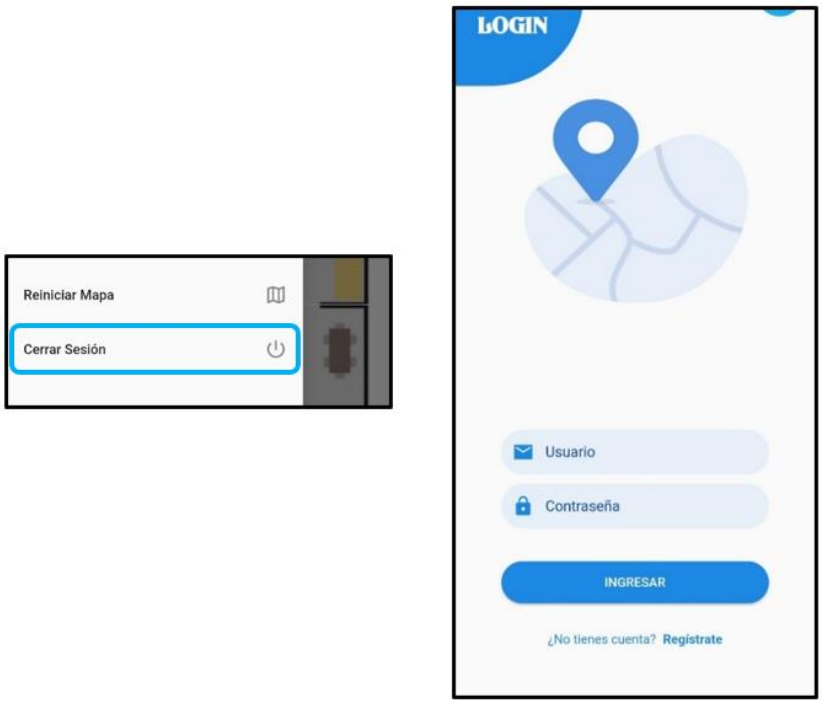


Figura 50 Función del botón para cerrar sesión
Fuente: Investigador

Despliegue de notificaciones

Las notificaciones son enviadas al dispositivo del usuario una vez se haya detectado la señal de los faros, estas conforman parte de los widgets programados dentro de Flutter y son activadas mediante el asistente virtual en la acción de escáner.

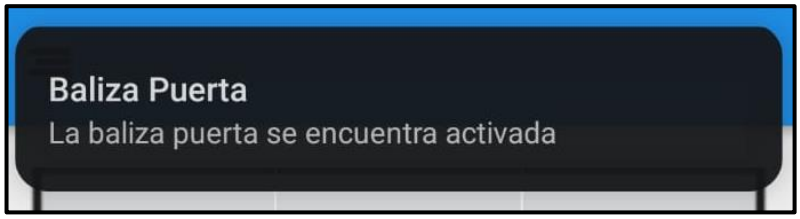


Figura 51 Despliegue de la notificación al detectar un faro
Fuente: Investigador

Descargar la aplicación

Para el proceso de descarga de la aplicación se generó un código QR que redirige a la página de descarga, con el fin que el proceso sea rápido y sencillo. Además, que los códigos QR pueden ser impresos en folletos o carteles para promover el uso y acceso rápido a la aplicación.

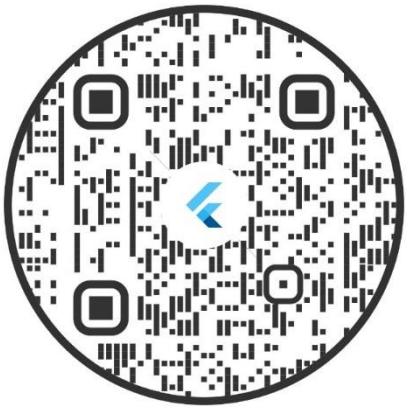


Figura 52 Código QR de descarga de la aplicación
Fuente: Investigador

3.2.3. Manual de usuario

Se realizó un manual de usuario en donde se explica de forma detallada el uso de la aplicación móvil, así como los requerimientos de esta, guiando al usuario desde el proceso de descarga he instalación. Ayudando así a familiarizarse con las funciones que ofrece el aplicativo. Aquella guía se encuentra especificada en la sección de anexos.

3.2.4. Pruebas de funcionamiento del asistente virtual para navegación dentro de un espacio cerrado mediante generación de rutas.

Una vez concluido el prototipo de asistente virtual de navegación y generación de rutas para espacios cerrados se procede a realizar pruebas de funcionamiento de la aplicación móvil, estas pruebas se las realiza por partes para verificar que todos los procesos de la aplicación móvil funcionen de manera correcta.

3.2.4.1.Prueba de registro de usuario y almacenamiento en la base de datos

Para verificar que los datos ingresados en la página de registro de usuario se guardan correctamente dentro de la base de datos del servidor se realizó pruebas de registro en donde ingreso los datos de un nuevo usuario.

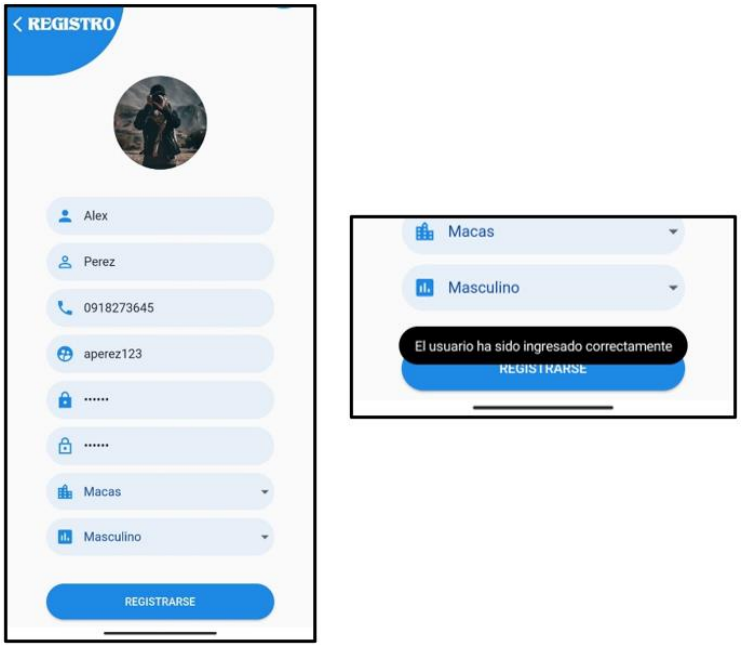


Figura 53 Prueba de registro de usuario
Fuente: Investigador

Una vez llenados los datos y pulsado en el botón de registrarse, la aplicación muestra un mensaje notificando que los datos han sido ingresados correctamente. Esto se comprueba accediendo a la base de datos y observando el nuevo usuario creado.

	ID_USU	NOM_USU	APE_USU	TEL_USU	NOM_USU_LOG	CON_USU	IMG_USU	ID_GEN_PER	ID_CIU_PER	ID_ROL_PER	FECHA_RE...
▶	63	Carolina	Guzmán	0987654321	cguzman1234	8d969eef6e...		2	1803	2	2022-12-08
	65	Luis	Perez	0987654321	lperez4526	8d969eef6e...		1	1801	2	2022-12-14
	66	Lucas	Tañeda	0987654321	ltaneda1234	8d969eef6e...		1	1802	2	2022-12-14
	69	Amuro	Toru	0987654321	atoru1234	8d969eef6e...		1	1801	2	2022-12-14
	70	Jonathan	Nunez	0995567333	jnunez1159	8d969eef6e...		1	1801	1	2022-12-14
	75	Carmen	Bonilla	0999999999	cbonilla1234	8d969eef6e...		2	1701	2	2023-01-15
	77	Melanie	Nunez	0999999999	mnunez1234	8d969eef6e...		2	1802	2	2023-01-17
	78	Lucas	Castillo	0987654321	lcastillo1234	8d969eef6e...		1	1701	2	2023-01-21
	79	Daniel	Bonilla	0995567333	dbonilla1234	8d969eef6e...		1	1802	2	2023-01-21
	80	Alex	Perez	0918273645	aperez123	8d969eef6e...		1	1401	2	2023-01-22
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 54 Usuario nuevo reflejado dentro de la base de datos
Fuente: Investigador

Al iniciar sesión con el nuevo usuario dentro de la aplicación se toman los datos ingresados en el registro y se los muestra en la pantalla de opciones.

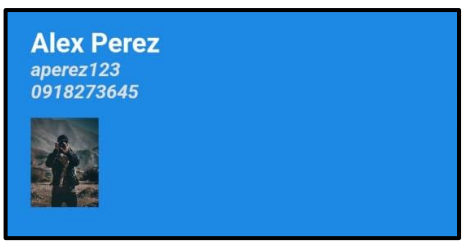


Figura 55 Datos del registro mostrados en la aplicación
Fuente: Investigador

3.2.4.2.Prueba de editar perfil y modificación en la base de datos

Para la prueba de edición de los datos se tomó como referencia el mismo usuario creado, al cual se modificó los datos de número de teléfono, ciudad y foto de perfil.

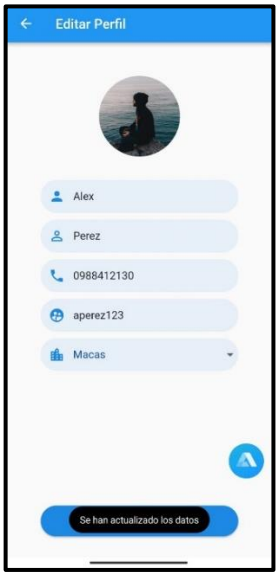


Figura 56 Prueba de edición del perfil
Fuente: Investigador

Una vez actualizados los datos estos se ven reflejados en la base de datos.

	ID_USU	NOM_USU	APE_USU	TEL_USU	NOM_USU_LOG	CON_USU	IMG_USU	ID_GEN_PER	ID_CIU_PER	ID_ROL_PER	FECHA_RE...
	63	Carolina	Guzmán	0987654321	cguzman1234	8d969eef6e...		2	1803	2	2022-12-08
	65	Luis	Perez	0987654321	lperez4526	8d969eef6e...		1	1801	2	2022-12-14
	66	Lucas	Tañeda	0987654321	ltaneda1234	8d969eef6e...		1	1802	2	2022-12-14
	69	Amuro	Toru	0987654321	atoru1234	8d969eef6e...		1	1801	2	2022-12-14
	70	Jonathan	Nunez	0995567333	jnunez1159	8d969eef6e...		1	1801	1	2022-12-14
	75	Carmen	Bonilla	0999999999	cbonilla1234	8d969eef6e...		2	1701	2	2023-01-15
	77	Melanie	Nunez	0999999999	mnunez1234	8d969eef6e...		2	1802	2	2023-01-17
	78	Lucas	Castillo	0987654321	lcastillo1234	8d969eef6e...		1	1701	2	2023-01-21
	79	Daniel	Bonilla	0995567333	dbonilla1234	8d969eef6e...		1	1802	2	2023-01-21
▶	80	Alex	Perez	0988412130	aperez123	8d969eef6e...		1	1401	2	2023-01-22
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 57 Datos actualizados dentro de la base de datos
Fuente: Investigador

Al regresar a la aplicación se pueden ver los datos actualizados en la parte del menú de opciones.

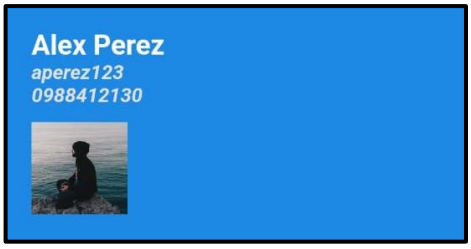


Figura 58 Datos actualizados mostrados en la aplicación
Fuente: Investigador

3.2.4.3.Pruebas de detección de Faros a través de comando de voz

Las pruebas de detección de los faros se las realizado en la parte central y la parte lateral izquierda del lugar debido a que estas zonas son las que presentan la mayor cantidad de rutas.

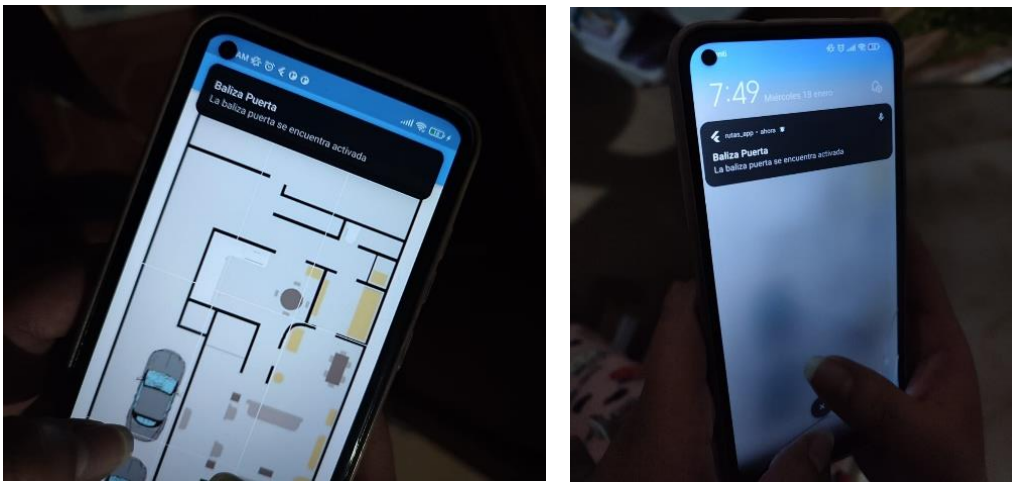


Figura 59 Pruebas de detección de los Faros
Fuente: Investigador

Los resultados obtenidos se muestran en la siguiente tabla en donde se detalla el lugar de prueba el número de faros detectados y el orden en que aparecen las notificaciones.

Tabla 18 Resultados de las pruebas de detección de faros

N°	Lugar de prueba	1° Faro	2° Faro	3° Faro	Primera notificación	Segunda notificación	Tercera notificación
1	Parte inferior	Detectado	Detectado	No Detectado	1° Faro	2° Faro	-
2	Parte central	Detectado	Detectado	Detectado	2° Faro	1° Faro	3° Faro
3	Parte superior	No detectado	Detectado	Detectado	3° Faro	2” Faro	-
4	Parte inferior izquierda	Detectado	No detectado	Detectado	1° Faro	3° Faro	-
5	Parte central izquierda	Detectado	Detectado	Detectado	3° Faro	2° Faro	1° Faro
6	Parte superior izquierda	No Detectado	Detectado	Detectado	3° Faro	2° Faro	-

Fuente: Investigador

En la siguiente tabla de muestra los valores de RSS medidos tras las pruebas de detección de Faros.

Tabla 19 Valor RSS en las pruebas realizadas

N° de prueba	Distancia hasta 1° Faro	RSS 1° Faro	Distancia hasta 2° Faro	RSS 2° Faro	Distancia hasta 3° Faro	RSS 3° Faro
1	3 m	-77,16 dBm	11 m	-93,24 dBm		-
2	10 m	-92,73 dBm	2 m	-75,06 dBm	11 m	-95,18 dBm
3		-	12 m	-92,53 dBm	2 m	-70,23 dBm
4	5.5 m	-83,15 dBm		-	10 m	-94,88 dBm
5	12 m	-95,06 dBm	10 m	-89,19 dBm	11 m	-92,19 dBm
6		-	12 m	-93,41 dBm	2 m	-62,03 dBm

Fuente: Investigador

En el siguiente gráfico de dispersión se puede observar los puntos típicos a donde tiende cada faro ya ubicado en el lugar, esta dispersión se debe a que cada dispositivo se encuentra en condiciones diferentes del espacio sin embargo su relación de distancia y RSSI oscila entre los valores calculados. El rango de perdidas aceptables para que los faros sean detectados van desde los -60 dBm hasta los -96 dBm ya que a partir de ese rango el receptor ya no detecta la señal.

