Implementación de un entorno de bajo coste para telemetría en vehículos

---

MEMORIA PRESENTADA POR:

Ferran Esparza Micó

GRADO DE INGENIERÍA INFORMÁTICA

Convocatoria de defensa: *Julio 2020.*

Implementación de un entorno de bajo coste para telemetría en vehículos

Ferran Esparza Micó

Resumen
En este TFG se pretende implementar un entorno de bajo coste que permita obtener mediciones sobre los diferentes parámetros de interés para la caracterización del comportamiento de un vehículo mientras éste se encuentra en funcionamiento. Las medidas realizadas se enviarán a la nube para que el usuario, una vez terminada la sesión de telemetría, pueda consultar los resultados de la prueba.
Los requerimientos del sistema de telemetría son:

- Sistema de sensorización en tiempo real de, al menos, tres parámetros
- Sistema local de almacenamiento de datos, situado en el mismo vehículo a probar
- Envío inalámbrico de los datos desde el sistema de sensorización al sistema de almacenamiento local
- Sistema global de almacenamiento de datos, en la nube, basado en Firebase y Google Cloud Platform
- Envío inalámbrico de los datos desde el sistema de almacenamiento local al sistema de almacenamiento global, una vez finalizada la sesión de telemetría
- Implementación de una aplicación de usuario para consulta de todos los datos almacenados

Dirección
Pau Micó

Contenidos

1 Introducción

1.1 Antecedentes

Cualquier persona ha visto en su vida una carrera ya sea de Fórmula 1, Moto GP, etc. donde los pilotos cuentan con sofisticados sistemas de monitorización para mantener vigiladas las constantes y el rendimiento de sus vehículos durante carreras o entrenamientos. En cambio la gente entusiasta, pero que compite o le gusta practicar este tipo de deportes a un nivel más modesto, no puede contar con sistemas accesibles para medir y analizar luego como se comporta su vehículo.

Como estudiante de Ingeniería Informática en el campus de Alcoy y como amante del mundo motor, me propuse como trabajo de final de grado el tratar de ofrecer una solución viable a este problema al que se pueden enfrentar fanáticos del motor o de los vehículos en general que traten de llevar un paso más allá su afición.

1.2 Objetivos

- Implementar un sistema de medición de métricas para vehículos.
- Implementar un sistema de almacenamiento y visualización de métricas.

1.3 Requerimientos

Los requerimientos a nivel de usuario son:

- Recopilación de métricas del vehículo como; velocidad, fuerzas G, temperatura de las ruedas y/o temperatura del motor.
- Revisión de la telemetría (datos) en tiempo real
- Tratamiento y análisis de los datos a posteriori (plataforma web de analíticas de datos)

Para llegar a resolver los requerimientos generales del desarrollo, a continuación se especifican una serie de requerimientos a nivel técnico:

- Dispositivo de control central. Este dispositivo será el encargado de recoger todos los datos que se vayan recopilando por parte de los distintos sensores y antenas para subirlo a un servidor donde se almacenará dichos datos y podrán ser posteriormente tratados y visualizados para poder extraer información.
- Conexión a la red. Se trata de poder comunicar el dispositivo de control central con el servidor donde se almacenan los datos. Para ello es posible emplear WiFi, redes móviles como GSM, 4G, etc. Esta conexión será importante a la hora de iniciar sesión en el perfil del usuario y a la hora de enviar los datos que se van recopilando al servidor.
- Placa de desarrollo y sensores. Por ejemplo: Arduino + batería + sensor de temperatura láser + bluetooth. Esta puede ser la misma que integra el dispositivo de control central u otra que complemente a dicho dispositivo para poder extraer datos desde diferentes partes del vehículo o aumentar la posibilidad de sensores de los que se dispone.

- Nube con posibilidad de almacenar datos en una base de datos y posteriormente poder procesar y mostrar información (Por ejemplo: Google Cloud Platform / AWS / Microsoft Azure, etc.). No es estrictamente necesaria una solución que abarque todos los aspectos, pero sí muy recomendable ya que puede facilitar el desarrollo y la integración de todos los componentes.

2 Anteproyecto

2.1 Estado del arte

En este apartado se va a analizar algunas de las aplicaciones que actualmente se encuentran en el mercado, es decir, disponible para cualquier usuario de un smartphone Android y que sus características principales sean bastante parecidas a las del proyecto que se pretende desarrollar.

No todas las aplicaciones que hay tienen que hacer exactamente lo mismo que el proyecto que se pretende desarrollar y esto puede ser una ventaja a la hora de sacar adelante un proyecto único y diferenciador.

2.1.1 Track Addict

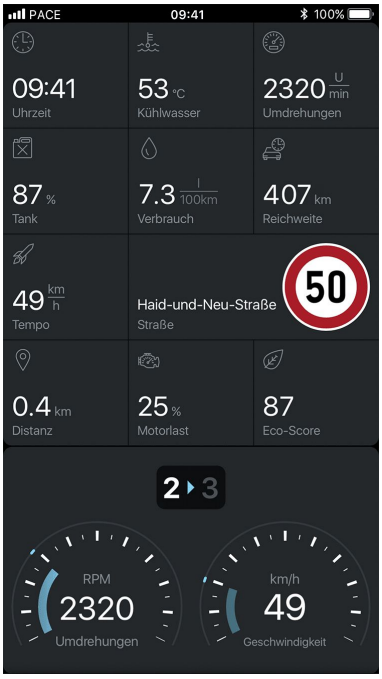
Se trata de una aplicación (link) para smartphone que se encarga de medir ciertas métricas según el tipo de actividad que se realice; carrera en circuito, cuarto de milla, salida libre,etc., graba en video mediante la cámara del dispositivo y calcula las vueltas en circuito, los tiempos a los que se realiza y mide la velocidad, la aceleración y las fuerzas G que actúan sobre el dispositivo. También permite emitir en directo como parte de la funcionalidad premium.
Todas las métricas y el video se almacenan para generar estadísticas. Dichas estadísticas podrán ser consultadas empleando la propia app, es decir, de forma local.

2.1.2 Torque Pro

Se trata de una aplicación para dispositivos Android (link) que se encarga de leer la información que ofrece el propio coche mediante el puerto OBD del mismo y un dispositivo OBD (como este) que transmita dicha información mediante la tecnología Bluetooth. También permite realizar grabaciones mediante la cámara además de aportar a la grabación la información obtenida del coche.

Es un sistema muy interesante para coches o vehículos que cuenten con un puerto OBD ya que se trata de un sistema barato y asequible para cualquiera, pero que su virtud también se convierte en su principal defecto. En el caso de requerir un sistema similar por ejemplo con una motocicleta no sería prácticamente posible. Si bien hay motocicletas que incluyen el estándar OBD, no es una práctica muy extendida. Tampoco sería posible su uso en patinetes, patinetes eléctricos, karts, entre otros.

2.1.3 Pace Car



Similar a Torque Pro, este sistema ofrece una solución integral, es decir incluye en su precio tanto la plataforma, la aplicación y el dispositivo OBD que se conecta al coche. También incluye mapas para trazar las rutas que se realizan. Permite realizar diagnosis del vehículo para poder saber las temperaturas de los aceites o los consumos que se están realizando.

El uso del sistema en sí es gratuito, pero obligan a comprar el dispositivo OBD de la propia compañía lo cual dispara el precio hasta los 119€ en Alemania.

También tiene las ventajas y problemas de ser exclusivo para vehículos con el estándar OBD quedando fuera todos aquellos vehículos más simples o que utilicen tecnologías alternativas al OBD.

2.1.4 GPS Race Timer

Se trata de una aplicación (link) destinada a medir y guardar el tiempo y la velocidad de ciertas carreras como los cuartos de milla. Se trata de una aplicación que funciona simplemente con el smartphone, es decir, no requiere dispositivos de terceros ni software adicional. En esta misma aplicación se pueden visualizar los resultados de las carreras.

Se trata de una aplicación que puede ser utilizada en diferentes vehículos independientemente de las tecnologías que incluya este. Por otra parte, los datos que recoge son escasos y solo pueden ser empleados en la propia aplicación.

2.1.5 Otras aplicaciones similares

- Motolog – combustible, gastos, traza tu ruta GPS (link)
- RaceChrono (link)

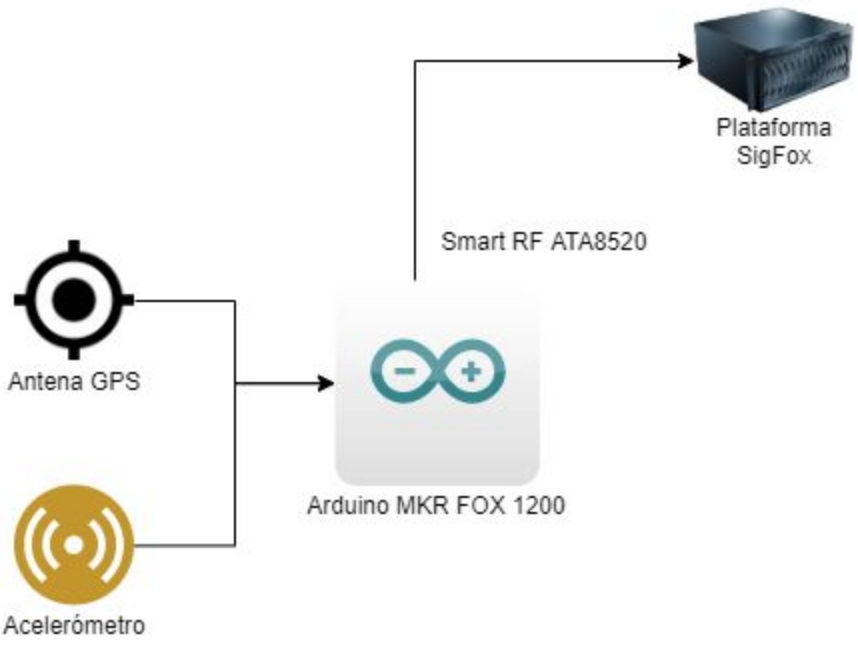
2.2 Estudio de propuestas

2.2.1 Propuesta 1

En esta propuesta se propone el uso de la tecnología SigFox junto a Arduino para el desarrollo de un entorno de telemetría para vehículos.

En primer lugar, el hardware necesario para desarrollar un sistema como el que se requiere:

- Arduino MKR FOX 1200: Este sería el corazón del dispositivo y desde donde se coordinarán los distintos sensores que se le conecten. La propia placa de desarrollo incluye un módulo de comunicación Sigfox vía radio a 868 MHz, concretamente un Smart RF ATA8520.
- Antena GPS (link): Esta sería la encargada de ubicar el vehículo para poder disponer de su velocidad y posición.
- Acelerometro: GY-521 6DOF MPU-6050: Se trata de un módulo acelerómetro que permitirá conocer las fuerzas en los tres ejes que se aplican al vehículo.
- Sensor de temperatura láser: GY-906 MLX90614: Se trata de un pequeño sensor de temperatura que se puede emplear a distancia para conocer la temperatura de lo que apunte tanto en modo ambiente, es decir sin especificar un objeto concreto como en modo fijo para conocer concretamente la temperatura que tiene un objeto al que está apuntando. Es interesante para conocer la temperatura de, por ejemplo las ruedas o los frenos del vehículo.



Boceto de como sería de forma aproximada la conexión

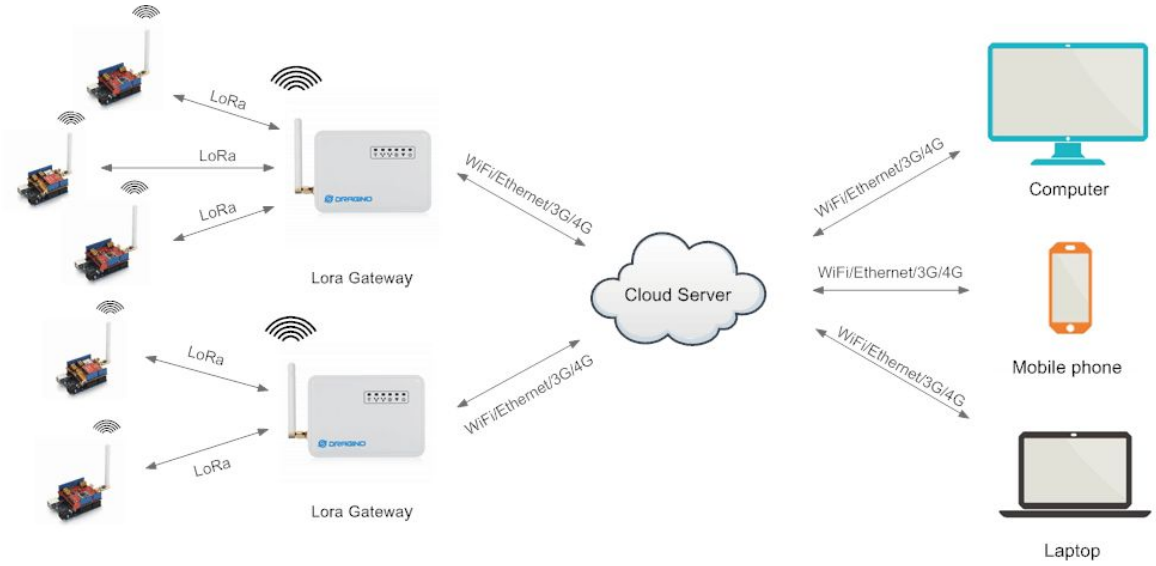
PRODUCTO	PRECIO
ARDUINO MKR FOX 1200	35€
AZDelivery Antena GPS	7,99€
GY-521 6DOF MPU-6050 Módulo Acelerómetro y Giro de 3 Ejes	2,19€
Sensor de temperatura: GY-906 MLX90614ESF	7,31€
TOTAL:	**52,49€**

En cuanto a la tecnología SigFox, no está pensada para enviar grandes cantidades de información, sino más bien para dispositivos pequeños que emiten periódicamente información sobre su entorno para ver como evoluciona o si su estado es correcto. Esta tecnología tiene una clara limitación en cuanto al límite de mensajes que permite enviar al día que serían 140 por dispositivo al día.
Esto plantea que la propuesta no sea viable pues se estima que el dispositivo pueda enviar de 2 a 5 mensajes por segundo recogiendo la información que ofrecen todos los sensores mencionados. Conociendo esta estimación, un sistema así sería interesante si se requiriera durante 70 segundos al día o incluso menos. Esto dificultará el uso extendido del proyecto a carreras de cuarto de milla y similares, pero este no es el propósito principal del proyecto.

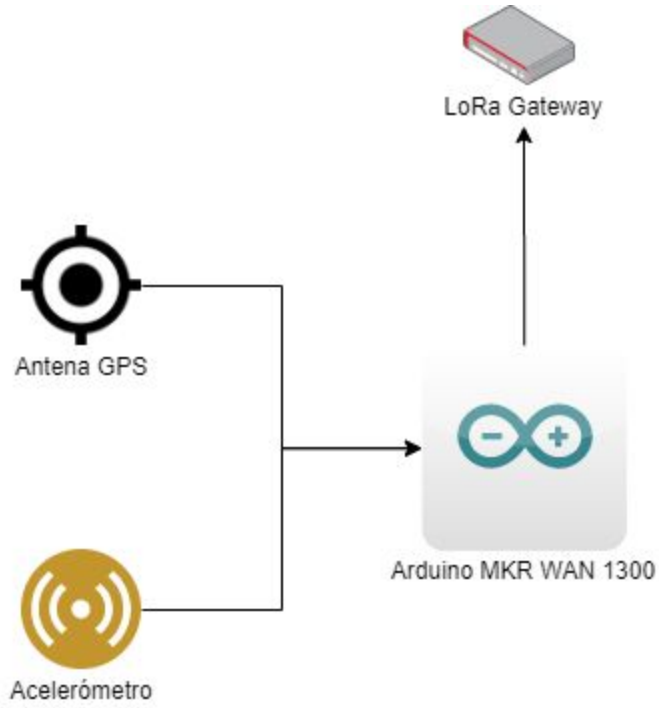
2.2.2 Propuesta 2

En esta propuesta se propone el uso de la tecnología LoRa junto a Arduino para el desarrollo de un entorno de telemetría para vehículos.
Esta propuesta comprende el uso de los distintos dispositivos compatibles con la tecnología de comunicación LoRaWAN y el uso de herramientas para la creación de gráficos con los que interpretar la información recopilada como puede ser Grafana o similares.
Esta propuesta comprende el uso de los distintos dispositivos LoRa, el uso de sensores y componentes compatibles y el uso de las redes LoRaWAN para la comunicación y analítica de datos.

En primer lugar, es interesante conocer cómo funciona la tecnología LoRa pues según esta se basará la comunicación entre los vehículos y el servidor en gran medida. Como se puede ver en la imagen inferior, además de las propias placas de desarrollo compatibles con LoRa es necesario un Gateway cerca para poder conectarse con los servidores. Este gateway ya se conectaría mediante diferentes tipos de conexión según el caso; WiFi, Ethernet, 3G,4G, etc. En caso de no disponer de ningún Gateway cercano por la zona por la que se emplearán los vehículos, sería necesario desplegar un LoRa Gateway propio para poder tener cobertura. Esto podría suponer un coste superior a los 60€ solo para poder desplegar los dispositivos.



Cada vehículo requeriría de ciertos componentes para poder recopilar y enviar datos que pasan a describirse de forma esquemática:



El listado de componentes necesarios sería el siguiente:

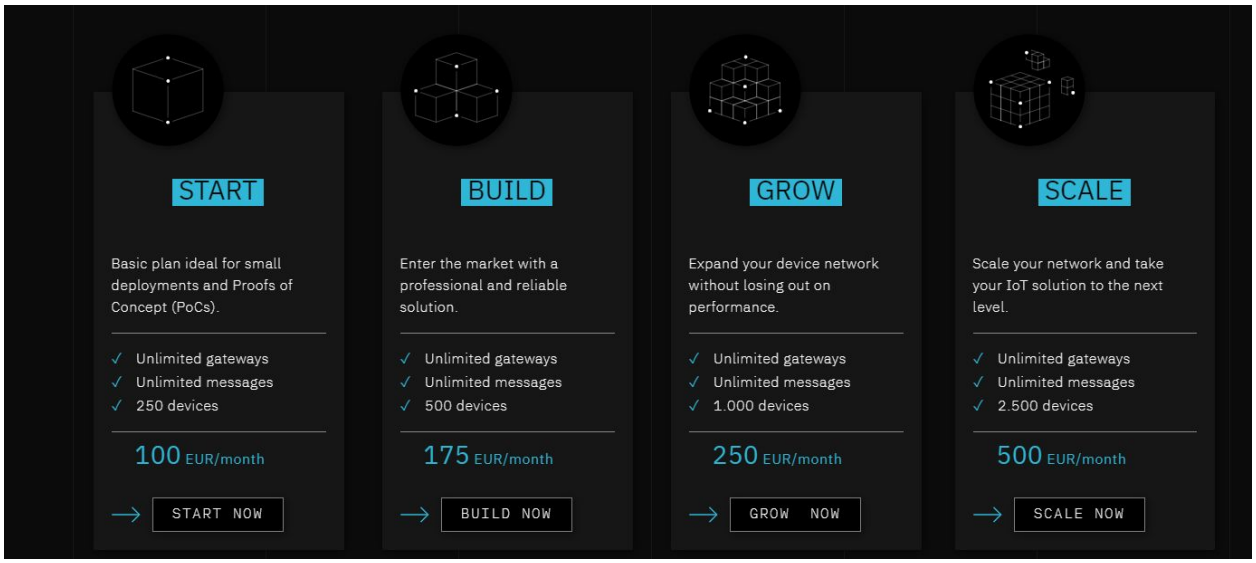
- Arduino MKR WAN 1300: Este sería el corazón del dispositivo y desde donde se coordinarán los distintos sensores que se le conecten. Esta placa de desarrollo incluye

el módulo de radio CMWX1ZZABZ para comunicarse mediante la tecnología LoRaWAN con una frecuencia entre 860 y 930 MHz.

- Antena GPS (link): Esta sería la encargada de ubicar el vehículo para poder disponer de su velocidad y posición.
- Acelerometro: GY-521 6DOF MPU-6050: Se trata de un módulo acelerómetro que permitirá conocer las fuerzas en los tres ejes que se aplican al vehículo.
- Sensor de temperatura láser: GY-906 MLX90614: Se trata de un pequeño sensor de temperatura que se puede emplear a distancia para conocer la temperatura de lo que apunte tanto en modo ambiente, es decir sin especificar un objeto concreto como en modo fijo para conocer concretamente la temperatura que tiene un objeto al que está apuntando. Es interesante para conocer la temperatura de, por ejemplo las ruedas o los frenos del vehículo.
- LoRa Gateway:LG01-P LoRa: Se encargará de conectar los distintos dispositivos LoRa que se desplieguen dentro de su rango a los servidores.

PRODUCTO	**PRECIO**
ARDUINO MKR FOX 1200	35€
AZDelivery Antena GPS	7,99€
GY-521 6DOF MPU-6050 Módulo Acelerómetro y Giro de 3 Ejes	2,19€
Sensor de temperatura: GY-906 MLX90614ESF	7,31€
LoRa Gateway	63,63€
TOTAL:	**52,49€/116,12€**

Además de los gastos en equipamiento físico, sería necesario disponer de una plataforma que comunique los dispositivos LoRa que empleemos con un servidor donde recopilar los datos como puede ser Loriot (link) pero también se requiere de una nube donde almacenar dichos datos como podría ser Azure, Google Cloud Platform, AWS, etc.



Esto supone un coste en mantenimiento de 100€ al mes solo para poder comunicar los dispositivos con el servidor, luego haría falta dicho servidor. Tomando como ejemplo la empresa Neodigit, la cual entre sus servicios incluye Servidores Cloud se podría estimar el coste de dicho servidor en torno a unos 29€ al mes pagando mes a mes.
Esto deja los costes de mantenimiento en unos 129€ al mes aproximadamente sin contar con posibles imprevistos o posibles ampliaciones de servicios.

2.2.3 Propuesta 3

En esta propuesta se propone el uso de dispositivos Arduino junto con un dispositivo Android y los sensores correspondientes. Este sistema se conectará a un servidor de base de datos empleando las tecnologías de red que dispone el dispositivo Android como pueden ser WiFi o el uso de los datos móviles (aunque se prevé que principalmente se empleen los datos móviles ya que será la red de mayor disponibilidad la mayoría de las veces) y se visualizarán las métricas obtenidas mediante la herramienta Grafana.

