PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS

Trabajo de Titulación

DESARROLLO DE UNA APLICACIÓN PARA EL CONTROL DE UN DISPOSITIVO ELECTRÓNICO PARA LA UBICACIÓN DE LA MASCOTA

AUTOR:

MIGUEL SEBASTIÁN VALENZUELA SÁNCHEZ

QUITO DM, 2021

Dedicatoria

Queridos Marcelo, Susana y Johanna,

Hoy, en el culmen de este viaje académico, me siento profundamente agradecido por tenerlos a mi lado. Vuestra inquebrantable dedicación, amor y apoyo han sido pilares fundamentales en mi vida y en el logro de este hito tan significativo.

Marcelo, mi padre ejemplar, gracias por ser mi guía constante, inspirándome con tu sabiduría, perseverancia y humildad. Tus palabras de aliento y consejos siempre me han impulsado a superar obstáculos y a alcanzar mis metas más audaces. Tu labor como modelo a seguir es inigualable, y esta tesis lleva impreso el reflejo de tu espíritu incansable.

Susana, mi madre excepcional, no existen palabras suficientes para expresar mi gratitud hacia ti. Tú has sido mi roca, mi mayor defensora y mi mejor amiga. Tu amor incondicional y tus sacrificios han sido fundamentales para mi desarrollo académico y personal. Cada vez que he dudado de mí mismo, tus palabras llenas de ternura y aliento me han dado la fuerza para seguir adelante. Esta tesis es un tributo a tu amor eterno y a tu inmensa dedicación.

Johanna, mi amada hermana, tu apoyo constante y tu cariño sincero han sido un faro en mi vida. Siempre has estado ahí para animarme, celebrar mis triunfos y consolarme en los momentos difíciles. Tu inteligencia y perspicacia me han inspirado a dar lo mejor de mí. Esta tesis no solo representa mis logros, sino también el amor fraternal que compartimos y la inspiración que encuentro en ti.

Miguel Sebastián Valenzuela Sánchez

Agradecimiento

En este momento tan importante de mi vida, quiero aprovechar estas líneas para expresar mi más profundo agradecimiento hacia ustedes. Vuestra dedicación, amor incondicional y apoyo inquebrantable han sido fundamentales en mi trayectoria académica y en la realización de esta tesis.

Este logro no solo es mío, sino también de ustedes. Cada página de esta tesis lleva impregnada vuestra influencia y enseñanzas. Vuestro amor y apoyo han sido la fuerza impulsora que me ha llevado a superar desafíos y a alcanzar nuevas alturas. No hay palabras suficientes para expresar mi gratitud hacia ustedes, pero espero que estas líneas sean un testimonio sincero de mi agradecimiento eterno.

Resumen

El proyecto consistió en el desarrollo de un sistema de seguimiento y monitoreo en tiempo real utilizando Arduino GPS GY-NEO6MV2 y la placa de desarrollo ESP32-Wrover-b. Se implementó un front end en HTML para la interfaz de usuario y un back end en Python para el procesamiento de datos. Se utilizó MQTT como servicio de mensajería ligera para la comunicación entre los dispositivos.

El Arduino GPS GY-NEO6MV2 fue utilizado para la obtención de datos de ubicación mediante el uso de satélites GPS. Estos datos fueron enviados al ESP32-Wrover-b, que actuó como el controlador principal del sistema. El ESP32 procesó los datos recibidos y los transmitió al back end utilizando MQTT.

El back end, programado en Python, recibió los datos de ubicación y los procesó para su visualización en tiempo real en el front end. El intercambio de datos entre el back end y el front end se llevó a cabo utilizando MQTT, lo que permitió una transferencia eficiente de datos.

En resumen, el proyecto logró desarrollar un sistema de seguimiento y monitoreo en tiempo real utilizando Arduino GPS, ESP32, HTML, Python y MQTT, proporcionando una interfaz amigable y una comunicación eficiente entre los dispositivos.

Palabras clave: Desarrollo, Sistema de seguimiento, Monitoreo en tiempo real, Arduino GPS GY-NEO6MV2, Placa de desarrollo, Comunicación eficiente de datos

Abstract:

The project involved the development of a real-time tracking and monitoring system using Arduino GPS GY-NEO6MV2 and the ESP32-Wrover-b development board. A front end was implemented in HTML for the user interface, and a back end was developed in Python for data

processing. MQTT was used as a lightweight messaging service for communication between the devices.

The Arduino GPS GY-NEO6MV2 was utilized to obtain location data through GPS satellites. These data were sent to the ESP32-Wrover-b, which acted as the main controller of the system. The ESP32 processed the received data and transmitted it to the back end using MQTT.

The back end, programmed in Python, received the location data and processed it for real-time visualization on the front end. Communication between the back end and the front end was achieved through MQTT, enabling efficient data transfer.

In summary, the project successfully developed a real-time tracking and monitoring system using Arduino GPS, ESP32, HTML, Python, and MQTT, providing a user-friendly interface and efficient communication between the devices.

Keywords: Development, Tracking System, Real-time Monitoring, Arduino GPS GY-NEO6MV2, Development Board, Efficient Data Communication.

Contenido

CAPITULO l

1.1 Justificación

Se realiza el presente tema dado que; actualmente el uso de redes sociales o afiches en postes es muy común, en las cuales se muestran las fotos de sus mascotas y con cierta cantidad de recompensa en esta, a cambio de información sobre la misma, pese a esto, se corre mucho riesgo de resultar defraudados por gente que se dedica a estafar a las personas, por lo general la pérdida de mascotas se da también por distracción del dueño en parques o donde saquen a pasearlas, a quienes les resulta muy difícil seguir el rastro de estas cuando salen corriendo, en otras ocasiones cuando se deja la puerta principal de la casa abierta; estas se escapan y no saben cómo volver a su hogar, y sus dueños en su desesperación las buscan sin éxito en la mayoría de casos.

La presente tesis tiene como meta el desarrollo de un dispositivo electrónico que permita localizar mascotas en tiempo real, con el objetivo de prevenir pérdidas o robos, proporcionando a los dueños la ubicación exacta de sus mascotas en todo momento. Este sistema utiliza tecnologías de posicionamiento global y ofrece la posibilidad de acceder a la ubicación desde cualquier dispositivo móvil, dado que ahora el teléfono celular es una necesidad en nuestras vidas, por eso, para facilidad de la búsqueda este proyecto se realizará en una app para que se muestre en cualquier dispositivo móvil, al poder implementar este tipo de tecnología, se logra tener un mejor control de nuestras mascotas.

1.2 Planteamiento del problema

Según la psicóloga (Sabater, 2015) en el artículo “Mi perro no es una mascota, es mi familia”, el término mascota se define como un animal domesticado que convive con los seres humanos. Las mascotas pueden tener diversos efectos positivos en diferentes aspectos, como en el ámbito psicológico, fisiológico, terapéutico y social. Además, los animales domésticos pueden ser parte importante de nuestra vida emocional, proporcionando beneficios como terapia,

protección contra enfermedades cardiovasculares, reducción del estrés, alivio de la sensación de soledad y fomento de la interacción social entre los propietarios y su entorno.

Las estadísticas mundiales afirman que un 56% posee al menos una mascota, a mayoría de las personas en Latinoamérica tienen una mascota, siendo los perros los más comunes, mientrasque, en Francia y Rusia, los gatos son los preferidos; en los Estados Unidos, aproximadamente el 68% de los hogares tienen al menos una mascota, lo que equivale a unos 85 millones de hogares. Por otro lado, en Argentina, México y Brasil se encuentran entre los países con los porcentajes más altos de tenencia de mascotas. (Campos, 2015)

PetHub, una empresa que ofrece etiquetas de identificación para mascotas ha compartido estadísticas preocupantes sobre la pérdida de animales. Según sus datos, aproximadamente una de cada tres mascotas se extravía en algún momento de sus vidas. Sin embargo, las tasas de recuperación son muy bajas, ya que menos del 2% de los gatos y solo del 15 al 20% de los perros extraviados logran regresar a su hogar. Frente a esta situación, muchas personas recurren a las redes sociales para difundir información sobre la desaparición de sus mascotas y ofrecen recompensas con la esperanza de encontrarlas. Lamentablemente, en muchos casos, los dueños son engañados y extorsionados, lo que agrava aún más la situación. (Becker, 2017)

Los robos de mascotas están vinculados a la reproducción forzada de los animales, con el fin de venderlos ilegalmente o para solicitar recompensas, aprovechándose del vínculo emocional que existe entre los animales y sus dueños. Según la secretaria de salud del Distrito Metropolitano de Quito, alrededor de 3 de cada 5 familias en el área urbana de Quito poseen una mascota, lo que representa aproximadamente 600 mil perros identificados y con dueños. (Telegrafo, 2015)

1.3 Objetivo general

- Desarrollar una aplicación móvil para el control de un dispositivo electrónico para la ubicación de la mascota.

1.4 Objetivos específicos

- Analizar la información sobre el dispositivo electrónico para la ubicación de mascotas.
- Desarrollar una interfaz de rastreo para el sistema electrónico de ubicación para la mascota
- Diseñar un prototipo electrónico de rastreo de mascotas.
- Implementar los beneficios de la utilización GPS para la ubicación de mascotas.

1.5 Antecedentes investigativos

Para iniciar este proyecto se ha llevado a cabo una basta investigación de varias fuentes bibliográficas donde se encontraron; artículos científicos, revistas científicas, papers digitales, entre otros, en busca de temas concordes a nuestra tesis, los cuales son esenciales.

A lo largo del tiempo, el uso de sistemas de posicionamiento global ha experimentado una evolución significativa, adaptándose a las demandas y necesidades de la sociedad. La combinación de esta tecnología con otros dispositivos electrónicos ha permitido el rastreo de objetivos a largas distancias, satisfaciendo así las diversas exigencias requeridas por la sociedad.

Un artículo de Purina llamado “La importancia del localizador GPS de perros en viajes” determina que la probabilidad de que una mascota se pierda en un viaje aumenta si acabamos de adoptar un cachorro, o cuando decidimos viajar con nuestra mascota a un lugar desconocido para él, sobre todo si su temperamento es asustadizo y puede reaccionar de forma imprevista ante estímulos como petardos, truenos, ruidos de ambulancia u otras situaciones estresantes.

La colocación de una placa identificativa con nuestro número de teléfono en su collar, y el hecho de que lleve microchip, puede ayudar en caso de perdida, pero esta solución está condicionada a que una persona lo encuentre, y decida contactarse con el propietario o bien acudir a un agente de seguridad para que puedan identificarlo. Afortunadamente el avance de la tecnología ofrece una tercera opción complementaria, fácil y muy tranquilizante para los dueños, un localizador GPS, un dispositivo de rastreo y seguimiento de tu mascota en tiempo real, que funciona con un mecanismo muy similar al GPS para vehículos.

Actualmente, puedes encontrar en el mercado muchos modelos distintos de localizadores GPS para perros, cada vez más personalizados para cubrir las necesidades de tu perro (raza, tamaño, nivel de actividad), con funciones que contemplan desde las características más básicas (peso del dispositivo, dimensión, diseño), hasta las más sofisticadas (resistencia al agua, linterna, posibilidad de compartir la información con otras personas e, incluso, programas para evaluar el nivel de ejercicio medio de tu perro, y su estado de salud). (Purina, 2018)

Durante la última década se han realizado varias investigaciones acerca de una aplicación para el rastreo de una mascota, teniendo en cuenta las más relevantes.

El seguimiento de mascotas en ciudades inteligentes es un desafío complejo. Los métodos tradicionales, como las etiquetas de aire, GPS y RFID, no logran proporcionar el nivel completo de monitoreo y seguimiento requerido para las mascotas, además de tener limitaciones y ser costosos. Sin embargo, el rápido desarrollo de Internet de las cosas (IoT) en las ciudades inteligentes ofrece la posibilidad de controlar e interactuar con las mascotas mediante tecnologías de IoT. Este estudio presenta un enfoque de seguimiento de mascotas utilizando transmisión de video y capacidades de aprendizaje profundo para detectar y clasificar el objeto de interés. (A. A. Hammam, 2018).

Recientemente se realizaron otras investigaciones más relacionadas con el desarrollo con Arduino, como el paper “Sistema de seguimiento de mascotas con LoRa” que propone un sistema de localización de mascotas basado en IoT mediante la combinación de un teléfono inteligente con un Arduino UNO. El sistema utiliza GPS, GSM y tecnología LoRa para rastrear el movimiento de la mascota en tiempo real. El dispositivo, integrado en el collar de la mascota, envía actualizaciones sobre su ubicación al propietario a través de SMS y se visualiza en un panel LCD. Además, el mapa de Google en el dispositivo móvil muestra la ubicación de la mascota. Esta solución surge debido a la necesidad de monitorear a las mascotas, ya que un porcentaje significativo de dueños pierden a sus mascotas en algún momento. El IoT ofrece una opción asequible y fácil de implementar para aplicaciones como esta, y su uso está extendido en diversas áreas, incluyendo la atención médica y las ciudades inteligentes. (C. Akhil, 2022).

El riesgo de perder mascotas, especialmente aquellas sin correa, puede deberse a que los animales se extravían o son objeto de robos irresponsables. La búsqueda manual en el entorno cercano y la colocación de carteles de anuncios en lugares estratégicos puede ser un proceso largo y poco efectivo, ya que no garantiza encontrar a la mascota de inmediato. Por lo tanto, se realizó una aplicación y un dispositivo de seguimiento para localizar a los animales, como el uso de un dispositivo basado en GPS. Este estudio tiene como objetivo implementar la tecnología de geofencing para el seguimiento de mascotas mediante un dispositivo basado en Arduino y una aplicación Android. Los resultados de las pruebas demuestran que la aplicación y el dispositivo funcionan de manera efectiva, permitiendo recibir notificaciones cuando una mascota sale de un límite de geocerca predefinido. (Hardyanto, 2021)

1.6 Alcance

En base a lo expuesto, las personas que tienen cualquier tipo de mascota corren el riesgo de que esta pueda extraviarse o ser robada, lo cual genera preocupación y la necesidad de encontrar

soluciones efectivas para su localización. Es por ello por lo que surge el interés en investigaciones y desarrollos relacionados con aplicaciones móviles para el rastreo de mascotas.

La finalidad de estas investigaciones es desarrollar dispositivos y aplicaciones que permitan a los propietarios conocer la ubicación en tiempo real de sus mascotas de manera rápida y eficiente, minimizando el uso de recursos, tanto sociales como económicos. Para lograrlo, se exploran diferentes tecnologías, siendo una de ellas la utilización de la plataforma Arduino.

Arduino es una plataforma de hardware de código abierto que permite desarrollar prototipos electrónicos de manera accesible y flexible. Su versatilidad y facilidad de uso la convierten en una opción popular para implementar sistemas de rastreo de mascotas. La combinación de Arduino con tecnologías como GPS, GSM o redes inalámbricas y sensores permite obtener información precisa sobre la ubicación y el estado de las mascotas en tiempo real.

El desarrollo de aplicaciones de rastreo de mascotas basadas en Arduino ofrece beneficios significativos, como una mayor precisión de ubicación, una mayor duración de la batería y una personalización más amplia de las funcionalidades según las necesidades de cada propietario.

1.7 Antecedentes

En la presente investigación, se ha realizado un análisis exhaustivo de diversos estudios e investigaciones relevantes que han sido de gran utilidad para el desarrollo de este proyecto. Entre las fuentes consultadas se incluyen repositorios digitales de diversas universidades, así como artículos y revistas científicas que abordan temas similares al propuesto. Estos recursos han sido fundamentales para la elaboración y enriquecimiento del presente trabajo.

Según el artículo 10 de la ordenanza metropolitana que determina las condiciones en las que se debe mantener a los perros y otros animales, las personas que se dediquen a la comercialización de perros deberán cumplir con los siguientes requisitos:

- Mostrar la documentación legal que autoriza la importación de dichos animales, cuando sea solicitada por las autoridades municipales.
- El proceso de adquisición incluye la inscripción de la persona que adquiere el perro, con información detallada sobre su nombre y la dirección donde el perro va a residir.
- Proporcionar la documentación de soporte necesaria a los usuarios que adquieran el producto, con el objetivo de que puedan familiarizarse con las características del mismo.

Luego de la reforma con la ordenanza de bienestar animal nº 0129 que se reformó en el 2020 dictamina en el artículo 3 que, en el contexto de la aplicación de esta norma metropolitana, se emplearán múltiples definiciones, que incluyen lo siguiente:

El artículo 388 del Reglamento al Código Orgánico del Ambiente establece que los establecimientos que se dediquen a la crianza, reproducción o venta de animales de compañía deben mantener un registro que incluya información detallada sobre las especies, el número de animales y cualquier otra información que sea requerida por el Gobierno Autónomo Descentralizado Municipal o Metropolitano del cantón correspondiente.

