Collar GPS: paseos a demanda

Guido D'Amore

***Resumen*—Pasear a nuestros perros es esencial para su salud, y muchas veces no tenemos el tiempo disponible para cumplir con dicha obligación. En este trabajo final de grado, se diseña una aplicación, utilizando Arduino IDE, que permite que un usuario solicite y trackee en vivo el paseo de su perro, pagando por ello; y a un usuario paseador, aceptar dicho paseo para cobrar por el servicio. Adicionalmente, para el trackeo en vivo, se diseñó un collar implementando las placas NODEMCU-8266 y el módulo de GPS Ublox Neo6, que proporciona la longitud y latitud en tiempo real (cada 3 segundos) para que la ubicación pueda ser representada gráficamente en un mapa en la aplicación IOT: BLYNK. Posteriormente, el recorrido del paseo es reconstruido utilizando la API 'ROAD INSPECTOR' de Google y figurado en Google Maps.**

I. Palabras Clave

Pasear, perros, paseadores, aplicación, app, GPS, localización, geolocalización

II. Nomenclatura

- API: Application Programming Interface
- GPS: Global Positioning System
- APP: Aplicación / Application
- GCBA: Gobierno de la Ciudad de Buenos Aires
- P2P: Peer to Peer
- IDE: Integrated development environment
- Usuario: Persona que contrata el servicio
- Usuario Paseador: Persona que provee el servicio a cambio de un pago.

III. Introducción

Las mascotas han sido, desde hace mucho tiempo, una compañía de privilegio para los seres humanos. Los paseos diarios y una alimentación variada y balanceada, son pilares esenciales en la salud de nuestros compañeros caninos. Cuando son cachorros, los paseos sirven para que empiecen a estimular sus sentidos del olfato, oído , y vista, con algo externo a lo que ven constantemente dentro de su hogar. Los cachorros, para permanecer sanos, deberían pasear cada 90 minutos Mientras que los perros adultos deberían salir por lo menos dos veces al día, con paseos no inferiores a los 45 minutos. [1].

Hoy en día, un porcentaje de las personas no tiene el tiempo físico disponible para cumplir con la responsabilidad de pasear a sus mascotas. Algunas otras, tienen imposibilidad de pasearlos por tanto tiempo como deberían. Pensemos, por ejemplo, en un abuelo que adoptó un perro cachorro: generalmente hablando, es muy probable que no tenga la energía para poder pasearlo cada 90 minutos como indican las buenas prácticas de cuidado.

Otro problema detectado, es que hay muchos individuos que cobran por dar paseos a los perros de terceros, faltan a la confianza de los dueños dejando a sus mascotas atadas en algún espacio público, sin que estos puedan gastar la energía acumulada, y no hay manera de poder darse cuenta de dicha actitud.
La idea de este trabajo es analizar y desarrollar un prototipo de aplicación que permita pasear perros a cambio de dinero, y que éste sea trackeado en vivo.

El usuario cargará en su UI la distancia y el tiempo aproximado que desea que su/sus perro pasee/n y la app calculará un precio aproximado a cobrar por el servicio, que estará directa y estrictamente relacionado tanto con las variables recientemente mencionadas como con la disponibilidad de paseadores al momento del pedido. El paseo se publicará en la UI de los paseadores, y aquel que se encuentre cerca y quiera tomar el trabajo, puede hacerlo siempre y cuando tenga una cantidad igual o inferior que 8 perros (contando al que acepta y debe pasar a buscar). El collar tendrá un sensor de contacto en la unión de los enganches para poder asegurarnos de que no sea quitado nunca durante el paseo. Ante la falta de contacto entre estos sensores, se disparará una alarma al usuario dueño del perro para que esté al tanto de la situación. Para este desarrollo, usaremos tanto hardware como software. Se diseñará un collar con un módulo de GPS que podrá ser utilizado para geolocalizar a los perros. El mismo será desarrollado utilizando una placa Nodemcu ESP8266 y un módulo GPS Ublox NEO 6M. Al ser ambos de un tamaño relativamente pequeño, pueden ser incluidos en la parte superior del collar (o pretal) del perro.