En primer lugar, el hardware necesario sería el siguiente:

- Dispositivo Android: Puede ser cualquier smartphone con el que ya cuente el usuario. Este debe disponer de bluetooth, acelerómetro, GPS y de forma bastante recomendable tarifa de datos móviles para disponer de conexión en cualquier sitio. Dichos requisitos se cumplen sin problema en cualquier smartphone con menos de 5 años de antigüedad.
- Placa de desarrollo Arduino: Esta placa se encargará de emplear otros sensores que no disponga el dispositivo Android. Por ejemplo un sensor de temperatura para controlar la temperatura de una parte del vehículo.
- Sensor de temperatura láser: GY-906 MLX90614: Se trata de un pequeño sensor de temperatura que se puede emplear a distancia para conocer la temperatura de lo que apunte tanto en modo ambiente, es decir sin especificar un objeto concreto como en modo fijo para conocer concretamente la temperatura que tiene un objeto al que está

apuntando. Es interesante para conocer la temperatura de, por ejemplo las ruedas o los frenos del vehículo.

- Módulo Bluetooth HC-06: Este módulo dotará de conectividad Bluetooth a la placa Arduino para poder comunicar los datos, que vaya recopilando, con el dispositivo Android el cual será el que en última instancia enviará todos los datos al servidor.

PRODUCTO	**PRECIO**
ARDUINO UNO	9,99€
Antena Bluetooth HC-06	7,98€
Sensor de temperatura: GY-906 MLX90614	7,31€
Smartphone Android (Orientativo)	Desde 92€
TOTAL:	**25,28€/117,28€**

El total superior contempla la propuesta de disponer ya de un dispositivo Android, con lo que el coste se reduce considerablemente y en caso de no disponer de un smartphone hay opciones asequibles como la que se ha dado de forma orientativa.

A esto, habría que sumarse el gasto de un servidor donde alojar todos los datos en una base de datos e instalar software que permite obtener gráficos y/o información acerca de dichos datos. Tomando como ejemplo la empresa Neodigit, la cual entre sus servicios incluye Servidores Cloud se podría estimar el coste de dicho servidor en torno a unos 29€ al mes pagando mes a mes.
Esto deja los costes de mantenimiento en unos 29€ al mes aproximadamente sin contar con posibles imprevistos o posibles ampliaciones de servicios.

2.3 Justificación

En este apartado se estudia el impacto económico (esto incluye horas de desarrollo, licencias software, contratación de servidores, etc ...) de las posibles propuestas presentadas en el apartado anterior.

2.3.1 Estimación de recursos

Estimación de recursos Propuesta 1:
En primer lugar, el hardware necesario para desarrollar un sistema como el que se requiere:

- Arduino MKR FOX 1200: Este sería el corazón del dispositivo y desde donde se coordinarán los distintos sensores que se le conecten.
- Antena GPS (link): Esta sería la encargada de ubicar el vehículo para poder disponer de su velocidad y posición.

- Acelerometro: GY-521 6DOF MPU-6050: Se trata de un módulo acelerómetro que permitirá conocer las fuerzas en los tres ejes que se aplican al vehículo.
- Sensor de temperatura láser: GY-906 MLX90614: Se trata de un pequeño sensor de temperatura que se puede emplear a distancia para conocer la temperatura de lo que apunte tanto en modo ambiente, es decir sin especificar un objeto concreto como en modo fijo para conocer concretamente la temperatura que tiene un objeto al que está apuntando. Es interesante para conocer la temperatura de, por ejemplo las ruedas o los frenos del vehículo.
-

Estos componentes serán necesarios para cada vehículo que se pretenda desplegar.

Por otra parte haría falta la contratación de los servicios de la propia plataforma SigFox cuyos precios son muy accesibles. El problema que se plantea en esta propuesta, es que la tecnología SigFox no está pensada para manejar una gran cantidad de mensajes diarios y por tanto limitaría de forma grave la cantidad de información diaria que se podría enviar desde los vehículos.

Estimación de recursos Propuesta 2:
El listado de componentes necesarios sería el siguiente:

- Arduino MKR WAN 1300: Este sería el corazón del dispositivo y desde donde se coordinarán los distintos sensores que se le conecten. Como ventaja frente a la propuesta 1, esta placa ya incluye antena con lo que la compra de componentes podría verse simplificada.
- Antena GPS (link): Esta sería la encargada de ubicar el vehículo para poder disponer de su velocidad y posición.
- Acelerometro: GY-521 6DOF MPU-6050: Se trata de un módulo acelerómetro que permitirá conocer las fuerzas en los tres ejes que se aplican al vehículo.
- Sensor de temperatura láser: GY-906 MLX90614: Se trata de un pequeño sensor de temperatura que se puede emplear a distancia para conocer la temperatura de lo que apunte tanto en modo ambiente, es decir sin especificar un objeto concreto como en modo fijo para conocer concretamente la temperatura que tiene un objeto al que está apuntando. Es interesante para conocer la temperatura de, por ejemplo las ruedas o los frenos del vehículo.
- LoRa Gateway:LG01-P LoRa: Se encargará de conectar los distintos dispositivos LoRa que se desplieguen dentro de su rango a los servidores.

Estos componentes serán necesarios para cada vehículo que se pretenda desplegar.

Sería necesario disponer de una plataforma que comunique los dispositivos LoRa que empleemos con un servidor donde recopilar los datos como puede ser Loriot (link) pero también se requiere de una nube donde almacenar dichos datos como podría ser Azure, Google Cloud Platform, AWS, etc. Esta parte puede encarecer los gastos del proyecto haciendo que otras propuestas sean más atractivas para entornos de bajos recursos.

Tomando como ejemplo los servicios de Loriot para la gestión de las comunicaciones LoRaWAN y la empresa Neodigit para la contratación de un servidor los costes mensuales para mantener toda la infraestructura ascienden a 129€ cada mes.

Estimación de recursos Propuesta 3:
Los recursos físicos para esta propuesta están descritos en el apartado 2.3.2 Impacto económico.
Esta propuesta requeriría además de un servidor donde almacenar los datos en una base de datos y montar el software que permita ver las métricas. Actualmente existen soluciones que permiten tener todo el software necesario montado en un único servicio o una máquina virtual. Esto se detalla mejor en el apartado 2.3.2 .
En este caso no haría falta pagar ningún tipo de plataforma que permita vincular la nube con los dispositivos montados en los vehículos.

2.3.2 Impacto económico

Impacto económico Propuesta 1:

PRODUCTO	**PRECIO**
ARDUINO MKR FOX 1200	35€
AZDelivery Antena GPS	7,99€
GY-521 6DOF MPU-6050 Módulo Acelerómetro y Giro de 3 Ejes	2,19€
Sensor de temperatura: GY-906 MLX90614ESF	7,31€
Plataforma SigFox	19,50€/año
TOTAL:	**71,99€**

La tabla superior incluye el gasto que supondría en hardware para poder desplegar un sistema empleando la Propuesta 1 para cada vehículo.
A este gasto en material habría que añadir el gasto de contratación de los servicios de la plataforma Sigfox los cuales pueden llegar a ser de precios módicos como 16,13€ al año por dispositivo, IVA no incluído(19,5€ IVA incluído).

Impacto económico Propuesta 2:
Las tablas inferiores incluyen el gasto que supondría en hardware para poder desplegar un sistema empleando la Propuesta 2. El total refleja en la primera tabla el caso en que no hiciera falta desplegar un Gateway propio, sino que ya se cuenta con cobertura y el segundo precio

sería el caso peor o que no se cuenta con cobertura y que por tanto haría falta adquirir y desplegar un LoRa Gateway.

A esto habría que añadir un gasto mensual en mantenimiento del servidor o servidores que hacen falta para almacenar la información y emplearla en la creación de gráficos.
Tomando como ejemplo la empresa Neodigit, la cual entre sus servicios incluye Servidores Cloud se podría estimar el coste de dicho servidor en torno a unos 29€ al mes pagando mes a mes.
También sería necesaria una plataforma LoRaWAN que comunique los dispositivos LoRa con el servidor contratado. Un ejemplo podría ser Loriot (link) donde se dispone de dicho servicio a partir de 100€ mensuales sin limitaciones de mensajes, que era el principal problema de la propuesta 1.
Por tanto, el coste de iniciar dicha propuesta podría considerarse de entre 52,49 y 116,12€ para pagar el material necesario para cada vehículo y con unos gastos mensuales de 129€ entre el servidor y la plataforma LoRaWAN.

PRODUCTO	**PRECIO**
ARDUINO MKR FOX 1200	35€
AZDelivery Antena GPS	7,99€
GY-521 6DOF MPU-6050 Módulo Acelerómetro y Giro de 3 Ejes	2,19€
Sensor de temperatura: GY-906 MLX90614ESF	7,31€
Servidor Cloud	29€/mes
Loriot (link)	100€/mes
TOTAL:	**181,49€**

PRODUCTO	**PRECIO**
ARDUINO MKR FOX 1200	35€
AZDelivery Antena GPS	7,99€
GY-521 6DOF MPU-6050 Módulo Acelerómetro y Giro de 3 Ejes	2,19€

Sensor de temperatura: GY-906 MLX90614ESF	7,31€
Servidor Cloud	29€/mes
Loriot (link)	100€/mes
LoRa Gateway	63,63€
TOTAL:	**245,12€**

Impacto económico Propuesta 3:
Las tablas inferiores incluyen el gasto que supondría en hardware para poder desplegar un sistema empleando la Propuesta 3. El total refleja en la primera tabla el caso mejor, es decir, en el caso que ya se cuente con un smartphone propio y la segunda tabla sería en caso de no disponer de un smartphone o de requerir uno dedicado. En tal caso refleja el precio sumando un smartphone accesible.

A esto habría que añadir un gasto mensual en mantenimiento del servidor o servidores que hacen falta para almacenar la información y emplearla en la creación de gráficos.
Tomando como ejemplo la empresa Neodigit, la cual entre sus servicios incluye Servidores Cloud se podría estimar el coste de dicho servidor en torno a unos 29€ al mes pagando mes a mes.
Por tanto, el coste de iniciar dicha propuesta podría considerarse de 25,28€ para pagar el material necesario por vehículo en la mayoría de los casos, teniendo como casos peores un coste de 117,28 para disponer de un smartphone y con unos gastos mensuales de 29€.

PRODUCTO	**PRECIO**
ARDUINO UNO	9,99€
Antena Bluetooth HC-06	7,98€
Sensor de temperatura: GY-906 MLX90614	7,31€
Servidor Cloud	29€/mes
TOTAL:	**54,28€**

PRODUCTO	PRECIO
ARDUINO UNO	9,99€
Antena Bluetooth HC-06	7,98€
Sensor de temperatura: GY-906 MLX90614	7,31€
Servidor Cloud	29€/mes
Smartphone Android (Orientativo)	Desde 92€
TOTAL:	**146,28€**

2.3.3 Propuesta final

Respecto a la Propuesta 1, si bien los costes son asumibles y es capaz de ofrecer un sistema barato y asequible, contar con la gran limitación en cuanto a envío y recepción de mensajes hacía inviable la propuesta para un sistema de este tipo.

En base a las propuestas expuestas en los apartados anteriores, he decidido implementar la propuesta número 3 ya que las tecnologías empleadas ya me son conocidas y, dadas mis circunstancias, cuenta con el menor coste de adquisición.

Al disponer de un smartphone Android he empleado dicho dispositivo para llevar a cabo algunas de las prácticas de laboratorio a lo largo de la universidad que requerían el desarrollo de algún programa.
Por otra parte también hemos desarrollado en prácticas de laboratorio algunos pequeños dispositivos con Arduino junto con un módulo Bluetooth para comunicar la placa de desarrollo con un smartphone.

Todo ello me ha llevado a decidirme por emplear tecnologías a las que ya estaba relativamente familiarizado y que por tanto me permitirían agilizar el proceso de aprender los conocimientos necesarios y la adquisición de documentación para poder avanzar en su desarrollo.

3 Implementación

3.1 Entorno de desarrollo

El entorno de desarrollo tiene varios puntos que se deben abordar:
Por una parte se encuentra el entorno de desarrollo Android para la parte de la aplicación. En este apartado se pretende emplear Android Studio por su extenso uso, su facilidad de uso y amplia documentación. En el punto 3.1.2 se entra más en detalle acerca del IDE Android Studio pero a grandes rasgos, se trata de un entorno que incluye absolutamente todo lo necesario para desarrollar aplicaciones Android. Desde el editor de código, debugger, hasta un emulador completo de Android para hacer tests de las aplicaciones.
Por otra parte, para el dispositivo Arduino se requerirá su propio IDE para programar y cargar en memoria el programa que se encargará de recopilar la temperatura y enviar los datos vía Bluetooth al smartphone con la app principal. En el punto 3.1.2 se entra más en detalle acerca del IDE Arduino pero a grandes rasgos, se trata de un editor de código que comprueba algunos errores de código y facilita subir el código a las placas Arduino y leer mediante el puerto serie la información que devuelven las placas Arduino.

3.1.1 Control de versiones

- **GIT**:
  Es un sistema de control de versiones distribuido, de código abierto, diseñado para gestionar los diversos cambios que se realizan sobre elementos de algún proyecto o configuración del mismo. Además, nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida.
  En este proyecto, se utilizará un sistema de control de versiones online para evitar perder datos en el caso de que el ordenador tuviera problemas y para poder revertir algún cambio que no funcione como era esperado o restaurar errores al desarrollar nuevas funcionalidades.
- **GitHub:**
  Es una plataforma donde alojar código o repositorio empleado para el control de versiones del software facilitando el trabajo en equipo y la gestión de diferentes versiones de un mismo software. Aquí será donde se aloje el código, de forma privada, para evitar pérdidas y problemas a la hora de trabajar tanto si el ordenador de trabajo tuviera problemas o si fuera necesario restaurar errores al ir añadiendo funcionalidad.
  La plataforma funciona bajo el lenguaje Ruby on Rails. Desde junio de 2018 esta plataforma es propiedad de Microsoft.

3.1.2 Entornos de desarrollo integrados

- **Android Studio**
  Este es el entorno referente para el desarrollo y mantenimiento de aplicaciones para dispositivos Android de forma nativa. Permite trabajar con GIT y plataformas de control de versiones como GitHub de forma integrada y agiliza la creación de código.
  Desarrollado por la propia Google como el entorno oficial, se encuentra publicado desde diciembre de 2014. Permite programar empleando XML para la parte visual y tanto Java como Kotlin para la lógica. Kotlin es el lenguaje que está impulsando Google para sustituir a Java dentro de su entorno aunque Java todavía cuenta con muchísima más popularidad pero sobretodo documentación.
  Permite desarrollar aplicaciones para smartphone, tablet, televisores inteligentes y wearables.

- **Visual Studio Code**
  Se trata de un editor de código bastante ágil que dispone de una gran cantidad de plugins que hacen que se pueda adaptar a las necesidades del desarrollador casi al 100%. Entre estos plugins, se pueden encontrar atajos para la sintaxis de distintos lenguajes de programación, herramientas que integran GIT como también compiladores, debuggers, entre otros.
  Se trata de un editor con soporte para la gran mayoría de lenguajes web y de programación como Python, PHP, Ruby, Perl, etc.
  Desarrollado por Microsoft como solución de editor de código open source permite el desarrollo de distintos lenguajes de programación y frameworks como Angular, React JS o Vue JS.

- **Arduino IDE**
  Es el entorno diseñado específicamente para desarrollar los scripts que emplearán los dispositivos arduino y compatibles. Actualmente se puede descargar desde la Microsoft Store facilitando su puesta a punto y la adquisición de los programas base.

3.1.3 Lenguajes de programación

- **Java.** Si bien Kotlin parece que ha llegado para desplazar a Java como lenguaje lógico en Android, Java sigue siendo una opción muy robusta y fiable debido a la documentación que se puede encontrar en gran cantidad de medios para formarse o resolver problemas. Java es un lenguaje de programación orientado a objetos lanzado de la mano de Sun Microsystems, actualmente bajo la propiedad de Oracle. Se desarrolló con la idea de crear un lenguaje que, escrito una sola vez fuera capaz de funcionar en distintos entornos y en cualquier contexto. Todas estas características lo hacen un lenguaje casi idóneo para el desarrollo de aplicaciones Android las cuales pueden llegar a ejecutarse en centenares de distintos dispositivos.

- **XML.** XML o Lenguaje de Marcado Extensible es el lenguaje que se emplea a la hora de programar la parte visual de Android. Empleando este lenguaje o un editor visual basado en él (en Android Studio se contemplan ambas opciones) se puede maquetar y crear toda la parte visual de una aplicación o entorno Android.
- **JavaScript.** Se trata de un lenguaje ampliamente usado en el entorno web y que es capaz de aportar más funcionalidad a las páginas web convencionales apoyando desde el 'backend' todo el proceso web. Actualmente también se ha extendido su uso a nivel de servidor. Es decir se emplea tanto en clientes web como servidores. Este lenguaje será muy útil a la hora de desplegar un servicio web que permita analizar los datos recopilados y mostrar gráficas.
- **CSS3.** Hojas de estilo en cascada (por sus siglas en inglés CSS) es un lenguaje empleado para dar estilo a las webs, es decir que se "vean más bonitas".
- **PHP.** Se trata de uno de los lenguajes más extendidos en el mundo web. De propósito general aunque empleado normalmente por parte del servidor y oculto de cara al cliente. Funciona de forma similar a JavaScript pero está mucho más extendido dentro del mundo web. Permite generar toda la lógica de un sitio web de forma totalmente opaca al usuario que lo emplea.
- **C/C++.** Se trata del lenguaje que emplea Arduino internamente para funcionar. Difiere en algunos puntos del propio lenguaje en sí pero se trata de casos puntuales para adaptarse al funcionamiento del hardware o hacerlo más fácil.

3.1.4 Servidor de bases de datos

- **Azure SQL Server.** Este servidor será el encargado de recoger todos los datos que se irán recopilando durante las sesiones con la app para posteriormente poder ser visualizadas y analizadas por algún gestor de datos web. Se tratará pues, de una base de datos relacional que se apoya en la sintaxis SQL. Forma parte de la suite para cloud de Microsoft Azure.
- **Google Cloud SQL**. Este servidor será el encargado de recoger todos los datos que se irán recopilando durante las sesiones con la app para posteriormente poder ser visualizadas y analizadas por algún gestor de datos web. Se tratará pues, de una base de datos relacional que se apoya en la sintaxis SQL. Forma parte de los servicios de la suite para cloud de Google Cloud Platform.
- **mariaDB**. Este servidor será el encargado de recoger todos los datos que se irán recopilando durante las sesiones con la app para posteriormente poder ser visualizadas y analizadas por algún gestor de datos web. Se tratará pues, de una base de datos relacional que se apoya en la sintaxis SQL.
  Se trata de una bifurcación de MySQL desarrollado por los mismos creadores tras vender MySQL. Su propuesta se basa en mantener la compatibilidad en todo lo posible con su anterior producto y puede ser utilizada exactamente igual que MySQL en la mayoría de casos
- **Firebase Realtime Database**. Forma parte de la plataforma Firebase de Google. Se trata de una base de datos NoSQL en tiempo real que permite enviar y recibir datos de forma muy sencilla y rápida. Puede ser una opción muy interesante para gestionar las

cuentas y perfiles de los usuarios ya que facilita la integración con servicios como 'Iniciar sesión con Google' y tiene mucha documentación sobre cómo implementarlo en una aplicación Android.

3.1.5 Servidor web

- **Apache.** Se trata de un servidor HTTP de código abierto multisistema empleado para alojar páginas y otros servicios web.

3.1.6 Servicios y herramientas en el servidor

- **Docker.** Se trata de una tecnología de creación y gestión de contenedores de software autocontenido, es decir, con todo el software que incluye el contenedor está listo para funcionar. Docker emplea el Kernel Linux y algunas funciones de este para poder ejecutar de forma independiente todos los contenedores. Este es el factor clave que identifica a esta tecnología; poder disponer de todo el software que se requiera de forma independiente entre sí de forma que sea muy fácil de instalar y gestionar. Mediante repositorios como DockerHub se facilita el acceso a imágenes ya creadas del software necesario.
- **Portainer.** Portainer es una herramienta web que facilita la creación y gestión de contenedores Docker de manera gráfica, es decir, sin necesidad de emplear la terminal más allá de su propia instalación. Esta herramienta permite gestionar diferentes espacios, es decir diferentes equipos siempre y cuando se pueda acceder a estos, gestionar volúmenes de almacenamiento, contenedores, entre otros.
- **Grafana.** Es una herramienta de software libre que facilita la visualización y formato de métricas desde diferentes fuentes. Esto permite poder crear varios cuadros de control para visualizar datos desde diferentes fuentes de datos entre ellas diferentes tipos de bases de datos como MySQL, PostgreSQL, InfluxDB y otros sistemas para obtener información sobre equipos informáticos como Telegraf.
- **No-IP.** Se trata de un proveedor de DNS dinámico. Este tipo de servicios permiten asignar un dominio a IPs dinámicas, direcciones IP que suelen encontrarse en domicilios. Puede ser interesante para poder levantar pequeños servidores domésticos o servidores de pruebas sin necesidad de pagar dominios ni IPs fijas.
- **SSH.** O Secure Shell, es el nombre de un protocolo y programa cuyo propósito es permitir el acceso o gestión remota de un servidor. Se trata de un programa que será necesario para gestionar tanto el servidor de pruebas como el servidor de producción.

3.2 Implementación práctica

En este apartado se va a explicar los diferentes aspectos que se han visto planificados o implicados en el desarrollo del proyecto.

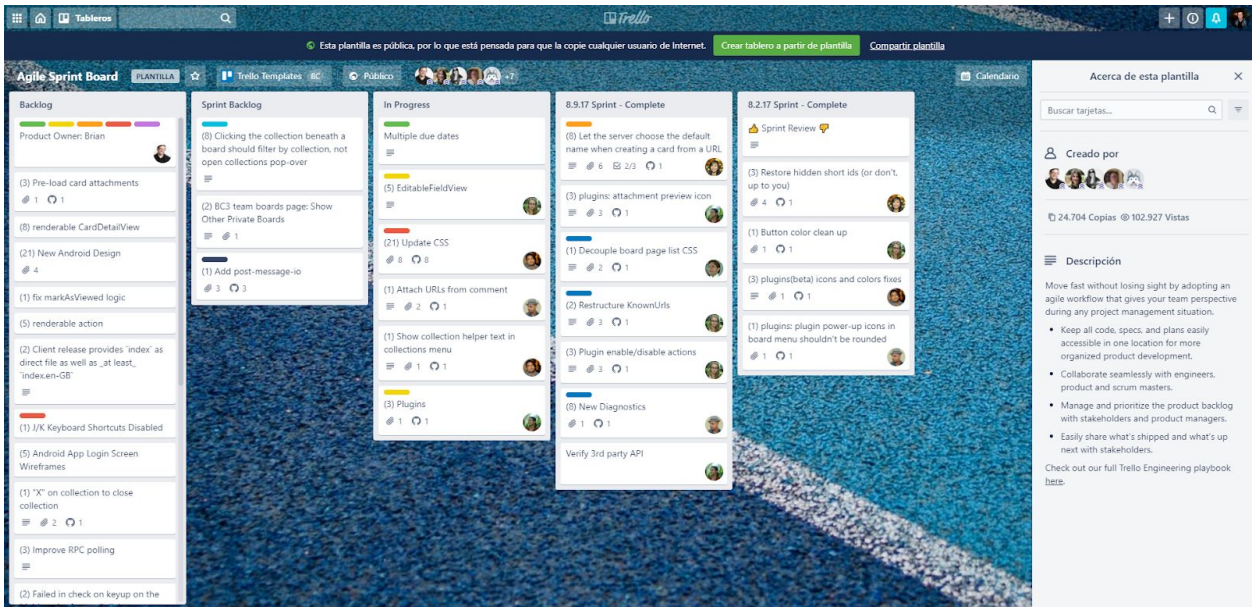
En primer lugar se analiza cómo se planificó desde un inicio el proyecto, metodología para su planificación y seguimiento y poder predecir cuando aproximadamente se cumplen las distintas partes del proyecto. Posteriormente se ataca el diseño del producto y su funcionamiento, se realizan distintas maquetas y se analizan distintas tecnologías y partes del código a destacar.