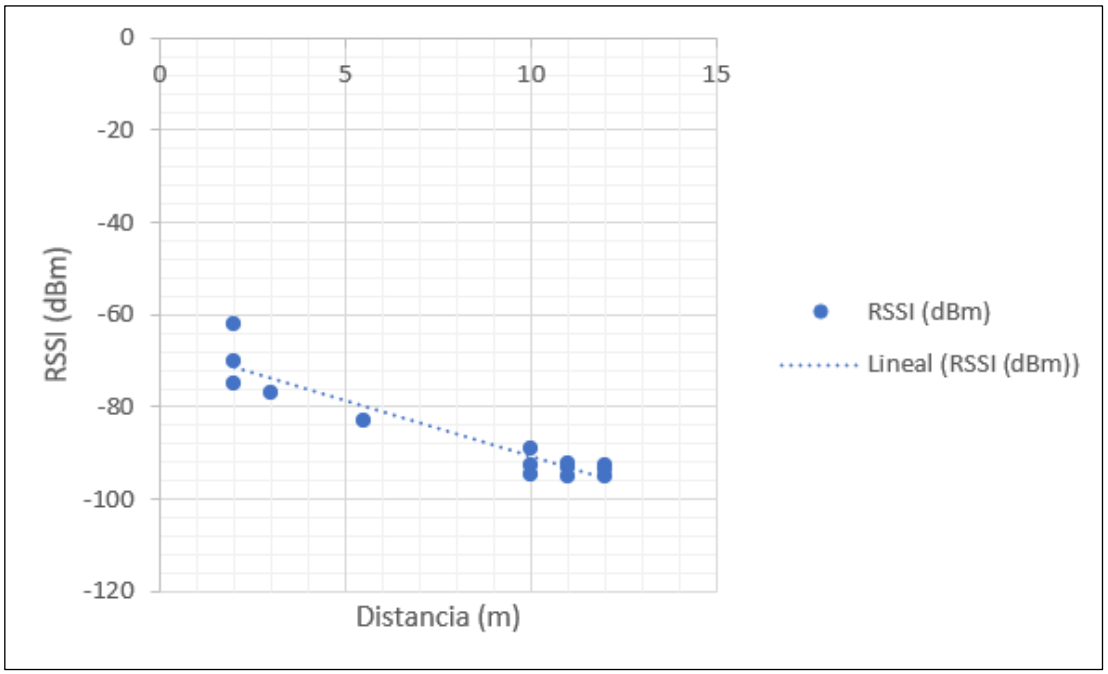


Figura 60 Gráfica de dispersión de la señal RSSI en relación con la distancia.
Fuente: Investigador

3.2.4.4.Pruebas de navegación y generación de rutas

Las pruebas de navegación y generación de rutas se las realiza dentro del espacio en donde se usa la ayuda del asistente virtual, así como la del buscador para generar la ruta a seguir.

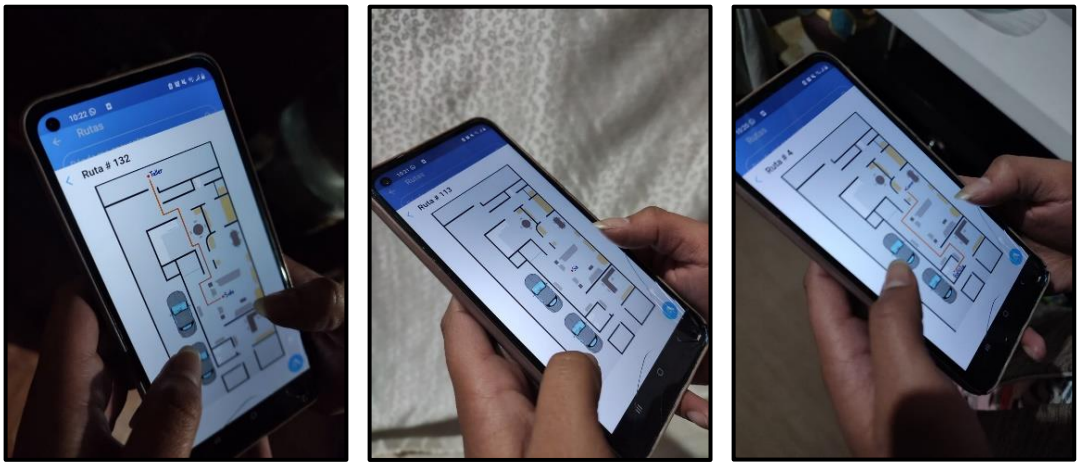


Figura 61 Pruebas de navegación y generación de rutas
Fuente: Investigador

Los resultados obtenidos se muestran en la siguiente tabla en donde se detalla el lugar de inicio y destino ingresados para las pruebas con el número de ruta generado y la respuesta tanto del asistente como la del generador de rutas.

Tabla 20 Resultados de las pruebas de navegación y generación de rutas

N°	Lugar inicio	Lugar destino	Número de ruta	Respuesta asistente	Ruta generada	Tiempo (s)	Efectividad (%)
1	Bodega	Sala	32	Si	Si	8	100
2	Cocina	Sala	43	Si	Si	6	100
3	Estudio	Baño 2	57	Si	Si	13	100
4	Garaje	Bodega	69	Si	Si	12	100
5	Baño 1	Jardín 2	30	Si	Si	5	100
6	Taller	Estudio	127	Si	Si	14	100
7	Jardín 3	Cocina	103	Si	Si	5	100
8	Sala	Taller	121	Si	Si	10	100
9	Baño 1	Comedor	4	Si	Si	10	100
10	Jardín 2	Baño 1	89	Si	Si	5	100
11	Jardín 1	Jardín 3	86	Si	Si	10	100
12	Taller	Garaje	128	Si	Si	10	100
13	Sala	Jardín 1	118	Si	Si	9	100
14	Jardín 3	Estudio	105	Si	Si	10	100
15	Garaje	Sala	76	Si	Si	10	100
16	Estudio	Jardín 3	64	Si	Si	13	100
17	Jardín 2	Estudio	94	Si	Si	5	100

18	Comedor	Baño 1	45	Si	Si	10	100
19	Baño 2	Estudio	16	No	Si	15	75
20	Jardín 1	Sala	87	Si	Si	8	100
21	Cocina	Baño 1	34	Si	Si	9	100
22	Bodega	Comedor	26	Si	Si	5	100
23	Baño 1	Estudio	5	Si	Si	4	100
24	Comedor	Taller	55	Si	Si	10	100
25	Bodega	Sala	32	Si	Si	8	100
26	Taller	Cocina	125	Si	Si	5	100
27	Jardín 2	Taller	99	Si	Si	15	100
28	Cocina	Comedor	37	Si	Si	5	100
29	Baño 2	Sala	21	Si	Si	9	100
30	Jardín3	Jardín 1	107	Si	Si	9	100
31	Garaje	Baño 1	67	Si	Si	7	100
32	Estudio	Garaje	61	Si	Si	9	100
33	Taller	Comedor	126	Si	Si	7	100
34	Sala	Bodega	113	Si	Si	6	100
35	Comedor	Jardín 2	52	Si	Si	10	100
36	Baño 1	Baño 2	1	No	Si	15	75
37	Cocina	Garaje	39	Si	Si	11	100
38	Bodega	Jardín 1	29	Si	Si	13	100
39	Jardín 1	Estudio	83	Si	Si	5	100
40	Baño 1	Garaje	6	Si	Si	8	100
41	Comedor	Sala	54	Si	Si	6	100
42	Taller	Jardín 2	130	Si	Si	15	100
43	Estudio	Jardín 3	64	Si	Si	13	100
44	Sala	Cocina	114	Si	Si	6	100
45	Garaje	Baño 2	68	Si	Si	10	100
46	Baño 2	Taller	22	No	Si	5	75
47	Jardín 3	Taller	110	Si	Si	9	100
48	Bodega	Garaje	28	Si	Si	10	100
49	Taller	Baño 1	122	Si	Si	13	100
50	Cocina	Sala	43	Si	Si	6	100

Fuente: Investigador

3.2.4.5.Pruebas de estrés

Se usó la herramienta Apache JMeter para realizar pruebas de carga y rendimiento a la API en Node.js. Para esto se seleccionó la opción de bucle sin fin para realizar el mayor de número de solicitudes posibles.

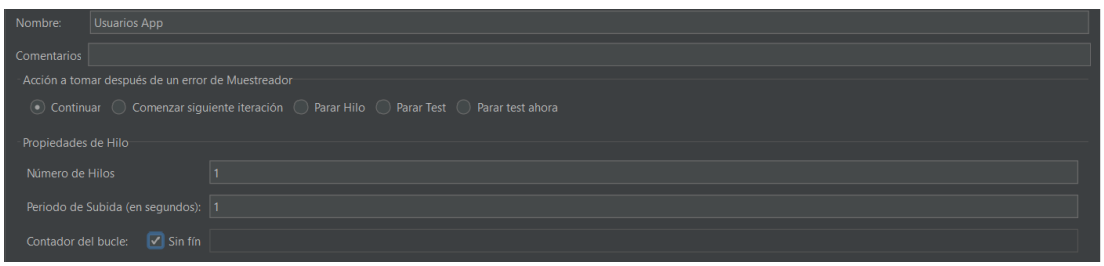


Figura 62 Prueba de rendimiento de la API

Fuente: Investigador

Para iniciar la prueba se configuró el protocolo HTTP la dirección url del servidor (API), el puerto 3000 que es donde se ejecuta el método HTTP (GET) y la ruta de la API a solicitar.

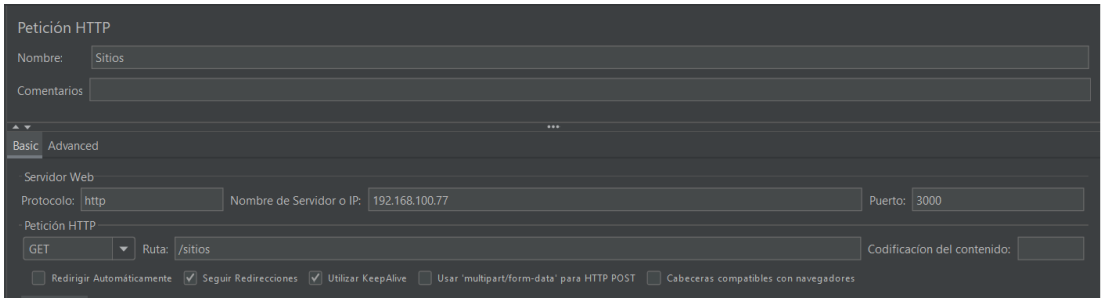


Figura 63 Configuración de los parámetros para las pruebas
Fuente: Investigador

Una vez iniciada la prueba se crean muestra con el fin de saturar la API, creando un total de 2943 muestras exitosas, teniendo su punto de quiebre en la muestra número 2944.

Muestra #	Tiempo de comie...	Nombre del hilo	Etiqueta	Tiempo de Muestr...	Estado	Bytes	Sent Bytes	Latency	Connect Time(ms)
2925	21:40:47.343	Usuarios App 1-1	Sitios	18		2344	127	18	0
2926	21:40:47.361	Usuarios App 1-1	Sitios	11		2344	127	11	0
2927	21:40:47.372	Usuarios App 1-1	Sitios	10		2344	127	10	0
2928	21:40:47.382	Usuarios App 1-1	Sitios	8		2344	127	8	0
2929	21:40:47.391	Usuarios App 1-1	Sitios	9		2344	127	9	0
2930	21:40:47.400	Usuarios App 1-1	Sitios	9		2344	127	9	0
2931	21:40:47.409	Usuarios App 1-1	Sitios	8		2344	127	8	0
2932	21:40:47.418	Usuarios App 1-1	Sitios	8		2344	127	8	0
2933	21:40:47.426	Usuarios App 1-1	Sitios	9		2344	127	9	0
2934	21:40:47.435	Usuarios App 1-1	Sitios	8		2344	127	8	0
2935	21:40:47.444	Usuarios App 1-1	Sitios	8		2344	127	8	0
2936	21:40:47.452	Usuarios App 1-1	Sitios	8		2344	127	8	0
2937	21:40:47.461	Usuarios App 1-1	Sitios	7		2344	127	7	0
2938	21:40:47.469	Usuarios App 1-1	Sitios	8		2344	127	8	0
2939	21:40:47.478	Usuarios App 1-1	Sitios	9		2344	127	9	0
2940	21:40:47.488	Usuarios App 1-1	Sitios	12		2344	127	12	0
2941	21:40:47.500	Usuarios App 1-1	Sitios	11		2344	127	11	0
2942	21:40:47.511	Usuarios App 1-1	Sitios	10		2344	127	10	0
2943	21:40:47.522	Usuarios App 1-1	Sitios	9		2344	127	9	0
2944	21:40:47.532	Usuarios App 1-1	Sitios	5		2701	0	0	0

Scroll automatically? Child samples? No. de Muestras 2944 Última Muestra 5 Media 9 Desviación 4

Figura 64 Número total de hilos que soporta la API
Fuente: Investigador

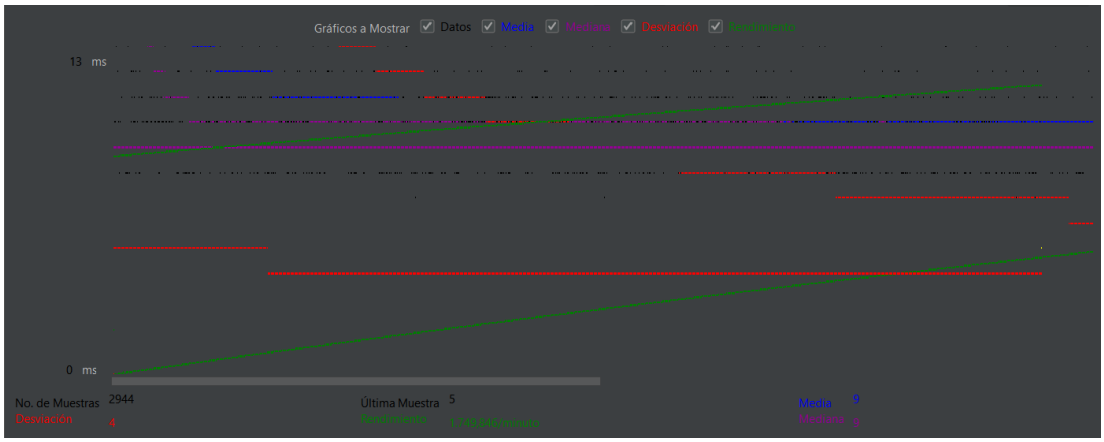


Figura 65 Resultados de las pruebas de rendimiento
Fuente: Investigador

El número de muestras (2944) indica que se realizaron muchas solicitudes en la prueba. Cuanto mayor sea el número de muestras, más precisa será la medición del rendimiento.