El IDE de arduino será el que usaremos para desarrollar el código que será subido a la placa NODEMCU y será la encargada de recibir, procesar y representar las señales GPS del módulo GPS en un mapa utilizando la aplicación de Blynk IOT.

IV. Caso de estudio

Una persona que se encuentra en su domicilio, localizado dentro de la capital federal, posee un impedimento para poder pasear a su perro. Esta persona es consciente de que el no hacerlo puede afectar la salud de su mascota y decide buscarla solución en internet, y encuentra una aplicación que permite darle este beneficio a su can, sin tener que salir físicamente de su domicilio. Por otro lado, hay una persona que está buscando generar ingresos principales o extras a su economía individual, que vive o está dispuesta a viajar hacia capital federal para trabajar, y es una persona que le gusta caminar y sabe tratar con las mascotas de las demás personas. Se decide implementar una solución que da una respuesta a ambos requerimientos: una aplicación que conecta a quien quiere

pasear a su perro, pero se encuentra imposibilitado (cualquiera sea la razón) con a una persona amante de los perros y de las caminatas diarias, que va a cobrar por ese servicio.

El siguiente trabajo busca proveer los detalles de la solución (creación de una app) propuesta.

V. Marco Teórico

V-A. *Global Positioning System (GPS)*

El GPS es un sistema de posicionamiento y navegación por satélite. Su constelación básica consta de 24 satélites, todos con relojes muy precisos, que transmiten continuamente ondas de radio a un receptor que puede utilizarlas para determinar su propia ubicación. Los satélites se mueven constantemente con respecto a la tierra, pero sus posiciones pueden ser obtenidas y utilizadas con mucha precisión por el receptor. Un dato muy especial en este traspaso de información es: la hora. Un receptor GPS suele tener un reloj incorporado que es mucho menos preciso que el de un satélite, pero gracias a la física y a la matemática, efectuando algunos cálculos desde el receptor se puede obtener y utilizar el tiempo que tarda en llegar la señal de radio de satélite para estimar aproximadamente la distancia entre el satélite y el receptor, y asi dar una ubicación aproximada del receptor.

Para esto, se busca involucrar por lo menos tres satélites, y utilizando sus intersecciones y el radio de los círculos generados por los cálculos efectuados en relación a las distancias, podemos obtener la ubicación aproximada del receptor [2] (ver figura 1)

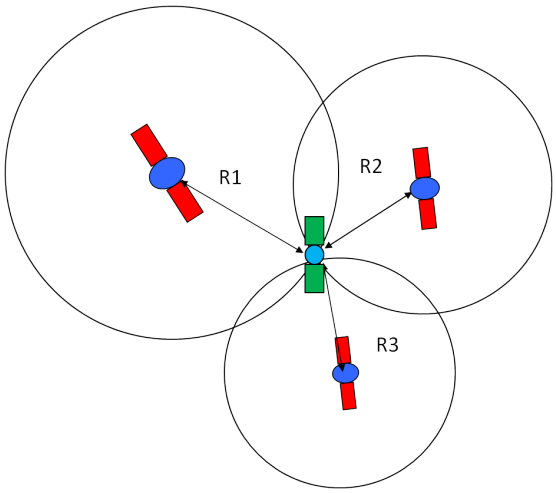


Figura 1. Funcionamiento de GPS, según NASA [3]

GPS, ofrece dos servicios distintos actualmente: El Servicio de Posicionamiento Estándar (SPS), que utiliza el código de adquisición gruesa (C/A) en la frecuencia L1, y el Servicio de Posicionamiento Preciso (PPS), que utiliza el código P(Y) en las frecuencias L1 y L2. El acceso al PPS está restringido al gobierno Estadounidense. El SPS es el que usamos frecuentemente y está disponible para todos los usuarios de forma continua y a nivel mundial, libre de cualquier cargo directo para el usuario. [3]

Este trabajo utilizará el servicio SPS.