También se añade al final de este apartado documentación acerca de Google Cloud Platform y Microsoft Azure, dos nubes que se plantearon como posibles para desempeñar los distintos servicios por parte del servidor requeridos.

3.2.1 Metodología empleada para el desarrollo

Para llevar a cabo de forma eficaz el desarrollo de todo el sistema se planifica bajo un kanban. Personalmente no lo llamaría metodología Kanban como tal, sino una metodología que he ido perfeccionando para mi mismo, para mi productividad personal, y que al menos para mi forma de trabajar y teniendo en cuenta el resto de ocupaciones que tenía he valorado que podía funcionar mejor.

En primer lugar se planificó un tablero de tareas en Trello, una herramienta muy útil para organizar tareas. Esto ayudó en primer lugar a esquematizar mejor la aplicación, ver qué partes hacían falta y empezar a priorizar algunas tareas frente a otras con un sistema de prioridades de colores (rojo,naranja,amarillo,verde) para saber qué tareas eran prioritarias y cuáles debían ser abordadas antes que otras opcionales o menos prioritarias y que a lo mejor dependen de otras más prioritarias.



La imagen superior es un ejemplo de plantilla disponible en Trello bastante parecida a lo que se implementó. En caso de ser necesario Trello puede usarse con equipos de personas.

Esta desgranación de tareas se realizó mediante un par de brainstormings analizando qué era lo que se requería para diseñar el sistema entero y valorando qué tecnologías o formas de abordar el problema podrían ser las más adecuadas. Estas tareas no fueron inamovibles, sino que entre algunas de ellas surgieron algunos problemas y aparecieron nuevas tareas, algunas de ellas prioritarias para subsanar problemas o corregir algunas deficiencias de tareas anteriores.

Al tratarse de una sola persona, decidí omitir todo lo relacionado a reuniones o revisiones periódicas, al terminar una tarea, esta se verifica si efectivamente funciona correctamente y por tanto realmente está terminada. Como lo anterior podría haber sido problemático para cuadrar los tiempos se decidió planificar un calendario aproximado para cada parte de las tareas dividiendo en grupos generales o bloques que serían los siguientes:

- Diseño de la app. Esto incluye crear el diagrama de flujo de toda la aplicación Android y la posterior programación de las distintas pantallas entre las que se reparte la funcionalidad.
- Análisis e implementación de los distintos sensores que se pueden manipular en Android así como algunas de las conexiones a Internet para acceder a Firebase. Se empezó a implementar Firebase en este punto ya que es una parte importante en el desarrollo del inicio de sesión y el registro. Además, al ser una tecnología de Google cuenta con documentación detallada que facilita su rápida implementación.
- Diseño y programación de Arduino. Esto incluye todo lo relacionado al montaje de los componentes de la placa de desarrollo y su gestión por Bluetooth.

- Diseño e implementación del Cloud. Esta fue la última parte que se llegó a implementar si bien sus primeros diseños se realizaron al inicio de todo. Se trataba de la última parte que se quería abordar ya que primero se requería la generación de datos para posteriormente tener ciertos datos sobre los que basarse al diseñar los dashboards de forma efectiva.

Se planificó el primer bloque para enero, coincidiendo con los exámenes residuales para poder disponer de tiempo suficiente para ir buscando documentación que pudiera ser de utilidad.

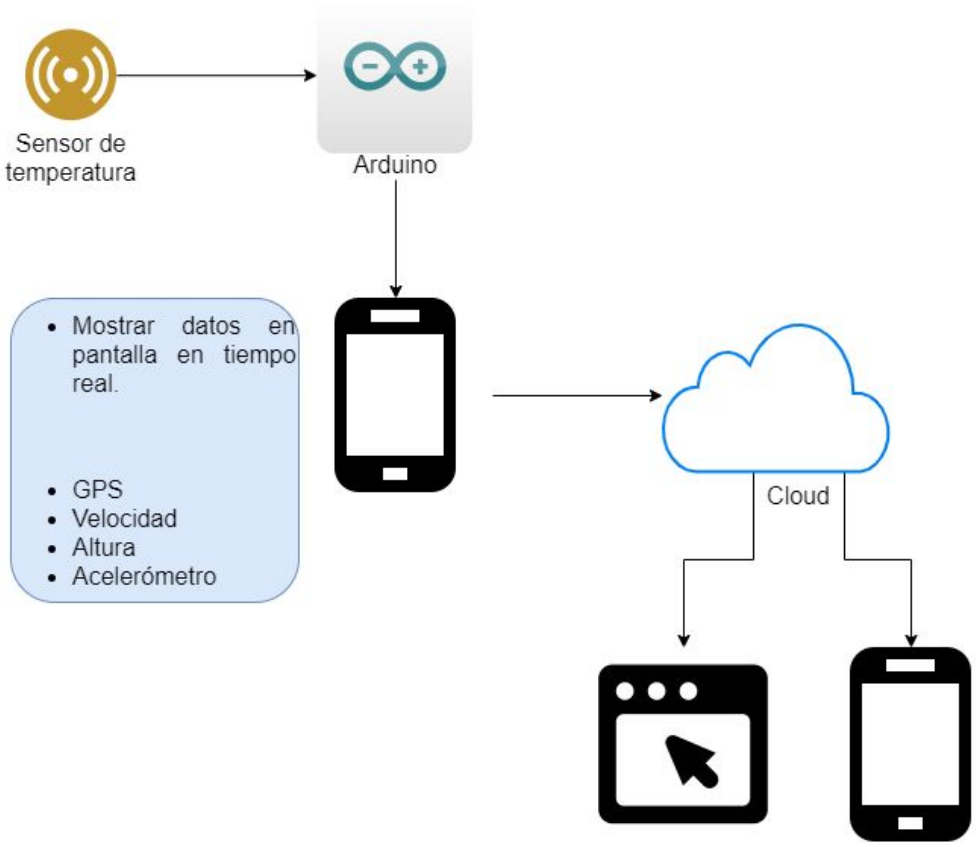
El segundo bloque se destinó a febrero, se estimó dicha duración ya que había que realizar un trabajo de investigación sobre cómo funcionan los distintos sensores y antenas del dispositivo, había que implementarlo todo y realizar tests para verificar que todo iba correctamente antes de empezar la implementación de la conexión con un dispositivo Arduino. También el tener los indicadores gráficos ya preparados se valoró que serían de gran ayuda para realizar dicha conexión con Arduino y poder verificar que los datos se transmitían de forma correcta.

El tercer bloque se planificó para marzo, entre sus tareas, se requería la compra de algunos componentes que faltaban como el sensor láser. Investigar el correcto montaje de la placa de desarrollo Arduino con todos sus componentes y empezar el desarrollo de la conexión tanto por parte del dispositivo Android como por parte de la placa Arduino con el módulo Bluetooth.

Finalmente, la implementación del cloud se planificó para Abril ya que se estimó oportuno disponer ya del resto del sistema para poder proveer al cloud de datos con los que hacer tests con la conexión a la base de datos como posteriormente tratar dicha información en alguna herramienta de generación de gráficos.

Por otra parte, pero no por ello menos importante, al obviar las revisiones que se suelen realizar al final de los sprints, se decidió realizar dichas revisiones al final de cada bloque de los anteriormente mencionados.

3.2.2 Diseño del producto



Se va a diseñar un sistema de telemetría para cualquier tipo de vehículo mediante el uso de un dispositivo Arduino y un dispositivo Android empleando los sensores correspondientes. Dichos dispositivos aprovecharán la conectividad a Internet que dispone el smartphone para enviar los datos recopilados a una base de datos situada en un servidor donde también se desplegará una herramienta para la visualización de dichas métricas.

Este sistema puede ser empleado en distintos tipos de vehículos como motocicletas, coches, patinetes,etc. Indistintamente de que sean vehículos de combustión interna o eléctricos.

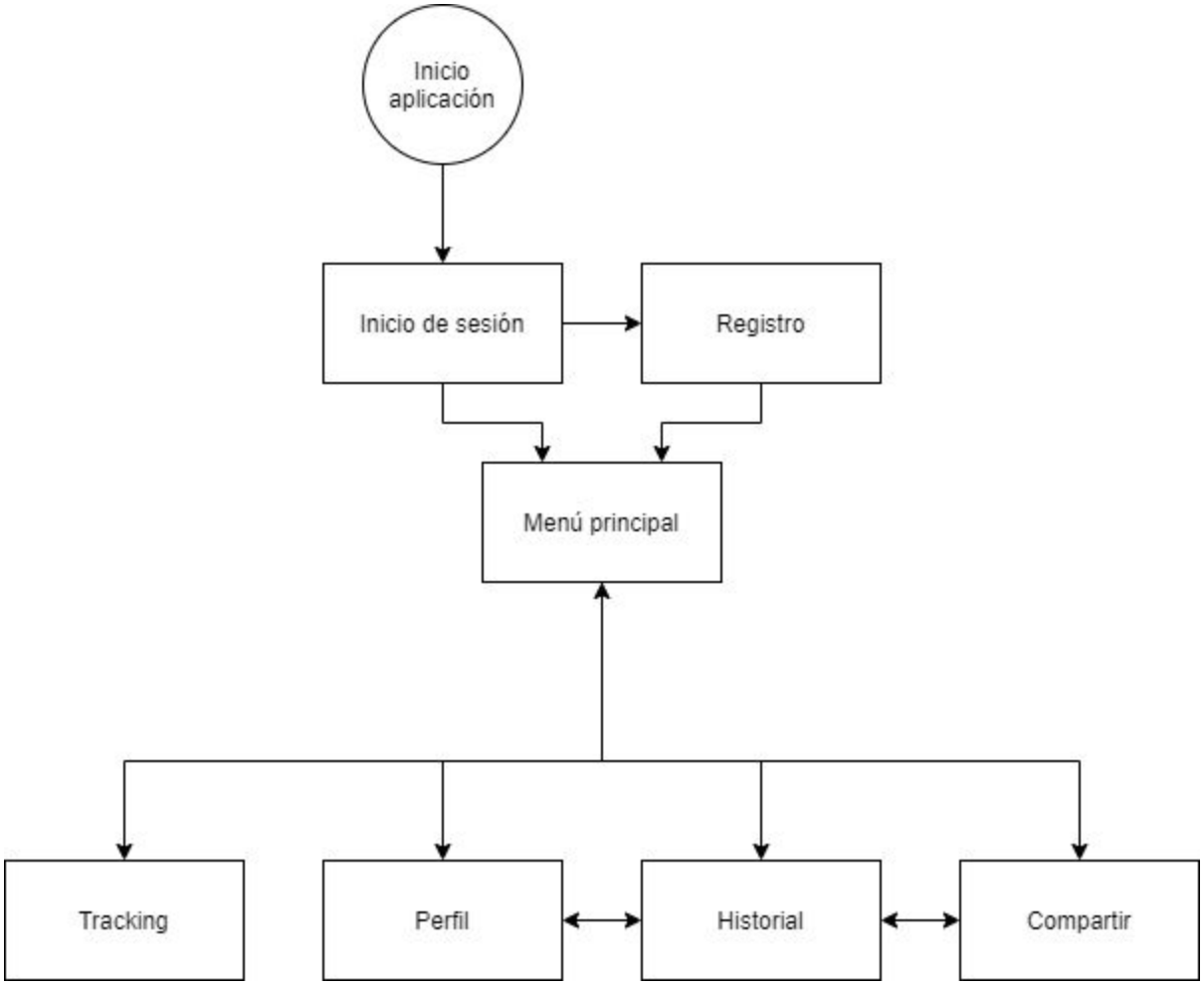
El dispositivo Arduino se encargará de medir la temperatura específica de una parte del vehículo como puede ser la temperatura de una rueda, de un disco de frenos, de la batería… Lo que se considere más oportuno en cada situación. Por otra parte transmitirá dicha temperatura al dispositivo Android empleando un módulo Bluetooth con el que será capaz de emparejarse y posteriormente ir enviando los datos.

El smartphone Android, además de conectarse recibir la información del Arduino, implementará todo el acceso a los servidores para poder registrarse y/o iniciar sesión para cada usuario, permitirá recopilar información de sus propios sensores y antenas, como el acelerómetro y el

GPS para poder conocer cómo se mueve el vehículo o a qué velocidad se mueve. Finalmente recogerá todos los datos que se han obtenido e irá tanto almacenándolos en local como enviandolos al cloud donde podrán ser utilizados con herramientas de generación de dashboards. Así mismo, la propia aplicación dispone del acceso a los dashboards para poder ser consultados.

El cloud se encargará tanto de almacenar todos los datos en una base de datos, como facilitar su visualización mediante dashboards con herramientas como pueden ser Grafana.

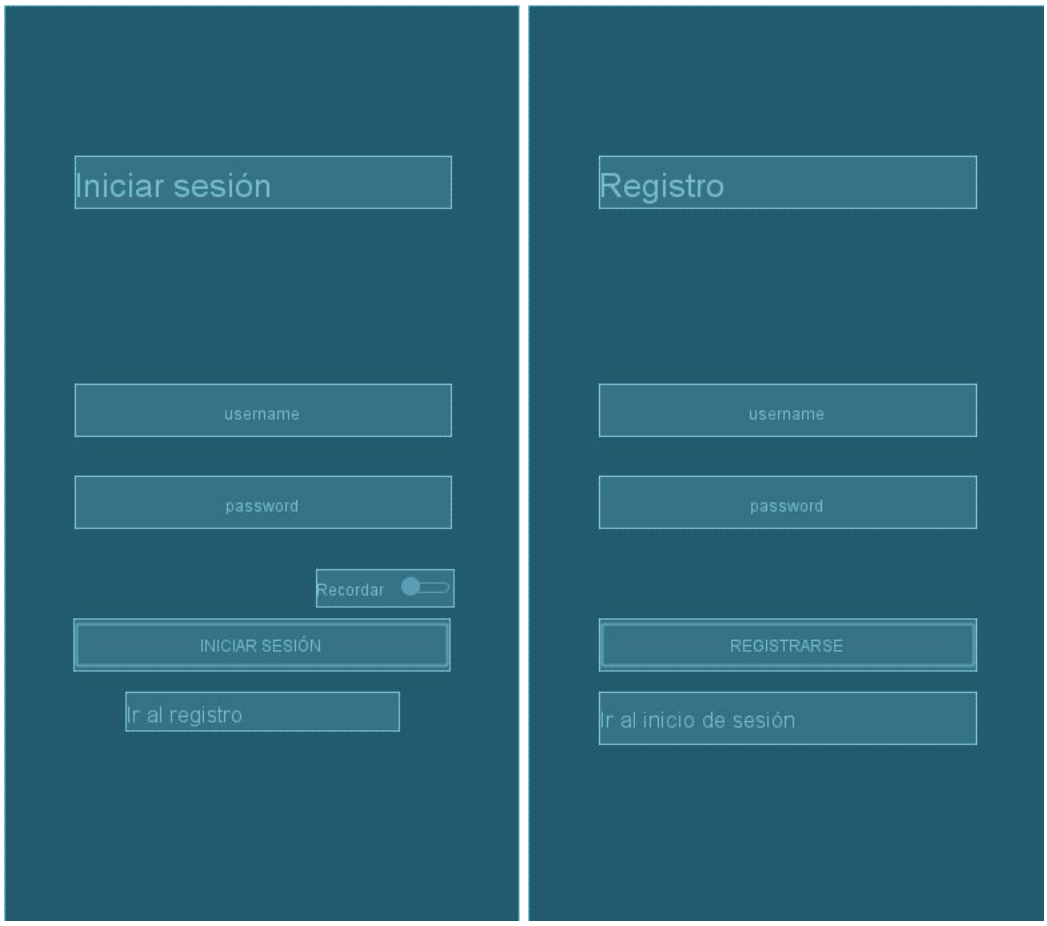
3.2.3 Diagramas de funcionamiento



En este diagrama se puede apreciar el flujo que puede seguir un usuario desde que inicia la app. Si bien el inicio de sesión puede realizarse de forma automática al marcar 'Recordar' durante el inicio de sesión. El inicio de sesión automático pasa por dicha pantalla sin que el usuario tenga que hacer nada. Por otra parte, como en Android el usuario puede abandonar la

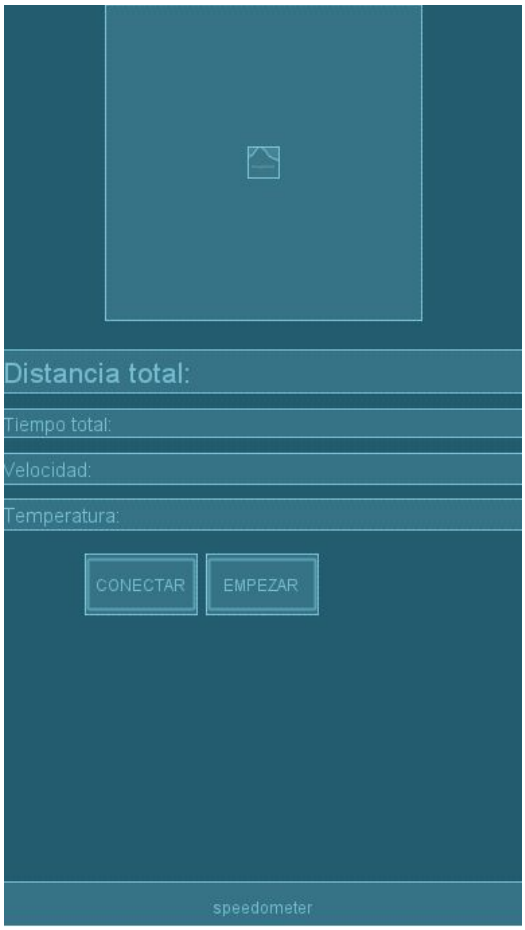
app en cualquier punto, simplemente volviendo al launcher, no se ha incluido un punto de salida.

IMAGENES DISEÑOS VISTAS



Pantalla Iniciar sesión Pantalla Registro

Estas serán las dos pantallas por las que deberá pasar inicialmente el usuario para crearse una cuenta e iniciar sesión cada vez. Opcionalmente, tendrá la posibilidad de recordar las credenciales y evitar tener que introducirlas cada vez que inicie la aplicación.



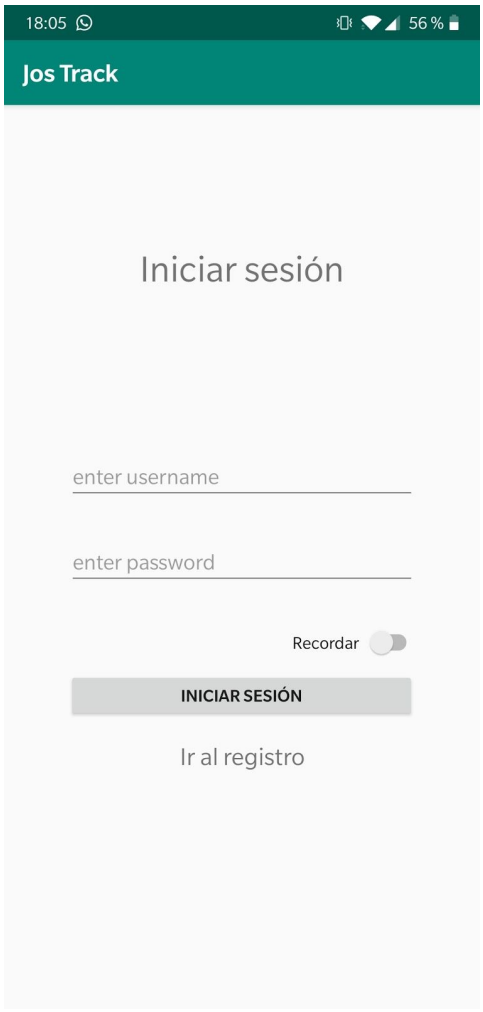
Pantalla Tracking

Pantalla Perfil

La pantalla del Tracking está pensada para que se puedan realizar los preparativos pertinentes antes de poner en marcha el vehículo y funcionar mientras se maneja el vehículo hasta que el usuario lo detiene y detiene también el tracking.

El perfil de usuario está pensado para complementar información sobre cada usuario de forma que facilite futuros desarrollos del sistema. También permite conocer el ID de la próxima sesión que realice el usuario y modificarlo en caso de ser necesario.

3.2.4 Maqueta



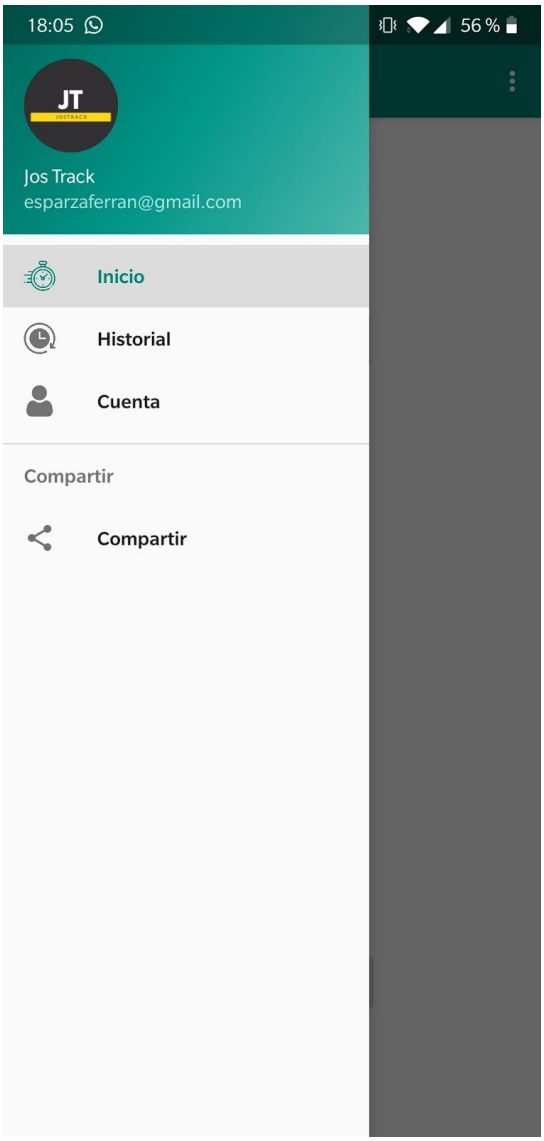
Pantalla Iniciar sesión

Pantalla Registro

La pantalla de inicio de sesión es la primera a la que accede el usuario. En caso de haber marcado 'Recordar' en el inicio de sesión anterior la aplicación iniciará sesión automáticamente e irá a la pantalla principal.
En caso de ser la primera vez que se inicia y no se dispone de cuenta, se puede pulsar en el botón 'Ir al registro' para acceder al formulario básico de registro. Más adelante, en la pantalla 'Perfil' se podrá completar el resto de información.