La última muestra registró un tiempo de respuesta de 5 segundos, lo que sugiere que hubo al menos una solicitud que tardó más de lo esperado en completarse.

La media de tiempo de respuesta fue de 9 segundos, lo que indica que la mayoría de las solicitudes tuvieron un tiempo de respuesta aceptable.

La desviación estándar de 4 indica que los tiempos de respuesta variaron ampliamente. Es posible que haya algunos casos extremos en los que el tiempo de respuesta fue mucho más largo o corto que el promedio.

El rendimiento de 1.749,846/minuto es un buen resultado y sugiere que la aplicación probada puede manejar una carga significativa de usuarios y solicitudes de manera eficiente.

La mediana de 9 segundos es similar a la media, lo que indica que no hubo una cantidad significativa de solicitudes con tiempos de respuesta extremadamente largos o cortos.

3.2.4.6.Pruebas con usuarios simultáneos

Para realizar una prueba de carga con JMeter, es necesario configurar un plan de prueba que incluya una serie de hilos (threads) que simulan usuarios que interactúan con la API. Cada hilo envía solicitudes a la API y mide el tiempo de respuesta y el rendimiento. A medida que se agregan más hilos, se aumenta la carga en la API y se puede determinar el número máximo de usuarios simultáneos que la API puede manejar sin degradar significativamente su rendimiento.

Para esta prueba de utilizó 1000 hilos, lo que sugiere una carga pesada a la API.

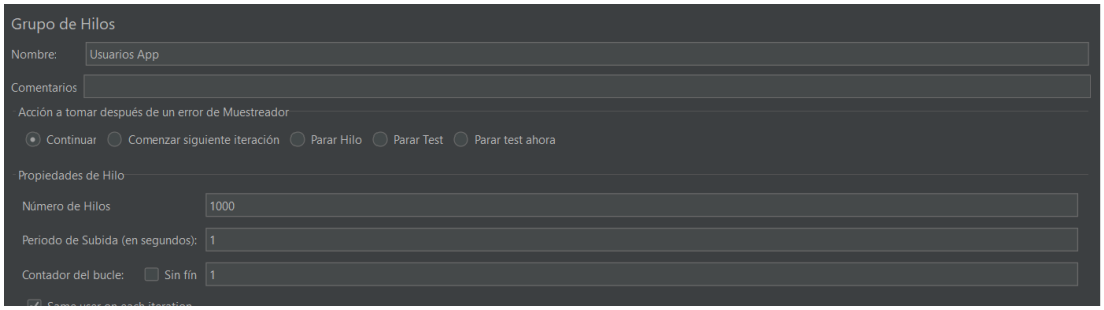
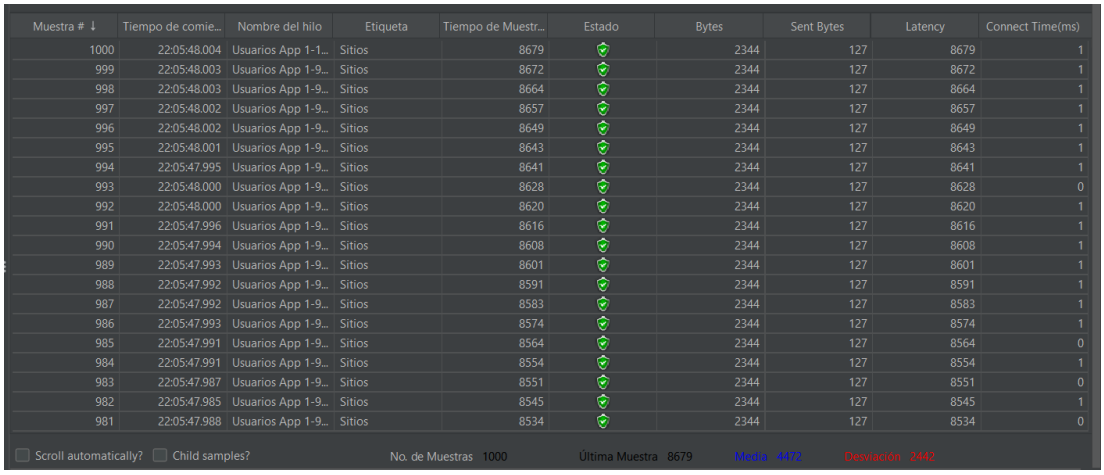


Figura 66 Configuración para pruebas con usuarios simultáneos
Fuente: Investigador



Muestra #	Tiempo de comie...	Nombre del hilo	Etiqueta	Tiempo de Muestr...	Estado	Bytes	Sent Bytes	Latency	Connect Time(ms)
1000	22:05:48.004	Usuarios App 1-1...	Sitios	8679		2344	127	8679	1
999	22:05:48.003	Usuarios App 1-9...	Sitios	8672		2344	127	8672	1
998	22:05:48.003	Usuarios App 1-9...	Sitios	8664		2344	127	8664	1
997	22:05:48.002	Usuarios App 1-9...	Sitios	8657		2344	127	8657	1
996	22:05:48.002	Usuarios App 1-9...	Sitios	8649		2344	127	8649	1
995	22:05:48.001	Usuarios App 1-9...	Sitios	8643		2344	127	8643	1
994	22:05:47.995	Usuarios App 1-9...	Sitios	8641		2344	127	8641	1
993	22:05:48.000	Usuarios App 1-9...	Sitios	8628		2344	127	8628	0
992	22:05:48.000	Usuarios App 1-9...	Sitios	8620		2344	127	8620	1
991	22:05:47.996	Usuarios App 1-9...	Sitios	8616		2344	127	8616	1
990	22:05:47.994	Usuarios App 1-9...	Sitios	8608		2344	127	8608	1
989	22:05:47.993	Usuarios App 1-9...	Sitios	8601		2344	127	8601	1
988	22:05:47.992	Usuarios App 1-9...	Sitios	8591		2344	127	8591	1
987	22:05:47.992	Usuarios App 1-9...	Sitios	8583		2344	127	8583	1
986	22:05:47.993	Usuarios App 1-9...	Sitios	8574		2344	127	8574	1
985	22:05:47.991	Usuarios App 1-9...	Sitios	8564		2344	127	8564	0
984	22:05:47.991	Usuarios App 1-9...	Sitios	8554		2344	127	8554	1
983	22:05:47.987	Usuarios App 1-9...	Sitios	8551		2344	127	8551	0
982	22:05:47.985	Usuarios App 1-9...	Sitios	8545		2344	127	8545	1
981	22:05:47.988	Usuarios App 1-9...	Sitios	8534		2344	127	8534	0

Figura 67 Resultado de cada hilo
Fuente: Investigador

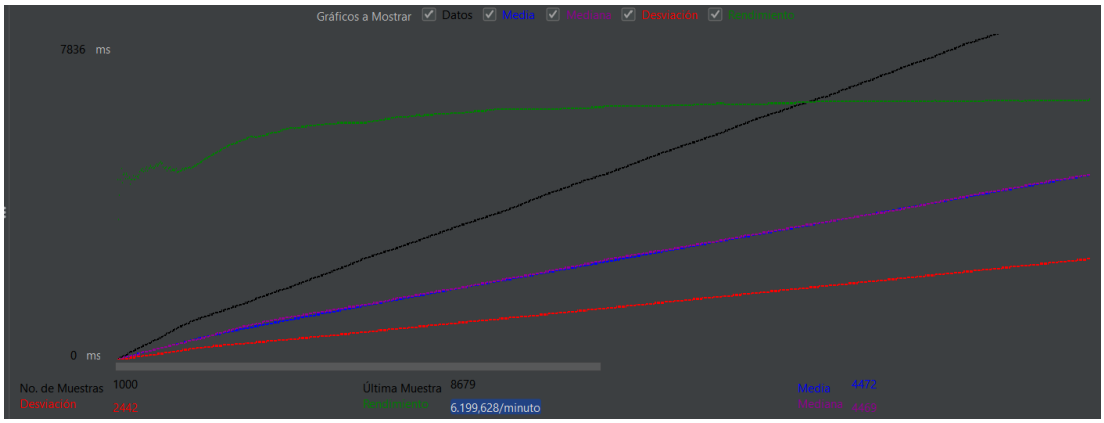


Figura 68 Resultado de las pruebas con usuarios simultáneos
Fuente: Investigador

La última muestra registró un tiempo de respuesta de 8697ms, lo que sugiere que al menos una solicitud tardó más de lo esperado en completarse.

La media de tiempo de respuesta fue de 4472ms, lo que indica que en promedio la API respondió en un tiempo razonable.

La desviación estándar de 2442 indica que hubo una variabilidad significativa en los tiempos de respuesta. Es posible que hayan existido solicitudes extremas que tardaron mucho más o mucho menos que el promedio.

El rendimiento de 6.199,628/minuto sugiere que la API puede manejar una carga significativa de usuarios y solicitudes de manera eficiente.

La mediana de 4469ms es similar a la media, esto indica que no hubo muchos casos extremos de tiempos de respuesta extremadamente largos o cortos.

Basándonos en los datos proporcionados, podemos concluir que la API puede soportar 1000 peticiones en simultáneo sin afectar significativamente el tiempo medio de respuesta.

3.2.5. Análisis de resultados.

En relación con los resultados obtenidos en las pruebas de escaneo de los faros se observa que en cada lugar la aplicación detecta al menos 2 Beacons mostrando como preferencia la notificación del faro ubicado en esa zona y mostrando la notificación del segundo faro después de tres segundos, sin embargo en la zona central de la casa la señal de los tres Beacons es reconocida y de igual forma las notificaciones se muestran en forma prioritaria según la ubicación de las pruebas, determinando así que cada zona está cubierta por al menos dos Beacons lo que demuestra que con tres dispositivos se logra cubrir un espacio de 28m de largo con 9m de ancho es decir un total de $252\ m^2$.

En base a esta premisa se hace una aproximación del terreno a 30m de largo y 10 m de ancho. Teniendo como resultado que cada Beacons abarca un área de $100\ m^2$. Esta información es útil para la aplicación del prototipo en otros espacios.

En la siguiente tabla, se muestra la cantidad de Beacons a usar en relación con el área del terreno. Cabe recalcar que este es un valor referencial ya que estos datos pueden verse afectados según la estructura del lugar a implementar.

Tabla 21 Número referencial de Beacons a usar según el área del terreno

Número de Beacons	Área del terreo
3	$300\ m^2$
4	$400\ m^2$
5	$500\ m^2$

6	$600\ m^2$
7	$700\ m^2$
8	$800\ m^2$
9	$900\ m^2$

Fuente: Investigador

Tras el análisis de las pruebas de navegación y generación de rutas se obtienen los siguientes resultados:

$$Eficacia_{asistente} = Resultado\ alcanzado * \frac{100}{Resultado\ previsto}$$

$$Eficacia_{asistente} = 47 * \frac{100}{50} = 94\ \%$$

$$Eficacia_{buscador} = Resultado\ alcanzado * \frac{100}{Resultado\ previsto}$$

$$Eficacia_{buscador} = 50 * \frac{100}{50} = 100\ \%$$

$$Eficacia_{Total\ generador\ de\ rutas} = \frac{Eficacia_{asistente} + Eficacia_{buscador}}{2}$$

$$Eficacia_{Total\ generador\ de\ rutas} = \frac{94\% + 100\%}{2} = 97\%$$

En donde se observa que tras 50 pruebas el asistente virtual respondió a 47 de estas, sin embargo, la generación de ruta mediante el buscador no presento ningún problema al mostrar la ruta debido a que no depende de factores externos al de su programación para funcionar, al ser estas dos formas de llegar al mismo resultado una se apoya en la otra teniendo un total de 97% de eficacia al momento de navegar y generar rutas dentro del lugar.

3.2.6. Presupuesto

En esta sección se detalla el costro de implementación del asistente virtual de navegación y generación de rutas en espacios cerrados, donde se tomó en consideración el costo de diseño y de los Beacons utilizados.

Para el costo de diseño de la aplicación se tomó como referencia el sueldo promedio de un ingeniero en telecomunicaciones el cual oscila entre los $700 según la

remuneración mensual por puesto impresa en el Art. 7 de la Ley Orgánica de Transparencia y Acceso a la Información Pública [55].

$$R_{Diaria} = \frac{R_{Mensual}}{Días_{Laborables}} = \frac{\$700}{20} = \$35$$

$$R_{Hora} = \frac{R_{Diaria}}{Horas_{Laborables}} = \frac{\$35}{8} = \$4.37$$

Para el desarrollo del aplicativo móvil con todos los archivos necesarios para su funcionamiento se emplea un tiempo estimado de 140 horas, esto multiplicado por la remuneración hora, da como resultado el presupuesto de diseño.

$$P_{diseño} = R_{Hora} * horas = \$4.37 * 140 = \$611.8$$

Como se muestra en la Tabla 14 de este documento el costo de cada Beacon es de $24.

Tabla 22 Presupuesto de construcción

Presupuesto				
N°	**Descripción**	**Costo unitario**	**Cantidad**	**Costo total**
1	Presupuesto de diseño	611.8	1	611.8
2	Beacons	24	3	72
			Valor Total	683.8

Fuente: Investigador

En aplicaciones futuras estos valores pueden variar dependiendo la calidad y detalle que se le desee incorporar a la construcción de la aplicación móvil, así como el modelo y cantidad de Beacons a utilizar.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se desarrolló una solución de navegación en interiores basada en dispositivos móviles que usa como tecnología de apoyo bluetooth de baja energía. Esto por medio de la aplicación móvil que integra un asistente virtual es capaz de graficar rutas a los usuarios permitiendo la navegación en espacios cerrados.

- Aplicar Beacons actuando como faros de manera distribuida en el plano permite la creación de un sistema de posicionamiento de bajo costo y fácil de usar, esto con la ayuda de la señal RSS que emiten estos dispositivos distinguidos por la dirección Mac de cada uno, las cuales aplicadas en espacios cerrados cubren rangos da hasta $100\,m^2$ por faro, este valor puede variar dependiendo la infraestructura del espacio a aplicarse.

- El asistente virtual creado en Alan Studio como complemento de Flutter se adapta a la aplicación móvil sin problema dando como resultado una eficacia del 95% al momento de reconocer la voz para presentar la ruta, esto debido a que originalmente fue entrenado en el idioma inglés, sin embargo, para solventar esta pérdida, la opción de mostrar rutas por medio del buscador presenta 100% de eficacia debido a que no depende de factores externos de su programación, lo que da como resultado un total de 97% por ciento de eficacia al proceso de graficar rutas dentro de la ampliación móvil.

- La construcción de un mapa dinámico a partir de matrices de imágenes aporta escalabilidad a la aplicación móvil ya que con esto se logra mantener la calidad de imagen en el mapa, así como posibilita la expansión a nuevas áreas, lugares o sitios de interés sin necesidad de reestructurar todas las partes ya creadas, sino más bien añadiendo las nuevas zonas al programa. Esto también ayuda a que la manipulación del mapa sea fluida ya que divide cada proceso en etapas a medida que se va construyendo.

4.2. Recomendaciones

- Se recomienda hacer uso de un framework multiplataforma como Flutter para el desarrollo de la aplicación móvil debido a que con una fuente de código se puede adaptar a demás sistemas operativos, consiguiendo de esta manera abarcar mayor cantidad de usuarios que puedan usar la aplicación.