V-B. *API (Application Programming Interface)*

Las API'S son usadas, generalmente, para exponer un set de datos y funciones para facilitar un contrato en las interacciones entre programas o compañías, delimitando el alcance y la información que puede ser consumida, pudiendo poner cierta información disponible de manera pública o dando acceso a ciertos usuarios para poder consultar información de forma autorizada bajo estándares de seguridad. [4]

V-C. *Aplicación (App)*

Es un producto que construyen los programadores y al que después le dan mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en distintas plataformas, contenido que se presenta a medida de que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos. La ingeniería de software está formada por un proceso, un conjunto de métodos (prácticas) y un arreglo de herramientas que permite a los profesionales elaborar software de cómputo de alta calidad. [5]

V-D. *MongoDB*

MongoDB es una base de datos de propósito general, no relacional. Es decir que no usa filas y columnas como las bases de datos relacionales. Es muy potente, flexible y escalable. Combina la capacidad de escalar con características tales como índices secundarios, consultas de rango, orden, agregaciones e índices geoespaciales. Esta base de datos, que está orientada a documentos, sustituye el concepto de "fila"por un modelo más flexible: el "documento". Al permitir la inclusión de documentos y matrices, el enfoque orientado a documentos permite representar relaciones jerárquicas complejas con un solo registro. Tampoco hay esquemas predefinidos: la clave y los valores de un documento no son de tipos o tamaños fijos. Sin un esquema fijo, resulta más fácil añadir o eliminar campos según las necesidades. En general, esto hace que el desarrollo sea más rápido, ya que los desarrolladores pueden iterar rápidamente. También es más fácil experimentar. Los desarrolladores pueden probar docenas de modelos para los datos y luego elegir el mejor para continuar. [6]

V-E. *Legal*

En un artículo publicado el 19 de Julio de 2017, se indican varios puntos a tener en cuenta que son de forma obligatoria para los paseadores [7] :

- No exceder el límite de 8 perros al mismo tiempo.
- Siempre debe llevarlos a todos con collar (o pretal) y correa. Y si el perro es potencialmente peligroso, debe colocarle el bozal
- No atar a los animales a árboles, monumentos, semáforos, postes o cualquier tipo de mobiliario urbano
- Llevar una escobilla y una bolsa de residuos para recoger la materia fecal.
- Los animales solamente podrán permanecer en el espacio público sin el empleo de correa cuando se encuentren en

los lugares exclusivamente autorizados para el uso de animales domésticos (por ejemplo, caniles)

No se observan impedimentos legales que eviten que un perro lleve un pretal/collar con localización de GPS incorporada y que se ofrezca un servicio de pago para proveer paseos diarios.

VI. Desarrollo

En esta sección, veremos en detalle como llegamos al desarrollo de la aplicación: explicaremos la lista de requerimientos, diagrama de clases, su base de datos, la configuración de las variables y veremos una vista aproximada de la UI, junto con la implementación tecnológica referida al funcionamiento del GPS que será mostrado en el aplicativo.

VI-A. Lista de Requerimientos funcionales

A continuación se detalla la lista de requerimientos funcionales, y no funcionales, necesarios para llevar la aplicación a cabo:

- La plataforma debe poder registrar usuarios de forma exitosa, incluyendo: Nombre, Apellido, E-mail, Método de pago, y cantidad, nombre y tamaño de los perros.
- La plataforma debe poder registrar usuarios paseadores de forma exitosa, incluyendo Nombre, Apellido, Email, Foto DNI frente y dorso, Certificado habilitante para paseador, método de cobro.
- La plataforma determinará, según la demanda del momento, cuál es el precio para el paseo según el kilometraje indicado, siendo 3km el mínimo.
- La plataforma permitirá al usuario solicitar un paseo para su/s perro/s de forma live, mostrando tiempo de paseo, distancia aproximada, y precio a abonar.
- La plataforma permitirá al usuario Paseador visualizar todos aquellos usuarios que deseen contratar el servicio, incluyendo el kilometraje deseado y la remuneración recibida por el paseo.
- La plataforma permitirá al usuario paseador elegir que paseos acepta, sin obligación de tomar paseos.
- La plataforma permitirá efectuar cobros con tarjeta de crédito/débito.
- La plataforma mostrará el acumulado total de todos los trabajos de un paseador durante una semana. Al octavo día se hará el depósito semanal en la cuenta indicada.
- La solución permitirá al usuario tener un trackeo en vivo del paseo de su mascota, mediante la utilización del módulo GPS incorporado el collar.
- La solución deberá enviar una señal de alerta en caso de que el collar sea abierto.

VI-B. Lista de Requerimientos no funcionales

- La interfaz de usuario debe poseer diseño responsive y user friendly.
- Se deben proporcionar mensaje de error que tengan una combinación de mensaje de error claro para el usuario y un código de error para proporcionar a soporte.
- La velocidad de respuesta a los comandos debe ser rápida para evitar frustración en la espera
- La contraseña para la creación de cuentas debe contener por lo menos un caracter especial, una minúscula, una mayúscula, un número y una extensión mínima de 8 caracteres.
- Después de 3 intentos fallidos en el inicio de cuenta, se bloqueará la cuenta y se enviará al mail registrado un link para la reactivación
- El collar/pretal deberá tener incorporado un sensor de apertura en el broche de apertura.
- La aplicación deberá funcionar tanto en Android como en Ios
- La aplicación debe estar disponible desde las 7hs de la mañana (inicio del primer paseo) hasta las 22hs (inicio del último paseo). La hora de finalización depende de la distancia del último viaje aceptado.
- Los datos entre las aplicaciones, la API y la base de datos, será encriptado punta a punta para evitar interceptaciones de información en las comunicaciones.

VI-C. Arquitectura del sistema

Expondremos y recibiremos todos los datos de los paseos a través de una API. La estructura pensada para poder cumplir con los requerimientos de la aplicación es la siguiente :

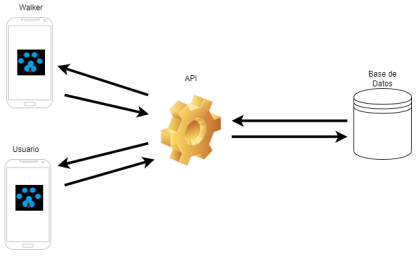


Figura 2. Estructura aplicación

VI-D. UML - Diagrama de clases

El diagrama de clases se efectuó pensando en que cada cuenta puede ser tanto de una persona que desea contratar un servicio, como de una persona que quiere proveerlo. El modelo se desarrolla en base a la clase ”paseo”, que será la

principal, y contará con los datos de ambos usuarios, el/los perros que paseen, el monto a pagar, el método de pago, el progreso del paseo y la ubicaciòn actual.

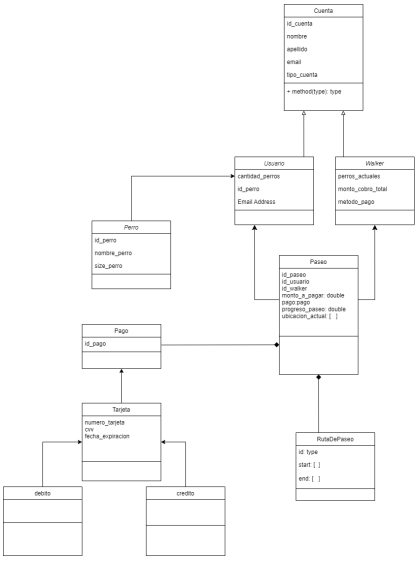


Figura 3. Diagrama UML propuesto

VI-E. Hardware GPS

Se unió la placa Nodemcu Esp8266 con el módulo de GPS Ublox Neo6. Para esto, se conectó la antena utilizando un pigtail con un conector mini PCI para radio frecuencia, y posteriormente se soldaron y conectaron los 4 pines del módulo mediante un cable dupont hembra-hembra de la siguiente manera:

Módulo GPS	NODEMCU
VCC	3.3V
GND	GND
RX	D1
TX	D2

VI-F. Software GPS

Para el desarrollo del proyecto, utilizaremos el IDE ARDUINO que viene específicamente diseñado para desarrollar proyectos IOT con este tipo de placas. El módulo GPS nos va a enviar señales que serán recibidas por la ESP8266, y graficadas posteriormente en un mapa dentro de Blynk: IOT.

Iniciamos esta sección incluyendo la parte del código que nos permite efectivamente recibir y guardar las señales necesarias para poder localizar la ubicación del collar. Los valores con los que vamos a trabajar principalmente, son: latitud, longitud y orientación.

Utilizaremos la librería ”TinyGPS++”, desarrollada para trabajar con módulos GPS, para guardarlos los datos nombrados anteriormente en variables y trabajar posteriormente con dichos valores.

```
void displayInfo()
{
  if (gps.location.isValid() )
  {
    float latitude = (gps.location.lat());     //Guardamos la latitud y la longitud
    float longitude = (gps.location.lng());

    Serial.print("LAT:  ");
    Serial.println(latitude, 6);  // Imprimimos los valores de Lat y Long
    Serial.print("LONG: ");
    Serial.println(longitude, 6);
    Blynk.virtualWrite(V1, String(latitude, 6));
    Blynk.virtualWrite(V2, String(longitude, 6));
    myMap.location(move_index, latitude, longitude, "GPS_Location");
    spd = gps.speed.kmph();               //tomamos la velocidad del módulo gps a la que está moviendose
       Blynk.virtualWrite(V3, spd);

       sats = gps.satellites.value();    //tomamos números de satélites que se utilizan
       Blynk.virtualWrite(V4, sats);

       bearing = TinyGPSPlus::cardinal(gps.course.value()); // tomamos la orientación de la dirección del modulo gps
       Blynk.virtualWrite(V5, bearing);
  }

 Serial.println();
}
```

Figura 4. Código gps - latitud y longitud

Luego de guardar los datos necesarios, pasamos a Blynk: una plataforma de IOT, muy útil para controlar de manera sencilla este tipo de placas de desarrollo.

Empezaremos por configurar ”labeled values”, nativos de Blynk. Estos sirven para representar datos en etiquetas de forma gráfica en la UI. A continuación, se mostrará únicamente la configuración de la etiqueta de longitud, pero la configuración será la misma para todos los valores que necesitamos, indicando distintos puertos virtuales donde estos trabajaran (ver figura 5)

Para configurar un Labeled Value vamos empezar por identificarlo con un nombre, elegir el puerto Virtual con el cual trabaja (en este caso usará el V2), y por último la taza de refresco (Push). Es escencial que todos los configuremos con una taza de refresco de entre 1 y 3 segundos para poder obtener una posición actual actualizada y precisa. Caso contrario, podríamos estar trabajando con datos desactualizados.