Si se accede al registro por equivocación o por cualquier otro motivo se desea volver al inicio de sesión, se dispone de un botón en la parte inferior al igual que para acceder al registro.

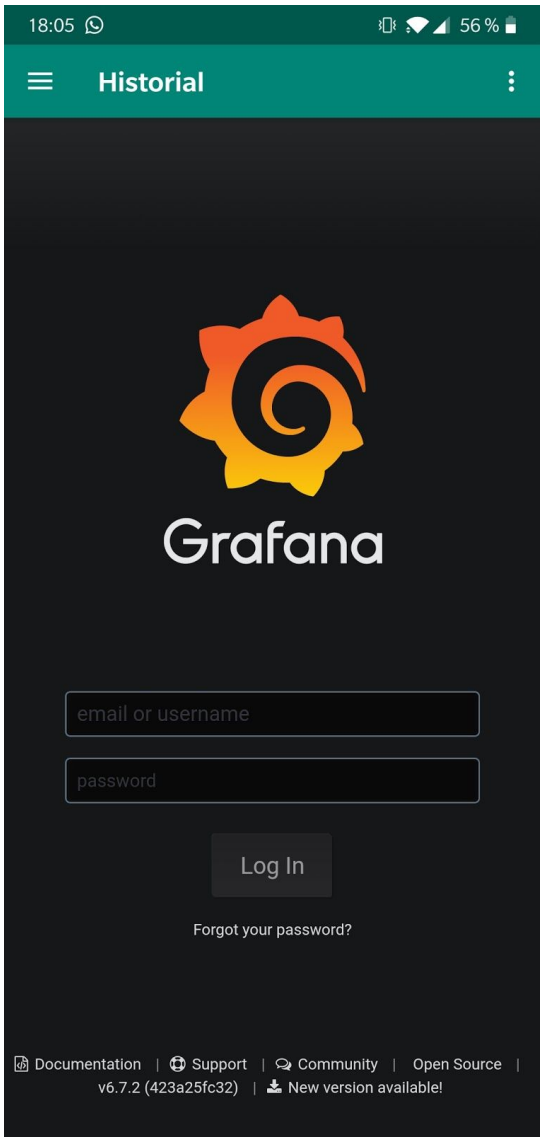
Pantalla Principal



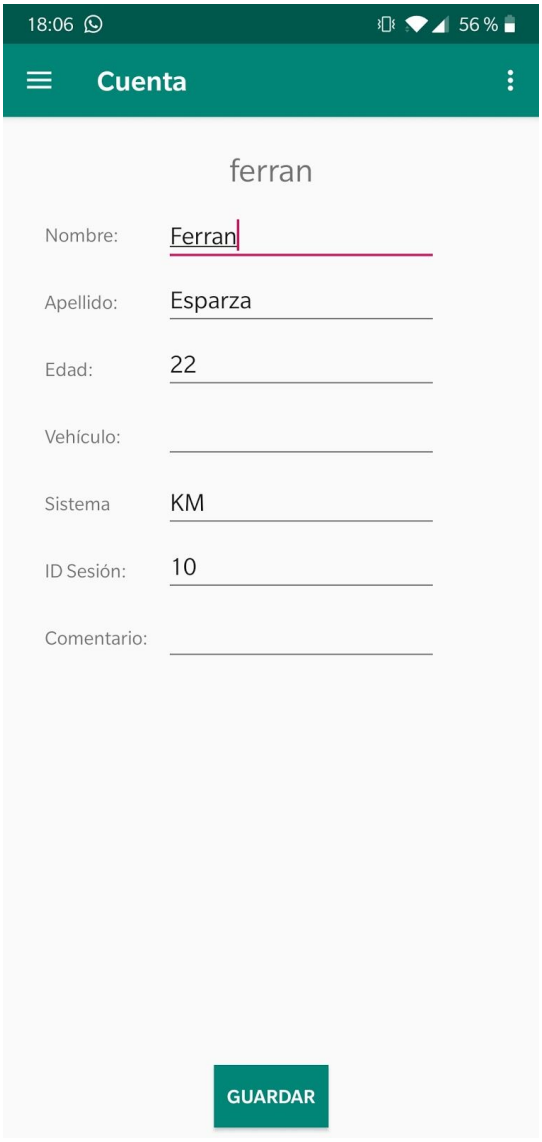
Menú lateral

Una vez se accede a la pantalla principal se tiene acceso a todo el resto de la aplicación. Deslizando desde la parte izquierda de la pantalla se accede al menú lateral desde donde se puede navegar entre el resto de las pantallas.

Para cerrar la sesión y volver al inicio de sesión o al registro, se dispone de un botón de 'Cerrar sesión' en la parte superior derecha tras pulsar sobre el botón que despliega el menú.



Pantalla Historial

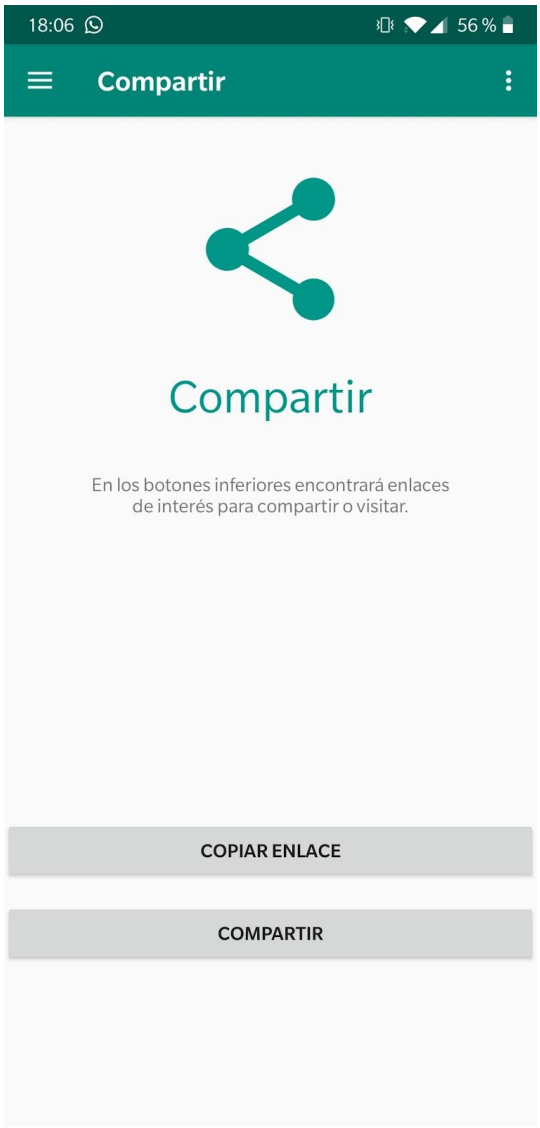


Pantalla Cuenta

En el historial se da acceso mediante una WebView a Grafana. Aquí será necesario iniciar sesión y una vez dentro se podrán consultar todos los datos como si se tratara de un ordenador.

Por otra parte tenemos la pantalla del perfil, donde se completa la información del usuario. Dicha información está enfocada a ampliar la funcionalidad del sistema en futuros desarrollos, como podría ser aumentar la interacción entre usuarios o hacer más social la aplicación. Como detalle a destacar, en el perfil de cada usuario se encuentra el campo ID Sesión. Este campo está pensado para poder ver y/o modificar el ID que tendrá la próxima sesión de tracking.

Desde ambas pantallas se puede volver a la pantalla principal o ir a otra mediante el menú lateral.



Pantalla Compartir

Pantalla Tracking

La pantalla 'Compartir' está pensada para poder compartir los enlaces de interés del sistema, como puede ser el link de descarga del apk o el acceso a Grafana.
Dichos links están todos agrupados dentro de uno solo. Este link global puede copiarse al portapapeles del dispositivo o compartir directamente utilizando otra app.

Desde ambas pantallas se puede volver a la pantalla principal o ir a otra mediante el menú lateral.

3.2.5 Sensores y antenas empleados

Sensores y antenas empleados por el dispositivo Android
El dispositivo Android, al ser cada vez más completos en cuanto a funcionalidad y componentes, permite usar la antena GPS, el acelerómetro y la red móvil como apoyo al GPS. También se emplea dicha red móvil para comunicarse con la nube y transmitir los datos.También emplea el bluetooth para la comunicación con la placa Arduino.

- Antena GPS
- Sensor Acelerómetro
- Antena Bluetooth: versión 5.0
- Módem.

La marca o modelo de dichos componentes por la parte de Android son irrelevantes, pues las APIs con las que se puede desarrollar en Android ya contemplan su de una forma generalizada y unificada para los distintos dispositivos. En todo caso, para las pruebas se emplea un OnePlus 6

- GPS: IZat Gen8C
- Acelerometro: Bosch BMI160
- Bluetooth: Bluetooth 5.0
- Modem: Snapdragon X20 LTE. Se trata del módulo dentro del smartphone encargado de las comunicaciones móviles como los datos móviles.
- Antena wifi: Qualcomm WCN3990 2x2 802.11ac Wi-Fi

Sensor y antena empleados por el dispositivo Arduino
Por su parte, el dispositivo basado en Arduino servirá para medir la temperatura mediante un sensor de temperatura láser que se podrá enfocar al objeto o parte del vehículo deseado.

Sensor Láser: GY-906 MLX90614
Antena Bluetooth: HC-06

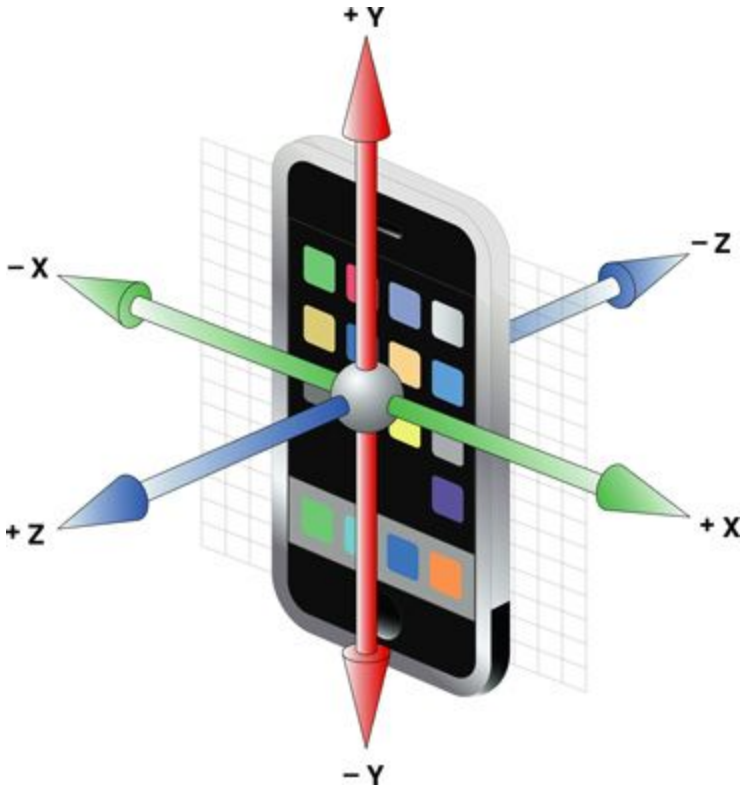
3.2.6 Calibración del acelerómetro

Es interesante calibrar el acelerómetro para tratar de conseguir datos lo más fiables posible. Antes de realizar una recopilación de datos, la aplicación pide al usuario calibrar el acelerómetro del smartphone dentro del vehículo. Este paso es muy recomendable para poder disponer de datos lo más fiables posible.

Al acceder a la aplicación, se lanza un aviso emergente de que es necesario calibrar el acelerómetro.

Una vez se pulsa sobre 'CALIBRAR' la aplicación almacena la posición en la que se encuentra el smartphone para posteriormente aplicar una compensación a los datos que vaya recopilando durante la sesión.

Dependiendo de cómo se fije el smartphone dentro del vehículo tendrá unos valores base u otros. Si se tratase de comparar diferentes sesiones se arrastraría cierto error derivado de contar con el dispositivo en distintas posiciones que afectarían a la fiabilidad. Calibrando el acelerómetro se pretende evitar esto mismo y contar con un entorno base lo más neutro posible en cuanto a acelerómetro se refiere.

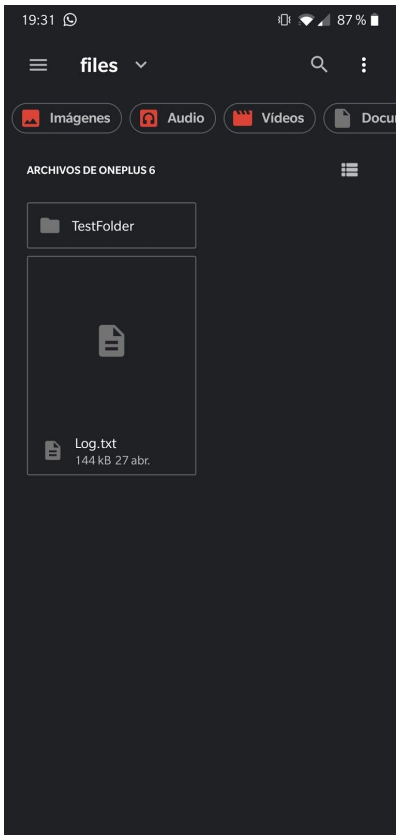


Por una parte, el sensor siempre calcula la fuerza de la gravedad que ejerce su influencia hacia el interior de la Tierra (9,81 m/s^2) en posición neutral, es decir, cuando el vehículo se encuentra en reposo. Esta fuerza puede afectar a uno de los ángulos, o varios dependiendo de la posición del smartphone dentro del habitáculo y por tanto podría 'desplazar' todos los datos por no haber realizado una calibración inicial. En el futuro, los datos de diferentes sesiones podrían perder fiabilidad debido a la falta de dicha calibración ya que el smartphone podría ver alterada su posición debido a diversas circunstancias, como por ejemplo que pudiera molestar al piloto en primera instancia o que se encontrara en una situación de peligro, cambio de soporte por otro más o menos robusto, etc.

3.2.7 Almacenamiento local de los datos

El smartphone irá guardando de forma local los registros que se van creando. Estos registros juntarán los propios generados por el smartphone como los recogidos de la placa Arduino. Esto permitirá que el smartphone tenga una copia en caso de tener una conexión con la base de datos desfavorable pero que no se pierdan datos, sino que estos estén almacenados en local y posteriormente se suben a la nube donde podrán ser procesados.

Los registros se van guardando en un archivo llamado log.txt en la ruta /Android/data/com.ferran.jostrack/files.



Los registros se van almacenando de forma ordenada dentro de un mismo archivo (Log.txt).

3.2.8 Tecnologías software empleadas en la aplicación Android

Para llevar a cabo todo el desarrollo de la aplicación Android se ha recurrido a ciertas librerías de terceros con las que conseguir ciertas funcionalidades ahorrando la necesidad de tener que desarrollar lo mismo desde cero.

Al abordar primero la parte visual de la aplicación, uno de los primeros problemas a los que se llega es como representar los datos que se van consiguiendo, entre ellos la velocidad.

Para esto, mediante una librería que se puede encontrar en GitHub es bastante sencillo colocar el velocímetro en pantalla. No funciona la visualización previa en el editor, por lo que se deben hacer pruebas para encontrar su posición y proporciones óptimas.
Pero por otra parte, la gestión lógica es muy sencilla:

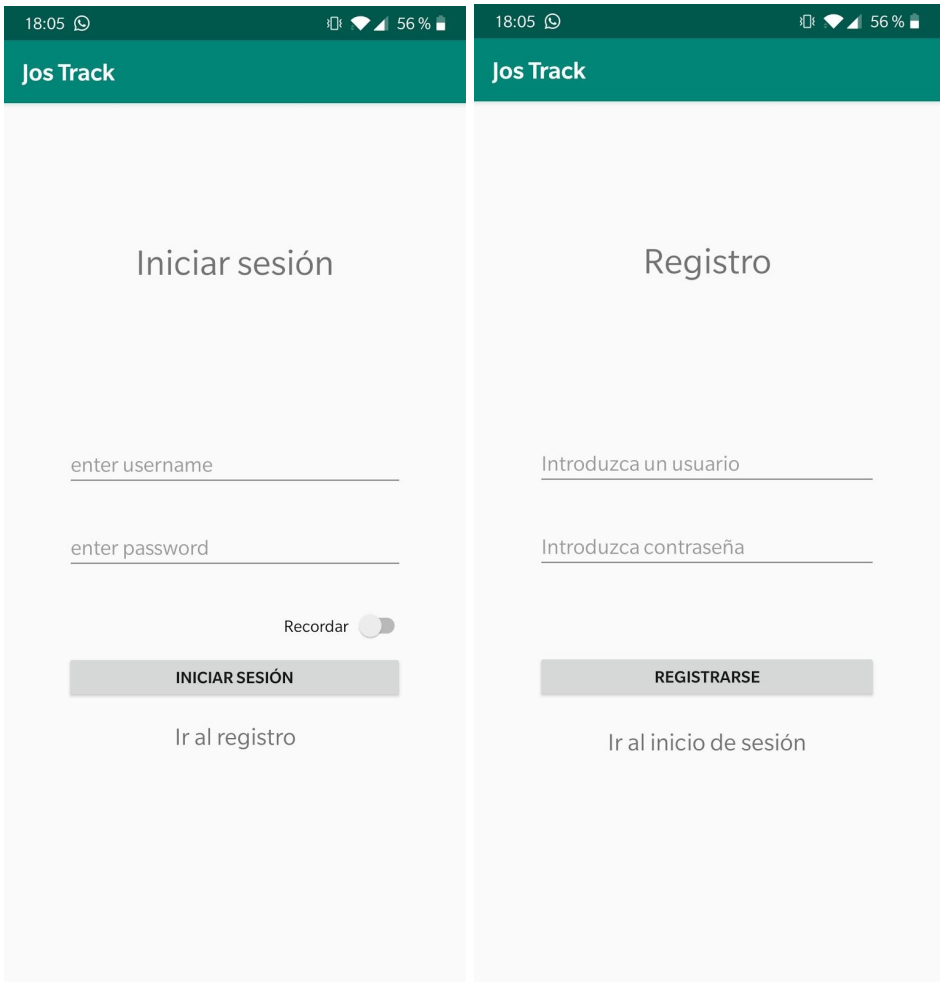
```
// configure value range and ticks
  speedometer.setMaxSpeed(300);
  speedometer.setMajorTickStep(30);
  speedometer.setMinorTicks(2);

  // Configure value range colors
  speedometer.addColoredRange(30, 140, Color.GREEN);
  speedometer.addColoredRange(140, 180, Color.YELLOW);
  speedometer.addColoredRange(180, 400, Color.RED);
```

Mediante una serie de variables que incluye cada velocímetro que se cree se puede configurar prácticamente todo. Los rangos de velocidades, la velocidad máxima que representa el velocímetro y los colores que se asocian a cada rango. Este punto es bastante interesante para poder adaptar el rango a las necesidades de cada proyecto.

Y para que el velocímetro sepa qué velocidad representar, se puede llamar al método setSpeed para que asigne la posición correcta del velocímetro.

Esta librería puede ser usada para la velocidad, como es el caso, pero podría ser empleada para otros datos ya que no extrae de ningún sitio por si mismo la velocidad sino que debe ser el desarrollador el que le proporcione el origen de los datos.

Para conectar con la base de datos de usuarios localizada en Firebase es bastante sencillo siguiendo la documentación que ofrece Google acerca de su servicio.



Por una parte, el Login se realiza pidiendo a Firebase un archivo JSON en el que se incluye la información del usuario correspondiente, se comprueba si coinciden las credenciales y en caso de ser correctas se almacenan para poder recuperar posteriormente más información o modificarla en el perfil.

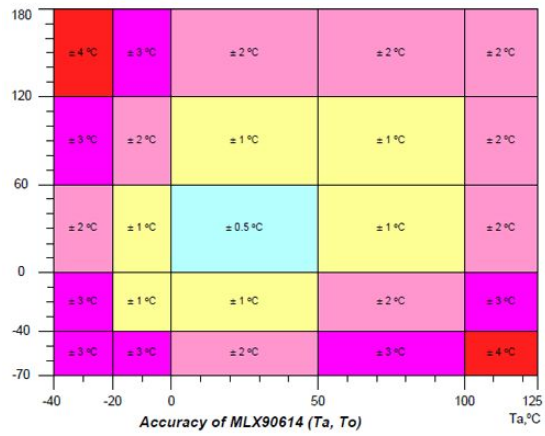
Para el registro de usuarios se recurre a la librería Volley para recoger la información introducida por el usuario en el formulario de registro y se envía a Firebase. Este mismo proceso se sigue para modificar el perfil de los usuarios desde la pantalla de Perfil.

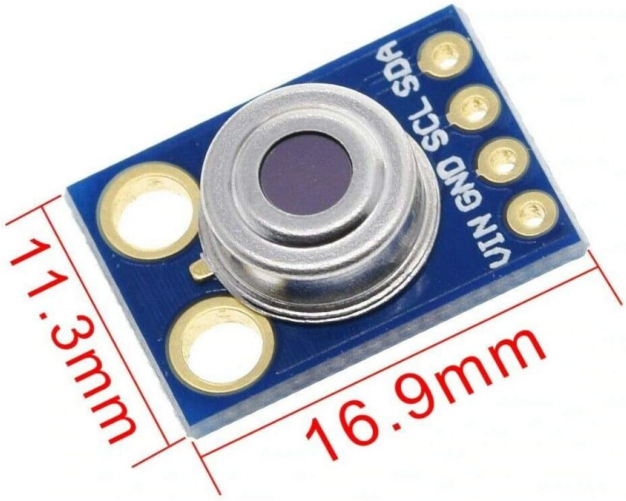
Volley además de ayudar en la comunicación con Firebase también se encarga de la comunicación con la base de datos mariaDB en la que se almacenan los datos de las sesiones de los usuarios al volante. Esta biblioteca HTTP facilita el envío y recepción de información por Internet de forma que la app simplemente debe realizar solicitudes al servidor mediante Volley.