“El artículo 7, de las prohibiciones de los sujetos responsables establece las siguientes prohibiciones:

22.- Realizar actos de comercio, actividades, operaciones, negociaciones mercantiles, ventas u actos similares que impliquen negociaciones a título de animales vivos, que se lleven a cabo habitual u ocasionalmente en el espacio público de forma ambulatoria, así como por aquellos establecimientos físicos privados de comercio, unidades económicas, entes dotados de personería jurídica, que lleven a cabo estos actos u operaciones mercantiles entre los que se incluyen los sitios virtuales en los que se ofrezcan animales vivos. La Unidad de Bienestar

Animal, procederá de oficio, a fin de tomar las medidas y llevar a cabo las acciones que tiendan a la custodia y protección de estos animales conforme las disposiciones consagradas, garantizadas en la Constitución de la República del Ecuador, los convenios internacionales, la normativa nacional o metropolitana vigentes y en los protocolos existentes. En estas actuaciones se podrá actuar coordinadamente con los organismos, entidades pertinentes y los agentes del orden a fin de atender estas circunstancias." (Municipalidad de Quito, 2020)

"24.- Criar, reproducir o vender animales en criaderos no autorizados ni registrados ante la autoridad competente." (Municipalidad de Quito, 2020)

Un estudio realizado por Felipe Robledo Rueda en el 2015 llamado "GPS mascotas alternativa tecnología para perros y gatos", durante la investigación, se pudo identificar el nivel de demanda en el mercado respecto a la necesidad o deseo de contar con un sistema de seguimiento constante para las mascotas. Los resultados indican un alto interés por parte de las personas en adquirir un dispositivo de rastreo. Sin embargo, también se evidenció la falta de conocimiento y manejo de esta tecnología, lo cual representa un aspecto crucial a considerar y mejorar en el desarrollo del proyecto. (Rueda, 2015)

En este apartado, se presentarán los conceptos esenciales para comprender el funcionamiento y operatividad del sistema en cuestión. Se explicará cómo se integra con otras tecnologías y aplicaciones, permitiendo una visión clara del rendimiento del dispositivo durante el proceso de rastreo.

1.8 Electrónica

Es una rama de la física cuyo fin es controlar el flujo de los electrones o de cualquier partícula cargada eléctricamente, se puede decir que la electrónica se complementa con el análisis de los electrones y su aplicación para los principios en contextos diferentes: los circuitos electrónicos logran el manejo de la energía eléctrica a través de la conversión y distribución de esta.Un

sistema eléctrico se forma por sensores que reciben señales físicas y las convierten en señales de corriente. (Millahual, 2020)

1.9 Componentes electrónicos

Hay diferentes maneras de identificar y clasificar los componentes electrónicos, por su estructura física; discretos e integrados, según el material de fabricación; semiconductores, no semiconductores y por su funcionamiento; activos y pasivos.

- **Activos**

  Son componentes electrónicos que pueden manejar el flujo de corriente, en la primera generación de estos, existían las válvulas, utilizadas en el radio o televisión.

- **Pasivos**

  Son componentes electrónicos que se encargan de establecer una conexión entre dos componentes de clasificación activos para asegurar que estas señales eléctricas puedan transmitirse. (Millahual, 2020)

1.10 Circuitos Electrónicos

Son circuitos eléctricos que poseen dispositivos como transistores, válvulas entre otros, pueden realizar funciones complejas utilizando cargas eléctricas. Es una asociación de varios componentes que cuando funcionan en conjunto, realizan cierto tratamiento de las señales eléctricas.

1.11 Placas de desarrollo

El primer microcontrolador comercial, introducido por Intel en 1971, marcó un hito importante en la historia de la informática. Inicialmente, estos microcontroladores eran chips de 4 bits, pero pronto evolucionaron hacia diseños de 8 bits, como el chip 8008, que sentó las bases para

el desarrollo de las primeras computadoras personales. Paralelamente, surgieron otros procesadores como el Z80 y el 6502, que también desempeñaron un papel importante en el avance de la tecnología informática.

En aquellos tiempos, el acceso a las herramientas de desarrollo estaba limitado debido al alto costo de los equipos de hardware y los sistemas de desarrollo eléctrico. Sin embargo, con la aparición de las placas de desarrollo, esta situación comenzó a cambiar. Las placas de desarrollo, como Arduino, se convirtieron en plataformas accesibles que democratizaron el acceso a las herramientas de desarrollo. Estas placas permitieron a los entusiastas y profesionales de la electrónica y la programación experimentar, crear prototipos y desarrollar proyectos sin las restricciones económicas anteriores.

Arduino, en particular, se volvió extremadamente popular debido a su simplicidad y versatilidad. Ofrecía una interfaz fácil de usar y una amplia gama de componentes y sensores compatibles, lo que facilitaba a los usuarios la construcción de diversos proyectos. Además, Arduino contaba con una comunidad activa y colaborativa que compartía conocimientos, ejemplos de código y proyectos, lo que fomentaba el aprendizaje y la creatividad.

En resumen, el surgimiento de las placas de desarrollo, como Arduino, democratizó el acceso a las herramientas de desarrollo de hardware y eléctrico, permitiendo a las personas experimentar, prototipar y desarrollar proyectos de manera más accesible y económica. Estas placas abrieron un mundo de posibilidades para la innovación y el aprendizaje en la electrónica y la programación, y su popularidad continúa creciendo hasta el día de hoy.

CAPITULO ll MARCO TEORICO

2.1 Herramientas tecnológicas para diseñar un prototipo electrónico de rastreo de mascotas

2.1.1 Arduino

Arduino se refiere específicamente a las placas y componentes físicos que componen esta plataforma de desarrollo electrónico de código abierto. El diseño del hardware es una parte fundamental de la filosofía de Arduino, ya que busca proporcionar una solución accesible y versátil para usuarios de todas las edades y niveles de experiencia en electrónica. A continuación, se describen los aspectos más destacados del hardware de Arduino:

Placas Arduino: La plataforma ofrece una variedad de modelos de placas, cada uno con diferentes características y capacidades para satisfacer las necesidades específicas de los proyectos. Algunos modelos populares incluyen Arduino Uno, Arduino Mega, Arduino Nano y Arduino Due, entre otros.

Microcontroladores: Cada placa Arduino está basada en un microcontrolador, que es el cerebro del sistema. Los microcontroladores utilizados en las placas Arduino provienen de diferentes fabricantes, siendo el más común el fabricante Atmel/Microchip.

Pines de E/S: Las placas Arduino cuentan con pines de Entrada/Salida (E/S) que permiten conectar sensores, actuadores y otros dispositivos externos para interactuar con el entorno. Estos pines se utilizan para leer y escribir datos digitales o analógicos.

Puerto USB: La mayoría de las placas Arduino cuentan con un puerto USB que se utiliza para cargar el código (sketch) desde el Arduino IDE y también para establecer una comunicación serie con otros dispositivos.

Regulador de voltaje: Cada placa Arduino incluye un regulador de voltaje que permite alimentar la placa con una fuente de voltaje más alta (como una batería o fuente de alimentación) y proporcionar el voltaje necesario para el funcionamiento adecuado del microcontrolador y otros componentes.

Conectividad: Algunos modelos de Arduino incorporan características adicionales, como conectividad inalámbrica a través de módulos Wi-Fi o Bluetooth, lo que permite una mayor interacción con otros dispositivos y servicios en la nube.

Herramientas de programación y depuración: Además del hardware, Arduino proporciona una amplia gama de herramientas de programación y depuración a través de su entorno de desarrollo integrado (Arduino IDE), lo que facilita la creación y carga de código en la placa.

En conclusión, el hardware de Arduino es una combinación de placas, microcontroladores y componentes físicos que permite a los usuarios desarrollar proyectos electrónicos de manera accesible y versátil. Su diseño flexible y sus numerosas características han hecho que Arduino sea una plataforma popular y ampliamente utilizada en proyectos de electrónica, robótica, automatización y muchas otras aplicaciones creativas e innovadoras.

2.1.2 GPS

El GPS (Sistema de Posicionamiento Global, por sus siglas en inglés) es una tecnología de navegación por satélite que posibilita la determinación exacta de la ubicación geográfica de un receptor en cualquier lugar del mundo. Este sistema se basa en una red de satélites en órbita terrestre, desarrollada por el Departamento de Defensa de los Estados Unidos. A través de señales emitidas por los satélites, el receptor puede calcular su posición y proporcionar datos precisos de latitud, longitud y altitud. El GPS ha revolucionado numerosas áreas como la navegación, la geolocalización de dispositivos móviles, la cartografía y la logística. Su uso se

ha vuelto común en aplicaciones de rastreo de vehículos, seguimiento de mascotas, navegación en automóviles y dispositivos móviles, entre otros. Gracias a la constante mejora y popularización de esta tecnología, el GPS se ha convertido en una herramienta fundamental para diversas aplicaciones en el ámbito personal y empresarial en todo el mundo.

El GPS funciona mediante la triangulación de señales de al menos cuatro satélites GPS que envían señales de radio al receptor. El receptor recibe estas señales y calcula el tiempo que tarda cada señal en llegar, lo que le permite determinar la distancia a cada satélite. Utilizando esta información de distancia y la posición conocida de los satélites, el receptor puede calcular su propia ubicación geográfica con una alta precisión.

El GPS tiene una amplia gama de aplicaciones, desde la navegación en automóviles y dispositivos móviles hasta el rastreo de objetos o personas, la cartografía, la geodesia y la meteorología. Proporciona información precisa de latitud, longitud y altitud, así como la velocidad y dirección de movimiento.

El GPS ha revolucionado la forma en que nos desplazamos y nos ubicamos en el mundo. Su precisión y disponibilidad global lo hacen una herramienta invaluable en muchos campos y ha facilitado la vida diaria de las personas en todo el mundo.

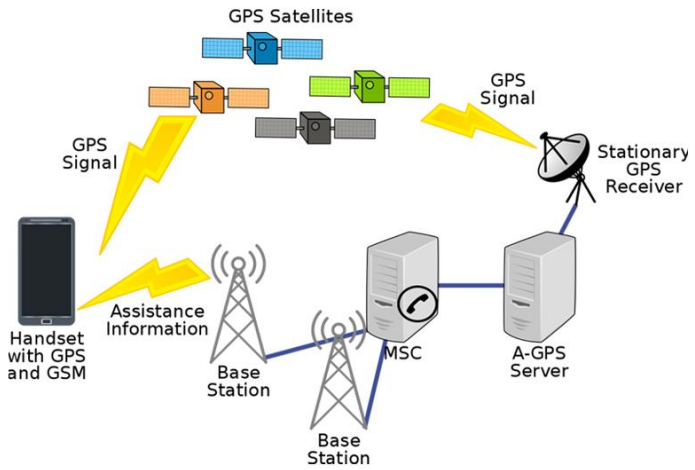


Figura 1: Funcionamiento del GPS. En Picoloro. Fuente: Huertas Caicedo, R. A. (s.f.). Ecoculturismo. Recuperado de http://picoloro.co/gps-que-es-como-funciona-aplicaciones/

2.1.3 GSM

GSM (Global System for Mobile Communications) - Sistema Global para Comunicaciones Móviles; es una norma ampliamente aceptada y establecida de telecomunicaciones móviles desarrollado inicialmente por el Grupo Especial de Servicios Móviles en Europa (GSM) a principios de los años 80. Es el sistema de comunicaciones móviles más ampliamente adoptado en todo el mundo.

GSM es una tecnología digital que utiliza una combinación de división de tiempo (TDMA) y división de frecuencia (FDMA) para permitir la transmisión de voz y datos de manera eficiente. Este estándar establece las especificaciones técnicas para la red de telefonía móvil, los protocolos de comunicación y los servicios ofrecidos.

Las características clave de GSM incluyen:

- **Comunicaciones digitales**

  GSM utiliza tecnología digital para codificar y transmitir la voz, lo que proporciona una mayor calidad de sonido en comparación con los sistemas analógicos anteriores.

Portabilidad: GSM permite la portabilidad de los dispositivos móviles, lo que significa que los usuarios pueden cambiar de un operador a otro manteniendo el mismo teléfono móvil.

- **Roaming internacional**

Facilita el roaming internacional, lo que permite a los usuarios utilizar sus dispositivos móviles en redes de operadores de diferentes países.

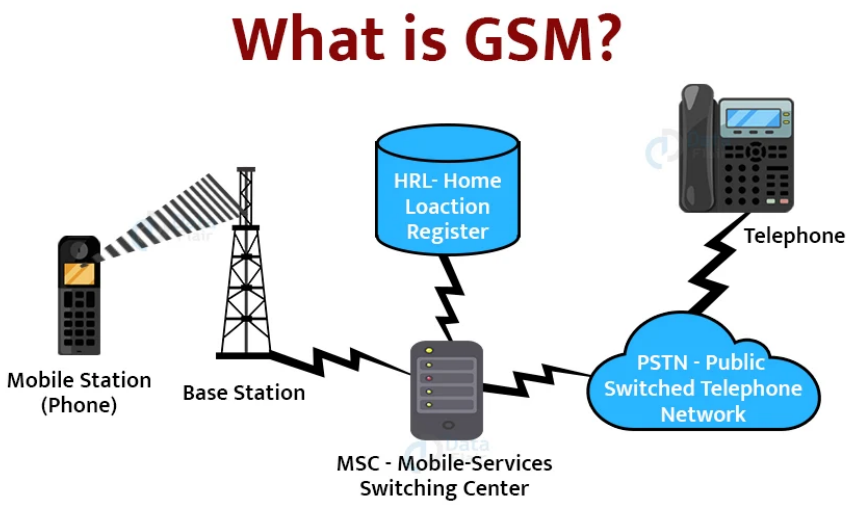
Mensajes de texto: GSM fue pionero en la funcionalidad de mensajería de texto (SMS), que habilita a los usuarios para enviar y recibir mensajes de texto cortos.

- **Transferencia de datos**

GSM también admite servicios de transferencia de datos, como la navegación por internet en dispositivos móviles y el intercambio de correos electrónicos.

GSM ha sido el precursor de tecnologías como 3G, 4G y 5G, sentando las bases para el desarrollo y mejora de las comunicaciones móviles. A lo largo del tiempo, estas generaciones sucesivas han expandido la cobertura, mejorado la calidad de las comunicaciones y transformado la forma en que nos conectamos y accedemos a la información en la era digital. El legado del GSM perdura en el avance constante hacia una conectividad más rápida, confiable y ubicua en el mundo móvil actual. Aunque ha sido reemplazado en gran medida por tecnologías más avanzadas, aún se utiliza en muchos países y proporciona una amplia cobertura y compatibilidad global.

Esquema de funcionamiento del GSM



. *Figura 2: Funcionamiento del GSM. Fuente: Data Flair* CDMA vs GSM – Difference Between GSM and CDMA*Recuperado de* *https://data-flair.training/blogs/cdma-vs-gsm/***2.1.4 ESP32-WROVER-B**

El ESP32-WROVER-B es una tarjeta de desarrollo que utiliza el microcontrolador ESP32 fabricado por Espressif Systems. Es una versión mejorada del módulo ESP32-WROOM-32 con algunas características adicionales.

El ESP32-WROVER-B incluye un potente procesador de dos núcleos Xtensa LX6 de 32 bits, con una frecuencia de operación de hasta 240 MHz. También cuenta con 520 kilobytes de memoria RAM interna y 4 megabytes de memoria flash, lo que proporciona suficiente capacidad para almacenar programas y datos.

Una de las características distintivas del ESP32-WROVER-B es la inclusión de un módulo de memoria PSRAM externa de 8 MB. Esto proporciona una memoria adicional para almacenar grandes cantidades de datos o imágenes, lo que lo hace especialmente útil en aplicaciones que requieren un mayor espacio de almacenamiento.

Además, el ESP32-WROVER-B dispone de conectividad Wi-Fi y Bluetooth integrada, lo que permite la comunicación inalámbrica con otros dispositivos y el acceso a servicios en la nube. Además, cuenta con una diversidad de interfaces periféricas, incluyendo puertos UART, SPI,

I2C, ADC y PWM, entre otros, lo que facilita la conexión con diferentes sensores, actuadores y otros componentes electrónicos.

El ESP32-WROVER-B es compatible con el entorno de desarrollo de Arduino y se puede programar utilizando el IDE de Arduino. También es compatible con otros entornos de desarrollo, como MicroPython y ESP-IDF (Espressif IoT Development Framework), lo que brinda flexibilidad a los desarrolladores para elegir la plataforma de programación que mejor se adapte a sus necesidades.

En resumen, el ESP32-WROVER-B es una placa de desarrollo versátil y potente que combina un procesador de alto rendimiento, conectividad inalámbrica, memoria adicional y una amplia gama de interfaces periféricas. Es una excelente opción para proyectos que requieren capacidad de procesamiento, almacenamiento y conectividad en un solo dispositivo.

2.1.5 Modulo GPS GY-NEO6MV2

El GPS GY-NEO6MV2 es un módulo GPS basado en el chip NEO-6M de la compañía u-blox. Es ampliamente utilizado en proyectos de electrónica y robótica para agregar capacidades de geolocalización a dispositivos como Arduino.

Este módulo se compone de un chip GPS y una antena integrada en un pequeño circuito impreso. Proporciona información de posición, velocidad y tiempo utilizando señales de satélite. Algunas de sus características principales son las siguientes:

- **Precisión:** puede proporcionar una precisión de hasta varios metros en condiciones normales de recepción de señal. Sin embargo, la precisión real puede variar dependiendo de diversos factores, como la ubicación, la interferencia de señales y las condiciones atmosféricas.
- **Conexión:** El módulo se comunica con otros dispositivos a través de una interfaz serial (UART) utilizando líneas de transmisión y recepción de datos.