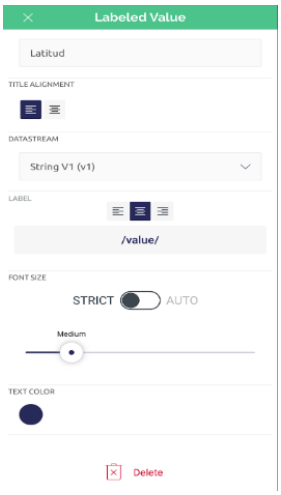


Figura 5. Configuración Blynk Longitud

Se agrega además el mapa nativo de Blynk que permite mostrar en vivo la ubicación del dispositivo. El mismo será configurado en el puerto virtual 0, y recibirá los datos configurados en los ”labeled value”que mostramos anteriormente (ver figura 6):

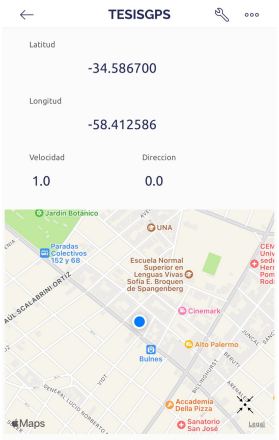


Figura 6. Configuración Blynk Mapa

VI-G. APIS

En esta sección se hablará de dos APIS distintas: La primera es la generada con Javascript para este trabajo, que será core para la conexión entre la base de datos y los dispositivos; y la segunda es la nativa de Google Maps, que servirá para recrear el recorrido efectuado por el paseo una vez que este finalice.

VI-G1. API Custom: La API custom nombrada en el punto anterior, incluye la creación, modificación, obtención y borrado de los paseos. Si bien dentro del código de la API se encuentran incluidas todas las funciones nombradas, a continuación haremos foco en la creación y modificación de los paseos, que son las que en una primera instancia tomarán más importancia.

Para la creación de los paseos, utilizaremos el verbo HTTP POST. Necesitaremos contar los siguientes datos de forma obligatoria en la llamada a la API:

- id_usuario: deberá ser del tipo integer
- id_walker: deberá ser del tipo integer
- monto_a_pagar: deberá ser del tipo Float
- pago: deberá ser del tipo String, solo acepta ”debito.º çredito”
- inicio (*): deberá ser un array con dos valores: latitud y longitud
- ubicacion_actual: deberá ser un array con dos valores: latitud y longitud
- progreso_paseo (**): deberá ser calculado con ambas ubicaciones

(*) Es el punto de übicación_actual”desde donde se solicita el paseo. (**) Son valores otorgados por el módulo de GPS en conjunción con una función de calculo para otorgar el resultado buscado

Para poder medir el progreso actual, tendremos en cuenta tres puntos muy importantes: Latitud, longitud y dirección del módulo de GPS. Recibiremos cada 5 segundos la ubicación estimada (Latitud y Longitud) del módulo de GPS del collar del perro. Esta ubicación será importante para medir las distancias recorridas por el paseador. Para calcular las distancias, haremos lo siguiente: Siempre y cuando la orientación del GPS no cambie, estaremos recorriendo distancia en linea recta. Iremos sumando en una variable dicha distancia recorrida. Considerando que los paseos nunca son 100 % en linea recta, aquí es donde entra en juego la orientación del módulo GPS: Cada vez que tengamos un cambio de orientación, en conjunción con un recorrido de más de 30 metros, cerraremos la cuenta en linea recta y abriremos otra. La sumatoria de todas las cuentas en linea recta al llegar al kilometraje solicitado por el usuario, pondrá un fin al paseo del perro.

Adicionalmente a la creación, contaremos con dos transacciones que usaran PUT , y que servirán para actualizar el paseo según corresponda:

VI-G2. API de Google Maps: Para poder utilizar la API de Google maps, es necesario crear un proyecto. El que será utilizado para este trabajo lleva el nombre de ”Tesis - Guido Damorez cuenta con una API KEY (AIzaSyDujk3C5QI7wGWwC7pwSp9K_HFZi5Oaob0) que sirve para identificarse y estar habilitado para utilizar cualquier de las API’S ofrecidas por Google. La ROAD INSPECTOR API”fue la elegida para el desarrollo del presente trabajo.