3.2.9 Lógica del script empleado en Arduino

Código empleado para diseñar la lógica que operará el sistema Arduino para recopilar y transmitir datos.
En primer lugar, cada vez que se enciende el dispositivo Arduino realiza las configuraciones iniciales del módulo Bluetooth empleando Comandos AT estándar para definir los baudios, el nombre por el que podrán reconocerlo el resto de dispositivos y el PIN para poder conectarse.

Una vez queda ya configurado el Bluetooth para poder atender una conexión, la placa de desarrollo empieza a enviar por el puerto serie la temperatura que va recopilando del sensor de temperatura láser.





En las imágenes superiores se puede ver por una parte como es el sensor físicamente, su apariencia física. Por otra parte se puede ver la precisión de dicho sensor según como se enfoque al objeto o zona deseada. Cuanto más centrado esté el objeto con el sensor más precisa será la medición cuando se pretenda captar una medición precisa de un objeto.

El sensor de temperatura láser GY-906 MLX90614 , una vez empieza a funcionar recoge la temperatura de dos formas a la vez. Por una parte recoge mediciones del ambiente en general al que enfoca con un ángulo entre 5º y 80º con lo que se puede utilizar para hacer mediciones

de qué temperatura hay en el ambiente. Y la segunda forma que es la que interesa de forma puntual a un objeto concreto.
Mediante una instrucción u otra se puede recuperar en el Arduino la temperatura ambiente o la temperatura puntual. Para la primera, bastaría con emplear la instrucción "*mlx.readAmbientTempC()*" siendo "mlx" la variable inicializada que hace referencia al sensor y para la segunda se requiere la instrucción "*mlx.readObjectTempC()*".

Para enviar la temperatura al smartphone Android es necesario que ambos dispositivos estén vinculados antes. Para ello, el dispositivo Arduino debe tener ya el programa cargado y estar inicializado.
Desde el menú Bluetooth del smartphone, se debe buscar el dispositivo JosTrack, nombre asignado durante la inicialización del dispositivo y vincularlo mediante el PIN asignado en el código.
Una vez realizado este paso, empleando el botón 'CONECTAR' en la pantalla de Tracking de la aplicación Android se realiza la conexión automática entre los dos dispositivos. Se sabrá que se ha realizado con éxito ya que la aplicación Android lanzará una notificación avisando de que se ha realizado la conexión de forma exitosa y por parte del Arduino, el led que incorpora el módulo HC-06 que estaba parpadeando se habrá quedado encendido de forma fija.

Una vez realizado el emparejamiento, los datos de la temperatura empiezan a ser enviados desde el dispositivo Arduino al smartphone cada segundo. En caso de ser necesaria mayor precisión a lo largo del tiempo bastaría con reducir el delay introducido en el loop de Arduino. Pero también es importante igualar dicho delay en el código del Bluetooth de la aplicación Android pues sino la información puede llegar incompleta.

3.2.10 Tecnologías software empleadas en el Cloud

Empleando Docker se instalan y administran las dos herramientas principales instaladas en el Cloud. En este punto, todavía en pruebas, se emplean haciendo uso de un servidor doméstico con una Raspberry Pi.

Para ello se quema la imagen ISO de Raspbian en una tarjeta MicroSD y se realiza la configuración de red necesaria para conectar el servidor a Internet. En este caso empleando el WiFi que incorpora la propia Raspberry.
Una vez el servidor está disponible, es necesario conectarse por SSH para poder gestionarlo a distancia.

Desde la terminal, se instala en primer lugar Docker. Para ello primero actualizaremos todos los repositorios con:
sudo apt-get update
sudo apt-get upgrade

Se eliminan posibles versiones que pudieran haber anteriores:
sudo apt-get remove docker docker-engine docker.io

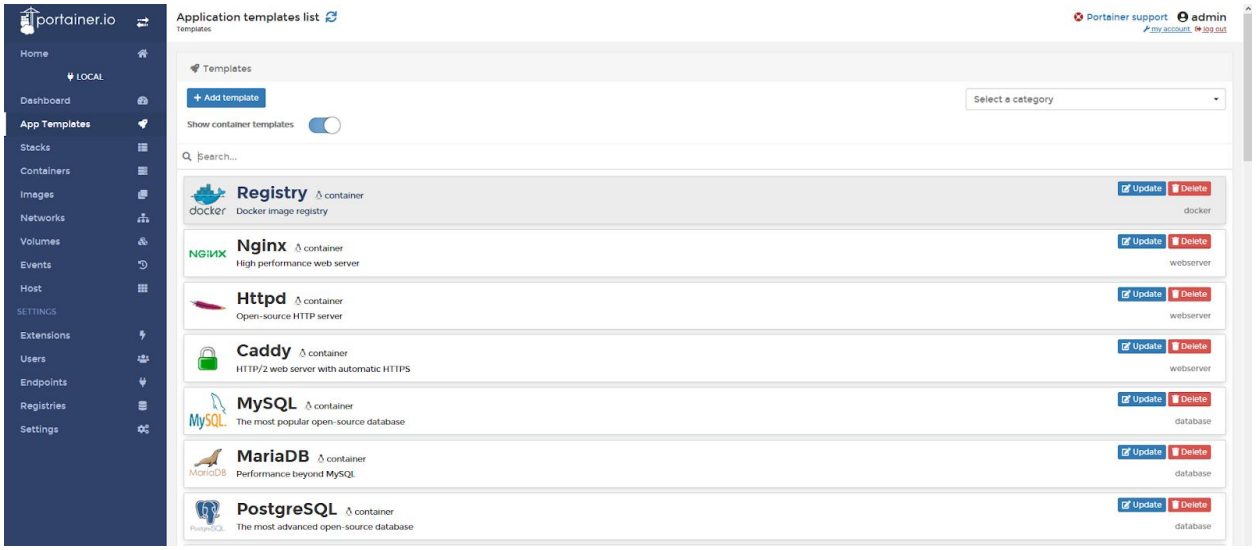
Y finalmente se instala Docker:
sudo apt install docker.io

También se debe configurar para que se inicie cada vez que se arranque el servidor:
sudo systemctl start docker
sudo systemctl enable docker

Una vez se tiene Docker, el servicio Portainer puede ser bastante útil para facilitar la instalación de los contenedores. Se requerirá un contenedor de MySQL o mariaDB y otro para Grafana. Para ello primero se creará un volumen para almacenar datos:
docker volume create portainer_data

Para luego poder instalar Portainer:
docker run -d \
--name portainer \
--restart=always \
-p 9000:9000 \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data portainer/portainer

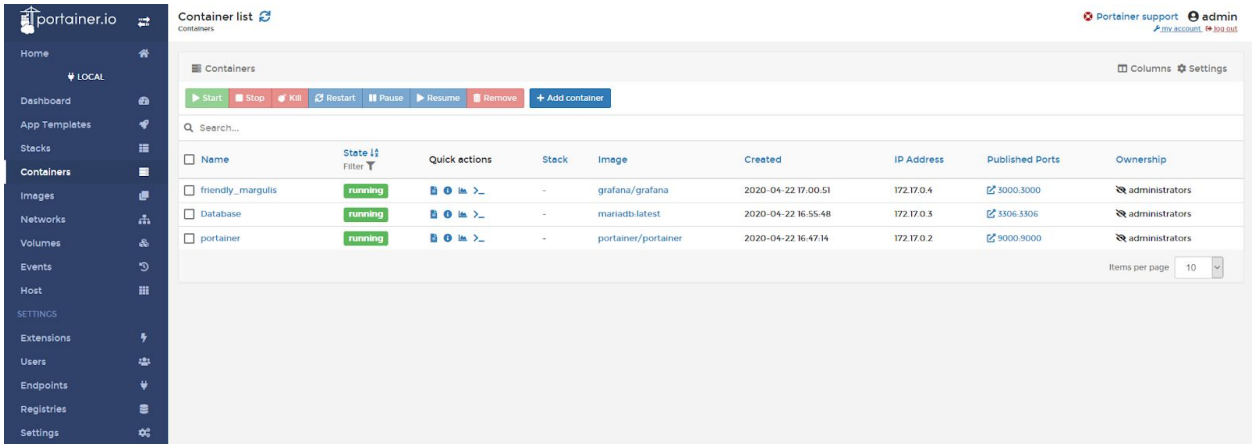
Una vez finalice ya se puede acceder desde el navegador con la dirección de la máquina+el puerto (en este caso le hemos asignado el puerto 9000). La primera vez que se acceda requerirá una contraseña para la cuenta de administrador y ya se podrán gestionar los contenedores.



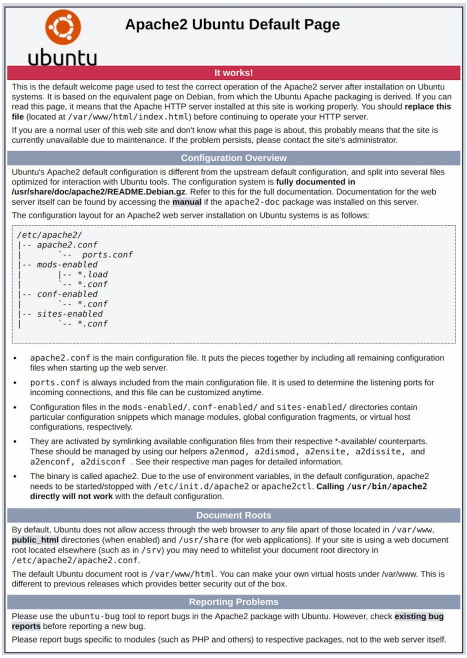
Mediante Portainer simplemente es necesario acceder a las plantillas y desplegar un contenedor con la plantilla que se requiera. Una de esas plantillas que se emplean es la de mariaDB para desplegar la base de datos donde se guarda todos los datos del proyecto.

Pero sigue siendo posible emplear Docker mediante terminal si no se quiere o no se puede emplear una plantilla para desplegar un contenedor. Para desplegar Grafana es tan sencillo como lanzar en la terminal:
docker run -d -p 3000:3000 grafana/grafana

Como detalle, los contenedores desplegados por terminal directamente con Docker, pueden verse y manipularse mediante Portainer:



Por otra parte, se requiere un servidor web capaz de ejecutar código PHP. Para ello se instala Apache y PHP. Posteriormente se subirán los scripts que se requieren para la plataforma.
En primer lugar, para instalar apache simplemente se requiere un comando en la terminal:
sudo apt install apache2
Una vez termina el proceso de instalación se puede acceder a la dirección IP mediante el navegador para comprobar que realmente se ha instalado y funciona correctamente.

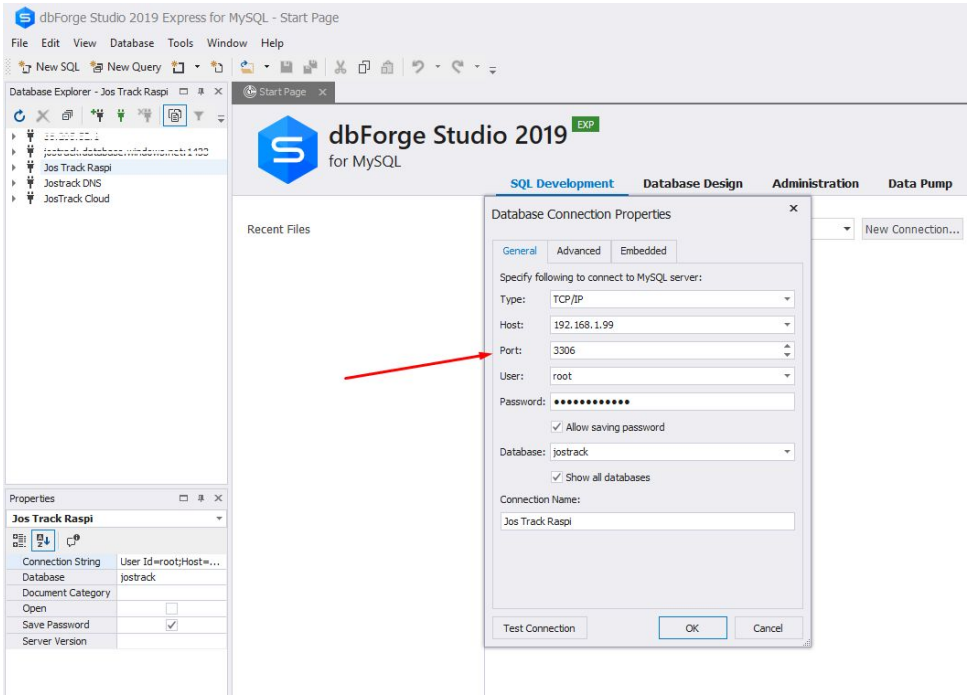


Pero este servidor todavía requiere de PHP para poder ejecutar archivos PHP:
sudo apt install php libapache2-mod-php php-mysql

A partir de aquí ya se pueden colocar archivos PHP o con código PHP en el servidor web.
Para ello, en la página por defecto de Apache, que ha mostrado antes se puede ver que las páginas html u otros archivos deben ir en la ruta **/var/www/html/** como el propio index.html. Aquí es donde deben colocarse tanto el *db_conection.php* como el script *asd.php* para realizar los inserts. El archivo asd realiza los insert basándose en la conexión que tiene establecida el *db_conection*. Siguiendo este esquema se podrían crear otros scripts basándose en el mismo archivo para la conexión según se requiera.

Al estar empleando un servidor de pruebas en un domicilio normal y corriente, es necesario gestionar la salida a Internet. Para ello se instala la herramienta No-IP con la que asignar a la IP externa de la red local un dominio con el que poder apuntar desde otro dispositivo fuera de la red local. Este o estos dispositivos serán los smartphones Android que se empleen para las pruebas.
Importante recalcar que en el router de la red local es necesario abrir el puerto 3306 ya que es el que se emplea en el servidor doméstico para acceder a la base de datos.
También es importante recordar que los dominios gratuitos de No-IP deben ser renovados de forma mensual para poder seguir utilizándolos.

Para la conexión con el servidor de base de datos se emplea DB Forge Studio Express. Una vez lanzado el programa, se crea la conexión con el servidor de base de datos y se abre automáticamente una pestaña donde poder trabajar con las consultas SQL.



En esta pestaña es donde se ejecuta el código que se ha diseñado para generar la base de datos y la tabla Tracks. Dentro de dicha tabla será donde se almacenen todos los datos de las sesiones al volante.

tracks		
PK	tracking_id	INT(20)
PK	user_id	VARCHAR(255)
PK	tracking_sequence	INT(20)
	distance	float(53)
	X	float(53)
	Y	float(53)
	Z	float(53)
	height	float(53)
	speed	INT(20)
	duration	float(53)
	temperature	float(53)
	sys_date	DATETIME

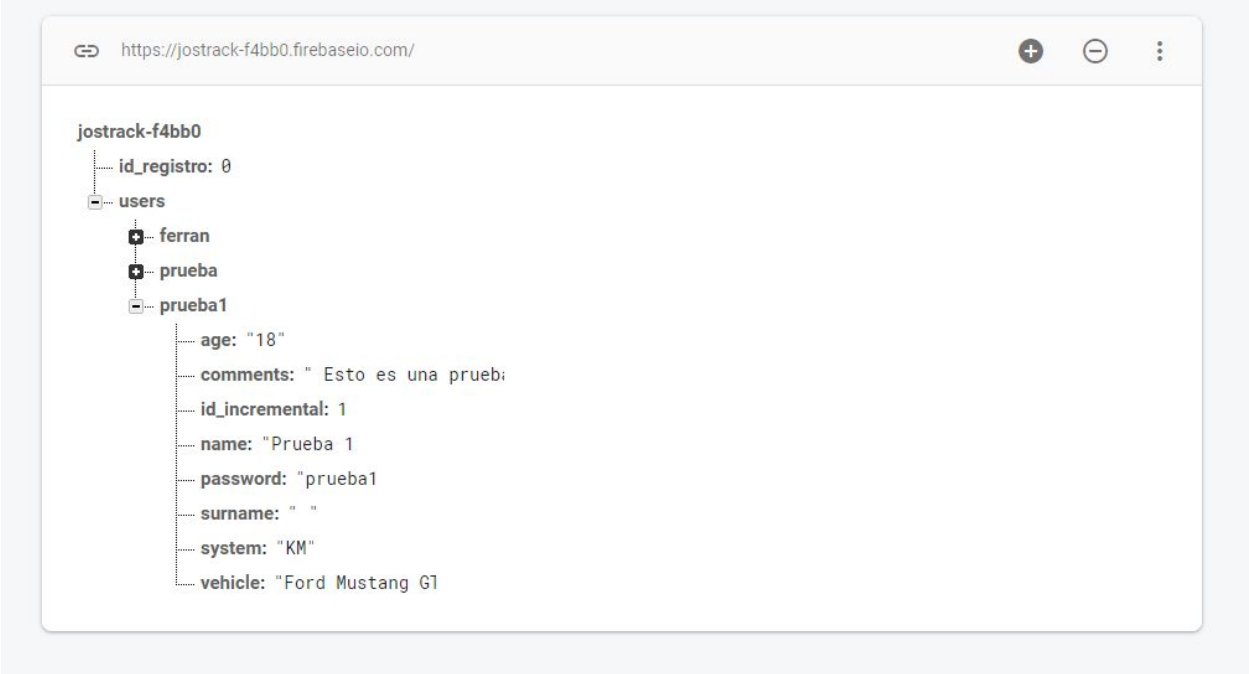
Como se puede apreciar en el esquema, la tabla en la que se almacenan todos los registros de los usuarios se identifica cada registro mediante tres claves primarias. Estas claves primarias permiten saber qué usuario ha realizado el registro, en qué entrenamiento y la secuencia concreta del entrenamiento. Con estos identificadores se podrá posteriormente filtrar de forma adecuada los registros según usuario y entrenamiento de forma que se muestren los gráficos en Grafana de forma correcta.

3.2.11 Firebase

El servicio Firebase dispone entre todas sus herramientas de Firebase Realtime Database. Una base de datos NoSQL. Esto permite crear esquemas no relacionales e interactuar de diferente forma entre el cliente y el servidor. Los datos se almacenan en formato JSON y pueden ser sincronizados en tiempo real entre el servidor y los distintos clientes. Esto permite ganar mucha agilidad a la hora de enviar y recibir datos. Como el inicio de sesión y el registro son dos partes críticas se llegó a la conclusión que emplear Firebase sería muy beneficioso.

Emplear Firebase abre las puertas a poder implementar en un futuro el inicio de sesión automático de Google mediante la cuenta de Google que se disponga en el dispositivo. Al emplear ya Firebase, también es posible sacar provecho en un futuro del resto de herramientas que ofrece como las analíticas para el comportamiento de los usuarios, la distintas herramientas de calidad para comprobar el correcto funcionamiento de la aplicación, hosting, almacenamiento de archivos, etc.

Volviendo a Firebase Realtime Database, los datos se almacenan en forma de árbol JSON y como tal deben ir estructurados los distintos registros que se vayan almacenando. En este proyecto solamente se emplea para almacenar los perfiles de los usuarios con lo que su estructura es simple y clara.

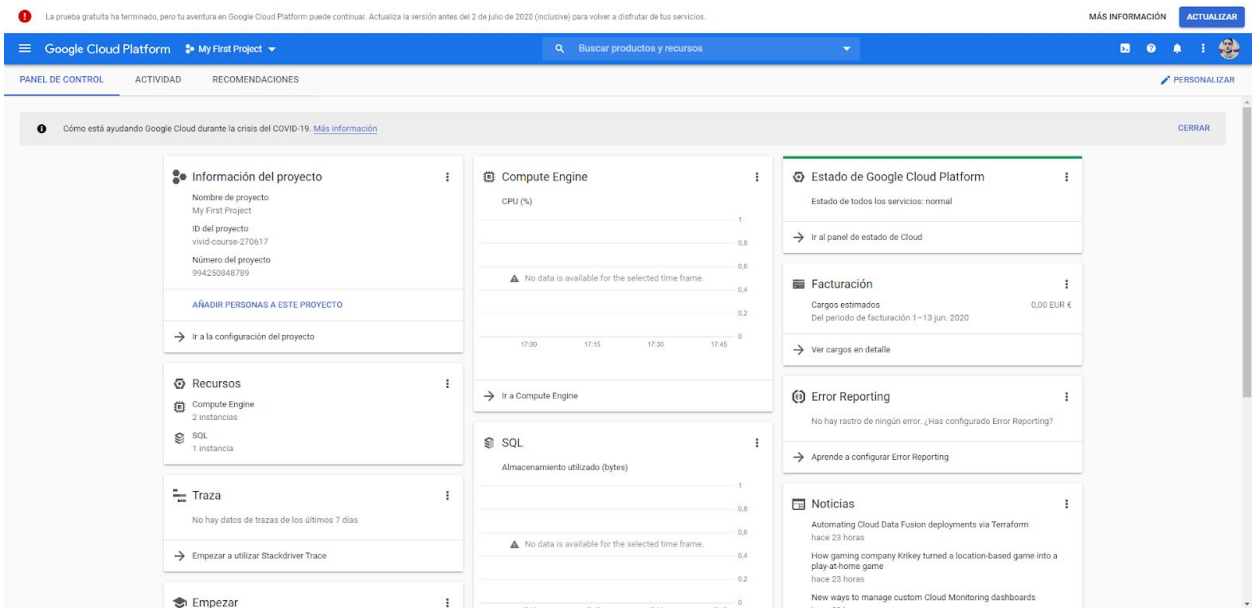


Tal y como se puede ver en la captura superior, la base de datos está conformada enteramente por la rama 'users'. Para acceder a dichos datos simplemente sería necesario apuntar a la URL

https://jostrack-f4bb0.firebaseio.com/users/ para poder acceder a todos los usuarios. Esto permite realizar el registro y los inicios de sesión. Para acceder a los registros de un determinado usuario se realizaría realizando la misma concatenación a la URL: https://jostrack-f4bb0.firebaseio.com/users/prueba1/, por ejemplo. Una vez conseguidos los registros, la aplicación ya los recupera o actualiza según sea necesario.

3.2.12 Google Cloud Platform / Microsoft Azure

Antes de escoger como nube a la empresa Neodigit, se hicieron algunas pruebas con Google Cloud Platform y Microsoft Azure como nube que integrara todos los servicios a nivel servidor que se requieren.
La primera prueba fue realizada con Google Cloud Platform. Google regala 300 USD en crédito para su plataforma.



Google Cloud Platform ofrece una gran variedad de servicios para casi cualquier propósito. El problema viene cuando se activan varios el precio puede llegar a dispararse agotando rápidamente dicho crédito y superando el crédito de prueba en pocos meses.