- **Alimentación:** El GY-NEO6MV2 requiere una alimentación de 3.3V a 5V, lo cual lo hace compatible con casi todas las tarjetas de desarrollo, como Arduino.
- **Protocolo de comunicación:** El módulo admite el protocolo de comunicación NMEA (National Marine Electronics Association), que es un estándar ampliamente utilizado para la transmisión de datos GPS. Esto permite la fácil integración del módulo con otros dispositivos y sistemas.

  Para utilizar el módulo GY-NEO6MV2, es necesario conectarlo correctamente a una placa de desarrollo, como Arduino, y utilizar una biblioteca compatible para acceder a los datos de posición y otros parámetros proporcionados por el módulo.

  Es importante destacar que la recepción de señal GPS puede verse afectada por la presencia de obstáculos físicos, como edificios altos o árboles densos, así como por condiciones climáticas adversas. Además, el tiempo de adquisición de señal inicial puede ser prolongado si el módulo se utiliza en una ubicación nueva o después de un largo período de inactividad.

  En resumen, el GPS GY-NEO6MV2 es un módulo confiable y ampliamente utilizado para la adición de capacidades de geolocalización en proyectos electrónicos y de robótica.

2.1.6 Módulo UPS Bateria 18650

El módulo UPS (Sistema de Alimentación Ininterrumpida) con batería 18650 es un componente electrónico diseñado para proporcionar una fuente de energía de respaldo a dispositivos o sistemas en caso de un corte o fluctuación en el suministro eléctrico. Estos módulos utilizan baterías 18650, que son baterías recargables de litio-ion conocidas por su alta capacidad de almacenamiento de energía y su tamaño compacto.

El propósito principal de un módulo UPS con batería 18650 es mantener el funcionamiento continuo de un dispositivo o sistema, incluso cuando se produce una interrupción en el suministro eléctrico. Cuando la alimentación de red está disponible y estable, el módulo UPS carga las baterías 18650 para mantenerlas en su máximo nivel de carga. En el caso de un corte de energía, el módulo UPS automáticamente conmuta a la alimentación de la batería, lo que proporciona un suministro de energía ininterrumpido a los dispositivos conectados.

Algunas características comunes de los módulos UPS con batería 18650 incluyen:

- **Circuitos de Protección:** Estos módulos incorporan circuitos de protección para garantizar una carga segura de las baterías y evitar situaciones de sobrecarga, sobre descarga y cortocircuitos.
- **Indicadores LED:** Algunos módulos tienen indicadores LED que muestran el estado de carga de la batería, el estado de la alimentación y cualquier problema que pueda surgir.
- **Tiempo de Respaldo:** La capacidad de tiempo de respaldo depende del número y capacidad de las baterías 18650 utilizadas en el módulo, lo que determina cuánto tiempo puede mantener el suministro eléctrico durante un corte de energía.
- **Capacidad de Carga:** Algunos módulos UPS con batería 18650 también pueden proporcionar la funcionalidad de cargar dispositivos conectados, como teléfonos móviles u otros dispositivos electrónicos, durante un corte de energía.
- **Conexiones Eléctricas:** Los módulos UPS proporcionan terminales o conectores para la conexión de dispositivos o sistemas que requieren respaldo de energía.

Estos módulos UPS con batería 18650 se utilizan comúnmente en sistemas de seguridad, servidores, computadoras, dispositivos de comunicación y otros equipos electrónicos críticos que necesitan una fuente de energía continua y fiable. Además, su diseño compacto y eficiente

los convierte en una opción popular para aplicaciones que requieren una protección efectiva contra cortes de energía y fluctuaciones en la red eléctrica.

2.1.7 Protoboard

Placa de pruebas que permite la inserción de elementos electrónicos y cables para construir diversos tipos de circuitos sin requerir soldadura, es usado para realizar pruebas experimentales, si la prueba resulta positiva el circuito a prueba se construye de manera permanente.

Para el desarrollo de la aplicación móvil se utilizará la metodología ágil SCRUM la cual se caracteriza en dividir el proyecto en pequeñas secciones cumpliendo así un ciclo iterativo e incremental, y así poder tomar resultados de como avanza el proyecto en cada una de las diferentes etapas.

2.1.8 Baterías de litio

Las baterías de litio, también identificadas como baterías de iones de litio (Li-ion), son una tecnología de almacenamiento de energía muy común y ampliamente utilizada en diversos dispositivos electrónicos y aplicaciones portátiles. Estas baterías son reconocidas por su gran capacidad de almacenamiento de energía, su larga duración, su ligereza y su diseño compacto, lo que las convierte en una opción ideal para una amplia variedad de dispositivos móviles, como smartphones, tablets, laptops, cámaras digitales y aparatos electrónicos de consumo.

Las baterías de litio se componen de celdas individuales que contienen un electrolito líquido o sólido y dos electrodos: un ánodo de litio y un cátodo de otro material, como óxido de cobalto, óxido de manganeso o fosfato de hierro y litio. Durante la descarga, los iones de litio se mueven del ánodo al cátodo a través del electrolito, generando una corriente eléctrica que alimenta el dispositivo. Durante la carga, los iones de litio se desplazan en sentido contrario, moviéndose desde el polo positivo (cátodo) hacia el polo negativo (ánodo).

Una de las principales ventajas que ofrecen las baterías de litio reside en su elevada cantidad de energía por unidad de volumen, lo que implica que pueden almacenar una cantidad considerable de energía en proporción a su tamaño y peso. Esta característica las convierte en una elección ideal para dispositivos portátiles que demandan una prolongada duración de la batería. Además de eso, las baterías de litio presentan una baja tasa de autodescarga, lo que indica que su pérdida de energía es más lenta cuando no están en uso, lo que las convierte en una opción adecuada para dispositivos que se utilizan ocasionalmente.

Otra ventaja significativa es su vida útil. Las baterías de litio tienen una duración mayor en comparación con otras tecnologías de baterías, recargables, lo que significa que pueden soportar un mayor número de ciclos de carga y descarga antes de que su capacidad disminuya significativamente. Esto resulta en una mayor durabilidad y una vida útil más prolongada de los dispositivos alimentados por estas baterías.

Sin embargo, también existen algunas desventajas asociadas con las baterías de litio. Una de ellas es su sensibilidad a las condiciones extremas, como altas temperaturas, que pueden reducir su rendimiento y acortar su vida útil. Las baterías de litio necesitan un mecanismo de protección que evite cargas o descargas excesivas, lo que puede incrementar la complejidad y el precio de los dispositivos que las emplean.

2.1.9 Tarjeta SIM

Una tarjeta SIM (Subscriber Identity Module) es un pequeño dispositivo que se utiliza en los teléfonos móviles y otros dispositivos compatibles con redes celulares. Es una tarjeta de circuito integrado que almacena de manera segura la información de identificación y autenticación del usuario.

La tarjeta SIM contiene información esencial, como el número de teléfono, la identidad del operador de red, la configuración de seguridad y las claves de cifrado. También almacena

contactos, mensajes de texto y otros datos relacionados con la comunicación móvil. Esta información se utiliza para autenticar al usuario en la red móvil y permitirle acceder a los servicios de voz, mensajería y datos.

Además de los teléfonos móviles, las tarjetas SIM también se utilizan en otros dispositivos, como tablets, módems USB y dispositivos de rastreo GPS. Permiten la conectividad y comunicación a través de la red celular, lo que facilita la realización de llamadas, el intercambio de mensajes y la conexión a la red en dispositivos portátiles (móviles).

En resumen, una tarjeta SIM es un componente esencial para la comunicación móvil. Almacena la información de identificación y autenticación del usuario, permitiendo el acceso a los servicios de red celular en teléfonos móviles y otros dispositivos compatibles.

CAPITULO lll METODOLOGÍA

3.1 Metodología para la investigación

La implementación del plan del proyecto se llevará a cabo mediante la recopilación de información de las investigaciones previamente mencionadas, con el objetivo de desarrollar una aplicación para el rastreo de mascotas utilizando Arduino y cumpliendo con los requisitos mínimos establecidos. En este contexto, la tesis se desarrollará bajo la metodología de investigación aplicativa-cualitativa, la cual busca resolver la problemática planteada anteriormente.

La metodología cualitativa es un enfoque de investigación que se concentra en obtener una comprensión detallada y descriptiva de fenómenos sociales y humanos. Este método implica la recopilación y análisis de datos no numéricos, como palabras, imágenes o comportamientos, con el objetivo de obtener una visión profunda de los temas de estudio. En lugar de enfocarse en la cuantificación de datos, la metodología cualitativa se basa en la interpretación y el análisis de los aspectos cualitativos y significativos de los fenómenos investigados. Este enfoque permite a los investigadores explorar y capturar la riqueza y complejidad de los sujetos estudiados, brindando una comprensión más holística y contextualizada de los mismos. Al utilizar técnicas como entrevistas, observación participante y análisis de contenido, la metodología cualitativa proporciona una perspectiva enriquecedora y valiosa para la investigación en diversas áreas de estudio.

Esta metodología busca explorar significados, perspectivas y contextos para obtener una comprensión más completa y rica de la realidad estudiada. Según Denzin y Lincoln (2011), pioneros en el campo de la investigación cualitativa, esta metodología "se ocupa de las descripciones, las palabras, los detalles y las narrativas que nos ayudan a entender el mundo social en que vivimos". (Denzin, 2011)

Para nuestro proyecto se realizó iteración y retroalimentación donde se realizaron pruebas y prototipos con usuarios reales para obtener retroalimentación cualitativa sobre la experiencia de uso y realizar mejoras iterativas en el diseño y funcionalidad de la aplicación.

Además, se utilizará la investigación aplicada, que tiene como objetivo resolver problemas prácticos y aplicar los conocimientos adquiridos para abordar situaciones reales. Se enfoca en generar soluciones concretas y prácticas, basadas en la evidencia y el análisis de datos, para mejorar procesos, productos o servicios en diferentes campos de aplicación (Hernández Sampieri, 2014), para abordar el problema principal de desarrollar una aplicación de rastreo de mascotas.

La investigación aplicada en el desarrollo de una aplicación para el control de un dispositivo electrónico para la ubicación de la mascota consiste en utilizar conocimientos existentes y soluciones previas para diseñar y crear una aplicación funcional y efectiva. Se realiza una revisión de literatura para conocer las tecnologías y enfoques relevantes. Luego, se diseña y desarrolla un prototipo basado en esa información, el cual se somete a pruebas y validación con usuarios reales para obtener retroalimentación. Con base en los resultados, se hacen mejoras y ajustes en la aplicación antes de su lanzamiento. Posteriormente, se realiza un monitoreo continuo para evaluar su rendimiento y realizar mejoras según las necesidades de los usuarios. La investigación aplicada asegura que la aplicación sea eficiente, útil y satisfactoria para los usuarios en el control y ubicación de sus mascotas..

3.2 Metodología de desarrollo del software

Una metodología de desarrollo de software es un conjunto estructurado de técnicas y metodologías que permite abordar de manera sistemática cada etapa del ciclo de vida de un proyecto de desarrollo de software. Proporciona un enfoque detallado y completo para ejecutar, gestionar y administrar el proyecto con altas probabilidades de éxito. Este enfoque incluye un

proceso sistemático para diseñar, implementar y mantener un producto de software desde su concepción hasta el logro de sus objetivos.

La implementación de una metodología de desarrollo de software trae consigo varios beneficios. Por un lado, permite optimizar los procesos y productos de software, mejorando la calidad y eficiencia en el desarrollo. Además, proporciona métodos y guías claras para planificar y ejecutar el proyecto de software de manera estructurada. Al seguir una metodología, se establece qué, cómo y cuándo se deben realizar las actividades a lo largo del desarrollo y mantenimiento del proyecto.

Es importante destacar que existen diferentes metodologías disponibles, cada una con sus propias fortalezas y debilidades. No todas las metodologías son adecuadas para todos los proyectos, ya que su elección depende de consideraciones técnicas, organizativas, de diseño y de equipo. Cada metodología se adapta mejor a ciertos tipos de proyectos, teniendo en cuenta las necesidades y características específicas de cada caso.

3.2.1 Metodología tradicional

Para el desarrollo de software de calidad abarca una cantidad enorme de actividades, las mismas tienen un gran papel en la elección de la metodología para el proyecto.

Las metodologías tradicionales son vulgarmente conocidas como metodologías pesadas; estas se concentran en llevar documentación exhaustiva de cada paso que se va realizando dentro proceso del proyecto, tanto como la planificación y control de este; tienen un orden riguroso de trabajo sobre el proceso de desarrollo de software finalizando en un software eficiente y eficaz. Esta metodología resalta la planificación total de todo el trabajo que se tiene planificado y cuando este detallado se comienza el ciclo para el desarrollo del producto de software.

Otra de las características dentro de este enfoque es el alto costo que se tiene al implementar un cambio y no ofrecer una solución para proyectos con un entorno volátil.

3.2.2 Metodologías agiles

En base a las metodologías tradicionales se genera un nuevo punto de vista llamado, métodos agiles, este nace como una respuesta a los problemas que se suelen dar en las metodologías tradicionales, tiene dos principios fundamentales, retrasar decisiones y planificación adaptativa.

El retrasar las decisiones tanto como sea posible de manera responsable tiene una gran ventaja tanto para la empresa como para el cliente, dado que siempre permite tener la satisfacción en el cliente y finalmente en el éxito del producto.

Las ventajas de retrasar las decisiones son:

- Reducir el costo en decisiones de gran cantidad de recursos económicos invertidos.
- Disminuye el número de cambios que se dan durante el desarrollo del proyecto.
- Cualquier cambio por más pequeño, tendrá un costo mínimo.

La planificación adaptativa nos brinda la flexibilidad y capacidad de adaptarnos a los cambios y situaciones que puedan surgir para cualquier cambio que pueda surgir en el método que seguimos para desarrollar nuestros proyectos. Consiste en tomar decisiones a medida que avanza la iniciativa o el trabajo planificado, convirtiéndolo en un serie de proyectos interconectados más pequeños. De esta manera, podemos ajustar y adaptar nuestras estrategias según las necesidades cambiantes y las condiciones del entorno, lo que nos proporciona mayor flexibilidad y capacidad de respuesta. Al dividir el proyecto en tareas más manejables, podemos abordar cada aspecto de manera más eficiente y

eficaz, lo que resulta en una mayor probabilidad de éxito en el logro de nuestros objetivos. Esta metodología nos ayuda a ser más ágiles y a responder rápidamente a los desafíos y oportunidades que surgen durante el desarrollo del proyecto.

3.3 Diferencias entre metodología tradicionales y agiles

Las diferencias entre las metodologías ágiles y tradicionales radican en sus enfoques de gestión y desarrollo de proyectos. Las metodologías tradicionales, al igual que el enfoque de desarrollo en cascada, se basan en una preparación exhaustiva desde el inicio del proyecto y una secuencia lineal de fases, con entregas finales al final de cada etapa. En cambio, las metodologías ágiles, como Scrum o XP, adoptan un enfoque iterativo e incremental, priorizando la adaptabilidad y la flexibilidad. Las metodologías ágiles promueven una comunicación cercana entre el equipo y el cliente, entregas frecuentes y funcionales, así como la capacidad de ajustar y mejorar el producto continuamente en respuesta a los cambios y retroalimentación. Mientras que las metodologías tradicionales buscan minimizar los riesgos a través de una planificación detallada, las metodologías ágiles reconocen que el cambio es inevitable y se adaptan rápidamente para entregar valor y satisfacer las necesidades del cliente de manera más efectiva.

3.3.1 Tabla 1: Diferencia entre metodologías tradicionales y ágiles

Metodologías Tradicionales	Metodologías Agiles
Fundamentadas en directrices provenientes de normas establecidas en el contexto del ambiente de desarrollo.	Derivadas de métodos y procesos empleados en la generación de código.
Resistencia a las modificaciones.	Dispuestos a adaptarse a cambios a lo largo del proyecto.
Proceso más controlado con varias normas y políticas.	Proceso menos controlado, menos normas y principios.
El equipo de desarrollo mantiene una comunicación constante con el cliente mediante reuniones periódicas.	El cliente es considerado como un miembro esencial del equipo de desarrollo.
Mas roles.	Pocos roles.
Equipos extensos y geográficamente dispersos.	Equipos pequeños, con menos de 10 miembros, colaborando en el mismo lugar de trabajo.

Figueroa, R. G., Solís, C. J., & Cabrera, A. A. (febrero, 2007). Metodologías tradicionales vs. Metodologías Agiles. Recuperado de https://www.researchgate.net/publication/299506242_METODOLOGIAS_TRADICIONALES_VS_METODOLOGIAS_AGILES

3.4 Selección de Metodología

Tomando en cuenta los aspectos positivos y los aspectos negativos de las metodologías tradicionales y agiles, claramente por las ventajas otorgadas en la anterior tabla, se tomará la metodología ágil para el desarrollo de este proyecto.

3.5 Tabla 2: Comparación entre Metodolgía ágil y tradicional

Metodologías	Ágil	Tradicional
Ventajas	- Flexibilidad y adaptabilidad al cambio. - Enfoque en entregas frecuentes y tempranas. - Mayor colaboración entre equipo y cliente. - Retroalimentación constante y mejora continua. - Mayor satisfacción del cliente.	- Planificación detallada y estructurada. - Estimación precisa del tiempo y costo. - Documentación completa y detallada. - Mayor control sobre el proceso de desarrollo. - Adecuado para proyectos bien definidos.
Desventajas	- Falta de documentación exhaustiva. - Puede ser difícil estimar el tiempo y costo. - Requiere la implicación activa del cliente. - Puede ser desafiante para equipos grandes. - No es adecuada para proyectos muy grandes.	- Resistente a cambios durante el desarrollo. - Menos enfoque en la satisfacción del cliente. - Menor flexibilidad ante cambios. - Mayor riesgo de retrasos y costos excesivos. - Menor colaboración con el cliente.