Dicha funcionalidad opera con varios puntos GPS (longitud y latitud) distribuidos a lo largo de un mapa según fueron recibidos temporalmente. Tomando varios puntos sucesivamente, podemos ver el camino tomado por el módulo de GPS y reconstruirlo en la UI de la aplicación.

El ejemplo que colocamos a continuación, es el trackeo de un camino entre las coordenadas -34.5825258, -58.4200761 y -34.578329, -58.426272 , que se encuentran en el barrio de palermo, con el siguiente resultado:



Figura 7. coordenadas de api road

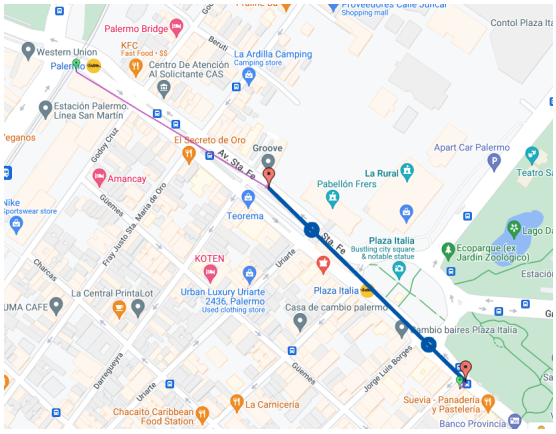


Figura 8. coordenadas del mapa marcadas con api road

VI-H. Vistas de la aplicación

Las siguientes imágenes (ver figuras 9 y 10) son una muestra parcial del total de vistas que contiene la UI de la aplicación y sirven solo a modo de representación general.

Iniciamos con la figura 9 donde encontramos las páginas de login. Aquí tenemos la posibilidad de iniciar sesión como Usuario, iniciar sesión como Walker o crear una nueva cuenta en caso de no tener una. A continuación se muestran los resultados de las páginas de inicio de sesión, tanto de una opción como de la otra, mostrando un diseño distinto para cada una: El diseño de los usuarios Walker, se destaca por tener la descripción ”4 Paws Walker.[en] la parte superior, mientras que la de usuario tiene un diseño que incluye el nombre de la aplicación y el lema ”paseos para tu mascota”.

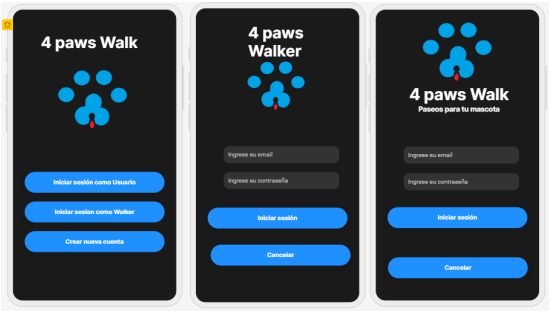


Figura 9. Vistas de la aplicación

Enfocándonos en la figura número 10, podemos ver el diseño de la UI del usuario: en la parte superior, podemos visualizar si es que sus perros están actualmente en un paseo y ver el porcentaje de avance del mismo. Se pueden configurar los medios de pago que deseemos (Tarjetas de crédito y débito), y agregar tantos 4 paws (perros) como queramos a la app. En la segunda imagen podemos ver como se configura el paseo: indicando que perro/s se incluyen en el paseo y los Kilómetros a pasear, junto con el precio final y un botón exclusivamente dedicado a solicitar el servicio. En la última vista, vemos que el Walker Çalvin Klein”pasará a buscar al perro en 8 minutos, lo paseará durante 45 minutos para completar 3km y el costo del paseo será de $2000.00.- ars