- Para aplicaciones en edificios u organizaciones se recomienda sacarle más provecho al uso de la tecnología beacon para interactuar con los visitantes, ya que aparte de ofrecer una solución a la navegación en espacios cerrados, también puede mandar notificaciones de ofertas o información importante acerca del lugar, esto dependiendo del caso al que se aplique.

- A medida que va ampliando el sistema de navegación en espacios cerrados y según las áreas u departamentos del lugar en donde se aplique el prototipo se recomienda hacer uso de diferentes tipos de Beacons y no limitarse el uso a una marca especifica, ya que existe variedad de dispositivos que se adaptan a cada necesidad y todos cumplen con su función básica para aplicarlo en proyectos de navegación.

- Para generar una mayor experiencia al usuario es recomendable añadir más acciones al asistente virtual dependiendo el lugar al que se aplique, personalizando así cada visita y expandiendo el uso de la aplicación a un gestor de espacios.

BIBLIOGRAFÍA

[1] Naciones Unidas, «Más de la mitad de la población vive en áreas urbanas y seguirá creciendo,» Naciones Unidas, 10 julio 2014. [En línea]. Available: https://www.un.org/development/desa/es/news/population/world-urbanization-prospects-2014.html#:~:text=Noticias-. [Último acceso: 12 septiembre 2022].

[2] La Hora, «Edificios inteligentes'son la nueva tendencia,» La Hora, 26 mayo 2006. [En línea]. Available: https://www.lahora.com.ec/cualquier-cosa-1/edificios-inteligentes-son-la-nueva-tendencia/#:~:text=El%20a%C3%B1o%20pasado%2C%20gan%C3%B3%20este. [Último acceso: 13 septiembre 2022].

[3] J. Garcés y F. Rubio, «Desarrollo de un asistente virtual móvil para potencializar la experiencia Turística Arquitectónica Patrimonial de la ciudad de Latacunga,» Universidad de las Fuerzas Armadas, Latacunga, 2018.

[4] A. Reyes, «Sistema de navegación en interiores por medio de dispositivos móviles y emisores de señal bluetooth,» Tecnológico Nacional de México, Tijuana, 2018.

[5] G. López y M. Rebollar, «Generación automática de rutas de Navegación en interiores utilizando la tecnología de beacons,» Tecnológico Nacional de México, Tijuana, 2019.

[6] F. Garibay, «Diseño e implementación de un asistente virtual (chatbot) para ofrecer atención a los clientes de una aerolínea mexicana por medio de sus canales conversacionales,» INFOTEC, Ciudad de México, 2020.

[7] H. Ibáñez, «Desarrollo de un chatbot para la recomendación de eventos o lugares de interés,» Universidad Politécnica de Valéncia, Valéncia, 2020.

[8] B. M. Flax, «Intelligent buildings,» *IEEE Communications Magazine,* vol. 29, nº 4, pp. 24-27, Abril 1991.

[9] A. H. Buckman, M. Mayfield y B. M. Beck, «What is a Smart Building?,» *Smart and Sustainable Built Environment,* vol. 3, nº 2, pp. 92-109, 2014.

[10] R. M. Alkan, H. Karaman y M. Sahin, «GPS, GALILEO and GLONASS satellite navigation systems & GPS modernization,» *Proceedings of 2nd International Conference on Recent Advances in Space Technologies*, pp. 390-394, 2005.

[11] F. Vejrazka, «Galileo and the Other Satellite Navigation Systems,» *17th International Conference Radioelektronika*, pp. 1-4, 2007.

[12] J. Spilker, Datos de navegación GPS. Sistema de posicionamiento global: teoría y aplicaciones, vol. 1, 1996, pp. 121-176.

[13] P. Puricer y P. Kovar, «Technical Limitations of GNSS Receivers in Indoor Positioning,» *Technical Limitations of GNSS Receivers in Indoor Positioning*, pp. 1-5, 2007.

[14] S. R. Misal, H. M. Prajwal, H. M. Niveditha, H. M. Vinayaka y S. Veena, «Indoor Positioning System (IPS) Using ESP32, MQTT and Bluetooth,» *Fourth International Conference on Computing Methodologies and Communication (ICCMC),* pp. 79-82, 2020.

[15] F. Potorti, A. Crivello, P. Barsocchi y F. Palumbo, «Evaluation of Indoor Localisation Systems: Comments on the ISO/IEC 18305 Standard,» *In Proceedings of the 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN),* pp. 1-7, 2018.

[16] Z. Tariq, D. Chemma, M. Kamran y I. Naqvi, «Non-GPS Positioning Systems.,» *ACM Comput,* nº 1-34, 2017.

[17] A. Basiri, E. Lohan, T. Moore, A. Winstanley, P. Peltola, C. Hill, P. Amirian y P. Silva, «Indoor location based services challenges, requirements and usability of current solutions.,» *Comput. Sci.,* Vols. %1 de %21-12, p. 24, 2017.

[18] G. Mendoza, J. Torres y J. Huerta, «A new location technique for the active office,» *MDPI,* vol. 19, nº 20, 2019.

[19] A. Tahat, G. Kaddoum y F. Gagnon, «Look at the Recent Wireless Positioning Techniques With a Focus on Algorithms for Moving Receivers,» *IEEE,* nº 4, pp. 6652-6680, 2016.

[20] C. Laoudias, A. Moreira, S. Kim, S. Lee y L. Wirola, «A Survey of Enabling Technologies for Network Localization, Tracking, and Navigation,» *IEEE Commun. Surv. Tutor.,* nº 20, pp. 3607-3644, 2018.

[21] M. O. Aqel, M. H. Marhaban, M. I. Saripan y N. B. Ismail, «Review of visual odometry: Types, approaches, challenges, and applications,» *SpringerPlus,* nº 5, p. 1897, 2016.

[22] A. Makki, A. Siddig, M. Saad y C. Bleakley, «Survey of WiFi positioning using time-based techniques,» *Comput. Netw.,* nº 88, pp. 218-233, 2015.

[23] R. Faragher y R. Harle, «Comput. Netw.,» *IEEE,* nº 33, pp. 2418-2428, 2015.

[24] F. Zafari, A. Gkelias y K. Leung, «Survey of Indoor Localization Systems and Technologies,» *IEEE Commun. Surv. Tutor.,* nº 21, pp. 2568-2599, 2019.

[25] S. He y S. Chan, «Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons.,» *IEEE Commun. Surv. Tutor.,* nº 18, pp. 466-490, 2016.

[26] Kontakt.io, «Beacon Buyer's Guide,» Kontakt.io, 2019.

[27] A. Basiri, E. Lohan, T. Moore, A. Winstanley, P. Peltola y C. Hill, «Indoor location based services challenges, requirements and usability of current solutions,» *Comput. Sci. Rev.,* nº 24, pp. 1-12, 2017.

[28] P. Davidson y R. Piche, «A Survey of Selected Indoor Positioning Methods for Smartphones.,» *IEEE Commun. Surv. Tutor.,* nº 19, pp. 1347-1370, 2017.

[29] M. Goel, A. Vasant y K. Anil, «Bluetooth Eddystone Ibeacon| Bluetooth Eddystone Ibeacon Suppliers-Minew,» *Latin American Journal of Solids and Structures,* vol. 11, nº 13, pp. 2497-2515, 2014.

[30] B. Rohrig, Smartphones, ChemMatters, 2015.

[31] M. Nosrati, R. Karimi y H. Hasanvand, «Mobile computing: principles, devices and operating systems,» *World Applied Programming,* vol. 2, nº 7, pp. 399-408, 2012.

[32] HappyFox, «Chatbots vs. Virtual Assistants – What's the Difference?,» HappyFox, [En línea]. Available: https://blog.happyfox.com/chatbots-vs-virtual-assistants-whats-the-difference/. [Último acceso: 27 octubre 2022].

[33] Hyro, «Voice Assistant (VA),» Hyro, [En línea]. Available: https://www.hyro.ai/glossary/voice-assistant-va. [Último acceso: 27 octubre 2022].

[34] J. Cuello y J. Vittone, Designing Movile Apps, Catalina Duque Giraldo, 2013.

[35] Clockwise, «What Is Cross-Platform Mobile App Development?,» Clockwise, 2021. [En línea]. Available: https://clockwise.software/blog/cross-platform-app-development/. [Último acceso: noviembre 2022].

[36] Dart, «Dart documentation,» Dart, [En línea]. Available: https://dart.dev/guides. [Último acceso: noviembre 2022].

[37] Nomtek, «What Is a Mobile App Backend and Does Your Mobile Application Need It?,» [En línea]. Available: https://www.nomtek.com/blog/mobile-app-backend. [Último acceso: noviembre 2022].

[38] nodeJS, «Acerca de Node.js,» Node.js, [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: noviembre 2022].

[39] Java, «What is Java technology and why do I need it?,» Java, [En línea]. Available: https://www.java.com/es/download/help/whatis_java.html. [Último acceso: noviembre 2022].

[40] Python, «Python Get Started,» Python, [En línea]. Available: https://www.python.org/. [Último acceso: noviembre 2022].

[41] W. Ella, «Node.js Vs JAVA Vs Python- How to Choose the Best Backend Tech Stack?,» [En línea]. Available: https://medium.com/quick-code/node-js-vs-java-vs-python-how-to-choose-the-best-backend-tech-stack-82833df0bb4e. [Último acceso: noviembre 2022].

[42] Estimote, «Technical specification of Estimote Beacons and Stickers,» Estimote, 2017. [En línea]. Available: https://forums.estimote.com/t/technical-specification-of-estimote-beacons-and-stickers/7297/1. [Último acceso: 15 noviembre 2022].

[43] Kontakt.io, «Anchor Beacon 2 Technical Specifications,» [En línea]. Available: https://store.kontakt.io/app/uploads/2021/09/Kontakt.io-Anchor-Beacon-2-Technical-Specification-1.pdf. [Último acceso: 15 noviembre 2022].

[44] Accent systems, «iBKS 105,» [En línea]. Available: https://accent-systems.com/es/producto/ibks-105/. [Último acceso: 15 noviembre 2022].

[45] IeBS, «Metodología Scrum,» IeBS, [En línea]. Available: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/. [Último acceso: 10 noviembre 2022].

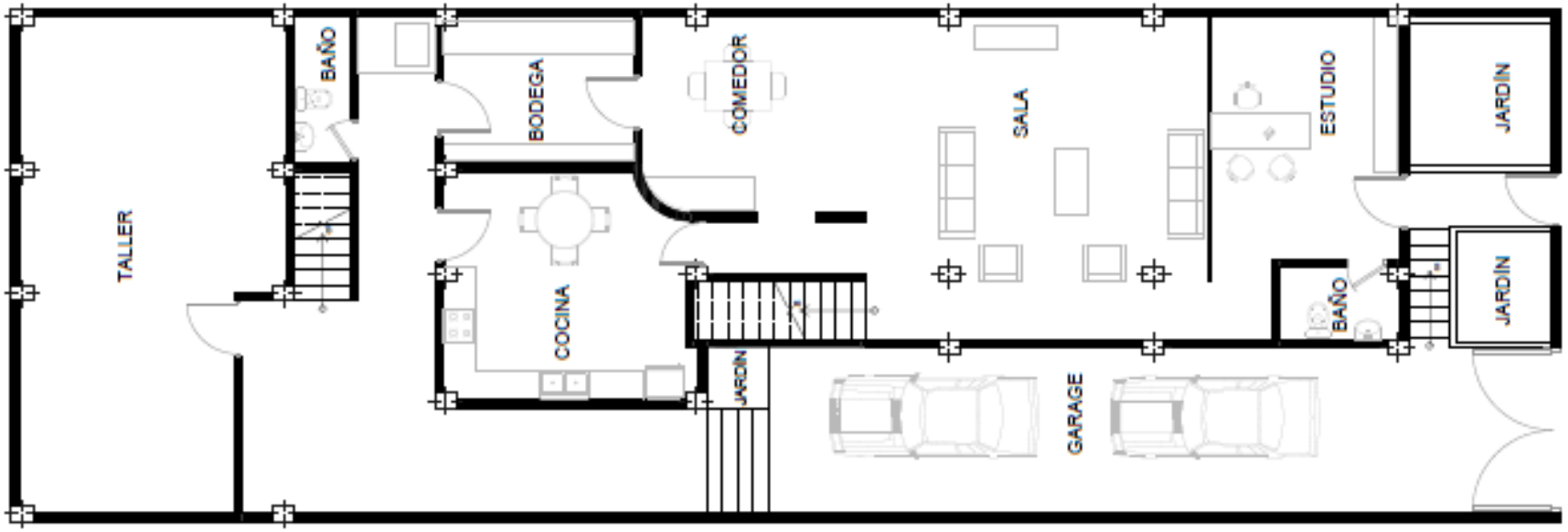
[46] 9.-1. ISO, «Ergonomic requirements for office work with visual display terminals (VDTs),» ISO, 2018.

[47] Medium, «Alan IA,» Medium, 2020. [En línea]. Available: https://alanvoice.medium.com/voice-enabling-you-flutter-app-complete-guide-of-how-you-can-add-a-voice-assistant-to-your-6307a78f65c2. [Último acceso: diciembre 2022].

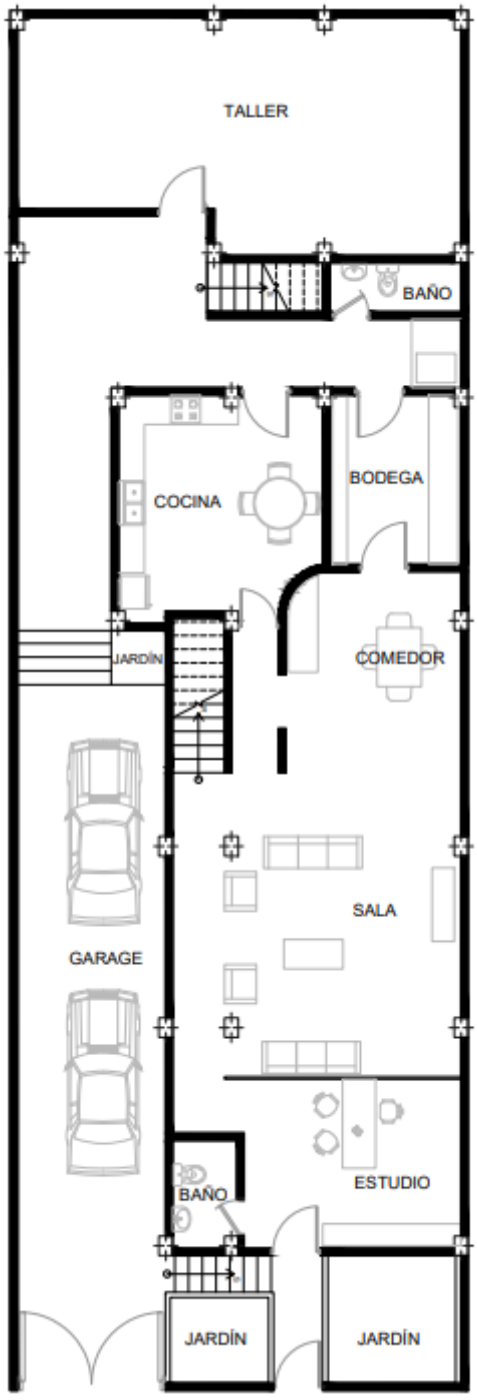
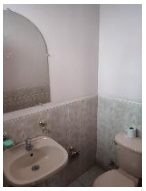
[48] LOTAIP, «Remuneración mensual por puesto,» 2019.