Valenzuela, M. (2023). Tabla 2: Comparación de Metodologías de desarrollo de software.

3.6 Tabla 3: Ventajas y desventajas de la metodología ágil

Ventajas	Desventajas
Mayor flexibilidad y adaptabilidad	Mayor dependencia de la colaboración y la comunicación efectiva
Entrega de valor temprana y continua	Requiere un equipo altamente colaborativo y autónomo
Mayor enfoque en la satisfacción del cliente	Puede ser difícil de gestionar en proyectos grandes o complejos

Mayor capacidad de respuesta a los cambios	Requiere un nivel de documentación menos exhaustivo
Mejora la comunicación y colaboración en el equipo	Puede requerir una mayor dedicación de tiempo y recursos
Mayor enfoque en la calidad del software	No es adecuada para proyectos con requisitos muy estables
Permite la identificación temprana de problemas y riesgos	Requiere una planificación y estimación más frecuente
Fomenta la retroalimentación y la mejora continuas	Puede requerir una mayor inversión en capacitación y formación

Valenzuela M. (2023). Tabla 3: Metodologías ágiles en la historia.

Es importante tener en cuenta que las ventajas y desventajas pueden variar según el contexto y la implementación específica de la metodología ágil. Además, las ventajas y desventajas pueden diferir entre las diferentes metodologías ágiles, como Scrum, Kanban, XP, etc.

3.6 Metodologías ágiles de desarrollo de software

A continuación, se presentarán brevemente algunas de las metodologías ágiles más destacadas.

3.6.1 XP- eXtreme Programming

Esta es la primera metodología ágil y la que impulsó al movimiento actual de las metodologías agiles. Surge como una nueva forma de abordar proyectos de desarrollo de software, proponiendo una metodología basada en la agilidad y la simplicidad. XP Ha demostrado ser una de las metodologías ágiles más exitosas en el desarrollo de software en los últimos tiempos, pues está diseñada para brindar el software que los clientes necesitan exactamente cuándo lo necesitan. Impulsa a los desarrolladores a responder de manera efectiva a los cambios en los requisitos que surgen de los clientes, incluso en las etapas más avanzadas del ciclo de vida del desarrollo.

3.6.1.1 Origen

Se originó durante los años 90, en respuesta a la necesidad de una metodología ágil de desarrollo de software que pueda adaptarse rápidamente y de manera

efectiva a los cambios en los requisitos del cliente, brindando una respuesta ágil y eficiente.

El término "Extreme Programming" fue acuñado por Kent Beck, un desarrollador de software estadounidense que había trabajado en varios proyectos de software de gran escala. Beck se dio cuenta de que ante el fracaso de proyectos de software debido a su rigidez e incapacidad para adaptarse a cambios en los requerimientos del cliente, se empezó a buscar soluciones para abordar este problema.

Beck desarrolló la metodología XP en la década de 1990 junto con otros líderes de la industria de software, como Ward Cunningham y Ron Jeffries. La metodología se basa en los principios de la simplicidad, la comunicación cercana y la retroalimentación constante.

En 1999, Beck publicó un libro llamado "Extreme Programming Explained: Embrace Change", en el que detallaba los principios y prácticas de la metodología XP. El libro fue muy bien recibido y ayudó a popularizar la metodología en todo el mundo.

Desde entonces, XP ha sido adoptado por muchas empresas y organizaciones en todo el mundo, y se ha convertido en una de las metodologías ágiles de desarrollo de software más populares y efectivas. Hoy en día, la metodología XP sigue evolucionando y mejorando, y se ha convertido en un enfoque popular para el desarrollo de software en equipos pequeños y grandes.

3.6.1.2 Modelo XP

La metodología XP se caracteriza por tener cuatro factores esenciales en cualquier proyecto de desarrollo de software, que son: tiempo, costo, calidad y

alcance. Estos elementos son considerados fundamentales y deben ser tomados en cuenta para lograr el éxito en el desarrollo del software.

Dentro del ciclo de vida de un proyecto XP, como en otras metodologías, entender lo que el cliente necesita, crear una solución, estimar el esfuerzo y entregar el producto final al cliente; sin embargo, el modelo XP plantea un ciclo de vida dinámico, el cual nos expone que los clientes no son capaces de definir sus requerimientos al inicio de un proyecto.

Por este motivo, el modelo XP trata de realizar ciclos de desarrollo cortos que se los llama iteraciones, con entregables funcionales al finalizar cada uno de estos ciclos. Durante cada iteración, se realiza un análisis completo del diseño, desarrollo y pruebas, siguiendo un conjunto de reglas que son características de la metodología XP. Por lo general, un proyecto que sigue XP se divide en 10 a 15 ciclos o iteraciones.

3.6.1.3 Roles XP

- **Desarrollador**: Es el miembro del equipo que es responsable de escribir código de alta calidad y funcionalidad. Los desarrolladores en XP trabajan en parejas y se turnan en el teclado para mejorar la calidad del código y reducir errores.
- **Cliente**: Es la persona o el grupo de personas que proporciona los requisitos y especificaciones del proyecto. Los clientes en XP tienen una comunicación cercana con el equipo de desarrollo y participan activamente en el proceso de desarrollo.
- **Tester**: Es el miembro del equipo que es responsable de las pruebas de calidad del software. Los testers en XP realizan pruebas automatizadas

y manuales para asegurarse de que el software cumpla con los requisitos del cliente y funcione correctamente.

- **Coach**: Es el miembro del equipo que es responsable de garantizar que la metodología XP se implemente correctamente. Los coaches en XP ayudan al equipo a mantenerse en el camino correcto, fomentan la comunicación y la colaboración y proporcionan orientación y apoyo a los miembros del equipo.

Además de estos roles principales, XP también fomenta la colaboración y el trabajo en equipo, por lo que todos los miembros del equipo deben estar dispuestos a contribuir en áreas que no sean específicas de su rol. El objetivo es lograr una mayor eficiencia, calidad y entrega de valor al cliente a través de la comunicación cercana y la colaboración de todo el equipo.

3.6.2 SCRUM

Scrum es una metodología ágil de software, es un conjunto de buenas prácticas para trabajar en conjunto con otros colaboradores (empresa), y con esto tener un mejor desenvolvimiento para poder realizar cualquier tipo de proceso para llegar a un producto final; según Schjwaber y Shutherlarnd añaden que es un proceso para construir productos, desde los años 90 se ha usado para la gestión de desarrollo de productos complejos, de define como una nueva forma y distinta de organizar el esfuerzo humano. (A. Delhij, 2015).

3.6.2.1 Origen

El concepto de Scrum tiene sus inicios en 1986 en Estados Unidos y también en Japón, las personas encargadas del desarrollo de software debían partir de una

serie de requisitos generales y su finalidad era poder publicar su producto en menor tiempo de lo esperado.

Se formalizo finalmente en 1995, creando un conjunto de reglas, o buenas prácticas con su punto principal en el desarrollo de software y lo bautizaron como Scrum, su nombre tiene origen en el deporte conocido como Rugby.

Desde entonces han sido una cantidad exorbitante de proyectos en todo el mundo que ha utilizado esta metodología.

3.6.2.2 Proceso de la metodología SCRUM

En un proyecto que se haya decidido utilizar la metodología Scrum se ejecuta en ciclos temporales cortos y de duración con un tiempo específico, este tiempo por lo regular suele ser de dos semanas, se divide en secciones, estas deben entregar un resultado integral y completo, que se trata de un incremento del producto final, el cual será proporcionado al usuario cuando lo solicite.

Scrum tiene una serie de actividades que son las siguientes:

Planificación del sprint.

- **Selección de requisitos:** El cliente que desea nuestro servicio presenta una lista de requisitos priorizada del proyecto, el equipo encargado del desarrollo toma los principales requisitos para poder completar un sprint con sus debidos involucrados.
- **Planificación de la iteración:** El equipo elabora una lista donde se encuentran las tareas necesarias y cuando se finaliza se cierra uno de tantos sprints, los integrantes de manera conjunta y miembros del equipo se autoasignan tareas para poder completar todos los requisitos.

Ejecución del sprint

- Cada día, el equipo se reúne para la sincronización del proyecto de aproximadamente 15 minutos, y se habla de lo que se adelantó del proyecto durante el día, o el día anterior dependiendo de la hora de las reuniones diarias, cada miembro del equipo va detallando su proceso en cada una de sus tareas.

Inspección y adaptación

- **Revisión:** El equipo muestra al cliente los requisitos cumplidos durante un sprint y, en base a los resultados presentados y los posibles cambios en el proyecto, el cliente puede realizar ajustes de manera objetiva desde el primer sprint.
- **Retrospectiva:** El equipo finaliza el proyecto dando a conocer cómo fue su experiencia con el proyecto, dando a conocer los pros y contras de las tareas divididas a cada miembro, y su experiencia con el cliente final.

3.6.2.3 Roles SCRUM

- **Product Owner:** Es el responsable de definir y priorizar los requisitos del producto, así como de comunicar las necesidades del cliente al equipo de desarrollo. El Product Owner trabaja estrechamente con el cliente o usuario final para asegurarse de que los requisitos del producto sean claros y estén actualizados en todo momento. Además, es responsable de mantener el Product Backlog, que es una lista priorizada de requisitos o características del producto.
- **Scrum Master:** Es el responsable de asegurar que el equipo de desarrollo siga las prácticas y procesos de Scrum y elimine cualquier

obstáculo que impida la entrega del trabajo. El Scrum Master trabaja para fomentar una cultura de mejora continua en el equipo y asegurarse de que las reuniones de Scrum, como la Daily Scrum y la Sprint Review, se realicen de manera efectiva.

- **Equipo de Desarrollo:** Está compuesto por un grupo de especialistas encargados de desarrollar y entregar el producto en un sprint determinado. Este equipo debe ser autoorganizado y multifuncional, con habilidades que abarquen todo el espectro del desarrollo de software, incluyendo análisis, diseño, programación, pruebas, etc. El equipo de desarrollo colabora estrechamente con el Propietario del Producto para comprender los requisitos del producto y programar las entregas dentro de los Sprints. Juntos, analizan y discuten los detalles de lo que se necesita implementar y planifican la ejecución de las tareas durante el período de desarrollo acordado. Esta colaboración facilita una comunicación efectiva y asegura que las metas del proyecto sean claras y alcanzables.

  En resumen, los roles en la metodología Scrum están diseñados para fomentar la colaboración, la comunicación efectiva y la toma de decisiones informada. El Product Owner se encarga de definir los requisitos y prioridades del producto, el Scrum Master garantiza que el equipo siga los procesos de Scrum y el Equipo de Desarrollo es responsable de desarrollar y entregar el producto en un sprint determinado.

3.6.2.4 Reuniones SCRUM

En la metodología ágil Scrum, las reuniones juegan un papel fundamental en la organización y el seguimiento del proyecto. Estas reuniones están diseñadas para fomentar la colaboración, la transparencia y la toma de decisiones basada en la inspección y adaptación continua. A continuación, se describen las principales reuniones de Scrum:

- **Sprint Planning (Planificación del Sprint):** Es una reunión que se lleva a cabo al comienzo de cada Sprint. En la reunión de planificación del Sprint, el equipo Scrum se congrega para establecer los propósitos y metas del Sprint, elige las historias de usuario que serán incluidas en el Sprint Backlog y traza un plan detallado para alcanzar dichos objetivos. Durante esta sesión, se determinan las tareas específicas que deben ser llevadas a cabo para el desarrollo exitoso de las funcionalidades deseadas.
- **Daily Scrum (Scrum Diario):** Se trata de un encuentro breve que se realiza diariamente, típicamente de unos 15 minutos, en la que el equipo Scrum se reúne para sincronizarse. Durante esta reunión, todos los integrantes del equipo comparten su progreso, lo que logró el día anterior, sus planes para el día actual y cualquier problema o impedimento que pueda afectar su avance. El propósito principal es mantener a todos informados y facilitar la colaboración y la resolución de problemas de manera oportuna.
- **El Sprint Review (Revisión del Sprint):** En esta reunión, que se realiza al final de cada Sprint, el equipo Scrum muestra al Product Owner y a los stakeholders el trabajo que ha sido realizado durante ese período.

Durante esta reunión, se muestra el incremento del producto desarrollado, y se busca obtener retroalimentación y comentarios para asegurar que el producto esté alineado con las expectativas y necesidades del cliente. Además, esta reunión brinda la oportunidad de revisar el Product Backlog y adaptar las prioridades o requerimientos según las necesidades del proyecto. Se revisan y se obtienen comentarios sobre las funcionalidades implementadas, y se pueden realizar ajustes en el Product Backlog.

- **Sprint Retrospective (Recapitulador del Sprint):** Es una reunión que ocurre tras la Sprint Review, donde el equipo Scrum realiza una reflexión sobre el Sprint recién concluido. Durante esta reunión, se evalúan tanto los aspectos favorables como aquellos que pueden ser mejorados en el proceso. Se identifican acciones de mejora que serán implementadas en el próximo Sprint. El objetivo es promover la mejora continua y el aprendizaje del equipo para optimizar su desempeño en futuras iteraciones. Además de estas reuniones principales, también pueden llevarse a cabo otras reuniones según las necesidades del proyecto, como reuniones de refinamiento del Backlog, para agregar detalles a las historias de usuario; o reuniones de revisión de producto, para obtener comentarios de los usuarios finales.

  Las reuniones en Scrum son oportunidades para la colaboración y la comunicación efectiva, lo que ayuda a mantener el enfoque en el objetivo del proyecto y facilita la toma de decisiones basada en la inspección y adaptación continua.

3.6.3 Prototipado

La metodología de prototipado es un enfoque de desarrollo de software que se basa en la creación de prototipos funcionales y tangibles del producto final. A través del prototipado, se busca obtener retroalimentación temprana de los usuarios y clientes, lo que permite realizar mejoras y ajustes antes de la implementación final.

3.6.3.1 Origen:

El origen de la metodología de prototipado se encuentra en los años 60 y 70, cuando surgieron los primeros enfoques iterativos en el desarrollo de software. A medida que las organizaciones y los desarrolladores buscaban formas más eficientes de desarrollar software, se comenzó a reconocer la importancia de obtener una retroalimentación temprana y continua del cliente. El prototipado se convirtió en una técnica efectiva para lograr este objetivo y se integró en el desarrollo ágil.

3.6.3.2 Características del prototipado:

Iterativo e incremental: La metodología de prototipado se basa en la construcción de prototipos iterativos y funcionales. Cada iteración se enfoca en desarrollar una parte del producto o una funcionalidad específica, lo que permite obtener una retroalimentación temprana y realizar mejoras continuas.

Enfoque centrado en el usuario: El prototipado se centra en las necesidades y expectativas del usuario final. Los prototipos se desarrollan con la intención de validar y refinar los requisitos del usuario, asegurando que el producto final cumpla con sus expectativas.

Comunicación efectiva: El prototipado facilita la comunicación entre los diferentes actores involucrados en el desarrollo del software, incluyendo desarrolladores, diseñadores y clientes. Los prototipos tangibles y funcionales permiten una comprensión más clara y precisa de las características del producto, lo que facilita la toma de decisiones y la colaboración efectiva.

Flexibilidad y adaptabilidad: La metodología de prototipado permite realizar cambios y ajustes rápidos en función de la retroalimentación obtenida. Esto permite una mayor flexibilidad y adaptabilidad durante el desarrollo del software, evitando la necesidad de realizar cambios costosos y significativos en etapas posteriores.

Enfoque en la validación: El objetivo principal del prototipado es validar y refinar las características y requisitos del software. Al obtener retroalimentación temprana y continua, se pueden realizar ajustes y mejoras para garantizar que el producto final cumpla con los objetivos establecidos.

3.7 Tabla 4: Comparación entre metodologías de desarrollo de software

	SCRUM	XP (Extreme Programming)	Prototipado
Ventajas	- Capacidad de adaptarse y ser flexible frente a cambios en los requerimientos o especificaciones del proyecto. - Enfoque en entregas iterativas y constantes. - Mayor involucramiento del cliente durante todo el proceso.	- Alta calidad del código debido a las prácticas de programación. - Entregas frecuentes y funcionales al final de cada iteración. - Enfoque en la satisfacción del cliente y la retroalimentación. - Fomenta la colaboración y la propiedad colectiva del código.	- Permite la validación temprana de requisitos y conceptos. - Facilita la comunicación y la comprensión entre cliente y equipo. - Rápido desarrollo de prototipos funcionales. - Identificación temprana de posibles problemas y mejoras. - Permite ajustes antes de avanzar a la implementación final.

	- Enfoque en la mejora continua y la retroalimentación. - Facilita la identificación temprana de problemas. - Fomenta la transparencia y la comunicación efectiva.	- Mayor involucramiento del cliente en las pruebas.	
Desventajas	- Requiere un equipo altamente colaborativo y autónomo. - Puede ser difícil de implementar en organizaciones grandes. - Requiere un Scrum Master capacitado y comprometido. - No se ajusta bien a proyectos pequeños o de corta duración.	- Requiere un equipo altamente especializado y comprometido. - Puede ser difícil de seguir en proyectos grandes y complejos. - Puede no ser adecuado para proyectos con requisitos cambiantes. - Requiere tiempo y esfuerzo para implementar todas las prácticas.	- Puede conducir a un código desordenado y mal estructurado. - Los prototipos pueden no representar fielmente el producto final. - Puede ser costoso si los cambios son frecuentes. - Requiere una buena planificación para evitar el "eterno prototipo".