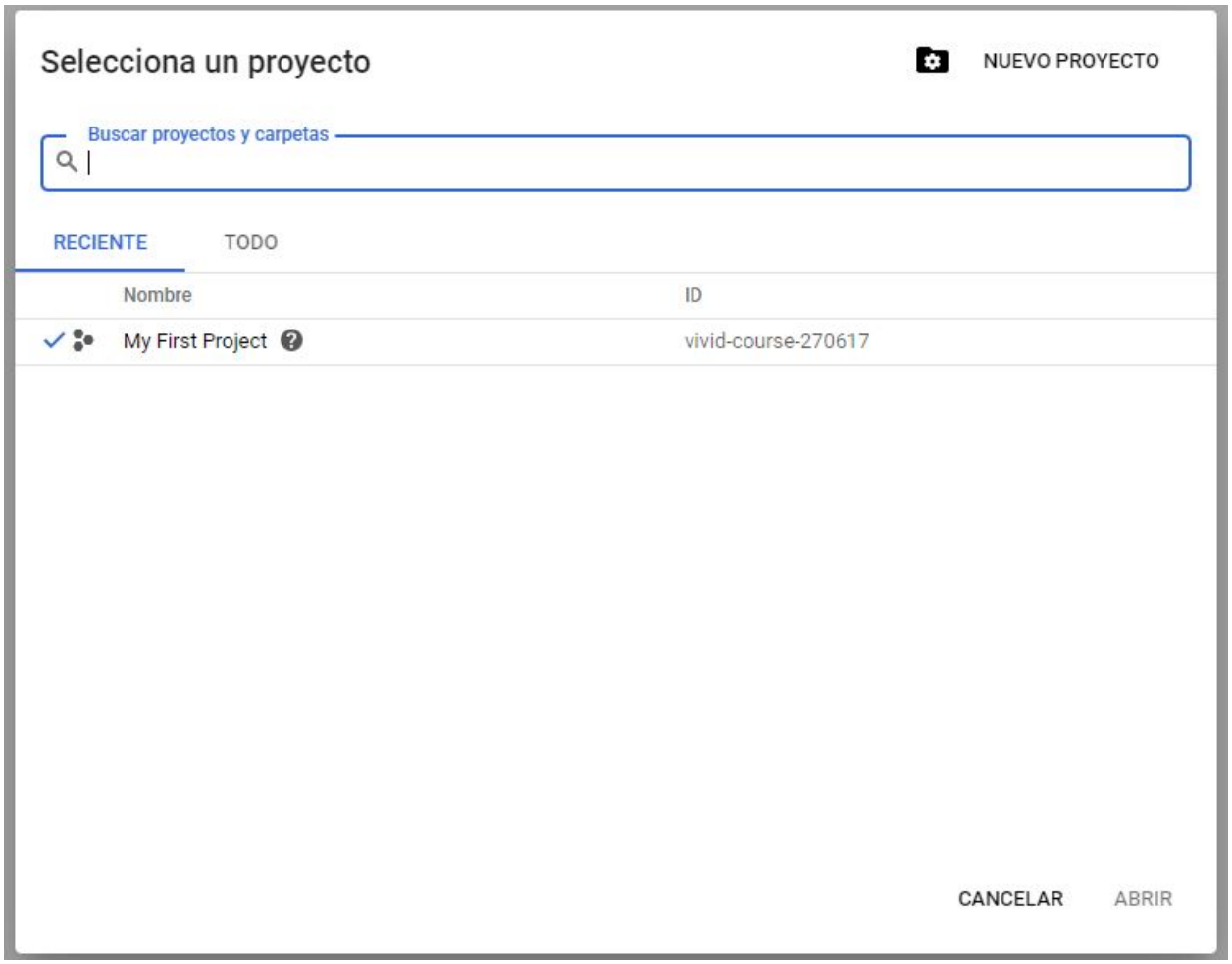
Al igual que con Neodigit, se tenía planificado desplegar el servicio de Cloud SQL y el servicio de Kubernetes para poder desplegar un servidor web y un contenedor con Grafana. Kubernetes es un sistema de código libre para la automatización del despliegue de distintos contenedores, ajuste de escala y manejo de aplicaciones en contenedores que fue originalmente diseñado por Google. Soporta diferentes entornos para la ejecución de contenedores, incluido Docker.

Producto	Tipo de recurso	Intervalo	Utilización	Importe (€)
Cloud SQL	DB standard Intel N1 1 VCPU running in EU (with 30% promotional discount)	1 may. - 31 may.	744 horas	46,37
Compute Engine	N1 Predefined Instance Core running in Americas	1 may. - 31 may.	1488.215 horas	43,37
Compute Engine	N1 Predefined Instance Ram running in Americas	1 may. - 31 may.	5580.808 gibibytes-hora	21,80
Compute Engine	Storage PD Capacity	1 may. - 31 may.	202.025 gibibytes-mes	6,34
Compute Engine	External IP Charge on a Standard VM	1 may. - 31 may.	1488.189 horas	2,74
Cloud SQL	Storage PD SSD for DB in EMEA	1 may. - 31 may.	10 gibibytes-mes	1,57
Compute Engine	Network Inter Zone Egress	1 may. - 31 may.	7.083 gibibytes	0,07
Cloud SQL	Storage PD Snapshot	1 may. - 31 may.	0.21 gibibytes-mes	0,02
Credit	External IPs will not be charged until July 1, 2020.	1 may. - 31 may.		-2,74
Compute Engine	Sustained Usage Discount	1 may. - 31 may.		-19,55
Credit	FreeTrial:Credit-01C78E-26BCD0-570777	1 may. - 31 may.		-99,99

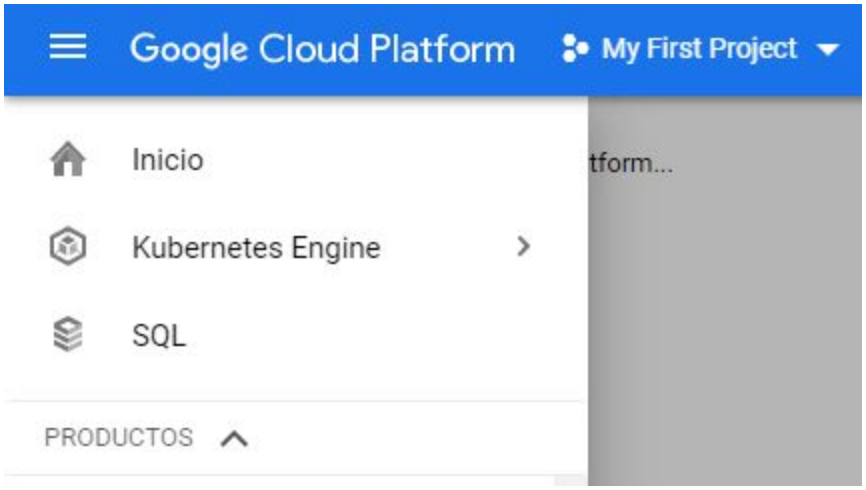
Subtotal en EUR	0,00 €
I.V.A. (21%)	0,00 €
Total en EUR	**0,00 €**

Como se puede apreciar en la factura superior, en el momento que se requiere el servicio de computación requerido para Kubernetes, el gasto empieza a dispararse. Por otra parte, el servicio de Cloud SQL también tiene un coste elevado teniendo en cuenta que solamente el servicio de Cloud SQL supera el precio de una máquina entera en Neodigit (29€ al mes).

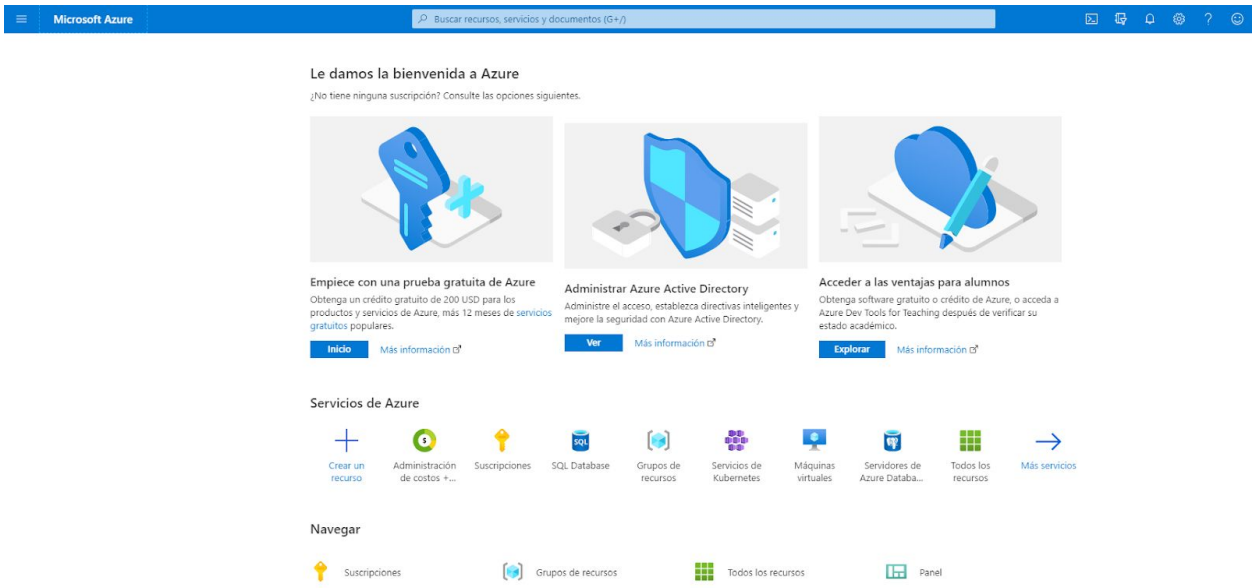
En el momento que se activa el servicio de Kubernetes, también se activan otros servicios necesarios y que siguen encareciendo la factura de dicha nube.



Para poder desplegar los servicios que se requieran es necesario crear un proyecto en primer lugar que englobe dichos servicios .
Una vez creado simplemente es necesario ir activando los servicios requeridos, en este caso Cloud SQL y Kubernetes.



En el caso de Kubernetes tiene gastos añadidos de computación, memoria y almacenamiento, entre otros, para para poder desplegar dicho servicio. Los gastos en detalle se pueden ver en la captura anterior. Dichas dependencias del servicio de Kubernetes se facturan de forma sencilla alertando el propio servicio que los requiere y pide el consentimiento del usuario para activarlo todo.



Por otra parte, Microsoft también ofrece una prueba de Azure de 170 USD en crédito para sus servicios o 12 meses, lo que se agote antes.
Microsoft ofrece algunos servicios parecidos a los de Google Cloud Platform entre ellos un servicio de SQL Database y Servicio de Kubernetes los que se requieren para desplegar la parte cloud del trabajo. Entre los servicios que ofrece Azure hay algunos enfocados a Windows como el Active Directory y permite desplegar directamente máquinas virtuales enteras.

Por desgracia y al igual que ocurrió con Google Cloud Platform, hay ciertos servicios que no consumen crédito, otros que consumen muy poco pero hay otros que disparan el gasto.
En este caso, conociendo lo que había ocurrido con Google Cloud Platform traté de desplegar los mínimos servicios posible, para ello se desplegó una máquina virtual donde instalar docker y un servidor web para poder montar todos los servicios que se requieren.

Si bien el gasto fue menor que el que se acumuló en Google Cloud Platform, al disponer de menos crédito, la prueba se agotó demasiado temprano.

Ferran
N.º de factura E0100AQ9YV

Microsoft Azure

Gastos de uso

Nombre	Tipo	Origen	Región	Consumido	Incluido	Facturable	Tasa	Valor
IoT Hub		Unidad S1 para Estándar		0,3871	0,0000	0,3871	21,0825	8,16
Virtual Machines	Serie Dv2/DSv2	D2 v2/DS2 v2	Oeste de Europa	529,2167	0,0000	529,2167	0,1147	60,69
Virtual Network	Direcciones IP	IP pública estática estándar		260,0000	0,0000	260,0000	0,0042	1,10
Log Analytics		Ingesta de datos	Oeste de Europa	2,2946	0,0000	2,2946	2,5215	5,79
Storage	Discos administrados SSD premium	Discos P10	Oeste de Europa	0,7263	0,0000	0,7263	18,2827	13,28
Load Balancer	Estándar	Datos procesados		0,1624	0,0000	0,1624	0,0042	0,00
Load Balancer	Estándar	Reglas de LB y reglas de salida incluidas		264,8000	0,0000	264,8000	0,0211	5,58

Subtotal	94,60
Total general	**94,60** EUR

Azure permite configurar según las prestaciones que se busquen cada servicio que se quiera activar. Aún así, tal y como se puede ver en la factura de marzo, el precio de contratar una máquina virtual es muy elevado y el período de prueba no ha permitido desarrollar todo el trabajo y mucho menos mantenerlo hasta su finalización.

Como conclusión de ambas nubes saco en claro que son dos solución muy robustas que ofrecen servicios para cualquier tipo de problema. Pero en caso de no haber sido utilizados antes la curva de aprendizaje y el coste pueden ser demasiado elevados si no se conocen dichas plataformas de antemano.

Para un proyecto de bajo presupuesto o que todavía debe validarse en el mercado considero que es mejor buscar otras alternativas como la solución que ofrece Neodigit y en un futuro valorar una migración a otro servicio más robusto y escalable según los requerimientos.

3.3. Pruebas

En este apartado se van a describir las distintas pruebas que se han realizado para comprobar el correcto comportamiento de todo el sistema que conforma el proyecto.

Para ello, en primer lugar se pasará a realizar distintas pruebas sobre cada implementación que se realice y su interacción con los componentes que debería comunicarse o actuar en conjunto. Cabe recordar que en el proyecto se conectan diferentes dispositivos físicos como el dispositivo Arduino y sus distintos módulos que lo acompañan con un dispositivo Android el cual debe acceder a diferentes sensores integrados en el propio dispositivo para finalmente enviar los diferentes datos a un servidor de bases de datos como también emplear Firebase para la autenticación de los usuarios.

Finalmente se realizan pruebas de integración, es decir, de cómo se integra el sistema entero en una prueba completa para verificar que el sistema va a actuar dentro de lo esperado.

3.3.1 De sistema

El sistema diseñado, debido a los distintos elementos que lo componen requiere de diferentes pruebas que verifiquen que cada parte funcione de forma correcta.

Pruebas de sensores en dispositivo Android

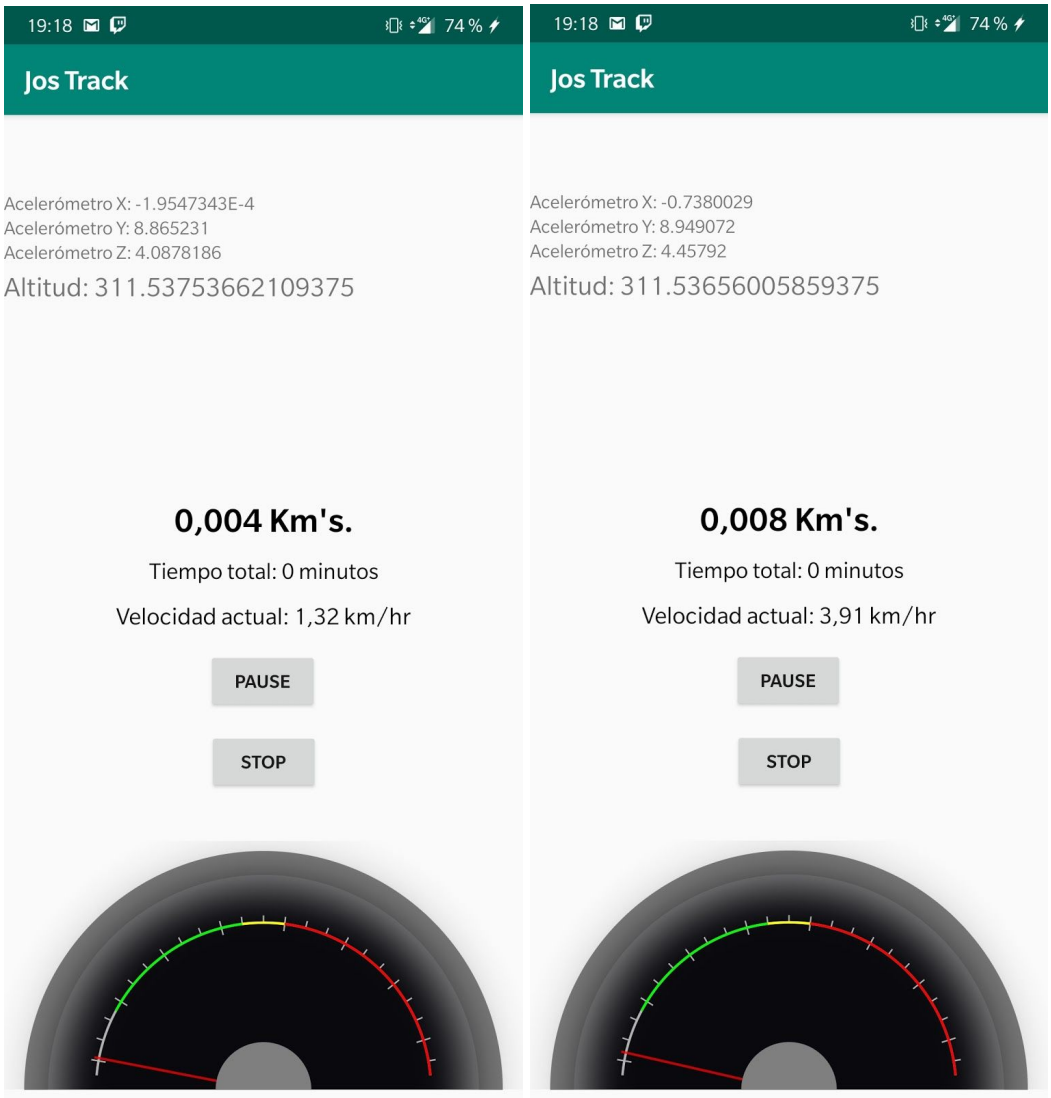
Para comprobar que todos los sensores dan la información correcta, se aprovecha directamente la versión preliminar de la aplicación Android. Esta versión todavía no cuenta con el indicador del acelerómetro pero mediante cajas de texto se pueden visualizar los valores y comprobar que estos son correctos.

El acceso a la información del acelerómetro es la más rápida de todas las que se precisan. Una vez la pantalla está iniciada ya se dispone de los valores en pantalla y se actualizan con una alta tasa de refresco según se mueva o se incline el smartphone.

Por otra parte, tanto la altitud, la velocidad y la distancia recorrida requieren encontrar los satélites GPS y encontrar la ubicación empleando dichos satélites GPS como también empleando las redes móviles y el WiFi para mejorar la velocidad de posicionamiento y su precisión. Esto hace que tarden unos segundos en aparecer en pantalla, pero una vez se muestran los valores en pantalla son bastante precisos. Como detalle importante, debido a la precisión del GPS, la velocidad puede no ser del todo exacta o marcar que se está en movimiento cuando realmente no se está moviendo. Se trata de un error bastante cercano al cero, en caso de estar totalmente detenido la velocidad se mantiene por debajo de los 3km/h y que no debería ser un inconveniente.

En este aspecto, es importante saber también que dentro de los edificios, la cobertura GPS tiene problemas y que por tanto, los problemas de imprecisión podrían verse aumentados. En

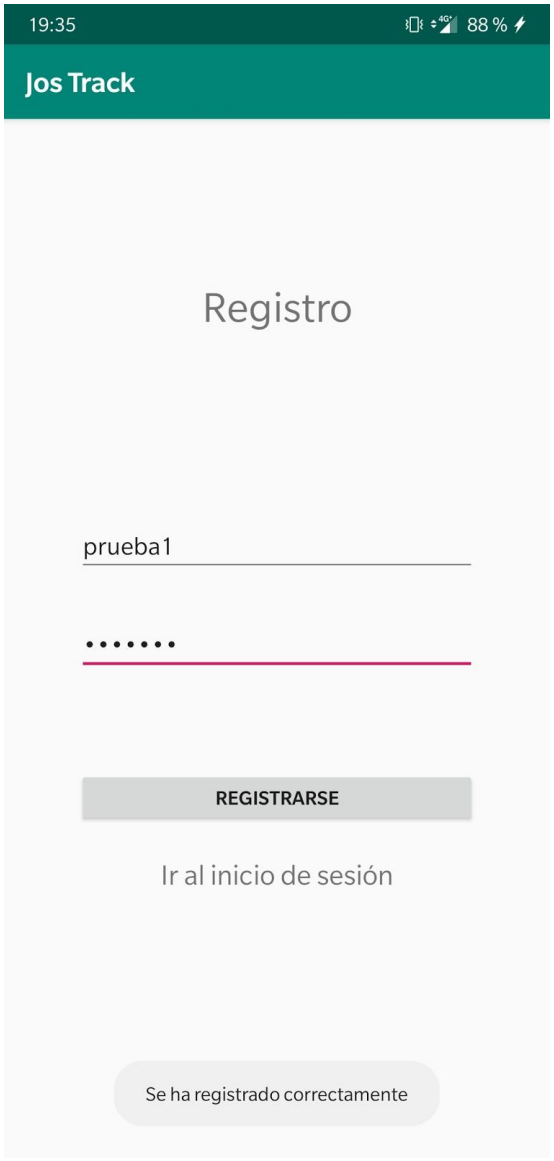
todo caso, el propósito con el que fue pensada la aplicación no era el de emplearse dentro de un edificio sino al aire libre.