ANEXOS

ANEXO 1: Planos del lugar a implementar el proyecto



ANEXO 2: Puntos de interés

Puntos de interés		Plano general
 Figura 69 Entrada principal	 Figura 70 Estudio	 Figura 71 Plano General del lugar a implementar el prototipo
 Figura 72 Baño 1	 Figura 73 Sala	
 Figura 74 Comedor	 Figura 75 Bodega	
 Figura 76 Cocina	 Figura 77 Baño 2	
 Figura 78 Taller	 Figura 79 Garaje	
 Figura 80 Jardín 1	 Figura 81 Jardín 2	

ANEXO 3: Tabla de rutas con su número, tiempo y distancia.

	Baño 1	Baño 2	Bodega	Cocina	Comedor	Estudio	Garaje	Jardín 1	Jardín 2	Jardín 3	Sala	Taller
Baño 1	-	Ruta # 12 t = 15 s d = 20 m	Ruta # 23 t = 9 s d = 17 m	Ruta # 34 t = 9 s d = 13 m	Ruta # 45 t = 10 s d = 11 m	Ruta # 56 t = 4 s d = 2 m	Ruta # 67 t = 7 s d = 7 m	Ruta # 78 t = 5 s d = 4 m	Ruta # 89 t = 5 s d = 4 m	Ruta # 100 t = 9 s d = 14 m	Ruta # 111 t = 6 s d = 8 m	Ruta # 122 t = 13 s d = 26 m
Baño 2	Ruta # 1 t = 15 s d = 20 m	-	Ruta # 24 t = 4 s d = 2 m	Ruta # 35 t = 5 s d = 5 m	Ruta # 46 t = 7 s d = 7 m	Ruta # 57 t = 13 s d = 16 m	Ruta # 68 t = 10 s d = 16 m	Ruta # 79 t = 15 s d = 22 m	Ruta # 90 t = 15 s d = 26 m	Ruta # 101 t = 9 s d = 15 m	Ruta # 112 t = 9 s d = 15 m	Ruta # 123 t = 5 s d = 9 m
Bodega	Ruta # 2 t = 9 s d = 17 m	Ruta # 13 t = 4 s d = 2 m	-	Ruta # 36 t = 5 s d = 7 m	Ruta # 47 t = 5 s d = 4 m	Ruta # 58 t = 10 s d = 17 m	Ruta # 69 t = 12 s d = 19 m	Ruta # 80 t = 13 s d = 20 m	Ruta # 91 t = 13 s d = 20 m	Ruta # 102 t = 10 s d = 18 m	Ruta # 113 t = 6 s d = 13 m	Ruta # 124 t = 7 s d = 13 m
Cocina	Ruta # 3 t = 9 s d = 13 m	Ruta # 14 t = 5 s d = 5 m	Ruta # 25 t = 5 s d = 7 m	-	Ruta # 48 t = 5 s d = 4 m	Ruta # 59 t = 10 s d = 14 m	Ruta # 70 t = 11 s d = 17 m	Ruta # 81 t = 13 s d = 19 m	Ruta # 92 t = 13 s d = 19 m	Ruta # 103 t = 5 s d = 16 m	Ruta # 114 t = 6 s d = 12 m	Ruta # 125 t = 5 s d = 9 m
Comedor	Ruta # 4 t = 10 s d = 11 m	Ruta # 15 t = 7 s d = 7 m	Ruta # 26 t = 5 s d = 4 m	Ruta # 37 t = 5 s d = 4 m	-	Ruta # 60 t = 9 s d = 12 m	Ruta # 71 t = 13 s d = 23 m	Ruta # 82 t = 10 s d = 15 m	Ruta # 93 t = 10 s d = 15 m	Ruta # 104 t = 15 s d = 22 m	Ruta # 115 t = 6 s d = 5 m	Ruta # 126 t = 10 s d = 17 m
Estudio	Ruta # 5 t = 4 s d = 2 m	Ruta # 16 t = 13 s d = 16 m	Ruta # 27 t = 10 s d = 17 m	Ruta # 38 t = 10 s d = 14 m	Ruta # 49 t = 9 s d = 12 m	-	Ruta # 72 t = 9 s d = 13 m	Ruta # 83 t = 5 s d = 5 m	Ruta # 94 t = 5 s d = 5 m	Ruta # 105 t = 10 s d = 18 m	Ruta # 116 t = 5 s d = 17 m	Ruta # 127 t = 14 s d = 27 m
Garaje	Ruta # 6 t = 7 s d = 7 m	Ruta # 17 t = 10 s d = 16 m	Ruta # 28 t = 12 s d = 19 m	Ruta # 39 t = 11 s d = 17 m	Ruta # 50 t = 13 s d = 23 m	Ruta # 61 t = 9 s d = 13 m	-	Ruta # 84 t = 6 s d = 9 m	Ruta # 95 t = 7 s d = 13 m	Ruta # 106 t = 4 s d = 6 m	Ruta # 117 t = 10 s d = 12 m	Ruta # 128 t = 10 s d = 20 m
Jardín 1	Ruta # 7 t = 5 s d = 4 m	Ruta # 18 t = 15 s d = 22 m	Ruta # 29 t = 13 s d = 20 m	Ruta # 40 t = 13 s d = 19 m	Ruta # 51 t = 10 s d = 15 m	Ruta # 62 t = 5 s d = 5 m	Ruta # 73 t = 6 s d = 9 m	-	Ruta # 96 t = 4 s d = 2 m	Ruta # 107 t = 10 s d = 16 m	Ruta # 118 t = 9 s d = 9 m	Ruta # 129 t = 15 s d = 27 m
Jardín 2	Ruta # 8 t = 5 s d = 4 m	Ruta # 19 t = 15 s d = 26 m	Ruta # 30 t = 13 s d = 20 m	Ruta # 41 t = 13 s d = 19 m	Ruta # 52 t = 10 s d = 15 m	Ruta # 63 t = 5 s d = 5 m	Ruta # 74 t = 7 s d = 13 m	Ruta # 85 t = 4 s d = 2 m	-	Ruta # 108 t = 13 s d = 18 m	Ruta # 119 t = 9 s d = 9 m	Ruta # 130 t = 15 s d = 28 m
Jardín 3	Ruta # 9 t = 9 s d = 14 m	Ruta # 20 t = 9 s d = 15 m	Ruta # 31 t = 10 s d = 18 m	Ruta # 42 t = 5 s d = 16 m	Ruta # 53 t = 15 s d = 22 m	Ruta # 64 t = 10 s d = 18 m	Ruta # 75 t = 4 s d = 6 m	Ruta # 86 t = 10 s d = 16 m	Ruta # 97 t = 13 s d = 18 m	-	Ruta # 120 t = 15 s d = 17 m	Ruta # 131 t = 9 s d = 10 m
Sala	Ruta # 10 t = 9 s d = 8 m	Ruta # 21 t = 9 s d = 15 m	Ruta # 32 t = 6 s d = 13 m	Ruta # 43 t = 6 s d = 12 m	Ruta # 54 t = 6 s d = 5 m	Ruta # 65 t = 5 s d = 17 m	Ruta # 76 t = 10 s d = 12 m	Ruta # 87 t = 9 s d = 9 m	Ruta # 98 t = 9 s d = 9 m	Ruta # 109 t = 15 s d = 17 m	-	Ruta # 132 t = 10 s d = 24 m
Taller	Ruta # 11 t = 13 s d = 26 m	Ruta # 22 t = 5 s d = 9 m	Ruta # 33 t = 7 s d = 13 m	Ruta # 44 t = 5 s d = 9 m	Ruta # 55 t = 10 s d = 17 m	Ruta # 66 t = 14 s d = 27 m	Ruta # 77 t = 10 s d = 20 m	Ruta # 88 t = 15 s d = 27 m	Ruta # 99 t = 15 s d = 28 m	Ruta # 110 t = 9 s d = 10 m	Ruta # 121 t = 10 s d = 24 m	-

ANEXO 4: Datasheet del Beacon IBS105

iBKS105 Datasheet

ABSTRACT

iBKS105 Technical Data

AUDIENCE

This document is primarily focused for engineers or other users with a technical profile

FEATURES

- Advertising **Beacon** Device
- **Bluetooth Low Energy®**
- Full **Eddystone** & **iBeacon** compatible
- 100% Configurable Parameters
- Firmware update **Over The Air** (OTA)
- Waterproof optional kit
- Logo and color customizable (MOQ)
- Provided with your own configuration (MOQ 50 units)
- No tools needed for maintenance
- Additional optional sensors available

Revision 3 | January 2017

accent systems

ANEXO 5: Código de los controllers del backend

Controlador Ciudad

```
import { getConnection, sql } from "../datatbase/connection.js";

export const getCiudades =  async (req, res) => {
    try{
        const pool = await getConnection();
        const result = await pool.request().query("SELECT ID_CIU,
NOM_CIU FROM CIUDAD");
        console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
};
```

Controlador Cliente

```
import { getConnection, sql } from "../datatbase/connection.js";
import crypto from 'crypto';
import { createRequire } from 'module';

export const getUsuario =  async (req, res) => {
    try{
        const pool = await getConnection();
        const result = await pool.request().query("SELECT * FROM
USUARIO");
        console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
};

export const registrarUsuario = async (req, res) => {
    try{
        const { nombre, apellido, telefono, nombreUsuario, clave,
imagen ,idGenero, idCiudad } = req.body;

        if(nombre == null || apellido == null || telefono == null ||
nombreUsuario == null || clave == null ||
            idGenero == null || idCiudad == null)
           return res.status(400).json({ msg: 'Bad Request'})
        else{
            var hash =
crypto.createHash('sha256').update(clave).digest('hex');
            const pool = await getConnection();
            await pool.request()
            .input('nombre', sql.VarChar, nombre)
            .input('apellido', sql.VarChar, apellido)
            .input('telefono', sql.VarChar, telefono)
            .input('nombreUsuario', sql.VarChar, nombreUsuario)
```

```
            .input('clave', sql.VarChar, hash)
            .input('imagen', sql.VarBinary, imagen)
            .input('genero', sql.Int, idGenero)
            .input('ciudad', sql.VarChar, idCiudad)
            .query("INSERT INTO
USUARIO(NOM_USU,APE_USU,TEL_USU,NOM_USU_LOG,CON_USU,IMG_USU,ID_GEN_P
ER,ID_CIU_PER,ID_ROL_PER,FECHA_REG_USU)" +

"VALUES(@nombre,@apellido,@telefono,@nombreUsuario,@clave,@imagen,@g
enero,@ciudad,2,GETDATE())");
        }

        res.json({nombre, apellido, telefono, nombreUsuario, clave,
imagen});
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
}

export const getUsuarioPorID = async (req, res) => {
    try {
        const { id } = req.body;
        const pool = await getConnection();
        const result = await pool.request()
                        .input('Id', id)
                        .query("SELECT * FROM USUARIO WHERE ID_USU =
@Id");
        console.log(result);
        res.send(result.recordset[0]);
    } catch (error) {
        res.status(500);
        res.send(error.message);
    }
}


export const login = async (req, res) => {
    try {
        var { usuario, clave } = req.body;
        const pool = await getConnection();
        clave =
crypto.createHash('sha256').update(clave).digest('hex');

        const result = await pool.request()
                        .input('usuario', sql.VarChar, usuario)
                        .input('clave', sql.VarChar, clave)
                        .query("SELECT * FROM USUARIO WHERE
NOM_USU_LOG = @usuario and CON_USU = @clave");
        res.json(result.recordset[0]);
        console.log(result.recordset[0]);
    } catch (error) {
        res.status(500);
        res.send(error.message);
    }
}
```

```
export const loginImage = async (req, res) => {
    try {
        var { usuario, clave } = req.body;
        const pool = await getConnection();
        clave =
crypto.createHash('sha256').update(clave).digest('hex');

        const result = await pool.request()
                        .input('usuario', sql.VarChar, usuario)
                        .input('clave', sql.VarChar, clave)
                        .query("SELECT IMG_USU FROM USUARIO WHERE
NOM_USU_LOG = @usuario and CON_USU = @clave");
        res.json(result.recordset[0]);
        console.log(result.recordset[0]);
    } catch (error) {
        res.status(500);
        res.send(error.message);
    }
}
```

Controlador Género

```
import { getConnection, sql } from "../datatbase/connection.js";

export const getGenero =  async (req, res) => {
    try{
        const pool = await getConnection();
        const result = await pool.request().query("SELECT * FROM
GENERO");
        console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
};
```