Valenzuela M. (2023). Tabla 4: Comparación entre metodologías de desarrollo de software

3.8 Tabla 5: Criterios entre Scrum, XP y Protipado 1-5

Criterios	Scrum	XP (Programación Extrema)	Prototipado
Flexibilidad	4	5	5
Comunicación	5	5	5
Planificación	4	4	3
Control de calidad	4	5	3
Prácticas técnicas	3	5	4
Gestión de riesgo	4	4	3
Integración continua	3	5	3

Adecuación a equipos pequeños	5	3	4
Adecuación a proyectos grandes	3	4	4
Enfoque en el negocio	4	3	5
Enfoque en la programación	2	5	4
Velocidad de entrega	4	4	4
Facilidad de implementación	4	4	5
Adaptación a cambios	5	5	5
Total	**56**	**62**	**62**

Valenzuela M. (2023). Tabla 5: Criterios de evaluación entre la metodología Scrum y XP.

3.9 Conclusión

En general, se puede observar en la tabla que las metodologías Scrum, XP y Prototipado tienen fortalezas y debilidades en diferentes áreas. Sin embargo, al analizar los criterios específicos, se puede destacar que el prototipado presenta varias características que lo hacen una opción a considerar.

En términos de flexibilidad, comunicación y adaptación a cambios, el prototipado muestra una alta calificación, lo que indica su capacidad para ajustarse a las necesidades cambiantes y fomentar una comunicación efectiva entre los miembros del equipo. Además, su enfoque en la iteración y la retroalimentación temprana proporciona una oportunidad constante de mejora y ajuste a lo largo del proceso de desarrollo.

En cuanto a la planificación, el prototipado se destaca por su enfoque flexible y adaptativo. No está limitado por una estructura rígida de planificación, lo que permite abordar los cambios y las necesidades del proyecto de manera más dinámica.

En términos de control de calidad, el prototipado se centra en la validación temprana, lo que ayuda a identificar y solucionar problemas rápidamente antes de la implementación final. Esto contribuye a la entrega de un producto de alta calidad.

La gestión de riesgos, el prototipado se enfoca en el enfoque iterativo y la retroalimentación temprana, lo que permite identificar y mitigar los riesgos a medida que se presentan.

En cuanto a la adecuación a equipos pequeños y proyectos grandes, el prototipado muestra una calificación alta en ambos casos. Su enfoque flexible y su capacidad para adaptarse a diferentes tamaños de equipo y proyectos lo hacen una opción viable en diferentes contextos.

Por tanto, el prototipado es una metodología que presenta varias características favorables en términos de flexibilidad, comunicación, adaptación a cambios, control de calidad, gestión de riesgos, adecuación a diferentes tamaños de equipo y proyectos. Estas características lo convierten en una opción a considerar para el desarrollo de software, especialmente cuando se busca una metodología ágil y adaptable que permita obtener resultados rápidos y ajustar el producto a medida que se avanza en el proceso de desarrollo.

3.10 Herramientas de programación

Un lenguaje de programación se compone de una serie de reglas e instrucciones que son empleadas para desarrollar programas de computadora. Estas normas definen cómo los comandos y datos deben ser estructurados y presentados al sistema informático para que pueda llevar a cabo las tareas requeridas. Los lenguajes de programación actúan como un medio de comunicación entre los desarrolladores y la máquina, permitiendo que se creen aplicaciones y software que resuelvan problemas específicos o realicen diversas funciones. Estas

instrucciones son escritas por los programadores en un formato específico que la computadora pueda entender y ejecutar. El lenguaje de programación permite al programador describir tareas o algoritmos a la computadora de una manera estructurada y fácil de entender.

Hay muchos lenguajes de programación diferentes, y cada uno tiene sus propias características y ventajas. Algunos lenguajes de programación son más adecuados para tareas específicas, mientras que otros son más generales y son aplicables en una extensa gama de contextos y situaciones.

El desarrollo de lenguajes de programación ha sido un proceso continuo desde los primeros días de la computación, se han creado nuevos lenguajes y se han mejorado los existentes para adaptarse a las cambiantes necesidades y tecnologías. Los lenguajes de programación son una herramienta esencial para los programadores y desarrolladores de software, ya que permiten la creación de aplicaciones, sistemas operativos, videojuegos, aplicaciones móviles, entre otros.

3.10.1 Visual Studio Code

Visual Studio Code (VS Code) es un popular entorno de desarrollo integrado (IDE) desarrollado por Microsoft que se ha convertido en una opción preferida para desarrolladores de Python debido a su amplia gama de características y su facilidad de uso.

Una de las ventajas para desarrollar aplicaciones en Python es su gran variedad de extensiones disponibles. VS Code cuenta con una amplia gama de extensiones específicas para Python que ofrecen características como resaltado de sintaxis, autocompletado inteligente, depuración, integración con sistemas de control de versiones, administración de entornos virtuales y mucho más. Estas extensiones hacen que el desarrollo en Python sea más eficiente y productivo.

Además, Visual Studio Code ofrece una interfaz de usuario intuitiva y personalizable. Permite organizar y gestionar proyectos de Python de manera sencilla, con funciones como la navegación de archivos, la búsqueda rápida y la capacidad de abrir múltiples pestañas y ventanas. También ofrece un potente editor de texto con características como resaltado de sintaxis personalizable, múltiples cursores, plegado de código y vista previa en vivo.

Otra característica destacada de VS Code es su integración con herramientas de control de versiones como Git. Esto facilita el seguimiento de cambios en el código, la colaboración con otros desarrolladores y la gestión de ramas y repositorios.

VS Code también es altamente personalizable a través de su sistema de configuración y la posibilidad de instalar temas y extensiones adicionales. Esto permite adaptar el entorno de desarrollo a las preferencias individuales y crear una experiencia de programación más cómoda.

3.10.1 Arduino IDE

Es un conjunto de herramientas y recursos de software diseñado específicamente para programar y cargar código en placas basadas en Arduino. Arduino es una plataforma de hardware de código abierto ampliamente utilizada en proyectos de electrónica y robótica, que permite a los usuarios crear dispositivos interactivos e inteligentes de manera accesible y asequible.

El Arduino IDE proporciona una interfaz amigable y sencilla para escribir, compilar y cargar programas en las placas Arduino. Está basado en el lenguaje de programación Wiring, que es una variante simplificada de C/C++ con bibliotecas y funciones específicas para interactuar con los periféricos y sensores de la placa.

Una vez que se ha escrito el código en el Arduino IDE, se puede cargar directamente en la placa Arduino a través de un cable USB. Esto permite que el dispositivo comience a funcionar según el programa cargado.

El Arduino IDE es ampliamente utilizado por estudiantes, entusiastas y profesionales para aprender sobre electrónica y programación, así como para prototipar y desarrollar proyectos prácticos y experimentales en diversas áreas, como domótica, robótica, automatización, entre otras.

3.11 Principales lenguajes para el desarrollo de aplicativos.

Los lenguajes de programación orientados a aplicaciones móviles son aquellos diseñados para crear aplicaciones destinadas a dispositivos móviles como smartphones y tablets. Estos lenguajes habilitan a los desarrolladores para crear aplicaciones compatibles con sistemas operativos móviles tales como Android y iOS.

Los lenguajes de programación utilizados en el desarrollo de apps móviles son similares a otros lenguajes de programación, pero tienen ciertas características que los hacen más adecuados para desarrollar aplicaciones móviles. Estos lenguajes se utilizan para crear aplicaciones que pueden interactuar con la cámara, el GPS, los sensores y otras funciones del dispositivo.

Los principales lenguajes de programación para aplicaciones móviles son Java, Kotlin, Swift, Objective-C, C# y Python. Java es un lenguaje de programación ampliamente empleado en el desarrollo de aplicaciones para la plataforma Android.

Estos lenguajes de programación ofrecen herramientas y características para que los desarrolladores creen aplicaciones móviles rápidas, eficientes y con una gran experiencia de usuario. Los programadores tienen la libertad de seleccionar el lenguaje de programación más adecuado según sus requerimientos y habilidades técnicas, con el propósito de construir aplicaciones móviles que sean compatibles con los principales sistemas operativos móviles.

3.11.1 Java

Java es un lenguaje de programación basado en la orientación a objetos que ha ganado mucha popularidad en el ámbito del desarrollo de aplicaciones móviles. Fue desarrollado por Sun Microsystems en la década de 1990 y es utilizado para crear aplicaciones móviles en el sistema operativo Android.

Java es un lenguaje multiplataforma, lo que significa que el código escrito en Java se puede ejecutar en diferentes plataformas sin necesidad de reescribir el código. Esto es posible gracias a la máquina virtual de Java (JVM), que interpreta el código Java y lo ejecuta en la plataforma en la que se está ejecutando.

Para desarrollar aplicaciones móviles en la plataforma Android, se emplea el Android SDK, que es un kit de desarrollo de software que proporciona las herramientas necesarias para crear aplicaciones. Android Studio es el entorno de desarrollo integrado (IDE) más recurrente que se utiliza para desarrollar aplicaciones en teléfonos móviles Android, el cual utiliza principalmente el lenguaje de programación Java, aunque también se puede optar por utilizar Kotlin u otros lenguajes de programación compatibles con la plataforma.

Java es conocido por su sintaxis clara y fácil de leer, lo que facilita la comprensión del código y reduce la posibilidad de errores. También es un lenguaje seguro, gracias a la verificación de tipos y a la gestión automática de memoria.

3.11.2 HTML

HTML (HyperText Markup Language) – Lenguaje de Marcas de Hipertexto es un lenguaje de marcado que se emplea para organizar el contenido de páginas web. A pesar de que su propósito original era crear sitios web estáticos, actualmente se utiliza de

manera extensa en el diseño de interfaces de aplicaciones para dispositivos móviles además de sitios web dinámicos.

HTML es empleado para establecer la organización y los elementos de la interfaz de usuario de una aplicación móvil. Permite crear contenedores, botones, campos de entrada, listas y otros elementos comunes en las interfaces de usuario.

Se emplea conjuntamente con CSS (Cascading Style Sheets) para dar estilo y formato al contenido de las páginas web, para separar la lógica y la presentación de una aplicación móvil. HTML se ocupa de establecer la organización y los elementos de la interfaz, en cambio, CSS se emplea para dar estilo y diseño a dichos elementos.

Se puede combinar con otros lenguajes y tecnologías para desarrollar aplicaciones móviles más avanzadas. Por ejemplo, se puede utilizar JavaScript para agregar interactividad y funcionalidad dinámica a una aplicación móvil HTML.

A través de APIs y bibliotecas JavaScript específicas, HTML permite acceder a las funciones del dispositivo móvil, como la cámara, el GPS, los sensores y el almacenamiento local. Esto permite desarrollar aplicaciones móviles HTML con funcionalidades similares a las aplicaciones nativas.

3.11.3 Objective-C

Objective-C es un lenguaje de programación de paradigma orientado a objetos que tiene una gran popularidad en el desarrollo de aplicaciones destinadas a las plataformas iOS y macOS. Fue desarrollado a principios de los años 80 por Brad Cox y Tom Love, y se convirtió en el lenguaje de programación principal para el sistema operativo macOS y la plataforma de desarrollo de software Cocoa de Apple.

Es una extensión del lenguaje de programación C, lo que significa que los programadores que conocen C pueden aprender rápidamente Objective-C. El lenguaje

se caracteriza por su capacidad para encapsular datos y código, lo que significa que los programadores pueden crear objetos y métodos que trabajan juntos para realizar tareas específicas. Además, Objective-C es conocido por su capacidad para soportar la herencia de clases, lo que permite a los programadores crear nuevas clases que heredan atributos y métodos de clases existentes.

Fue utilizado como el lenguaje de programación principal para el desarrollo de aplicaciones para iOS y macOS hasta la introducción de Swift en 2014. Sin embargo, todavía hay muchas aplicaciones existentes en la App Store que fueron escritas en este lenguaje, y algunos desarrolladores todavía prefieren utilizarlo debido a su familiaridad y el hecho de que todavía se mantiene y se actualiza por parte de la comunidad de desarrolladores.

3.11.4 C#

C# es un lenguaje de programación desarrollado por Microsoft que se utiliza para crear aplicaciones móviles, entre otros tipos de software. Es un lenguaje de programación orientado a objetos y está diseñado para ser seguro, flexible y escalable. C# se utiliza junto con el entorno de desarrollo integrado de Microsoft (IDE) llamado Visual Studio para crear aplicaciones móviles para la plataforma Windows y para la plataforma Xamarin, que permite crear aplicaciones para dispositivos móviles iOS y Android.

Su sintaxis es similar a otros lenguajes de programación como Java y C++. Los desarrolladores pueden utilizar C# para crear aplicaciones móviles nativas, que ofrecen un rendimiento superior y una mayor integración con el sistema operativo del dispositivo. También se puede utilizar para crear aplicaciones móviles multiplataforma mediante el uso de herramientas como Xamarin, que brinda a los desarrolladores la posibilidad de utilizar C# para desarrollar aplicaciones compatibles con iOS, Android

y Windows Phone. Es compatible con varias tecnologías y marcos de trabajo, como la plataforma .NET, ASP.NET y la biblioteca de clases de .NET, lo que lo hace adecuado para crear aplicaciones móviles empresariales y de gran escala.

3.11.5 Python

Python es conocido por su sintaxis legible y su extensa biblioteca, lo que lo hace adecuado para el desarrollo rápido de prototipos y aplicaciones. Además, existen frameworks como Kivy y BeeWare que permiten desarrollar aplicaciones móviles nativas con Python, lo que amplía aún más sus posibilidades en este ámbito. Aunque no es la elección más común, su facilidad de uso y flexibilidad lo convierten en una opción a considerar para desarrolladores que deseen utilizar un lenguaje de programación ampliamente reconocido y con gran potencial en el ámbito de las aplicaciones móviles.

Es importante tener en cuenta que, si bien Python ofrece estas posibilidades para el desarrollo de aplicaciones móviles, existen otros lenguajes y tecnologías más ampliamente utilizados y específicamente diseñados para este propósito, como Java y Kotlin para Android, y Swift para iOS. Sin embargo, En ciertos contextos y para ciertas categorías de aplicaciones móviles, Python se puede considerar como la mejor alternativa para la creación de aplicativos móviles, por su amplia biblioteca de herramientas que se adaptan a cualquier tipo de dispositivo móvil.

3.12 Tabla 6: Comparación de lenguajes de programación.

	Java	HTML	Objective - C	C#	Python
Ventajas	-Portabilidad -Orientación a objetos -Biblioteca estándar robusta -Seguridad -Comunidad y soporte	-Estructura sencilla -Compatibilidad -Integración con otros lenguajes -Amplia disponibilidad de recursos -SEO-friendly	-Compatibilidad con el ecosistema de Apple -Amplia comunidad y recursos -Herencia de C -Compatibilidad con código legado -Interoperabilidad con Swift	-Lenguaje orientado a objetos -Amplia biblioteca estándar -Compatibilidad multiplataforma -Herramientas de desarrollo robustas -Soporte para integración con otros lenguajes	-Sintaxis clara y legible -Amplia biblioteca estándar -Gran comunidad y recursos -Versatilidad -Fácil integración con otros lenguajes
Desventajas	-Rendimiento relativo -Curva de aprendizaje -Verbosidad	- Limitaciones de diseño -Falta de interactividad -Mantenimiento manual	-Sintaxis compleja -Curva de aprendizaje -Menor demanda laboral	Limitado al ecosistema de Microsoft -Curva de aprendizaje inicial -Dependencia de la plataforma .NET	-Velocidad relativa -Gestión de versiones -Consumo de recursos

Valenzuela M. (2023). Tabla 6: Comparación de lenguajes de programación.

3.13 Tabla 7: Tabla de ponderación de lenguajes de programación.

Criterio	Java	HTML	Objective – C	C#	Python
Facilidad de aprendizaje	3	4	2	3	5
Disponibilidad de librerías/API’s	4	1	3	4	5
Comunidad y soporte	4	3	3	4	5
Rendimiento	4	2	4	4	3
Portabilidad		5	2	3	5
Integración con	4	2	5	4	5

Dispositivos móviles					
Facilidad de depuración	4	2	3	4	5
Total	**27**	**19**	**22**	**26**	**33**

Valenzuela M. (2023). Tabla 7: Criterios de lenguajes de programación.

3.14 Conclusión

En base a la comparación de los lenguajes de programación Java, HTML, Objective-C, C# y Python para el desarrollo de aplicaciones, se ha llegado a la conclusión de que HTML es especialmente destacado para el desarrollo de aplicaciones web. HTML, aunque no es un lenguaje para programar completo, es esencial para establecer la estructura y el contenido de las páginas web.

En el enfoque sugerido, se utilizará Python como lenguaje de programación para el backend de la aplicación, ya que Python ofrece una sintaxis clara, una amplia biblioteca estándar, una gran comunidad y recursos, y una fácil integración con otros lenguajes. Por otro lado, HTML se utilizará como lenguaje principal para el frontend, permitiendo crear la estructura de las páginas web y proporcionando una amplia compatibilidad con los navegadores, facilidad de uso y amplia disponibilidad de recursos.