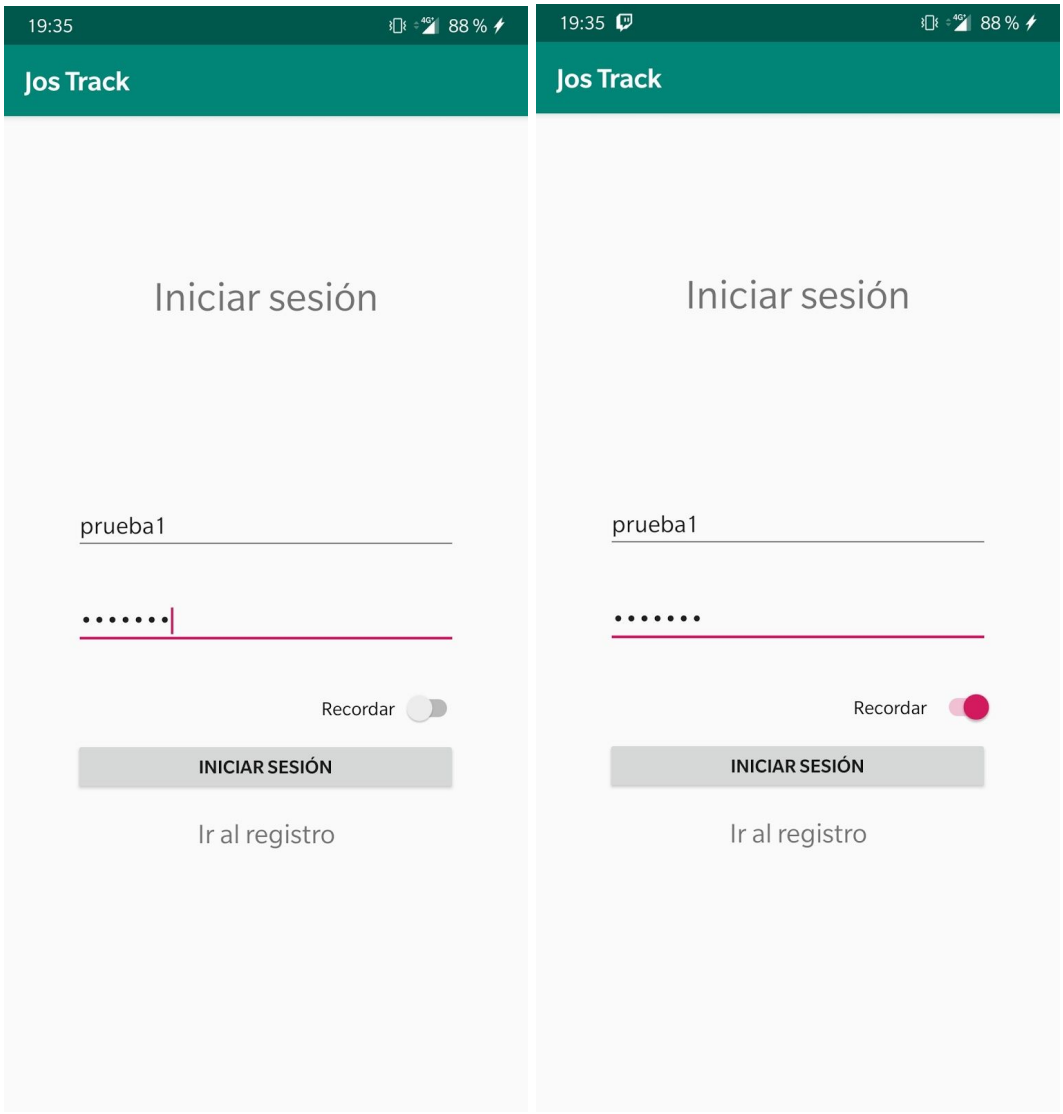
Pruebas con Firebase

En este punto, lo importante era comprobar por una parte, que los usuarios pueden registrarse, es decir, pueden introducir un nombre de usuario y una contraseña válidos y que no estén ya registrados.

Posteriormente también requieren iniciar sesión con las credenciales introducidas durante el registro para finalmente poder configurar sus perfiles y emplear con normalidad la cuenta.

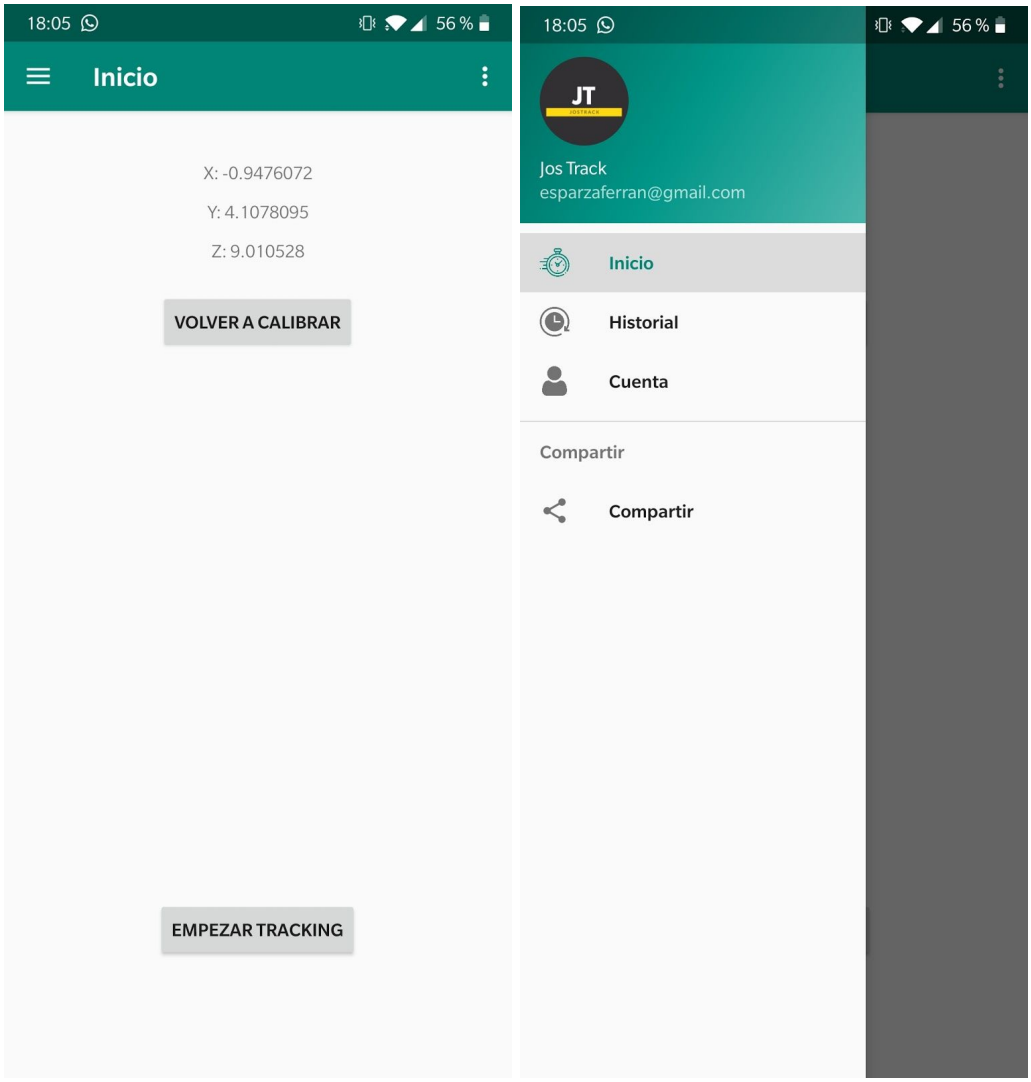


Una vez se accede al registro, el usuario debe introducir un nombre de usuario de al menos cinco caracteres, que no esté ya registrado. También debe introducir una contraseña para su cuenta de libre elección. Una vez se pulsa sobre el botón 'REGISTRARSE' se envía la petición al sistema de base de datos en tiempo real de Firebase y este comprueba el nombre de usuario antes de añadirlo. En caso de que dicho nombre de usuario esté ya registrado, devolverá error a la aplicación Android y esta lo notificará al usuario con una notificación flotante indicando que dicho usuario ya está registrado. En caso de ir todo el proceso de forma correcta, se recibirá en la aplicación una confirmación y esta lo notificará al usuario indicando que se ha registrado correctamente.



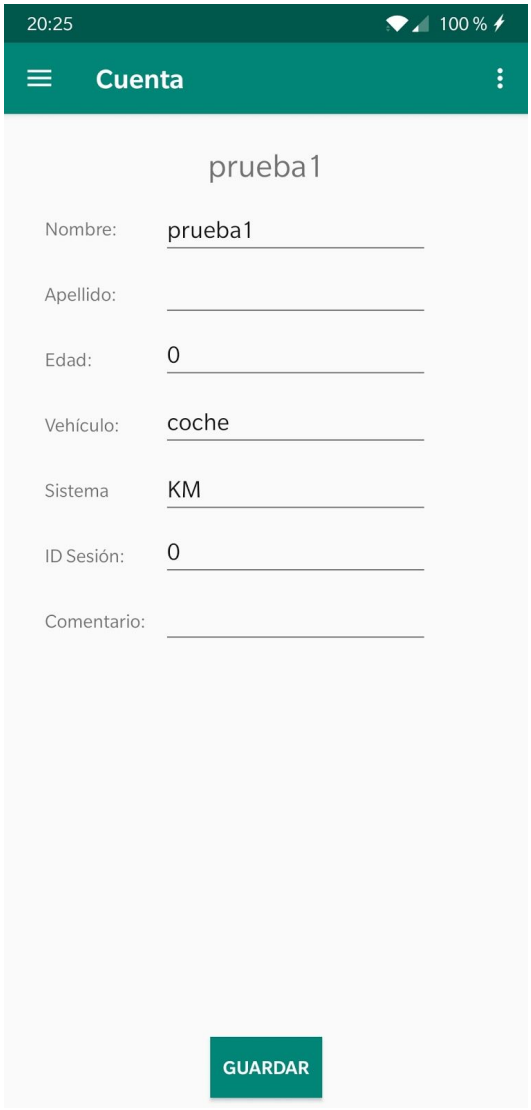
Una vez ya se dispone de una cuenta con la que iniciar sesión, el usuario ya puede tratar de iniciar sesión. Para ello este introduce sus credenciales anteriormente registradas y la aplicación comprueba junto con Firebase si dicho usuario existe y en tal caso si la contraseña también coincide. Si se cumple todo esto la aplicación almacena las credenciales para poder ser empleadas más adelante con otras funciones y lanza la pantalla principal de la aplicación.

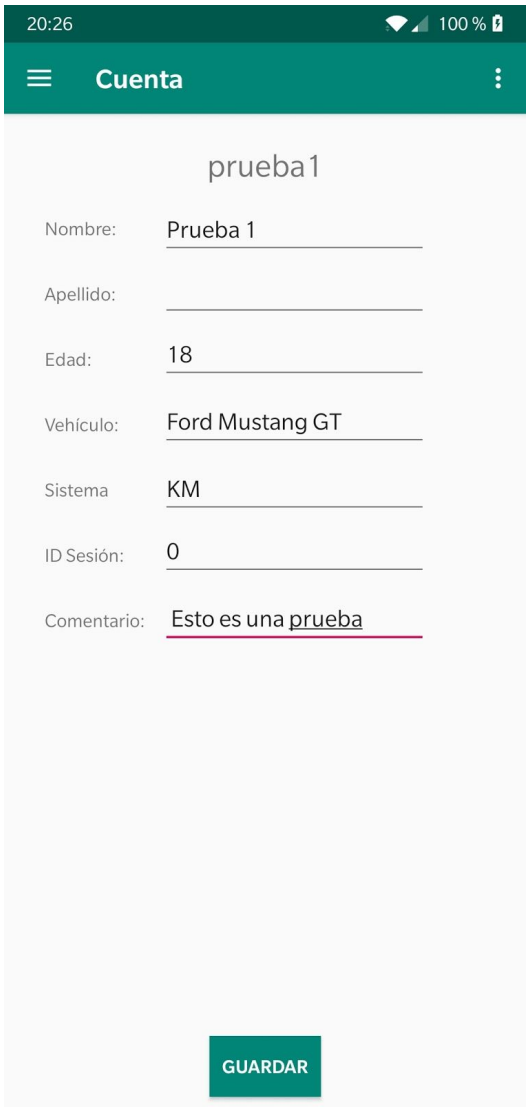
En caso de no querer iniciar cada vez, se dispone de un slider que puede ser activado antes de iniciar sesión y con el que, en caso de estar marcado, la aplicación comprueba cada vez que se abre las credenciales de forma automática e inicia sesión sin necesidad de que el usuario vuelva a introducir las credenciales nuevamente.



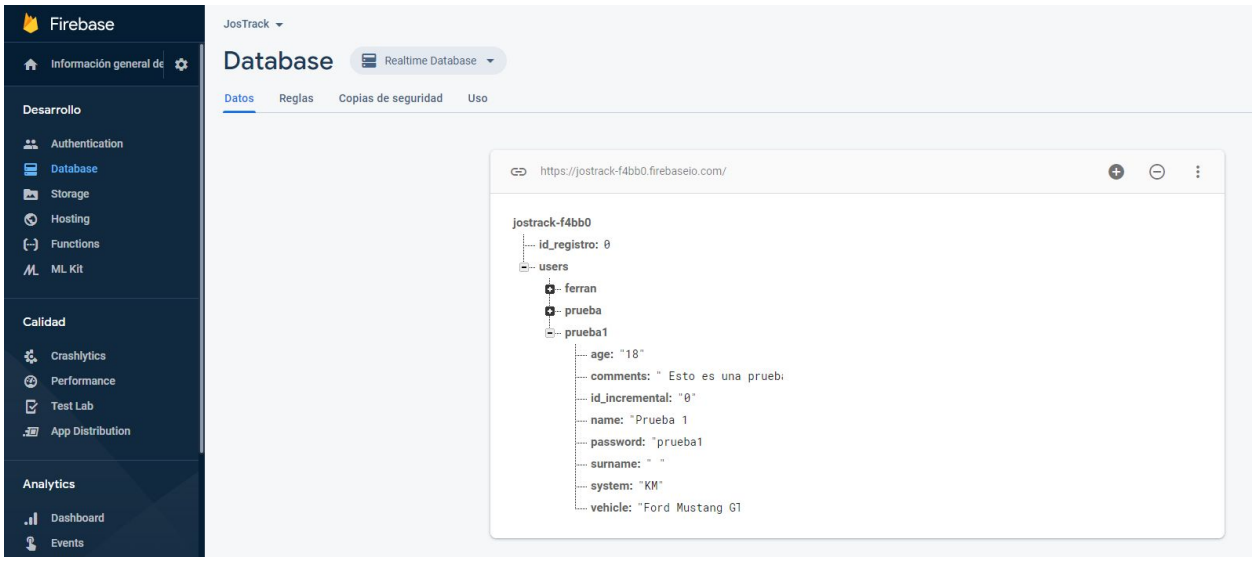
Si todo el proceso anterior se ha llevado de forma satisfactoria se lanza la pantalla principal desde donde se puede navegar por toda la aplicación. Entre sus funciones, permite cerrar la sesión mediante el menú superior derecho. En caso de pulsar dicho botón, la aplicación borra las credenciales de su memoria y vuelve al inicio de sesión para que el usuario vuelva a iniciar sesión con su cuenta o con otra.

Pero respecto a la funcionalidad de Firebase, también se puede acceder a la pantalla de Cuenta donde se puede seguir interactuando con el perfil del usuario y por tanto se sigue trabajando con Firebase.





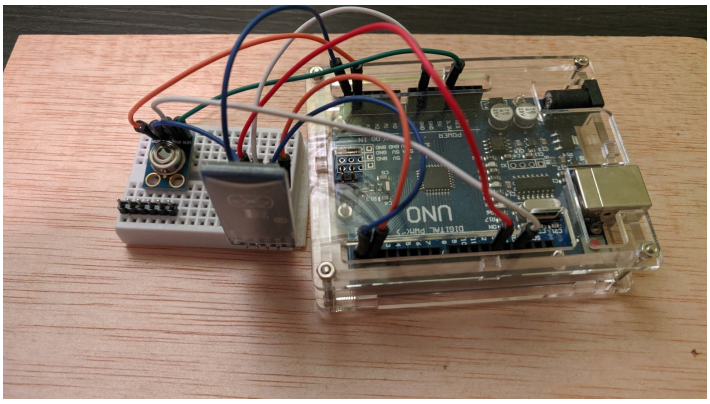
En la pantalla 'Cuenta' se puede completar y modificar el perfil de cada usuario como se puede apreciar en las capturas superiore. Una vez se han realizado los cambios pertinentes, el usuario debe pulsar el botón guardar para que se apliquen los cambios de forma efectiva.



Para comprobar que realmente se ha creado el perfil y/o se han llevado a cabo los cambios introducidos en el perfil desde la app se puede comprobar desde Firebase.
Accediendo a la ruta Database → Realtime database se puede acceder a la base de datos y comprobar los perfiles introducidos. En este caso, se puede apreciar que se ha creado correctamente el usuario y que dispone de toda su información tal y como se ha introducido en la aplicación Android. Estos cambios, se pueden ver a los pocos segundos y en caso de estar ya abierto Firebase se actualizan sin necesidad de refrescar la página.

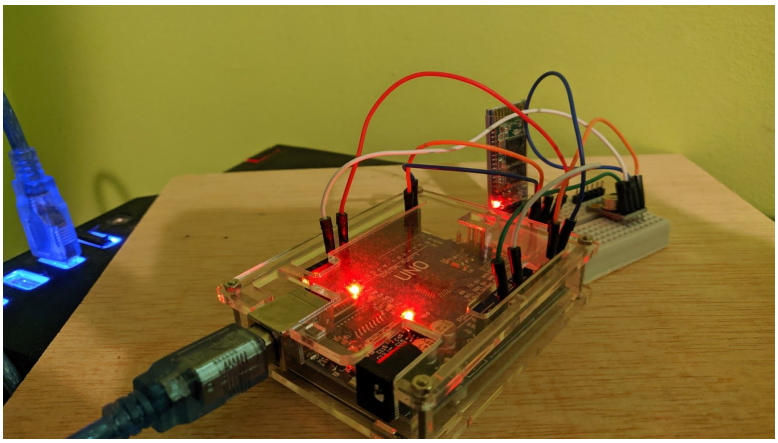
Pruebas con Arduino

En este punto, ya se contaba con el montaje básico y el script de la placa de desarrollo Arduino pero no se había integrado todavía la funcionalidad con el Bluetooth en la aplicación Android. Por tanto, para comprobar en primera instancia que el dispositivo Arduino funcionaba correctamente se procedió a utilizar una aplicación de terceros llamada 'Arduino bluetooth controller' disponible en Google Play Store de forma gratuita. Esta app permite conectarse mediante Bluetooth con dispositivos Arduino que cuenten con Bluetooth como puede ser el módulo HC-05 o el HC-06, como es el caso.



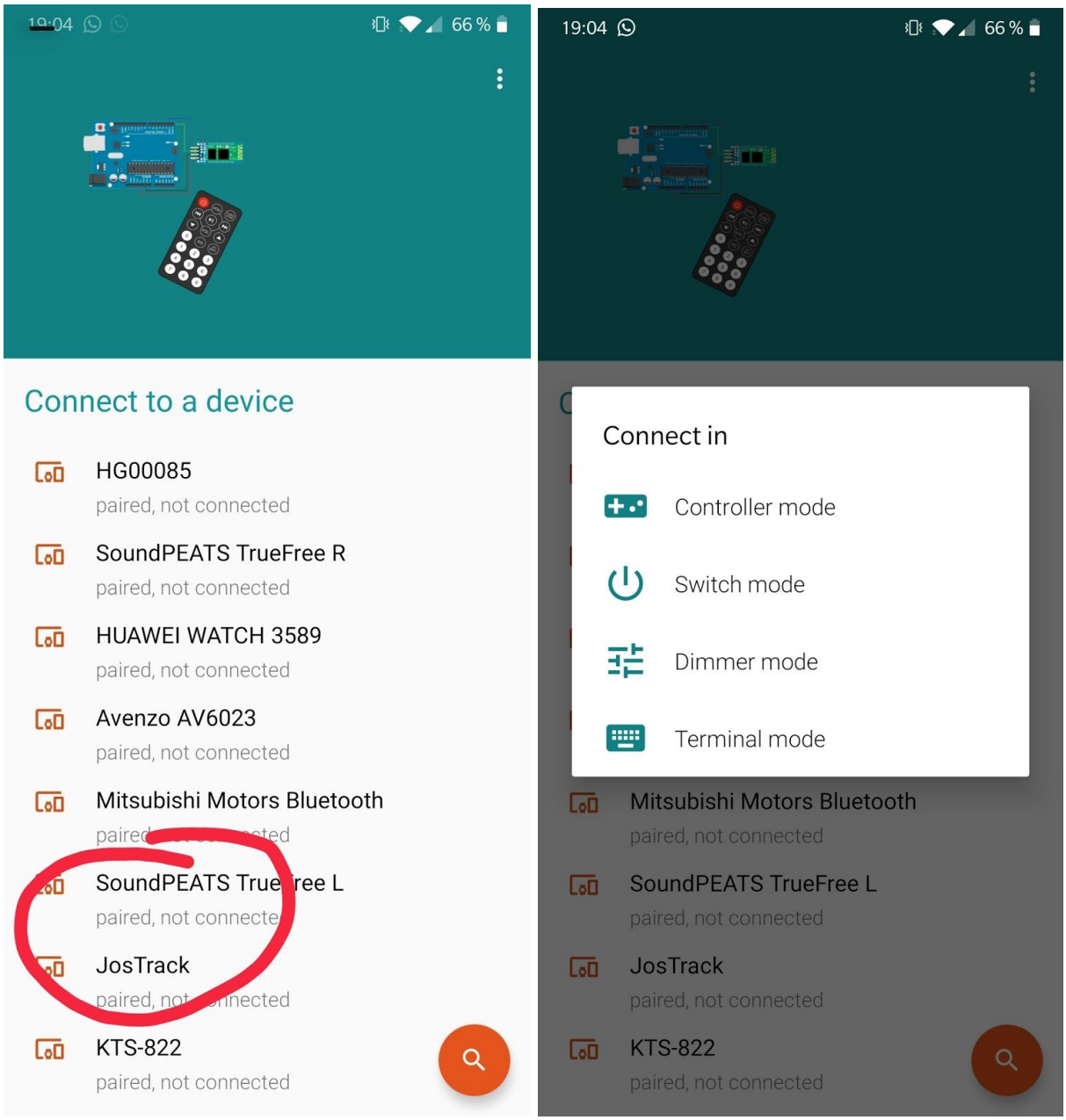
Una vez realizado el montaje el cual es bastante sencillo ya que tanto el sensor como el módulo Bluetooth no requieren de componentes extra más allá del correspondiente cableado para su correcto funcionamiento.

Posteriormente simplemente hace falta alimentar el dispositivo y cargar el script para que se ejecute.

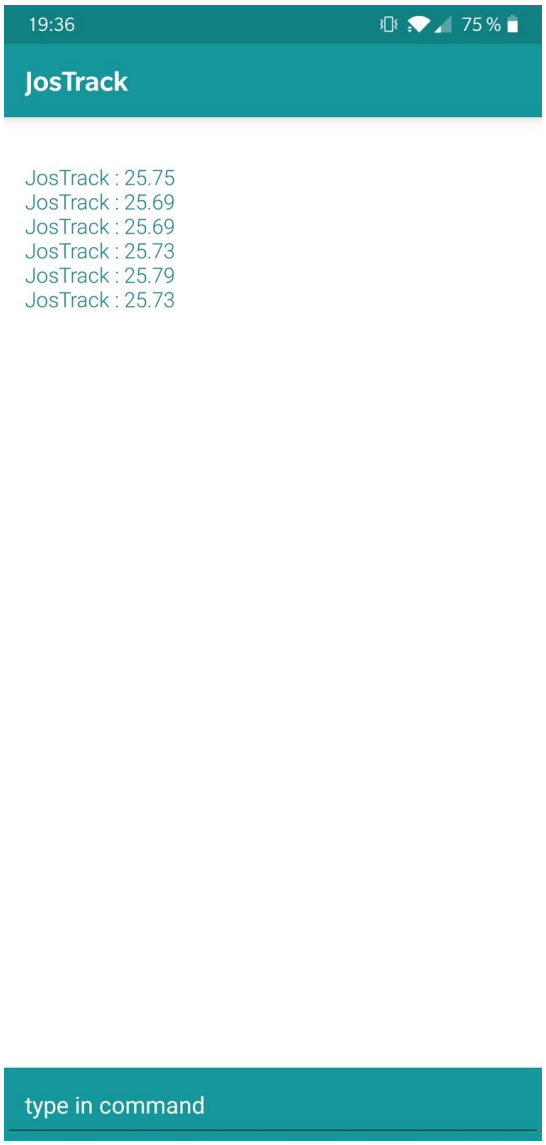


Con el script cargado ya se podrá realizar la conexión entre el dispositivo Android y el dispositivo Arduino al cabo de unos breves segundos. Esto es debido a que Arduino requiere configurar los parámetros AT para el Bluetooth. Una vez se configure todo, el led asociado al pin 13 incorporado en la propia placa de desarrollo se quedará encendido y el led del módulo bluetooth HC-06 empezará a parpadear indicando que está listo para poder ser emparejado.