Controlador Sitio

```
import { getConnection, sql } from "../datatbase/connection.js";

export const getDestinos = async (req, res) => {
    try {
        const { id } = req.body;
        const pool = await getConnection();
        const result = await pool.request()
                        .input('Id', id)
                        .query("SELECT ID_DET_RUT, S.NOM_SIT AS
SIT_DES, S.URL_IMG_SIT, S.ESP_SIT, DR.GIF_RUTA, DR.URL_SON,
DR.URL_SON_BAL, DR.URL_SON_CON\n"+
                                                "FROM
DETALLE_RUTA DR\n"+
                                                "INNER JOIN
SITIO S ON s.ID_SIT = dr.ID_SIT_DES\n"+
                                                "WHERE
DR.ID_SIT_ORI = @Id");
```

```
        console.log(result);
        res.send(result.recordset);
    } catch (error) {
        res.status(500);
        res.send('error.message');
    }
}

export const getDestinosAsistente = async (req, res) => {
    try {
        const { id } = req.body;
        const pool = await getConnection();
        const result = await pool.request()
                        .input('Id', id)
                        .query("SELECT ID_DET_RUT, S.NOM_SIT AS
SIT_DES, S.URL_IMG_SIT, S.ESP_SIT, DR.GIF_RUTA, DR.URL_SON,
DR.URL_SON_BAL, DR.URL_SON_CON\n"+
                                                            "FROM
DETALLE_RUTA DR\n"+
                                                            "INNER JOIN
SITIO S ON s.ID_SIT = dr.ID_SIT_DES\n"+
                                                            "WHERE
DR.ID_DET_RUT = @Id");
        console.log(result);
        res.send(result.recordset[0]);
    } catch (error) {
        res.status(500);
        res.send('error.message');
    }
}

export const Buscador = async (req, res) => {
    try {
        const { sitioOrigen, sitioDestino, idSitOri } = req.body;
        const pool = await getConnection();
        const result = await pool.request()
                        .input('IdSitOri', idSitOri)
                        .input('sitioOrigen', sitioOrigen)
                        .input('sitioDestino', sitioDestino)
                        .query("SELECT ID_DET_RUT, S.NOM_SIT AS
SIT_DES, S1.NOM_SIT AS SIT_ORI, S.URL_IMG_SIT, S.ESP_SIT,
DR.GIF_RUTA, DR.URL_SON\n"+
                                                            "FROM
DETALLE_RUTA DR\n"+
                                                            "INNER JOIN
SITIO S ON s.ID_SIT = dr.ID_SIT_DES\n"+
                                                            "INNER JOIN
SITIO S1 ON S1.ID_SIT = DR.ID_SIT_ORI\n"+
                                                            "WHERE
S1.NOM_SIT LIKE '%' + @sitioOrigen + '%' and S.NOM_SIT like '%' +
@sitioDestino + '%' and DR.ID_SIT_ORI = @IdSitOri;");
        console.log(result);
        res.send(result.recordset);
    } catch (error) {
        res.status(500);
        res.send(error.message);
```

```
    }
}


export const getVisitas = async (req, res) => {
    try {
        const { id } = req.body;
        const pool = await getConnection();
        const result = await pool.request()
                        .input('Id', id)
                        .query("SELECT * FROM VISITAS");
        console.log(result);
        res.send(result.recordset);
    } catch (error) {
        res.status(500);
        res.send('error.message');
    }
}


export const getSitios =  async (req, res) => {
    try{
        const pool = await getConnection();
        const result = await pool.request().query("SELECT * FROM
SITIO");
        console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
};

export const getSitiosMasVisitados =  async (req, res) => {
    try{
        const pool = await getConnection();
        const result = await pool.request().query("SELECT TOP 3
S.NOM_SIT, COUNT(V.ID_SIT_VIS) AS VISITAS, s.URL_IMG_SIT\n"+
                                                   "FROM VISITAS
V\n"+
                                                   "INNER JOIN SITIO
S ON S.ID_SIT = V.ID_SIT_VIS\n"+
                                                   "GROUP BY
S.NOM_SIT, S.URL_IMG_SIT\n"+
                                                   "ORDER BY
COUNT(V.ID_SIT_VIS) DESC");
        console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
};

export const getUsuariosConMasVisitas =  async (req, res) => {
    try{
```

```
        const pool = await getConnection();
        const result = await pool.request().query("SELECT TOP 3
U.NOM_USU + ' ' + U.APE_USU AS USUARIO, COUNT(V.ID_SIT_VIS) AS
VISITAS, U.IMG_USU\n"+
                                                  "FROM VISITAS
V\n"+
                                                  "INNER JOIN
USUARIO U ON U.ID_USU = V.ID_USU_VIS\n"+
                                                  "GROUP BY
U.NOM_USU, U.APE_USU, U.IMG_USU\n"+
                                                  "ORDER BY
COUNT(v.ID_SIT_VIS) DESC");
        //console.log(result);
        res.json(result.recordset);
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
};

export const registrarVisita = async (req, res) => {
    try{
        const { id_usuario, id_sitio } = req.body;

        if(id_usuario == null || id_sitio == null)
           return res.status(400).json({ msg: 'Bad Request'})
        else{
            const pool = await getConnection();
            await pool.request()
            .input('idUsuario', sql.Int, id_usuario)
            .input('idSitio', sql.Int, id_sitio)
            .query("INSERT INTO VISITAS(ID_USU_VIS,ID_SIT_VIS)" +
                   "VALUES (@idUsuario,@idSitio)");
        }

        res.json({id_usuario, id_sitio});
    }catch(error){
        res.status(500);
        res.send(error.message);
    }
}
```

ANEXO 6: Acciones del asistente virtual

```
intent('Hola', p => {
    p.play('(hola bienvenido)');
});

intent('Como te llamas', p => {
    p.play('(Mi nombre es Alan)');
});

intent('Rutas disponibles', p => {
    p.play({command: 'forward'})
});

intent('informacion de la cocina', p => {
    p.play({command: 'cocina'})
});

intent('ver los Reportes', p => {
    p.play({command: 'reportes'})
});

intent('informacion de las Gradas', p => {
    p.play({command: 'gradas'})
});

intent('informacion del primer faro', p => {
    p.play({command: 'balizapuerta'})
});

intent('informacion del segundo faro', p => {
    p.play({command: 'balizasala'})
});

intent('informacion del tercer faro', p => {
    p.play({command: 'balizagarage'})
});

intent('informacion del taller', p => {
    p.play({command: 'taller'})
});

intent('informacion del servicio dos', p => {
    p.play({command: 'baniodos'})
});

intent('informacion de la bodega', p => {
    p.play({command: 'bodega'})
});

intent('informacion del comedor', p => {
    p.play({command: 'comedor'})
});

intent('informacion del jardin tres', p => {
    p.play({command: 'jardintres'})
});

intent('informacion del garage', p => {
    p.play({command: 'garage'})
});

intent('informacion de la sala', p => {
    p.play({command: 'sala'})
});

intent('informacion del servicio uno', p => {
    p.play({command: 'baniouno'})
});

intent('informacion del estudio', p => {
    p.play({command: 'estudio'})
});

intent('informacion del jardin uno', p => {
    p.play({command: 'jardinuno'})
});

intent('informacion del jardin dos', p => {
    p.play({command: 'jardindos'})
});

intent('scanner', p => {
    p.play({command: 'escanear'})
});


intent('ver la ruta uno', p => {
    p.play({command: 'rutauno'})
});

intent('ver la ruta dos', p => {
    p.play({command: 'rutados'})
});

intent('ver la ruta tres', p => {
    p.play({command: 'rutatres'})
});

intent('ver la ruta cuatro', p => {
    p.play({command: 'rutacuatro'})
});
```

```
intent('ver la ruta cinco',
p => {
    p.play({command:
'rutacinco'})
});

intent('ver la ruta seis',
p => {
    p.play({command:
'rutaseis'})
});

intent('ver la ruta siete',
p => {
    p.play({command:
'rutasiete'})
});

intent('ver la ruta ocho',
p => {
    p.play({command:
'rutaocho'})
});

intent('ver la ruta nueve',
p => {
    p.play({command:
'rutanueve'})
});

intent('ver la ruta diez',
p => {
    p.play({command:
'rutadiez'})
});

intent('ver la ruta once',
p => {
    p.play({command:
'rutaonce'})
});

intent('ver la ruta doce',
p => {
    p.play({command:
'doce'})
});

intent('ver la ruta trece',
p => {
    p.play({command:
'trece'})
});

intent('ver la ruta
catorce', p => {
    p.play({command:
'catorce'})
});

intent('ver la ruta
quince', p => {
    p.play({command:
'quince'})
});

intent('ver la ruta
dieciseis', p => {
    p.play({command:
'dieciseis'})
});

intent('ver la ruta
diecisiete', p => {
    p.play({command:
'diecisiete'})
});

intent('ver la ruta
dieciocho', p => {
    p.play({command:
'dieciocho'})
});

intent('ver la ruta
diecinueve', p => {
    p.play({command:
'diecinueve'})
});

intent('ver la ruta
veinte', p => {
    p.play({command:
'veinte'})
});

intent('ver la ruta veinte
y uno', p => {
    p.play({command:
'veintiuno'})
});

intent('ver la ruta veinte
y dos', p => {
    p.play({command:
'veintidos'})
});

intent('ver la ruta veinte
y tres', p => {
    p.play({command:
'veintitres'})
});

intent('ver la ruta veinte
y cuatro', p => {
    p.play({command:
'veinticuatro'})
});

intent('ver la ruta veinte
y cinco', p => {
    p.play({command:
'veinticinco'})
});

intent('ver la ruta veinte
y seis', p => {
    p.play({command:
'veintiseis'})
});

intent('ver la ruta veinte
y siete', p => {
    p.play({command:
'veintisiete'})
});

intent('ver la ruta veinte
y ocho', p => {
    p.play({command:
'veintiocho'})
});

intent('ver la ruta veinte
y nueve', p => {
    p.play({command:
'veintinueve'})
});

intent('ver la ruta
treinta', p => {
    p.play({command:
'treinta'})
});

intent('ver la treinta
y uno', p => {
    p.play({command:
'treintayuno'})
});

intent('ver la treinta
y dos', p => {
    p.play({command:
'treintaydos'})
});
```

```
intent('ver la ruta treinta y tres', p => {
    p.play({command: 'treintaytres'})
});

intent('ver la ruta treinta y cuatro', p => {
    p.play({command: 'treintaycuatro'})
});

intent('ver la ruta treinta y cinco', p => {
    p.play({command: 'treintaycinco'})
});

intent('ver la ruta treinta y seis', p => {
    p.play({command: 'treintayseis'})
});

intent('ver la ruta treinta y siete', p => {
    p.play({command: 'treintaysiete'})
});

intent('ver la ruta treinta y ocho', p => {
    p.play({command: 'treintayocho'})
});

intent('ver la ruta treinta y nueve', p => {
    p.play({command: 'treintaynueve'})
});

intent('ver la ruta cuarenta', p => {
    p.play({command: 'cuarenta'})
});

intent('ver la ruta cuarenta y uno', p => {
    p.play({command: 'cuarentayuno'})
});

intent('ver la ruta cuarenta y dos', p => {
    p.play({command: 'cuarentaydos'})
});

intent('ver la ruta cuarenta y tres', p => {
    p.play({command: 'cuarentaytres'})
});

intent('ver la ruta cuarenta y cuatro', p => {
    p.play({command: 'cuarentaycuatro'})
});

intent('ver la ruta cuarenta y cinco', p => {
    p.play({command: 'cuarentaycinco'})
});

intent('ver la ruta cuarenta y seis', p => {
    p.play({command: 'cuarentayseis'})
});

intent('ver la ruta cuarenta y siete', p => {
    p.play({command: 'cuarentaysiete'})
});

intent('ver la ruta cuarenta y ocho', p => {
    p.play({command: 'cuarentayocho'})
});

intent('ver la ruta cuarenta y nueve', p => {
    p.play({command: 'cuarentaynueve'})
});

intent('ver la ruta cincuenta', p => {
    p.play({command: 'cincuenta'})
});

intent('ver la ruta cincuenta y uno', p => {
    p.play({command: 'cincuentayuno'})
});

intent('ver la ruta cincuenta y dos', p => {
    p.play({command: 'cincuentaydos'})
});

intent('ver la ruta cincuenta y tres', p => {
    p.play({command: 'cincuentaytres'})
});

intent('ver la ruta cincuenta y cuatro', p => {
    p.play({command: 'cincuentaycuatro'})
});

intent('ver la ruta cincuenta y cinco', p => {
    p.play({command: 'cincuentaycinco'})
});

intent('ver la ruta cincuenta y seis', p => {
    p.play({command: 'cincuentayseis'})
});

intent('ver la ruta cincuenta y siete', p => {
    p.play({command: 'cincuentaysiete'})
});

intent('ver la ruta cincuenta y ocho', p => {
    p.play({command: 'cincuentayocho'})
});

intent('ver la ruta cincuenta y nueve', p => {
    p.play({command: 'cincuentaynueve'})
});

intent('ver la ruta sesenta', p => {
    p.play({command: 'sesenta'})
});
```

```
intent('ver la ruta sesenta
y uno', p => {
    p.play({command:
'sesentayuno'})
});

intent('ver la ruta sesenta
y dos', p => {
    p.play({command:
'sesentaydos'})
});

intent('ver la ruta sesenta
y tres', p => {
    p.play({command:
'sesentaytres'})
});

intent('ver la ruta sesenta
y cuatro', p => {
    p.play({command:
'sesentaycuatro'})
});

intent('ver la ruta sesenta
y cinco', p => {
    p.play({command:
'sesentaycinco'})
});

intent('ver la ruta sesenta
y seis', p => {
    p.play({command:
'sesentayseis'})
});

intent('ver la ruta sesenta
y siete', p => {
    p.play({command:
'sesentaysiete'})
});

intent('ver la ruta sesenta
y ocho', p => {
    p.play({command:
'sesentayocho'})
});

intent('ver la ruta sesenta
y nueve', p => {
    p.play({command:
'sesentaynueve'})
});

intent('ver    la    ruta
setenta', p => {
    p.play({command:
'setenta'})
});

intent('ver la ruta setenta
y uno', p => {
    p.play({command:
'setentayuno'})
});

intent('ver la ruta setenta
y dos', p => {
    p.play({command:
'setentaydos'})
});

intent('ver la ruta setenta
y tres', p => {
    p.play({command:
'setentaytres'})
});

intent('ver la ruta setenta
y cuatro', p => {
    p.play({command:
'setentaycuatro'})
});

intent('ver la ruta setenta
y cinco', p => {
    p.play({command:
'setentaycinco'})
});

intent('ver la ruta setenta
y seis', p => {
    p.play({command:
'setentayseis'})
});

intent('ver la ruta setenta
y siete', p => {
    p.play({command:
'setentaysiete'})
});

intent('ver la ruta setenta
y ocho', p => {
    p.play({command:
'setentayocho'})
});

intent('ver la ruta setenta
y nueve', p => {
    p.play({command:
'setentaynueve'})
});

intent('ver    la    ruta
ochenta', p => {
    p.play({command:
'ochenta'})
});

intent('ver la ruta ochenta
y uno', p => {
    p.play({command:
'ochentayuno'})
});

intent('ver la ruta ochenta
y dos', p => {
    p.play({command:
'ochentaydos'})
});

intent('ver la ruta ochenta
y tres', p => {
    p.play({command:
'ochentaytres'})
});

intent('ver la ruta ochenta
y cuatro', p => {
    p.play({command:
'ochentaycuatro'})
});

intent('ver la ruta ochenta
y cinco', p => {
    p.play({command:
'ochentaycinco'})
});

intent('ver la ruta ochenta
y seis', p => {
    p.play({command:
'ochentayseis'})
});

intent('ver la ruta ochenta
y siete', p => {
    p.play({command:
'ochentaysiete'})
});

intent('ver la ruta ochenta
y ocho', p => {
    p.play({command:
'ochentayocho'})
});
```

```
intent('ver la ruta ochenta y nueve', p => {
    p.play({command: 'ochentaynueve'})
});

intent('ver la ruta noventa', p => {
    p.play({command: 'noventa'})
});

intent('ver la ruta noventa y uno', p => {
    p.play({command: 'noventayuno'})
});

intent('ver la ruta noventa y dos', p => {
    p.play({command: 'noventaydos'})
});

intent('ver la ruta noventa y tres', p => {
    p.play({command: 'noventaytres'})
});

intent('ver la ruta noventa y cuatro', p => {
    p.play({command: 'noventaycuatro'})
});

intent('ver la ruta noventa y cinco', p => {
    p.play({command: 'noventaycinco'})
});

intent('ver la ruta noventa y seis', p => {
    p.play({command: 'noventayseis'})
});

intent('ver la ruta noventa y siete', p => {
    p.play({command: 'noventaysiete'})
});

intent('ver la ruta noventa y ocho', p => {
    p.play({command: 'noventayocho'})
});

intent('ver la ruta noventa y nueve', p => {
    p.play({command: 'noventaynueve'})
});

intent('ver la ruta cien', p => {
    p.play({command: 'cien'})
});

intent('ver la ruta ciento uno', p => {
    p.play({command: 'cientouno'})
});

intent('ver la ruta ciento dos', p => {
    p.play({command: 'cientodos'})
});

intent('ver la ruta ciento tres', p => {
    p.play({command: 'cientotres'})
});

intent('ver la ruta ciento cuatro', p => {
    p.play({command: 'cientocuatro'})
});

intent('ver la ruta ciento cinco', p => {
    p.play({command: 'cientocinco'})
});

intent('ver la ruta ciento seis', p => {
    p.play({command: 'cientoseis'})
});

intent('ver la ruta ciento siete', p => {
    p.play({command: 'cientosiete'})
});

intent('ver la ruta ciento ocho', p => {
    p.play({command: 'cientoocho'})
});

intent('ver la ruta ciento nueve', p => {
    p.play({command: 'cientonueve'})
});

intent('ver la ruta ciento diez', p => {
    p.play({command: 'cientodiez'})
});

intent('ver la ruta ciento once', p => {
    p.play({command: 'cientoonce'})
});

intent('ver la ruta ciento doce', p => {
    p.play({command: 'cientodoce'})
});

intent('ver la ruta ciento trece', p => {
    p.play({command: 'cientotrece'})
});

intent('ver la ruta ciento catorce', p => {
    p.play({command: 'cientocatorce'})
});

intent('ver la ruta ciento quince', p => {
    p.play({command: 'cientoquince'})
});

intent('ver la ruta ciento dieciseis', p => {
    p.play({command: 'cientodieciseis'})
});
```

```
intent('ver la ruta ciento diecisiete', p => {
    p.play({command: 'cientodiecisiete'})
});

intent('ver la ruta ciento dieciocho', p => {
    p.play({command: 'cientodieciocho'})
});

intent('ver la ruta ciento diecinueve', p => {
    p.play({command: 'cientodiecinueve'})
});

intent('ver la ruta ciento veinte', p => {
    p.play({command: 'cientoveinte'})
});

intent('ver la ruta ciento veinte y uno', p => {
    p.play({command: 'cientoveinteyuno'})
});

intent('ver la ruta ciento veinte y dos', p => {
    p.play({command: 'cientoveinteydos'})
});

intent('ver la ruta ciento veinte y tres', p => {
    p.play({command: 'cientoveinteytres'})
});

intent('ver la ruta ciento veinte y cuatro', p => {
    p.play({command: 'cientoveinteycuatro'})
});

intent('ver la ruta ciento veinte y cinco', p => {
    p.play({command: 'cientoveinteycinco'})
});

intent('ver la ruta ciento veinte y seis', p => {
    p.play({command: 'cientoveinteyseis'})
});

intent('ver la ruta ciento veinte y siete', p => {
    p.play({command: 'cientoveinteysiete'})
});

intent('ver la ruta ciento veinte y ocho', p => {
    p.play({command: 'cientoveinteyocho'})
});

intent('ver la ruta ciento veinte y nueve', p => {
    p.play({command: 'cientoveinteynueve'})
});

intent('ver la ruta ciento treinta', p => {
    p.play({command: 'cientotreinta'})
});

intent('ver la ruta ciento treinta y uno', p => {
    p.play({command: 'cientotreintayuno'})
});

intent('ver la ruta ciento treinta y dos', p => {
    p.play({command: 'cientotreintaydos'})
});
```