Esta combinación de Python en el backend y HTML en el frontend proporcionará una solución completa y efectiva para el desarrollo de aplicaciones web. Python en el backend permitirá la implementación de la lógica de negocio, manejo de bases de datos, integración con servicios externos, entre otras funcionalidades. Mientras tanto, HTML en el frontend permitirá la creación de interfaces de usuario interactivas y atractivas.

3.15 Protocolo de mensajería ligeros

Los protocolos de mensajería ligero son protocolos diseñados para la comunicación eficiente entre dispositivos en entornos con recursos limitados, como redes de baja potencia y ancho de banda reducido. Estos protocolos se caracterizan por ser simples, livianos y consumir pocos

recursos, lo que los hace adecuados para aplicaciones de Internet de las Cosas (IoT) y sistemas embebidos.

Algunos ejemplos de protocolos de mensajería ligero son:

- MQTT
- CoAP
- AMQP
- XMPP

Estos protocolos de mensajería ligero proporcionan opciones flexibles y eficientes para la comunicación entre dispositivos en redes con recursos limitados. La elección del protocolo dependerá de las necesidades específicas de la aplicación, considerando factores como el ancho de banda, la escalabilidad, la seguridad y los requisitos de recursos.

A continuación, se definirán los principales protocolos de mensajería ligeros, con sus ventajas y desventajas respectivas.

3.15.1 MQTT (Message Queue Telemetry Transport):

Se trata de una plataforma de comunicación ligera, pensada para simplificar el intercambio de mensajes, eficiente y confiable en entornos de Internet de las cosas (IoT) y aplicaciones de mensajería instantánea. Su funcionamiento se basa en el modelo de publicación/suscripción, lo que significa que los dispositivos conectados actúan como "clientes" que pueden publicar mensajes en "topics" (temas) o suscribirse a temas para recibir mensajes específicos de interés.

El flujo de mensajes en MQTT es gestionado por un intermediario centralizado conocido como "broker". Los clientes (dispositivos) se conectan al broker a través de una red, como Internet, y pueden enviar y recibir mensajes a través de él. Cuando un

cliente publica un mensaje en un tema, el broker se encarga de enviar ese mensaje a todos los clientes que estén suscritos a ese mismo tema. Esto permite una comunicación uno a muchos, donde los datos pueden ser distribuidos de manera eficiente y selectiva a los destinatarios interesados.

La simplicidad de MQTT radica en su estructura liviana de cabeceras y la baja sobrecarga de datos, lo que lo hace ideal para dispositivos con recursos limitados, como sensores y dispositivos IoT con conexiones de ancho de banda reducido. Además, MQTT es conocido por su confiabilidad, ya que el broker puede almacenar temporalmente mensajes para clientes que estén desconectados y entregarlos una vez que se reconecten, asegurando que los datos no se pierdan.

3.15.2 CoAP (Constrained Application Protocol):

Se ha diseñado específicamente para atender las exigencias de los contextos relacionados con el Internet de las Cosas (IoT) y dispositivos con capacidades restringidas, como sensores y actuadores. Su objetivo principal es proporcionar una solución ligera y eficiente para la comunicación entre dispositivos en redes de baja potencia y baja capacidad de procesamiento, como las redes de sensores inalámbricos.

Opera sobre el protocolo de transporte UDP (User Datagram Protocol) y utiliza el modelo de interacción cliente/servidor similar a HTTP. El protocolo incluye cuatro tipos de mensajes: Confirmables, No confirmables, Acknowledgments y Resets. Estos mensajes se utilizan para realizar peticiones y obtener respuestas, y pueden tener distintos niveles de confiabilidad según el tipo de mensaje utilizado.

El funcionamiento de CoAP se basa en el uso de recursos identificados por URI (Uniform Resource Identifier), similar a las direcciones URL en HTTP. Cada dispositivo CoAP expone recursos que pueden ser accedidos mediante peticiones a

través de URIs específicos. Las operaciones de CoAP incluyen GET (obtener información), POST (crear recurso), PUT (actualizar recurso) y DELETE (eliminar recurso).

También incluye características como multicast, que permite enviar mensajes a varios dispositivos a la vez, y observación, que permite a los clientes suscribirse a recursos y recibir notificaciones automáticas de cambios.

Además, está diseñado para ser eficiente en el uso de la energía y el ancho de banda, utilizando cabeceras y opciones compactas para reducir la sobrecarga de datos. Esto lo convierte en una opción ideal para aplicaciones IoT que operan en entornos con limitaciones de recursos y conectividad.

3.15.3 AMQP (Advanced Message Queuing Protocol)

Es un protocolo de mensajería avanzado que facilita la comunicación entre aplicaciones y sistemas distribuidos, especialmente en entornos empresariales y de Internet de las cosas (IoT). Su objetivo principal es proporcionar una plataforma flexible y confiable para el intercambio de mensajes entre aplicaciones que se ejecutan en diferentes plataformas y lenguajes de programación.

Se basa en el modelo de interacción productor/consumidor, donde las aplicaciones pueden actuar como productores para enviar mensajes a una cola de mensajes, o como consumidores para recibir y procesar los mensajes de la cola. El protocolo establece una infraestructura de intermediarios, conocidos como "brokers", que actúan como intermediarios entre los productores y los consumidores.

El funcionamiento de AMQP se basa en el concepto de "exchanges" (intercambios) y "queues" (colas). Los productores envían mensajes a un exchange, que es responsable de enrutar los mensajes hacia una o varias colas, dependiendo de las reglas de

enrutamiento definidas. Luego, los consumidores pueden suscribirse a una cola específica para recibir y procesar los mensajes.

Garantiza una entrega confiable de mensajes, lo que significa que los mensajes enviados por los productores no se pierden, y los consumidores pueden confirmar la recepción de los mensajes al broker. Además, el protocolo admite diferentes tipos de mensajes, como mensajes persistentes que se almacenan en disco y mensajes no persistentes que se mantienen en memoria para una entrega más rápida.

3.15.4 XMPP (Extensible Messaging and Presence Protocol):

Es Un protocolo utilizado para la transmisión de mensajes y para indicar la disponibilidad de usuarios en tiempo real, diseñado para facilitar la comunicación instantánea y la presencia en línea en entornos de mensajería y colaboración. A menudo se le conoce como "Jabber", que es el nombre del proyecto inicial de código abierto que lo desarrolló.

Funciona bajo el principio de una red de federación descentralizada. En lugar de depender de un servidor centralizado, cada usuario se conecta a un servidor XMPP local o remoto. Estos servidores pueden comunicarse entre sí a través de una red federada, lo que permite a los usuarios de diferentes dominios interactuar y comunicarse sin problemas.

El protocolo XMPP admite varios tipos de mensajes, como mensajes de chat individuales y mensajes de grupo. Además de la mensajería, XMPP también maneja la presencia en tiempo real, lo que permite a los usuarios ver el estado en línea de sus contactos (por ejemplo, disponible, ocupado, ausente).

Una característica fundamental de XMPP es su capacidad para extenderse mediante el uso de extensiones o "XEPs" (XMPP Extension Protocols). Esto permite agregar

funcionalidades adicionales al protocolo base, como transferencia de archivos, voz y video, notificaciones, entre otras.

XMPP utiliza XML (eXtensible Markup Language) para el intercambio de mensajes, lo que facilita la interpretación y procesamiento de los datos. La comunicación se realiza a través de sockets TCP o en algunos casos a través de conexiones cifradas con TLS/SSL para garantizar la seguridad y privacidad de la información transmitida.

3.16 Tabla 8: Comparación entre protocolos de mensajería ligera.

Protocolo de mensajería ligera	MQTT	CoAP	AMQP	XMPP
Ventajas	-Eficiencia en la comunicación -Modelo de publicación/suscripción -Soporte para conectividad intermitente -Fácil integración con otros sistemas -Amplia adopción y comunidad	-Eficiencia en entornos con restricciones -Comunicación basada en REST -Soporte nativo para UDP -Extensibilidad y personalización	-Mensajería avanzada y robusta -Flexibilidad e interoperabilidad -Soporte para colas y enrutamiento avanzado	-Comunicación en tiempo real Interoperabilidad y federación -Extensibilidad y personalización
Desventajas	-Falta de seguridad nativa -Curva de aprendizaje inicial -Limitaciones en casos de uso específicos	-Menor madurez y adopción -Complejidad de configuración y administración -Escalabilidad limitada	-Mayor complejidad -Limitaciones de seguridad -Escalabilidad limitada	-Sobrecarga de ancho de banda -Complejidad de configuración -Menor adopción en comparación con otros protocolos

Valenzuela M. (2023). Tabla 8: Comparación entre protocolos de mensajería ligera.

3.17 Tabla 9: Ponderación entre protocolos de mensajería ligera.

Criterio	MQTT	CoAp	AMQP	XMPP
Eficiencia en la red	5	4	3	3
Consumo de energía	5	4	3	3
Facilidad de implementación	5	4	3	3
Seguridad	4	3	4	4
Escalabilidad	5	4	4	3
Comunidad y soporte	5	4	4	4
Total	**29**	**23**	**21**	**20**

Valenzuela M. (2023). Tabla 9: Criterios de los protocolos de mensajería ligera.

3.18 Conclusión

Basándonos en la última tabla de criterios de los protocolos de mensajería ligera, MQTT se destaca como la opción más adecuada para el desarrollo de una aplicación que brinde la ubicación exacta de una mascota. La calificación ponderada de MQTT es la más alta (29), superando a los otros protocolos como CoAP (23), AMQP (21) y XMPP (20).

Las principales ventajas que hacen a MQTT destacar son su alta escalabilidad, seguridad, eficiencia en la red, consumo de energía moderado, facilidad de implementación y su enfoque en la centralización del control, lo que lo convierte en una opción sólida y eficiente para la comunicación entre dispositivos en una aplicación de rastreo de mascotas.

Además, MQTT está diseñado para operar en redes de baja potencia y con recursos limitados, lo que lo hace ideal para dispositivos IoT y aplicaciones que requieren una transmisión de datos rápida, confiable y con baja sobrecarga.

En conclusión, si buscamos una solución eficiente, escalable y segura para una aplicación de rastreo de mascotas, MQTT es la opción recomendada basándonos en los criterios evaluados. Su rendimiento en los distintos aspectos lo convierte en una elección óptima para brindar la ubicación exacta de las mascotas de manera efectiva y confiable.

3.15 Modelo de arquitectura web

Los modelos de arquitecturas web son enfoques o estructuras que se utilizan para diseñar y desarrollar aplicaciones y sistemas web. Estos modelos proporcionan un marco de trabajo para organizar y estructurar los componentes de una aplicación web, así como las interacciones entre ellos.

Una arquitectura web eficiente y bien diseñada es fundamental para el desarrollo de aplicaciones web exitosas. Ayuda a garantizar la escalabilidad, la flexibilidad, la seguridad y el rendimiento de la aplicación. Además, facilita la colaboración entre los miembros del equipo de desarrollo y mejora la mantenibilidad y la evolución del sistema con el tiempo.

Los modelos de arquitecturas web se centran en la separación de responsabilidades y la modularidad de los componentes. Esto permite que diferentes partes de la aplicación se desarrollen de forma independiente y puedan ser modificadas o reemplazadas sin afectar al resto del sistema. También promueven la reutilización de código y la implementación de buenas prácticas de desarrollo.

Cada modelo de arquitectura web tiene sus propias características y ventajas, y la elección del modelo adecuado depende de los requisitos y objetivos específicos del proyecto. Algunos de los modelos de arquitecturas web más comunes incluyen el Modelo-Vista-Controlador (MVC), el Modelo-Vista-Presentador (MVP), el Modelo-Vista-ViewModel (MVVM) y la arquitectura en capas.

En resumen, los modelos de arquitecturas web son estructuras conceptuales que proporcionan una guía para el diseño y desarrollo de aplicaciones web. Estos modelos facilitan la organización, la flexibilidad y la mantenibilidad de las aplicaciones web, lo que a su vez contribuye al éxito y la eficiencia del desarrollo de software.

3.15.1 Modelo Cliente Servidor

El modelo cliente-servidor es un enfoque arquitectónico utilizado en el diseño de sistemas distribuidos, como las aplicaciones web. En este modelo, la comunicación y el intercambio de datos se realizan entre dos componentes principales: el cliente y el servidor.

El cliente es la parte de la aplicación que solicita y consume los servicios o recursos proporcionados por el servidor. Puede ser un navegador web, una aplicación móvil, un software de escritorio, entre otros. El cliente envía solicitudes al servidor y espera las respuestas correspondientes.

El servidor es el componente que procesa y gestiona las solicitudes realizadas por los clientes. Es responsable de proporcionar los servicios y recursos solicitados por el cliente. Se puede configurar un servidor para diversas funciones, por ejemplo, como servidor web, servidor de aplicaciones o servidor de bases de datos, entre otras opciones. Su principal labor consiste en recibir las solicitudes que provienen de los clientes, procesarlas y enviar las respuestas de vuelta a los mismos.

Una característica clave del modelo cliente-servidor es la distribución de responsabilidades y la comunicación mediante protocolos específicos, como HTTP en el caso de aplicaciones web.

Permite una arquitectura escalable y distribuida, donde múltiples clientes pueden conectarse a un servidor centralizado o a una red de servidores.

3.15.2 Modelo Punto a Punto

El modelo punto a punto es un enfoque de comunicación en redes donde los dispositivos se conectan directamente entre sí sin la necesidad de un dispositivo centralizado, como un servidor. En este modelo, los dispositivos se comunican entre sí de manera

individual, estableciendo conexiones directas y compartiendo información de manera punto a punto.

En un modelo punto a punto, cada dispositivo tiene la capacidad de actuar tanto como cliente como servidor. Pueden enviar solicitudes y recibir respuestas, estableciendo una comunicación bidireccional. Este modelo se utiliza comúnmente en redes peer-to-peer (P2P), donde los dispositivos se consideran iguales y tienen la capacidad de compartir recursos y servicios entre sí.

3.16 Tabla 10: Comparación de modelos de arquitectura web.

Modelo de Arquitectura	Cliente-Servidor	Punto a punto
Ventajas	-Separación de responsabilidades -Escalabilidad -Facilidad de mantenimiento -Mayor control y seguridad -Mejor rendimiento	-Descentralización -Escalabilidad -Eficiencia en la comunicación directa -Compartir recursos directamente entre dispositivos
Desventajas	-Dependencia del servidor -Mayor complejidad inicial -Costos de infraestructura y mantenimiento	-Configuración y mantenimiento complejos -Seguridad y protección de datos -Escalabilidad limitada

Valenzuela M. (2023). Tabla 10: Comparación de modelos de arquitectura web.

3.17 Tabla 11: Ponderación entre modelos de arquitectura web

Criterio	Cliente-Servidor	Punto a Punto
Escalabilidad	5	3
Seguridad	5	4
Eficiencia en la red	5	4
Consumo de energía	4	5
Facilidad de implementación	4	5
Centralización de control	5	2
Total	**28**	**23**

Valenzuela M. (2023). Tabla 11: Criterios de modelos de arquitectura web

3.18 Selección del modelo de Arquitectura web

Una razón por la cual se podría escoger el modelo cliente-servidor sobre los otros modelos es la escalabilidad y la gestión centralizada de datos. El modelo cliente-servidor permite separar la lógica de negocio y los datos en un servidor central, mientras que los clientes son responsables de la presentación y la interacción con el usuario.

El modelo cliente-servidor destaca por las siguientes razones:

- **Escalabilidad:** El modelo cliente-servidor permite escalar la capacidad del servidor de manera independiente de los clientes. Esto es especialmente útil en casos donde se espera un gran número de clientes que necesitan acceder y realizar operaciones en el servidor. Al escalar el servidor, se puede mantener un alto rendimiento y capacidad de respuesta para todos los clientes.
- **Centralización de datos:** En el modelo cliente-servidor, los datos se almacenan y gestionan en un servidor centralizado. Esto proporciona un control más efectivo sobre la integridad de los datos y facilita la implementación de políticas de seguridad y acceso. Además, los datos se pueden respaldar y administrar de manera más eficiente en un único punto central.
- **Mantenibilidad y actualizaciones:** Con el modelo cliente-servidor, las actualizaciones y modificaciones se pueden realizar en el servidor sin afectar directamente a los clientes. Esto facilita el mantenimiento y la evolución de la aplicación, ya que los cambios se pueden implementar de manera centralizada y distribuirse a los clientes de forma transparente.
- **Distribución geográfica:** El modelo cliente-servidor permite que los clientes se conecten al servidor a través de redes de área amplia (WAN) e Internet. Esto facilita la distribución geográfica de los clientes y su acceso remoto al servidor,

lo que resulta útil en entornos donde los usuarios están ubicados en diferentes lugares físicos.

Es importante destacar que la elección del modelo depende del contexto y los requisitos específicos del proyecto. Otros modelos, como el punto a punto, también tienen sus propias ventajas en diferentes situaciones. Por lo tanto, es necesario evaluar cuidadosamente los requisitos y las características del proyecto antes de seleccionar el modelo adecuado.