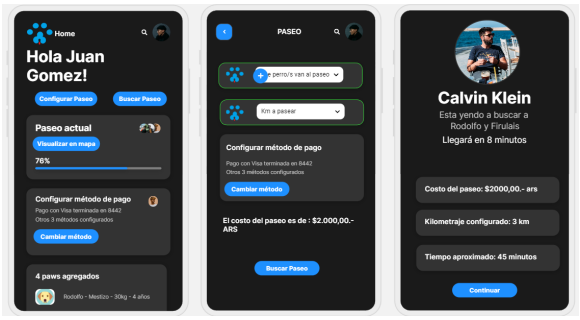


Figura 10. Vistas de la aplicación

VII. CONCLUSIÓN Y TRABAJOS FUTUROS

La placa Nodemcu y el módulo de GPS son realmente útiles para obtener la información necesaria para efectuar la aplicación tal como se pensó. Luego de soldarlos, este módulo de GPS envía las señales de latitud, longitud y orientación a la placa ESP8266, que se conecta por WIFI y transmite los datos a BLYNK, para que este haga la representación gráfica en un mapa y se pueda seguir la ubicación del dispositivo (perro) de forma live en un mapa.

Hacemos uso de La librería TinyGps++, que es nativa de Arduino IDE y muy potente, para tomar y guardar los datos provistos por ambas placas y poder utilizarlos luego. Está librería está diseñada especialmente para tratar este tipo de proyectos donde necesitamos identificar ciertos datos que son transmitidos de un módulo GPS hacia una placa de desarrollo de IOT.

La configuración de la NODEMCU ESP8266 no es para nada sencilla, y requiere de bastante lectura y pruebas. Hay que tener mucho cuidado de que dos de sus pines no estén en contacto, porque eso provoca cortos en la placa que devienen en errores en los datos recibidos.

El Push configurado dentro de BLYNK no debería ser inferior a los 3 segundos, porque eso provocaría que la información con la que estamos intentando relacionarnos de forma instantánea, no sea una representación aproximadamente fiel de la realidad.

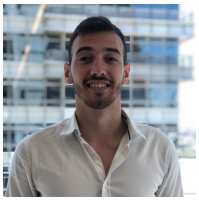
A continuación detallamos una lista de trabajos a futuro:

- No es posible comercializar un producto utilizando tanto las placas de desarrollo implementadas, como el IDE de Arduino. Razón por la cual, antes de sacar este producto y servicio al mercado, es necesario un rediseño basado sobre otras tecnologías.
- Actualmente la placa GPS trabaja con conexión WIFI. Sin dicha conexión, no podremos representar los datos dentro de BLYNK, por lo que es necesario efectuar la implementación de esta solución con conexión GSM, para dejar de depender de una conexión WIFI, y empezar a basarnos en conexiones inalámbricas móviles, tales como 4G, por ejemplo.

- Debemos embeber el mapa de Google Maps que trabaja recreando el camino efectuado por el perro dentro de la aplicación. Actualmente solo es visible desde la UI de la API de google.

- Es necesario hostear todos los requerimientos dentro de un servidor para poder empezar a proveer el servicio.
- Se deben agregar los sensores de contacto para evitar la apertura del collar/pretal del perro e implementar los métodos que permitan identificar este comportamiento para disparar la alerta cuando corresponda.

VIII. Biografía



Guido D'Amore Nacido en Capital Federal, Buenos Aires - Argentina, el 6 de Junio de 1993. Estudiante de la Licenciatura en Tecnología de la información en la Universidad de Palermo .

Referencias

[1] K. Sundance, "Dog training 101, step by step instructions for raising a happy, well-behaved dog," 2017.
[2] X.-W. CHANG and C. PAIGE, "Rest api design rulebook," 2003.
[3] T. Mai, "N.a.s.a web page," 2017. [Online]. Available: https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html
[4] M. Massé, "Rest api design rulebook," 2012.
[5] R. S. Pressman, "Ingeniería del software - un enfoque práctico," 2010.
[6] K. C. Shannon Bradshaw, Eoin Brazil, "Mongodb - the definitive guide," 2019.
[7] GCBA, "Normativa gcba para paseadores de perros," 2017.