ANEXO 7: Manual de usuario de la aplicación móvil

Asistente Virtual de Navegación y Generación de Rutas en Espacios Cerrados

INTRODUCCIÓN

El presente manual tiene como objetivo presentar al usuario final una guía básica para el uso de la aplicación móvil Ruttas_App así como familiarizarse con el entorno y la propuesta de solución a la navegación dentro de un espacio cerrado.

REQUERIMIENTOS

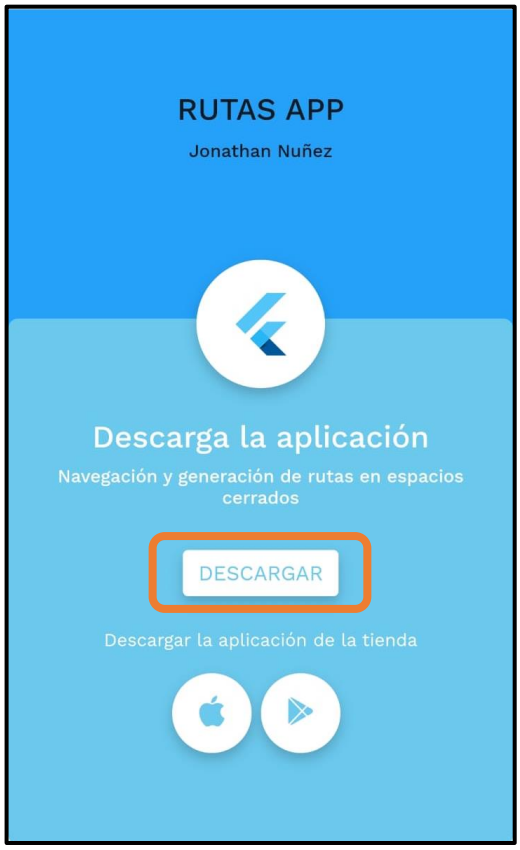
Los requerimientos mínimos para que la aplicación móvil funcione de manera correcta son los siguientes:

- Sistema Operativo Android 5+
- Almacenamiento mínimo disponible de 150 MB
- Bluetooth activado
- Acceso a fotos y cámara
- Conectividad a la red (Wi-Fi) servidor

Descargar la aplicación

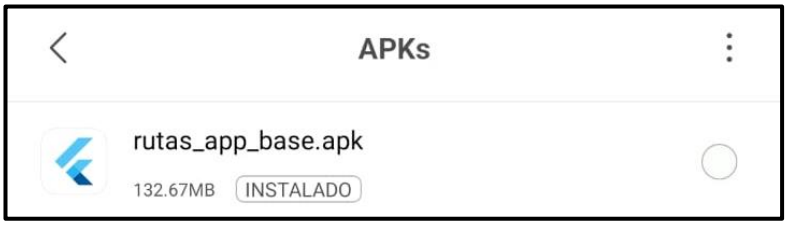
Para descargar la aplicación el primer paso a seguir es escanear el código QR que se muestra a continuación:

Este nos dirige a la siguiente página que presenta la aplicación, en la cual se debe presionar el botón “DESCARGAR”.



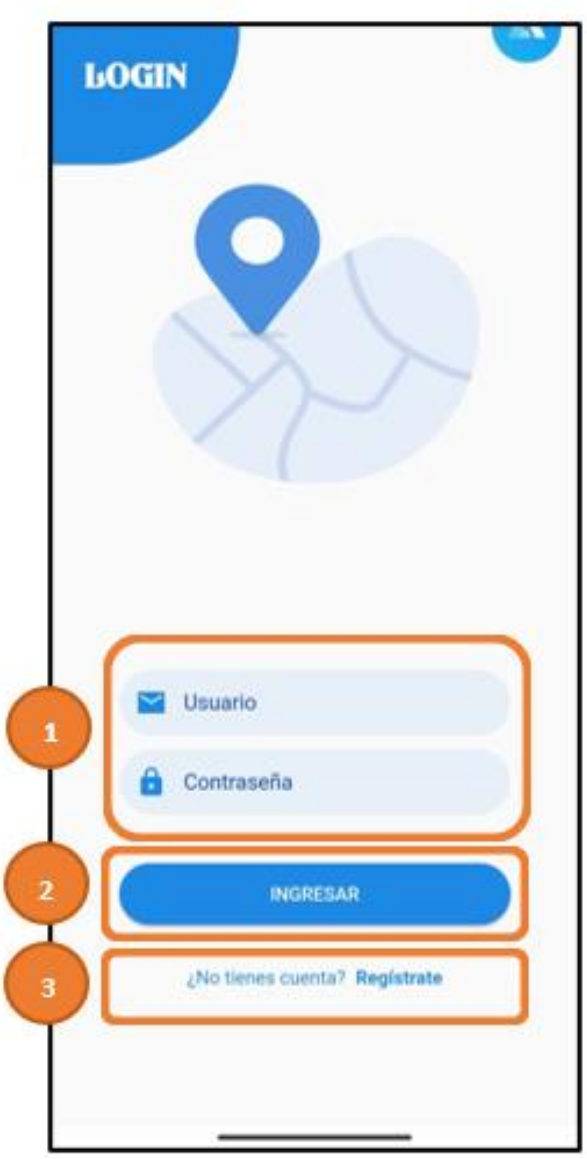
A continuación, se muestra el archivo rutas_app_base.apk el cual para iniciar su descarga se debe presionar en los tres puntos y elegir la opción de descarga

Una vez descargado, se procede con la instalación normal como cualquier aplicación.



Pantalla de inicio de sesión

Es la pantalla con la que inicia la aplicación, para iniciarla es necesario tener un usuario registrado.



1. En el caso de tener una cuenta registrada anteriormente, se deberá ingresar tanto el **usuario** como la **contraseña.**
2. El botón **ingresar** inicia la aplicación dependiendo si se encuentran bien ingresados los datos.
3. En caso de no tener una cuenta es posible **registrar** un nuevo usuario

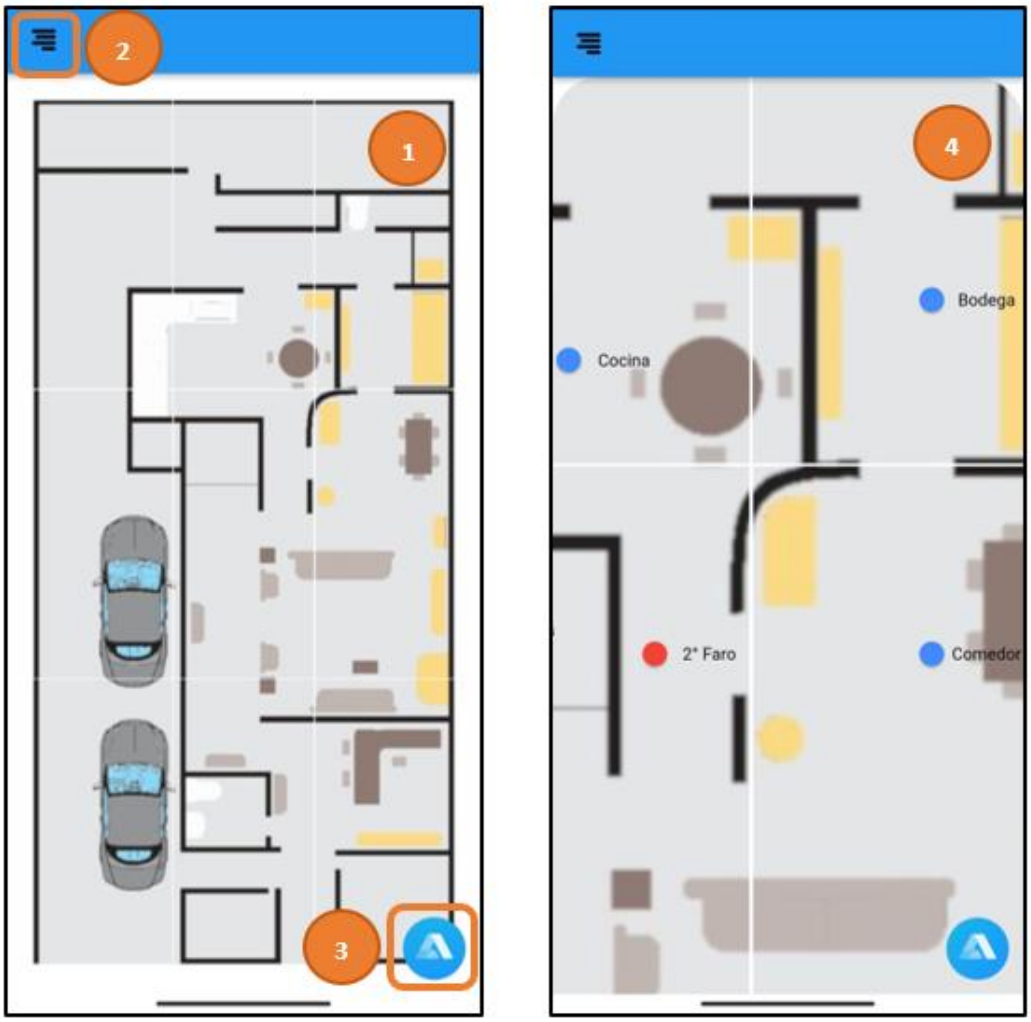
Pantalla de registro

En esta pantalla se ingresan todos los datos relevantes para registrar un nuevo usuario.

1. En este icono se selecciona la foto de perfil del usuario, la cual puede ser elegida de la galería o puede ser tomada en ese instante por la cámara.
2. Se deberá colocar el nombre del Usuario.
3. Se deberá colocar el apellido del Usuario.
4. Se deberá ingresar un número de teléfono.
5. Se escribe el nombre de usuario, este puede contener letras y números si así lo desea.
6. Ingresar una contraseña.
7. Ingresar nuevamente la contraseña, esta deberá coincidir con la primera.
8. En esta parte de selecciona la ciudad del visitante a partir de una lista desplegable.
9. En esta sección se selecciona el género.
10. Una vez completos los campos anteriores al presionar el botón registrarse se habrá creado el nuevo usuario.

Pantalla principal

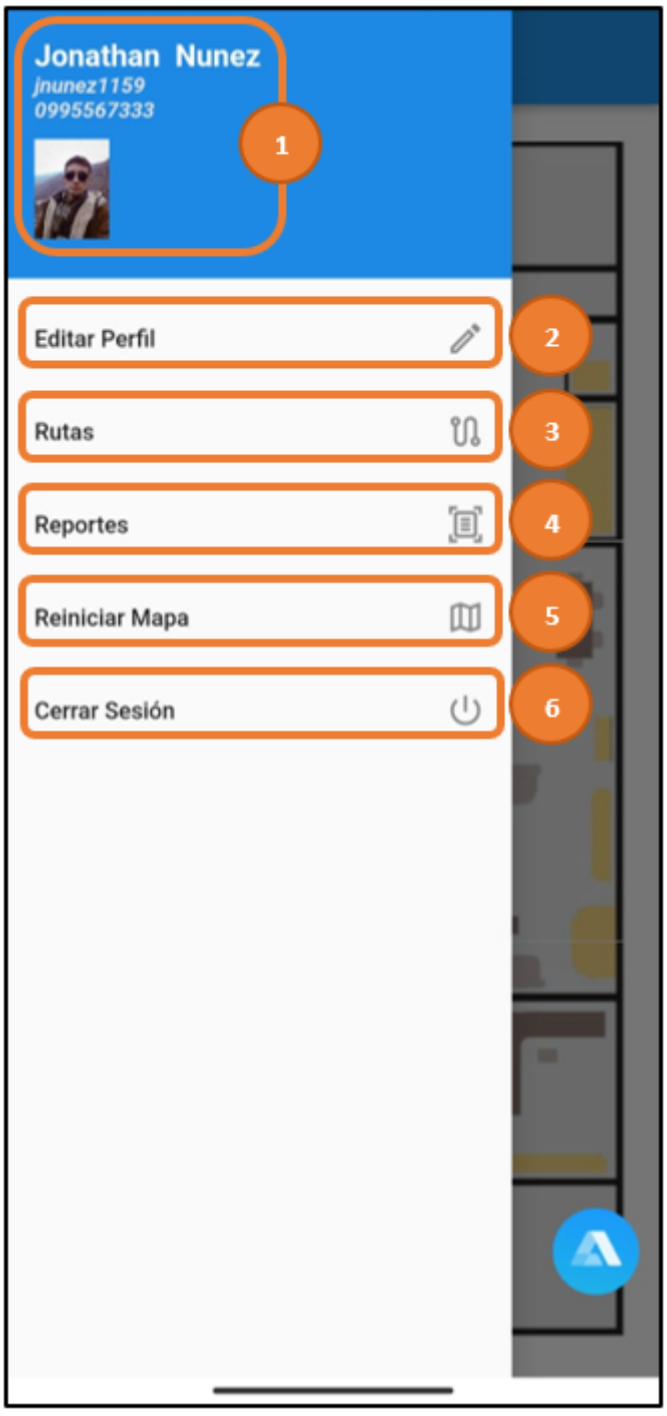
Una vez iniciada sesión correctamente, nos encontramos con la pantalla de inicio.