CAPITULO lV

4.1 Desarrollo de la aplicación, pantallas y como fueron diseñadas, usabilidad.

El desarrollo de una aplicación web para el rastreo de mascotas ofrece una solución tecnológica eficiente y práctica para que los propietarios puedan monitorear y localizar a sus mascotas de manera efectiva. Esta aplicación permite registrar y visualizar en tiempo real la ubicación de las mascotas a través de un sistema de geolocalización, brindando tranquilidad y seguridad a los propietarios al tener un control constante sobre el paradero de sus animales. En resumen, el desarrollo de una aplicación web para el rastreo de mascotas se convierte en una herramienta invaluable para garantizar la seguridad y bienestar de las mascotas, al tiempo que ofrece comodidad y control a los propietarios en el seguimiento de sus amados animales de compañía. Además, la aplicación web es de fácil acceso y compatibilidad con diferentes dispositivos, lo que permite a los usuarios rastrear a sus mascotas desde cualquier lugar y en cualquier momento.

4.2 Funcionalidades del aplicativo

En esta sección se Redactarán las principales funcionalidades que debería tener la aplicación.

- **Obtener Coordenadas GPS:** El aplicativo debe ser capaz de obtener las coordenadas GPS de la ubicación actual de la mascota utilizando el módulo GPS conectado a Arduino.
- **Visualización de la Ubicación:** Mostrar las coordenadas GPS de la mascota en tiempo real en un aplicativo, permitiendo que el usuario pueda ver la ubicación de su mascota.
- **Visualización de porcentaje de batería:** Mostrar el porcentaje de batería que posee la batería de litio mientras el dispositivo esté en uso.
- **Transmisión de Datos:** Transmitir las coordenadas GPS de la mascota de manera inalámbrica a un dispositivo receptor, como un teléfono móvil, utilizando un módulo de comunicación inalámbrica.

- **Batería y Consumo de Energía:** Implementar una gestión eficiente del consumo de energía para prolongar la duración de la batería y garantizar el funcionamiento continuo del rastreador de mascotas.
- **Botón de Encendido/Apagado:** Incluir un botón de encendido/apagado para controlar el funcionamiento del rastreador de mascotas y prolongar la vida útil de la batería.

4.3 Arquitectura del sistema Cliente – Servidor dentro del proyecto

Como se concluyó que se iba a trabajar con esta arquitectura, se detalla cómo funcionan las partes dentro de nuestra tesis:

Cliente:

- **Interfaz de usuario:** El cliente proporciona una interfaz de usuario a través de la cual los usuarios pueden interactuar con la aplicación. Puede ser una aplicación móvil, un navegador web u otro tipo de interfaz gráfica.
- **Envío de solicitudes:** El cliente envía solicitudes al servidor para solicitar servicios, recursos o realizar operaciones específicas.
- **Procesamiento local:** El cliente puede realizar cierto procesamiento local, como validación de datos, generación de informes o manipulación de la interfaz de usuario.
- **Recepción de respuestas:** El cliente recibe las respuestas del servidor y las muestra al usuario, ya sea como resultados de consultas, información actualizada o notificaciones.

Servidor:

- **Recepción de solicitudes:** El servidor recibe las solicitudes enviadas por el cliente y las procesa para determinar la acción a realizar.

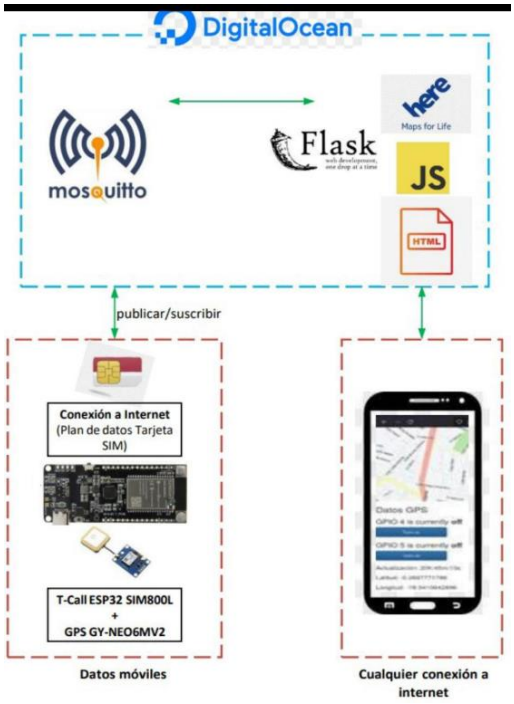
- **Procesamiento de solicitudes:** El servidor realiza el procesamiento principal de las solicitudes, lo que puede incluir acceso a bases de datos, ejecución de algoritmos, cálculos o interacciones con otros sistemas.
- **Gestión de recursos:** El servidor administra los recursos necesarios para satisfacer las solicitudes de los clientes, como el acceso a bases de datos, servicios externos o dispositivos periféricos.
- **Generación de respuestas:** El servidor genera las respuestas correspondientes a las solicitudes y las envía de vuelta al cliente. Esto puede incluir resultados de consultas, datos actualizados o mensajes de confirmación.

Es importante destacar que el modelo cliente-servidor se basa en la comunicación y cooperación entre el cliente y el servidor. El cliente solicita servicios al servidor, mientras que el servidor procesa esas solicitudes y devuelve las respuestas adecuadas. Esta división de responsabilidades permite una distribución eficiente del trabajo y facilita la escalabilidad y el mantenimiento de la aplicación.

En resumen, el modelo se compone del cliente, que interactúa con el usuario y envía solicitudes al servidor, y del servidor, que procesa las solicitudes y genera las respuestas correspondientes, en este caso coordenadas geográficas proporcionadas por la ubicación del Arduino GPS. Ambas partes trabajan en conjunto para ofrecer funcionalidades y servicios a los usuarios finales.

4.4 Flujo del Sistema

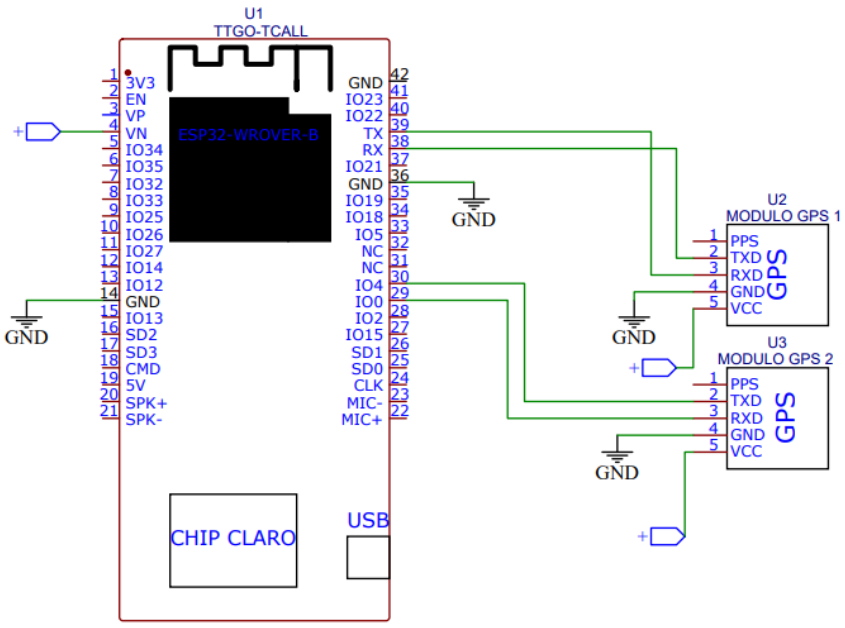
A continuación, se agregará una imagen del flujo de cómo funciona la aplicación.



Valenzuela, M. (2023). Figura 3: Arquitectura del sistema.

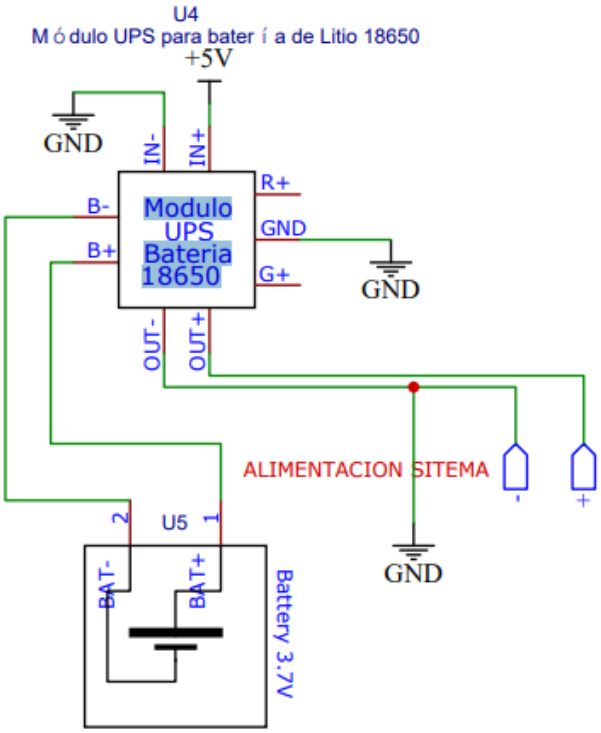
4.5 Esquema de conexión

Para la conectividad del módulo GPS se utilizaron dos módulos de Arduino GPS en caso de que falle cualquiera de estos, siendo el siguiente esquema la conexión entre los módulos y la placa ESP32-WROVER-B.



Valenzuela, M. (2023). Figura 4: Esquema de conexión entre la placa ESP32-WROVER-B y Módulos GPS.

Para que la placa esté alimentada se implementó una batería de litio 18650 que generan 5 voltios, siendo este el esquema que se realizó para la conexión entre la batería, el módulo de carga y la placa.

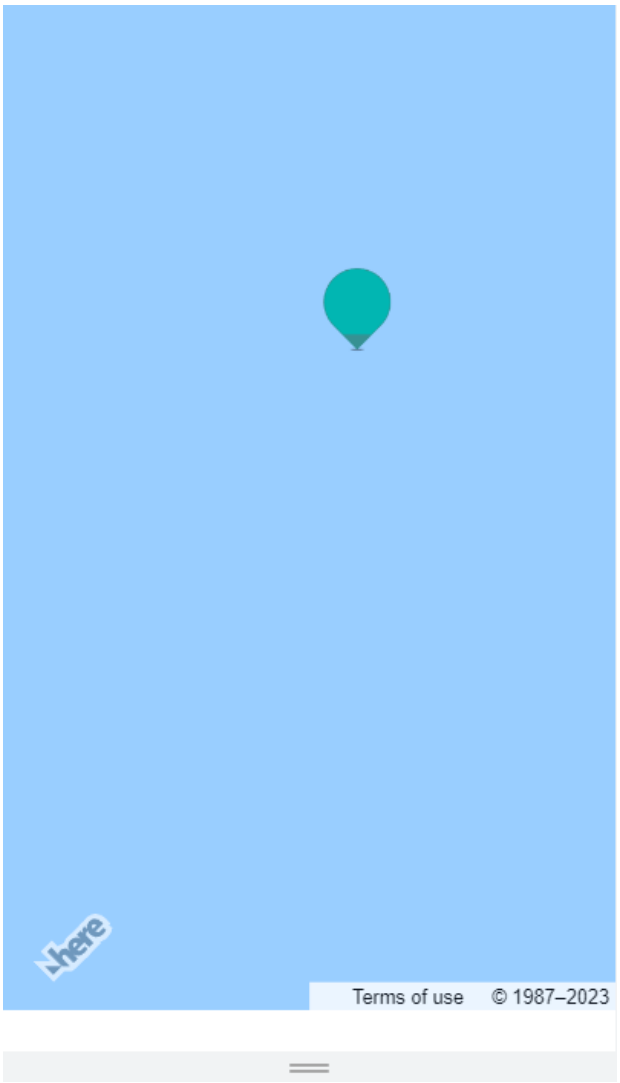


Valenzuela, M. (2023). Figura 5: Esquema de conexión entre la placa ESP32-WROVER-B, la batería de litio y el módulo UPS batería 18650.

CAPITULO V

5.1 Diseño del aplicativo

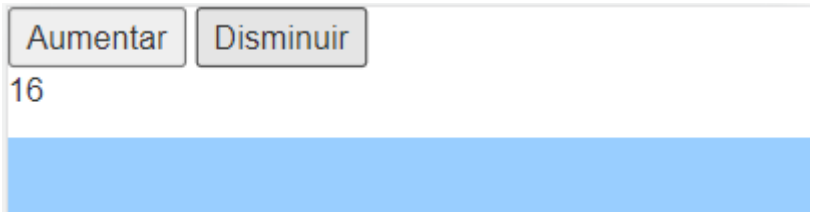
Para esta parte se utilizó HTML para realizar el front-end, en primer lugar, se tiene que importar un API de mapas de HERE dentro de nuestro código, teniendo como resultado la siguiente figura:



Valenzuela, M. (2023). Figura 6: Vector de Mapa de Here.

Con el objetivo de mejorar la usabilidad del proyecto, se incorporaron dos botones que permiten al usuario controlar el zoom del mapa de manera intuitiva y amigable. Estos botones facilitan la experiencia de navegación y brindan al usuario un mayor control sobre la

visualización de la información en el mapa con un indicador de la cantidad de zoom que el usuario ha realizado.



Valenzuela, M. (2023). Figura 7: Botones para aumentar o disminuir el zoom del mapa HERE.

Se agregaron dos botones el cual controla el encendido y apagado de ambos módulos GPS y en la parte inferior se agregó la hora de actuación, que nos da la hora de la ultima actualización y la latitud y longitud.

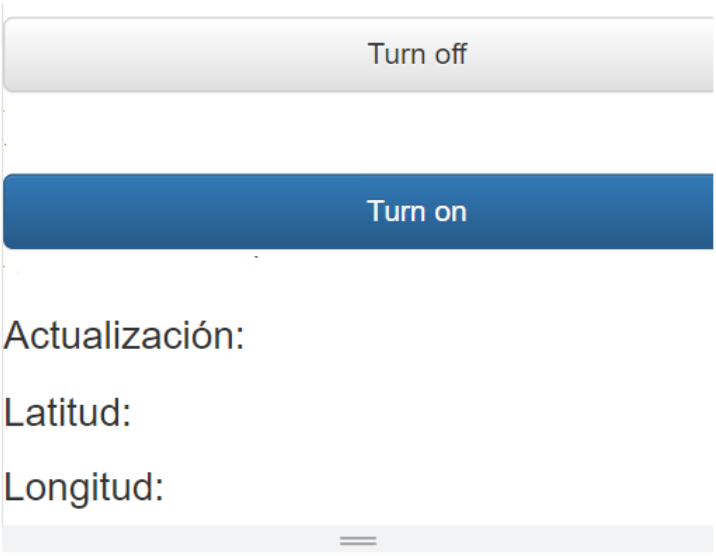


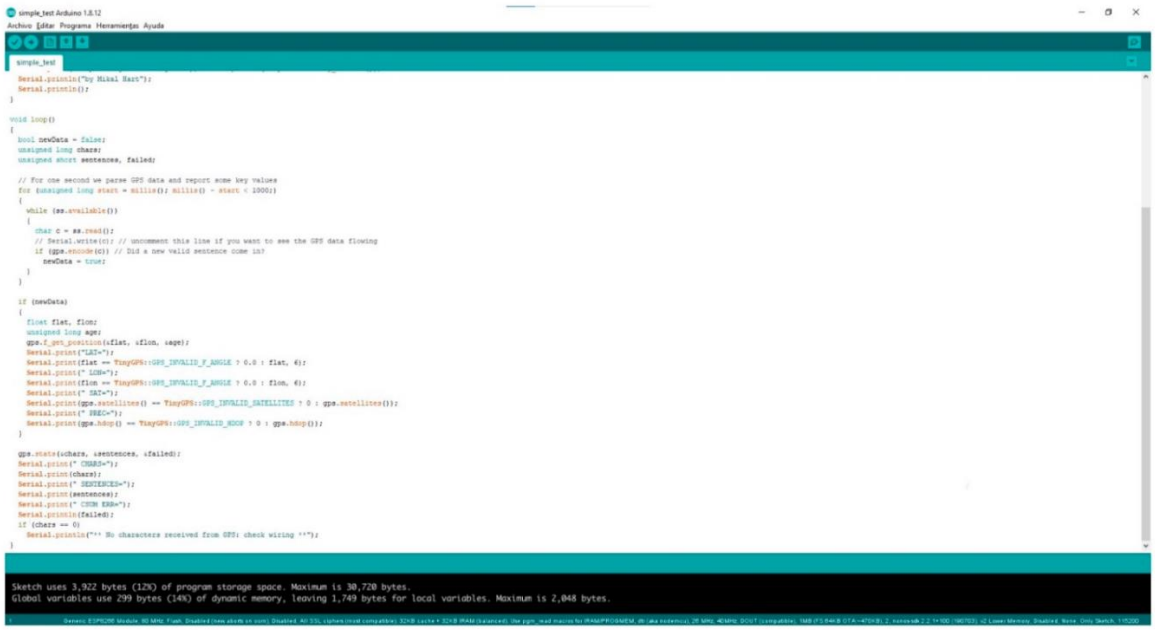
Figura 8 Botones Prender o apagar los Módulos GPS e información de ubicación y actualización.
Miguel Valenzuela (2023)

El aplicativo quedaría de la siguiente manera:

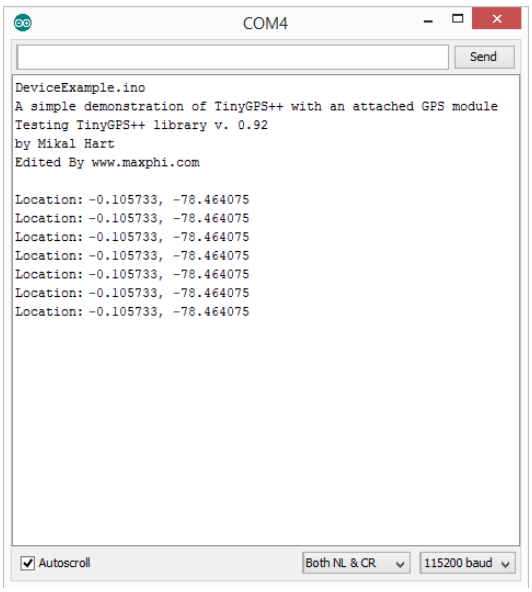
Valenzuela, M. (2023). Figura 9: Diseño del aplicativo final.

5.2 Desarrollo de la aplicación

Una vez realizada la conexión se procede a cargar la librería de Arduino TinyGPS para validar que nos trae los datos de posición correctamente, estos deberían ser latitud y longitud.



Valenzuela, M. (2023). Figura 10: Prueba del sistema con Ejemplo propio de Arduino GPS.



Valenzuela, M. (2023). Figura 11: Prueba del sistema con Ejemplo propio de Arduino GPS.

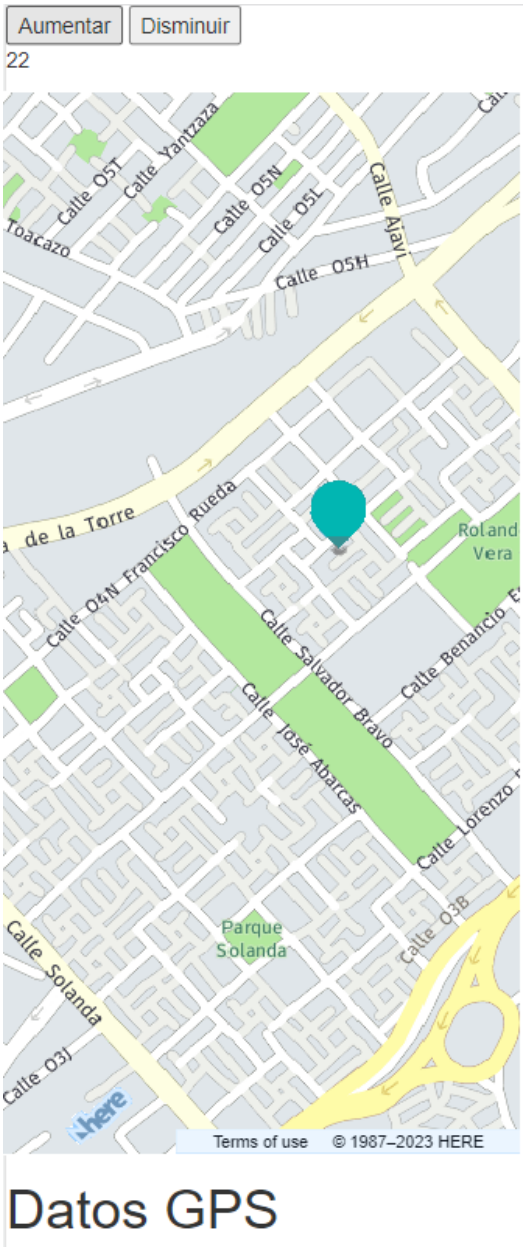
Para validar que son las coordenadas correctas se adjuntan las siguientes imágenes de validación de ubicación:

Valenzuela, M. (2023). Figura 12: Validación de ubicación donde se encuentra el Arduino GPS.

Como se observa, varia un poco la ubicación del Arduino GPS que de la ubicación marcada dentro de la aplicación Google Maps.

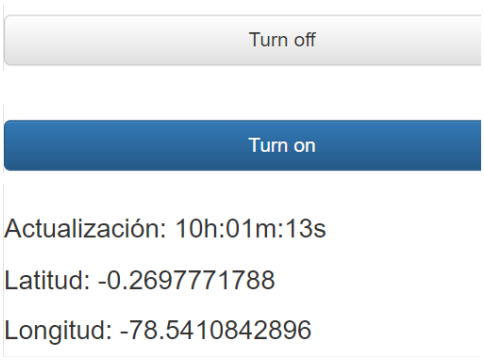
5.3 Pruebas del aplicativo

Para las pruebas finales del dispositivo de rastreo se realizaron pruebas al Sur de Quito, específicamente en el sector de la Javi.



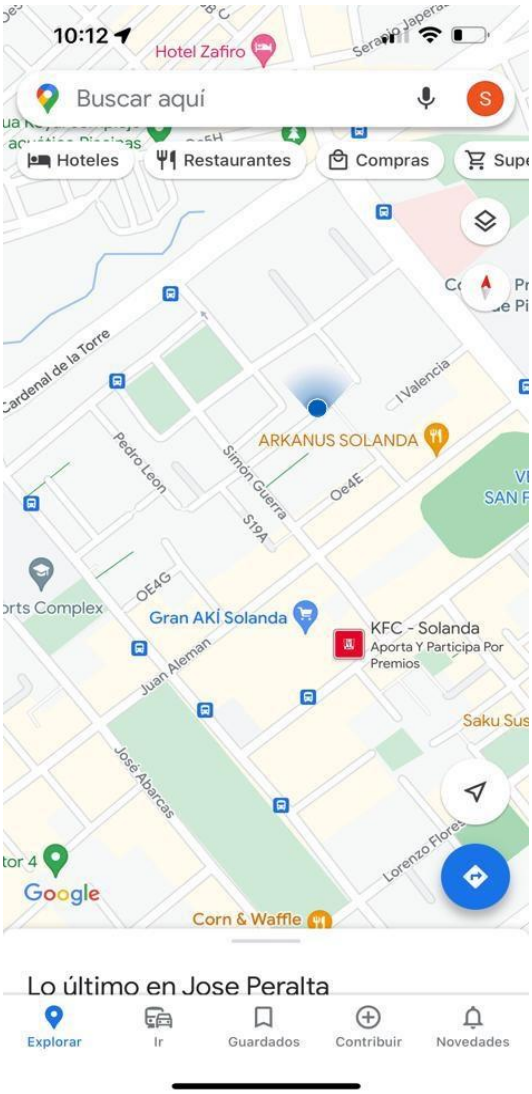
Valenzuela, M. (2023). Figura 13: Prueba de GPS ubicación por latitud y longitud.

Y en la parte inferior tenemos la información de la ubicación del mapa:



Valenzuela, M. (2023). Figura 14: Prueba de GPS ubicación por latitud y longitud.

En la siguiente figura observamos que se tiene un pequeño margen de error nuestro Módulo GPS al GPS de Google Maps

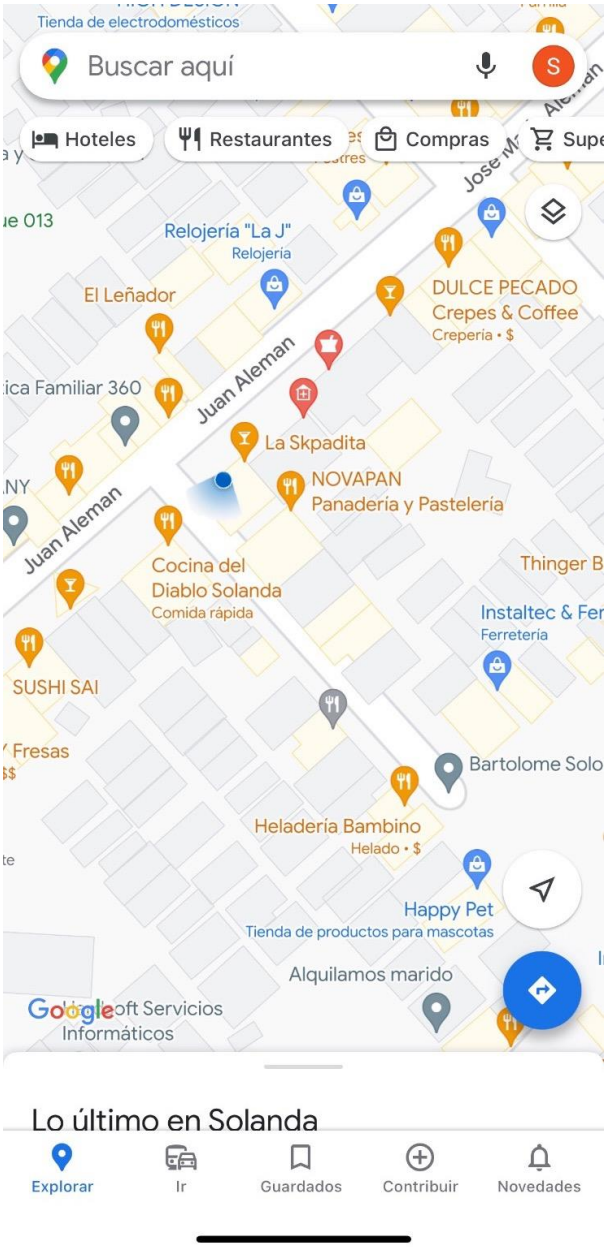


Valenzuela, M. (2023). Figura 15: Prueba Google Maps de la ubicación del Módulo GPS.

5.4 Margen de error

Para realizar esta prueba se mantuvo un lugar fijo dentro de una casa, que estaba ubicada en la Calle Bartolomé Solón al Sur de Quito.

Tendiendo así la ubicación de Google Maps:



Valenzuela, M. (2023). Figura 16: Prueba Google Maps para el cálculo de error.

Y nuestro módulo GPS nos reflejaba la siguiente ubicación:

Valenzuela, M. (2023). Figura 17: Distancia de ubicación real vs ubicación de GPS Arduino.

A continuación, se mostrarán tablas de comparación de pruebas en varios puntos de la parte del sur de Quito con el GPS Arduino con la ubicación de Google Maps, así podemos reconocer cual es la diferencia de distancia que se tiene del módulo GPS desarrollado con la ubicación de otras aplicaciones.

5.3 Tabla 12: Posiciones del Aplicativo desarrollado contra el GPS de Google Maps en Grados Decimales

Nro	Ubicación	Módulo GPS Desarrollado	GPS Google Maps

		Coordenadas en grados decimales		Coordenada en grados decimales	
		Latitud (DD)	Longitud (DD)	Latitud (DD)	Longitud (DD)
1	El Recreo	-0.252054	-78.522087	-0.252003	-78.522017
2	Redondel Atahualpa	-0.249051	-78.531669	-0.249094	-78.531709
3	Redondel Calzado	-0.253796	-78.531181	-0.249094	-78.531709
4	IESS Quito Sur	-0.259528	-78.524811	-0.249094	-78.531709
5	El Comercio	-0.27254	-78.527579	-0.272572	-78.527646
6	Av. Princesa Toa	-0.282059	-78.51651	-0.272572	-78.527646
7	Camino a Conocoto	-0.282222	-78.491462	-0.272572	-78.527646
8	Parque Conocoto	-0.291965	-78.478256	-0.272572	-78.527646
9	El Cinto	-0.248931	-78.521015	-0.247921	-78.569052
10	Solanda	-0.269789	-78.541069	-0.269767	-78.541087

Valenzuela M. (2023) Tabla 12: Diferencia entre grados decimales de la ubicación del módulo GPS desarrollado vs el GPS de Google Maps.

Nota. Elaboración propia

5.4 Tabla 13: Posiciones del Aplicativo desarrollado contra el GPS de Google Maps en metros

Nro	Ubicación	Distancia en metros			
		Modulo GPS Desarrollado		GPS Google Maps	
		Lat (m)	Long (m)	Lat (m)	Long (m)
1	El Recreo	0.4	0.8	0.7	1.53

2	Redondel Atahualpa	0.7	1.6	0.2	1.3
3	Redondel Calzado	1	0.1	1.4	0.13
4	IESS Quito Sur	0.9	0.15	1.1	0.18
5	El Comercio	0.4	0.89	0.8	1
6	Av. Princesa Toa	0.3	1	0.4	1.05
7	Camino a Conocoto	1.3	1.4	1.1	1.6
8	Parque Conocoto	1.7	1.2	0.8	1.5
9	El Cinto	0.4	0.6	0.6	0.4
10	Solanda	1	0.4	2.5	2

Valenzuela M. (2023) Tabla 13: Diferencia en metros de la ubicación del módulo GPS desarrollado vs el GPS de Google Maps.

Nota. Elaboración propia

Conclusiones

- En conclusión, el desarrollo de un aplicativo para rastrear a una mascota utilizando Python como lenguaje de programación para el backend, HTML para el frontend, y las herramientas de Putty, MQTT y la tecnología GSM, ofrece numerosas ventajas y posibilidades.
- Python es una elección sólida para el backend debido a su versatilidad, legibilidad y amplia comunidad de desarrolladores. Su amplio conjunto de bibliotecas y frameworks facilita el desarrollo de funcionalidades complejas, permitiendo un proceso de desarrollo eficiente y rápido.
- Por otro lado, HTML es un lenguaje estándar para la creación de interfaces de usuario en la web. Su estructura simple y su capacidad de integración con otros lenguajes y tecnologías lo convierten en una opción flexible y accesible para el frontend de la aplicación.
- La utilización de la herramienta Putty es fundamental para la conexión y administración remota del dispositivo Arduino utilizado en el rastreo de la mascota. Con Putty, se puede configurar y monitorear el dispositivo de manera eficiente, lo que facilita el mantenimiento y la solución de problemas.
- Cuando se trata de lograr una comunicación eficiente y en tiempo real entre el cliente y el servidor, no hay mejor opción que el protocolo MQTT, famoso por su ligereza y rapidez. Permite la transferencia rápida y confiable de datos, lo que resulta beneficioso para la actualización de la ubicación de la mascota y la interacción con el usuario.
- Además, la tecnología GSM ofrece una cobertura amplia y confiable en muchas áreas, lo que permite el acceso y seguimiento de la mascota en tiempo real, incluso en lugares remotos. Esta tecnología de comunicación móvil es fundamental para enviar y recibir datos relacionados con la ubicación de la mascota.

Recomendaciones

- En cuanto a las recomendaciones, es importante asegurarse de contar con una arquitectura bien diseñada y escalable, que permita el crecimiento y la integración de nuevas funcionalidades en el futuro. Además, se debe prestar atención a la seguridad de la aplicación, implementando medidas de autenticación y cifrado para proteger los datos sensibles.
- También se recomienda realizar pruebas exhaustivas tanto en el backend como en el frontend, para asegurarse de que la aplicación funcione correctamente y proporcione una experiencia de usuario fluida.
- En resumen, el desarrollo de un aplicativo para rastrear a una mascota utilizando Python en el backend, HTML en el frontend, y aprovechando las herramientas de Putty, MQTT y la tecnología GSM, ofrece una solución robusta y eficiente para el seguimiento y cuidado de las mascotas. Es importante considerar los aspectos de diseño, seguridad y pruebas para garantizar un funcionamiento óptimo y una experiencia positiva para los usuarios.

1. BIBLIOGRAFÍA

- 0128, O. M. (2004). Distrito Metropolitano de Quito. Obtenido de Ordenanza metropolitana 128: Ordenanza 128
- A. Delhij, R. v. (2015). La guía de eduScrum: “las reglas del juego”.
- Becker, D. K. (17 de 04 de 2017). Esta es la causa por la que muchas mascotas se pierden. Obtenido de Mascotas mercola: Artículo
- Caicedo, R. A. (31 de Enero de 2020). Picorolo. Obtenido de GPS: qué, cómo funciona y sus aplicaciones: Artículo
- Campos, J. M. (01 de 06 de 2015). ¿Qué porcentaje de hogares tienen mascotas? Obtenido de La Vanguardia: Artículo
- Millahual, C. P. (2020). Descubriendo Arduino. Buenos Aires.
- Purina. (2018). La importancia del localizador GPS de perros en viajes. Obtenido de Purina: Artículo
- Quito, D. M. (2020). Ordenanza Metropolitana Nº 019-2020. Obtenido de Distrito Metropolitano de Quito: Ordenanza 019-2020
- Rueda, F. R. (2015). Obtenido de Una alternativa tecnológica para perros y gatos: Tesis
- Sabater, V. (2015). Mi perro no es una mascota, es mi familia. Obtenido de La mente es maravillosa: Artículo
- Telegrafo, E. (30 de 08 de 2015). 3 de cada 5 familias tienen una mascota. Obtenido de El Telegrafo: Artículo
- Google Developers. (2021). Kotlin para programadores de Java. Recuperado de Página web
- Apple Inc. (s.f.). The Swift Programming Language. Recuperado el 28 de febrero de 2023, de Página web

- Objective-C. (s.f.). Recuperado el 28 de febrero de 2023, de Página web
- Microsoft. (2022). C# programming guide. Recuperado el 28 de febrero de 2023, de Página web
- Xamarin. (s.f.). Why C# is the best language for mobile app development. Recuperado el 28 de febrero de 2023, de Página web}
- A. A. Hammam, M. M. Soliman and A. E. Hassanein, "DeepPet: A Pet Animal Tracking System in Internet of Things using Deep Neural Networks," 2018 13th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 2018, pp. 38-43, doi: 10.1109/ICCES.2018.8639260.