1. Mapa dinámico general del edificio, se lo puede ampliar o alejar.
2. Este botón da paso a más opciones que se despliegan de la parte izquierda de la aplicación.
3. Botón para activar o desactivar el asistente virtual ALAN.
4. El mapa a medida que se lo vaya ampliando mostrara información del lugar como puntos azules y la ubicación de los faros como puntos rojos.

Opciones de la parte lateral izquierda

En este apartado se muestran las opciones que tiene el usuario para ejecutar en la aplicación.



1. En esta parte se muestra la información del usuario, así como su foto de perfil.

2. La opción editar perfil permite cambiar parámetros del ingreso de usuario.

3. El apartado de rutas presenta la lista de rutas disponibles

4. La opción de reportes no estará disponible para todos los usuarios solo para los administradores y mostrará información relevante para los mismos.

5. La función de reiniciar mapa sirve para reiniciar el mapa a su posición inicial.

6. Esta opción finaliza la sesión y nos devolverá a la pantalla de Login.

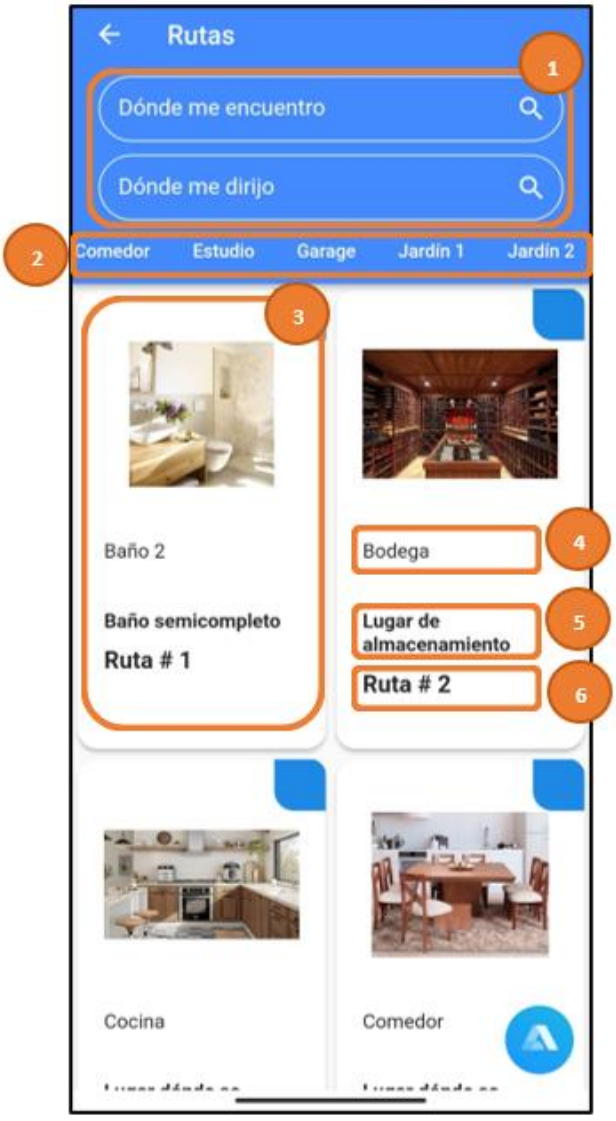
Editar perfil

En esta página se puede modificar los datos del usuario. Los campos disponibles para ser cambiados son:

1. Foto de perfil
2. Nombre de Usuario
3. Apellido de Usuario
4. Teléfono
5. Nombre de Usuario
6. Ciudad
7. Al momento de haber realizado los cambios deseados se procede a presionar el botón actualizar el cual guardara estos cambios y la vera reflejados en el próximo inicio de sesión.

Rutas

La página de rutas constituye una de las partes más importantes para la navegación dentro del edificio ya que muestra cada una de las rutas disponibles a elegir según se requiera.

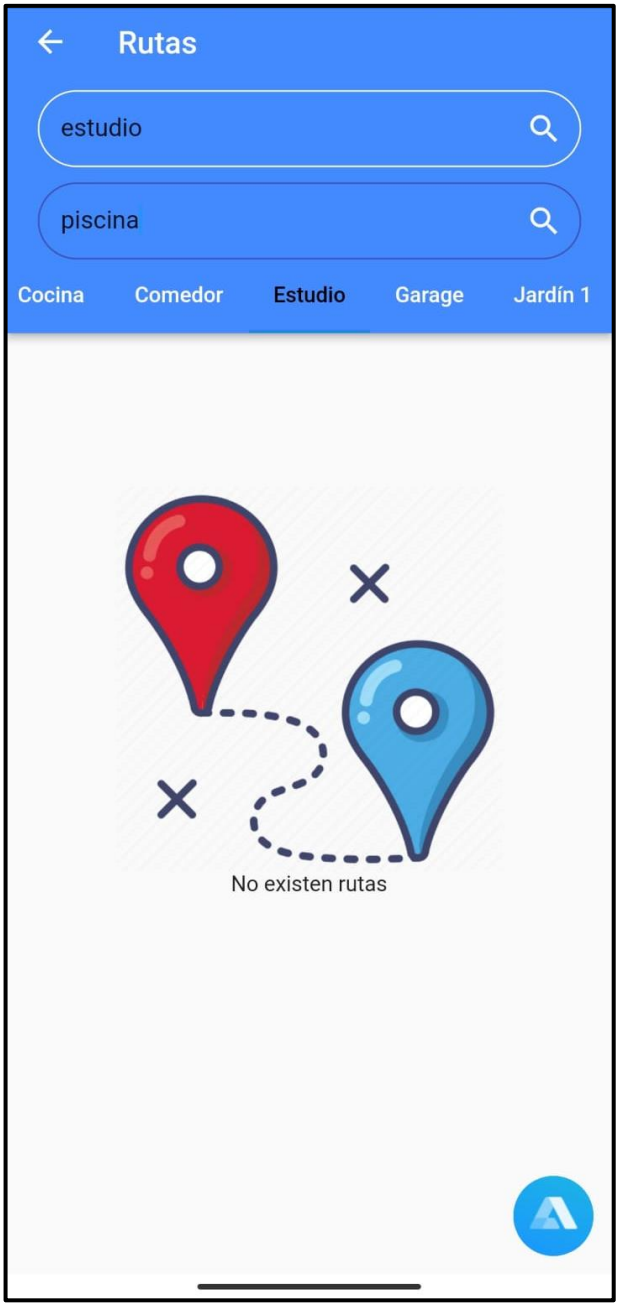


1. El buscador dinámico permite ingresar el lugar en donde nos encontramos y el lugar a donde nos queremos dirigir, haciendo así una búsqueda rápida de la ruta a seguir.
2. Otra forma de encontrar la ruta es haciendo uso de la lista en donde se encuentran ubicados todos los sitios de interés con todas las posibles rutas.
3. En cada lista se muestra un recuadro con el lugar mismo que funciona como botón que mostrara el camino a seguir.
4. Dentro del cuadro se encuentra la información de nombre del lugar
5. Se presenta una pequeña descripción del lugar
6. Representa el número de ruta establecida misma que servirá para activar el asistente virtual

Buscador dinámico

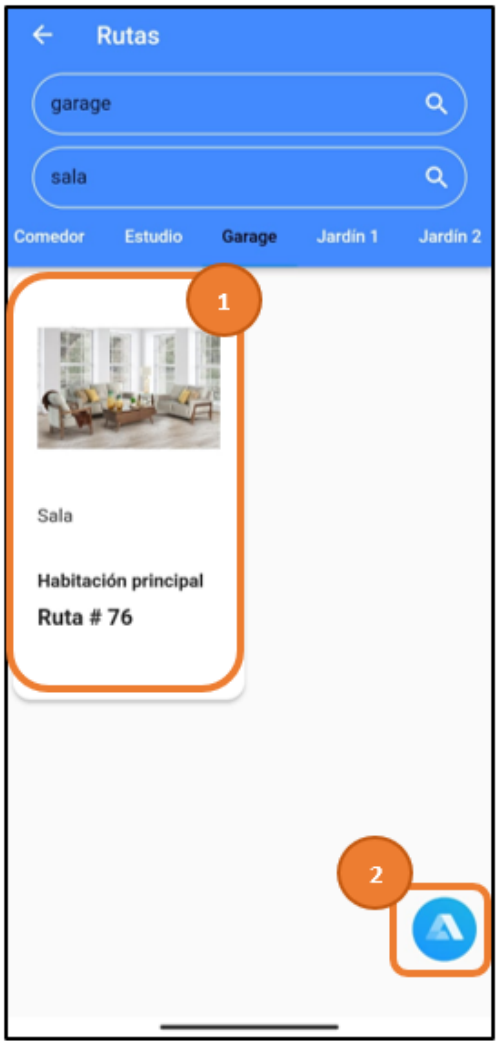
Caso 1: Ingresar mal un lugar

Al tratarse de un buscador dinámico la búsqueda la realiza al momento que se va escribiendo por lo que si se ingresa mal un valor o un lugar que no forma parte del edificio, mostrará un mensaje de ruta no existente.



Caso 2: Ingresar correctamente

En caso de haber ingresado correctamente tanto el lugar donde se encuentra como el lugar al que desea dirigirse, el buscador arroja la ruta correspondiente.

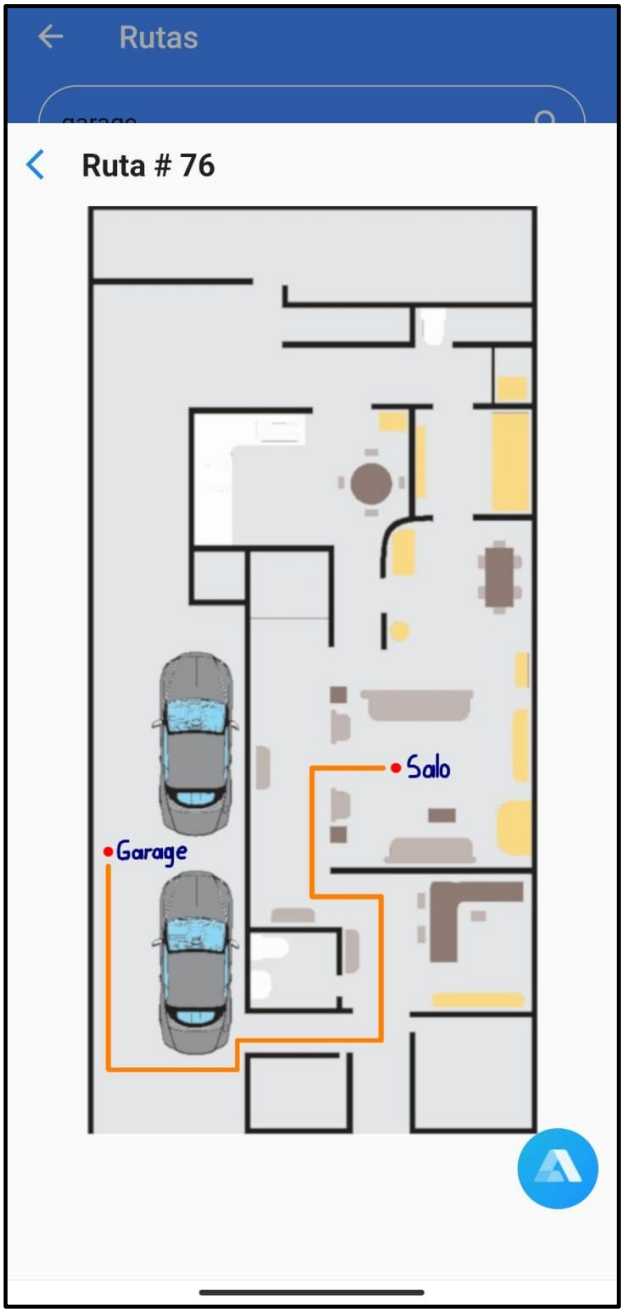


Para activar el mapa que dibujara la ruta se lo puede hacer de dos formas:

1. Presionando sobre el cuadro de la ruta
2. Activando el asistente virtual y pidiéndole que nos muestre la ruta a nuestro destino. Por ejemplo: “Muéstrame la Ruta 76”.

Generación de la ruta

Una vez activada la Ruta, se dibujará en el mapa la ruta a seguir desde el lugar donde se encuentra hasta el sitio de destino, de igual forma el asistente brinda información de tiempo estimado de llegada a su lugar de destino, como la detección de los faros para determinar el camino correcto.



Reportes

Como se mencionó anteriormente, este apartado solo se encuentra disponible para el o los usuarios administradores y en él se muestra información relevante para los mismos.

1. El apartado de sitios más visitados muestra en orden descendente los 3 sitios más visitados dentro del edificio con el número de visitas realizados al mismo.

2. El apartado de usuarios con más visitas muestra en orden descendente los 3 usuarios que realizan más visitas en el edificio.

Esta información permite al administrador del lugar tomar acción y gestionar de mejor manera el ambiente de acuerdo con sus intereses.

Funciones del asistente virtual

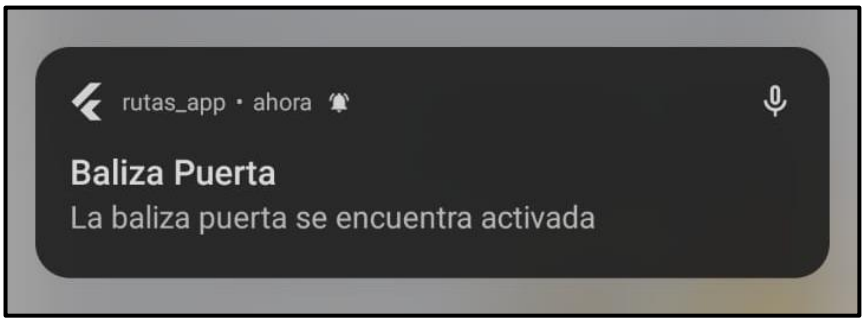
Como se pudo observar, el botón del asistente virtual aparece en la mayoría de las páginas de la aplicación y se puede hacer uso de este según se requiera. Dentro sus funciones más importantes se encuentran la de mostrar rutas y la función de escáner.

Mostrar la ruta

Como se mostró anteriormente con el comando de mostrar ruta el asistente virtual activará la ruta correspondiente, solo basta con decir “mostrar ruta (seguido del número de ruta)”.

Escáner de los faros

Esta función activa la opción de buscar los faros cercanos al lugar en el que se encuentre, recibiendo una notificación en el teléfono cada que se encuentre con un faro. Esto en conjunto con el mapa principal permite al usuario ubicarse dentro del mapa. Además, con el comando de ¨Información del faro¨ el asistente detalla los lugares que se encuentran alrededor del mismo.



Consideraciones finales

El éxito de una aplicación como esta se basa en la sencillez de uso. Sin embargo, cuando la aplicación cumple con este requisito de sencillez y practicidad, el éxito de su uso depende de la apropiación que el usuario haga de ella.

Es por eso que se hace la invitación a considerar este prototipo de aplicación como una opción para mejorar la experiencia del usuario al momento de realizar una visita a un edificio o espacio.