Para realizar el test por parte del dispositivo Android, se accede a Ajustes → Bluetooth → Emparejar nuevo dispositivo y seleccionar 'JosTrack'. Este será el dispositivo Arduino. Tras introducir el pin correctamente el smartphone ya tendrá registrado el dispositivo para poder ser empleado con la app 'Arduino Bluetooth Controller' que ya puede ser abierta.



Una vez abierta la app, se listan los dispositivos Bluetooth que haya vinculados con el smartphone, en este caso, se selecciona 'JosTrack' para conectarse con el dispositivo Arduino y en la ventana emergente se selecciona 'Terminal mode' para poder operar con el dispositivo como si se tratase de la terminal que dispone el Arduino IDE.
Una vez pulsada la opción, el led del módulo HC-06 que hasta este punto estaba parpadeando se queda encendido de forma fija indicando que está conectado correctamente y en la app se abre una terminal.

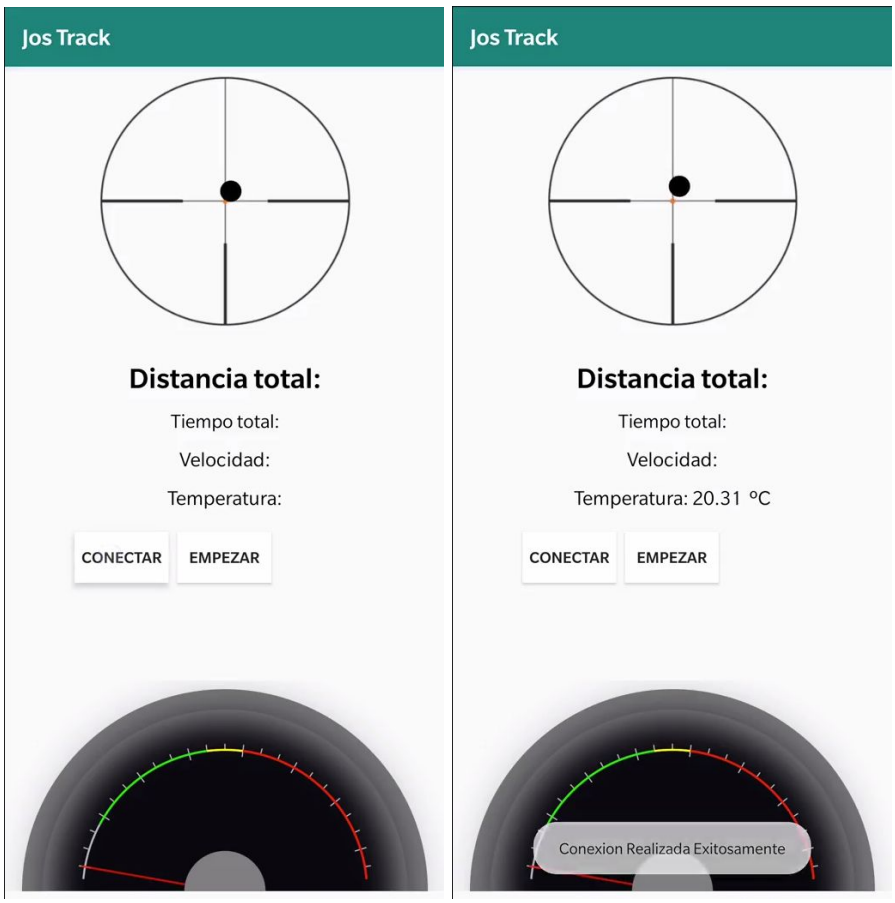


Una vez abierta la terminal, el dispositivo Arduino envía la temperatura que registra el sensor para un objeto concreto cada segundo y así es como se recibe en Android.
Se pueden poner objetos de diferente material para comprobar como la temperatura que registra el sensor es diferente.

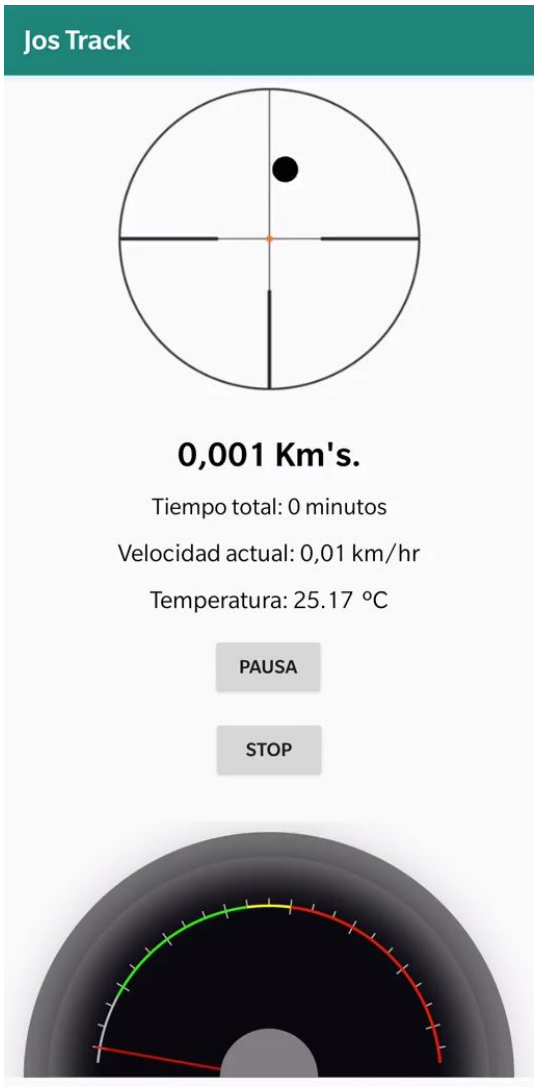
Con estas pruebas se puede llegar a la conclusión de que el dispositivo Arduino está realizando todas las tareas que se esperan de él de forma correcta y estos son capaces de llegar a un dispositivo Android. Esto, además de ser un avance en el proyecto, facilitará la implementación de la conexión Bluetooth en la aplicación Android.

Pruebas de conexión entre Arduino y Android

Una vez se dispone ya de la implementación del Bluetooth en Android es momento de comprobar que funciona correctamente y que interacciona de forma correcta con el dispositivo Arduino. Con el test anterior, se han podido descartar posibles problemas de incompatibilidad o de que realmente lo que funcione mal sea Arduino. En este caso se focaliza el test en cómo se conectan ambos dispositivos mediante la aplicación desarrollada y como esta recibe la información.



Para comprobar si la conexión se realiza correctamente se debe abrir la aplicación en el dispositivo Android acceder a la pantalla 'Tracking' y pulsar sobre el botón 'CONECTAR' tras unos breves segundos se mostrará un mensaje emergente indicando si la conexión se ha realizado de forma correcta o no tal y como se puede ver en las capturas superiores. Al estar ya en funcionamiento el dispositivo Android antes de realizar la conexión, este ya tiene datos sobre temperatura y una vez se realiza correctamente la conexión ya debería ser posible empezar a ver datos sobre temperatura en pantalla.



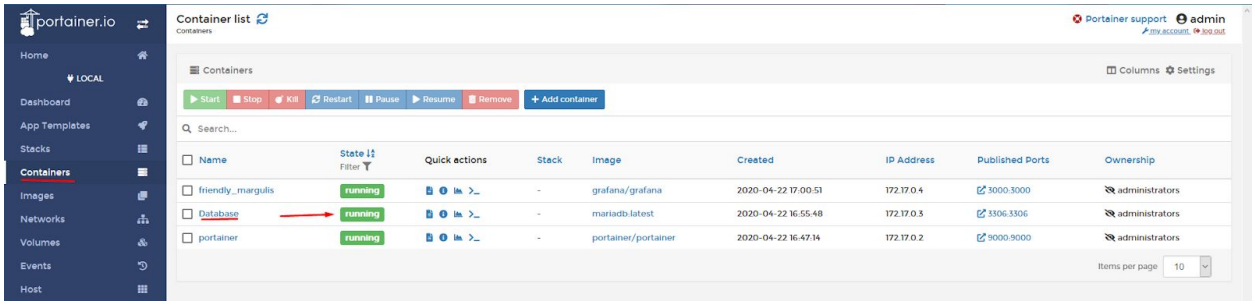
Para empezar el tracking como tal sería necesario pulsar sobre el botón 'EMPEZAR'. En este punto la aplicación lanza la localización GPS, empieza a registrar la duración total y a guardar los datos que se van recopilando.

Como dato de interés, es posible emplear la aplicación Android sin necesidad de registrar la temperatura. Es decir, en caso de tener algún tipo de problema con el dispositivo Arduino o simplemente no querer usarlo, la aplicación puede seguir siendo usada sin registrar datos de temperatura.

Pruebas de despliegue de mariaDB

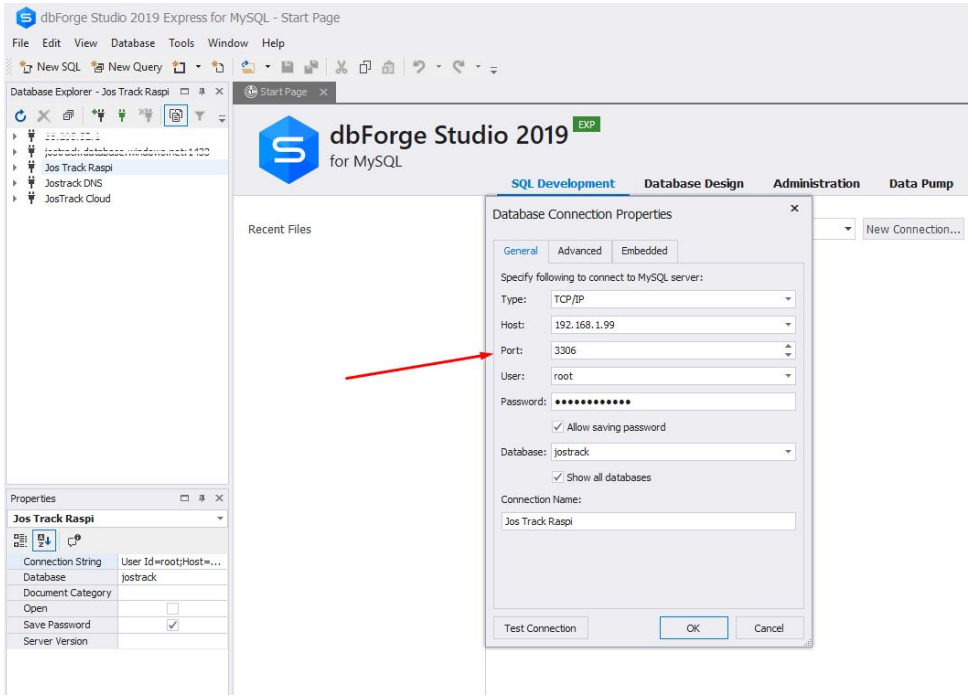
Como ya se ha mencionado anteriormente, mariaDB se ha decidido desplegar empleando la tecnología de contenedores Docker. La cual hace todo el proceso mucho más rápido y sencillo al venir todo lo necesario incorporado en una sola imagen. Para facilitar todavía más el proceso, se ha instalado Portainer para la gestión de las imágenes y contenedores, con lo que,

una vez desplegados, simplemente es necesario acceder al apartado 'Containers' de Portainer para comprobar el estado de los distintos contenedores.

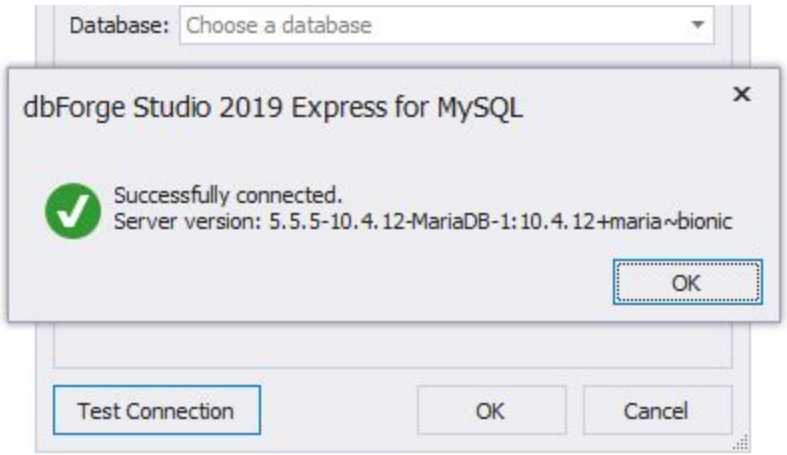


Tal y como se puede ver en la captura, los tres contenedores, entre ellos mariaDB están operativos. Esta es una buena forma de comprobar si los contenedores están operativos pero queda por comprobar que se haya realizado bien la configuración de red y que dicho contenedor puede recibir conexiones desde el exterior. Docker dispone de diferentes redes virtuales para configurar los contenedores. Según se configuren los contenedores, estos tendrán conexión al resto de contenedores solamente o también a Internet.

Para comprobar que la conexión es posible y además gestionar la base de datos se emplea el software gratuito DB Forge Studio Express para realizar las conexiones con la base de datos y gestionar la base de datos.



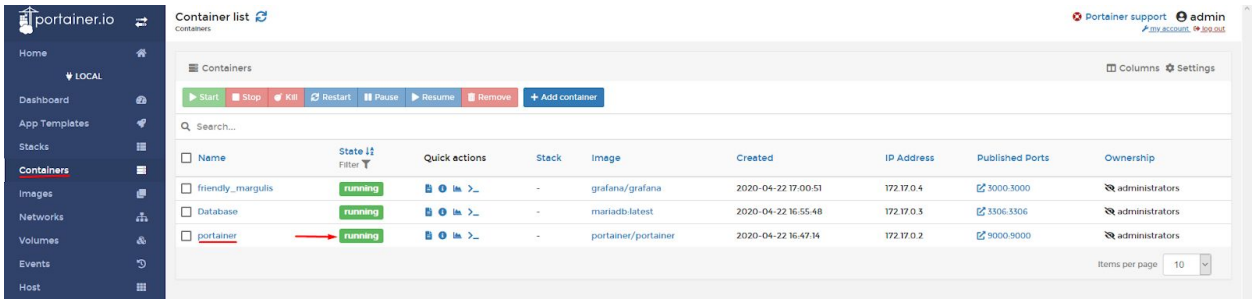
Como punto importante a la hora de crear la conexión con la base de datos es asegurarse que el puerto es el mismo que tiene el contenedor de mariaDB. Si no se le asigna el puerto concreto a la hora de crear el contenedor, Docker puede asignarle otro puerto y tener problemas de conexión por no apuntar posteriormente al mismo puerto. En este caso se ha empleado el 3306, el puerto por defecto para las conexiones con MySQL/mariaDB.



Con un 'Test Connection' se puede comprobar que efectivamente se tiene conexión con la base de datos. Con lo que solo queda pulsar 'OK' para guardar la conexión. Este test es importante realizarlo, además de la comprobación de los puertos empleados, tanto en el servidor de prueba como posteriormente en el servidor de producción para evitar posibles contratiempos por problemas en el servidor y poder descartar que este falle en caso de haber nuevas incidencias.

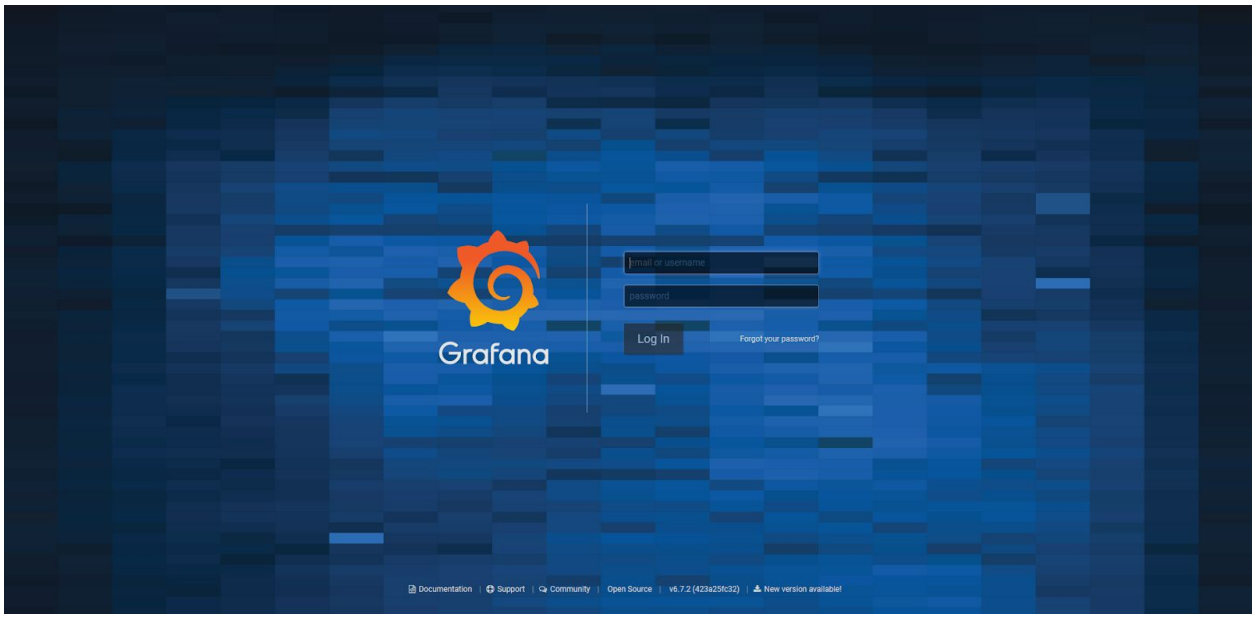
Pruebas de despliegue de Grafana

Como ya se ha mencionado anteriormente, Grafana se ha decidido desplegar empleando la tecnología de contenedores Docker. La cual hace todo el proceso mucho más rápido y sencillo al venir todo lo necesario incorporado en una sola imagen. Para facilitar todavía más el proceso, se ha instalado Portainer para la gestión de las imágenes y contenedores, con lo que, una vez desplegados, simplemente es necesario acceder al apartado 'Containers' de Portainer para comprobar el estado de los distintos contenedores.



Tal y como se puede ver en la captura, los tres contenedores, entre ellos Grafana están operativos. Esta es una buena forma de comprobar si los contenedores están operativos pero queda por comprobar que se haya realizado bien la configuración de red y que dicho contenedor puede recibir conexiones desde el exterior. Docker dispone de diferentes redes virtuales para configurar los contenedores. Según se configuren los contenedores, estos tendrán conexión al resto de contenedores solamente o también a Internet.

Para comprobar que efectivamente se puede acceder y usar grafana se debe emplear el navegador para acceder a la dirección IP del servidor o dominio junto al puerto 9000 tal y como puede verse en la captura anterior de Portainer.



Una vez cargue la página, la primera vez suele tardar un poco más de lo normal, ya se puede observar que se tiene conexión con Grafana y está listo para ser usado.

Pruebas de conexión entre PHP y mariaDB

Para facilitar el proceso de enviar datos a la base de datos desde la aplicación Android se decidió emplear como intermediario un script PHP que facilitase la transmisión de los datos sin necesidad de emplear conectores Java con propensión a fallar.
Dicho script se encuentra alojado en el servidor Apache del Cloud disponible para ser usado por diferentes dispositivos.

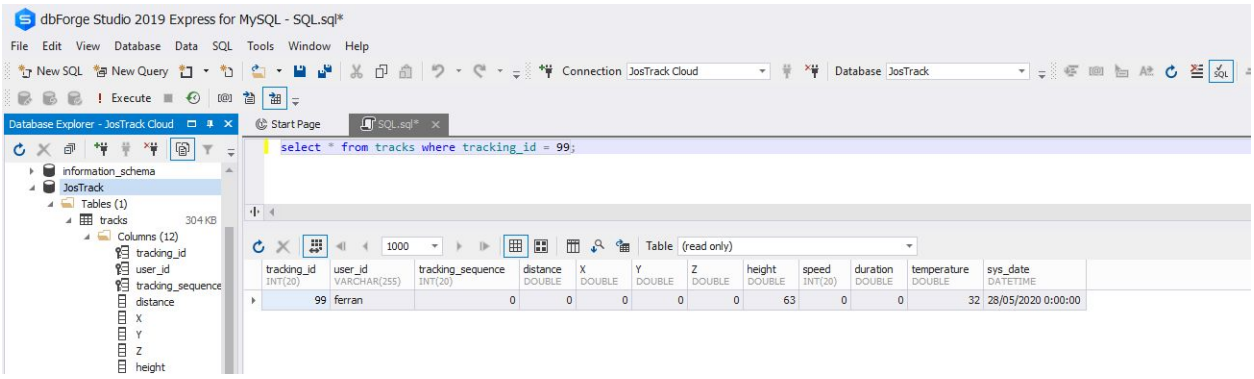
Para comprobar su funcionamiento, es necesario escribir la dirección IP seguido de los parámetros que requiere el script para poder ser insertados en la base de datos realizando un INSERT:

http://192.168.1.99/asd.php?tracking_id=99&user_id=ferran&tracking_sequence=0&distance=0&X=0&Y=0&Z=0&height=63&speed=0&duration=0&sys_date=2020-05-28&temperature=32

Como se puede ver en el enlace superior, se está apuntando al servidor de pruebas en local, concretamente al archivo php y se le pasan por parámetro los distintos campos que hay en la base de datos.
Si se trata de acceder a la dirección, el sistema responde con texto en el navegador indicando si la conexión ha sido correcta o si no ha sido correcta en caso de faltar algún campo o si se introduce de forma incorrecta.

Connected Successfully

Para verificar que el resultado verdaderamente es correcto se puede acceder a la base de datos mediante el DB Forge Studio Express y comprobar si se ha insertado correctamente la información.



Tal y como se puede observar, la información se ha insertado correctamente en la captura superior y la base de datos y el script PHP se encuentran listos para realizar pruebas de conexión entre Android y la base de datos y entre la base de datos y Grafana.

Pruebas de conexión entre Android y mariaDB empleando PHP

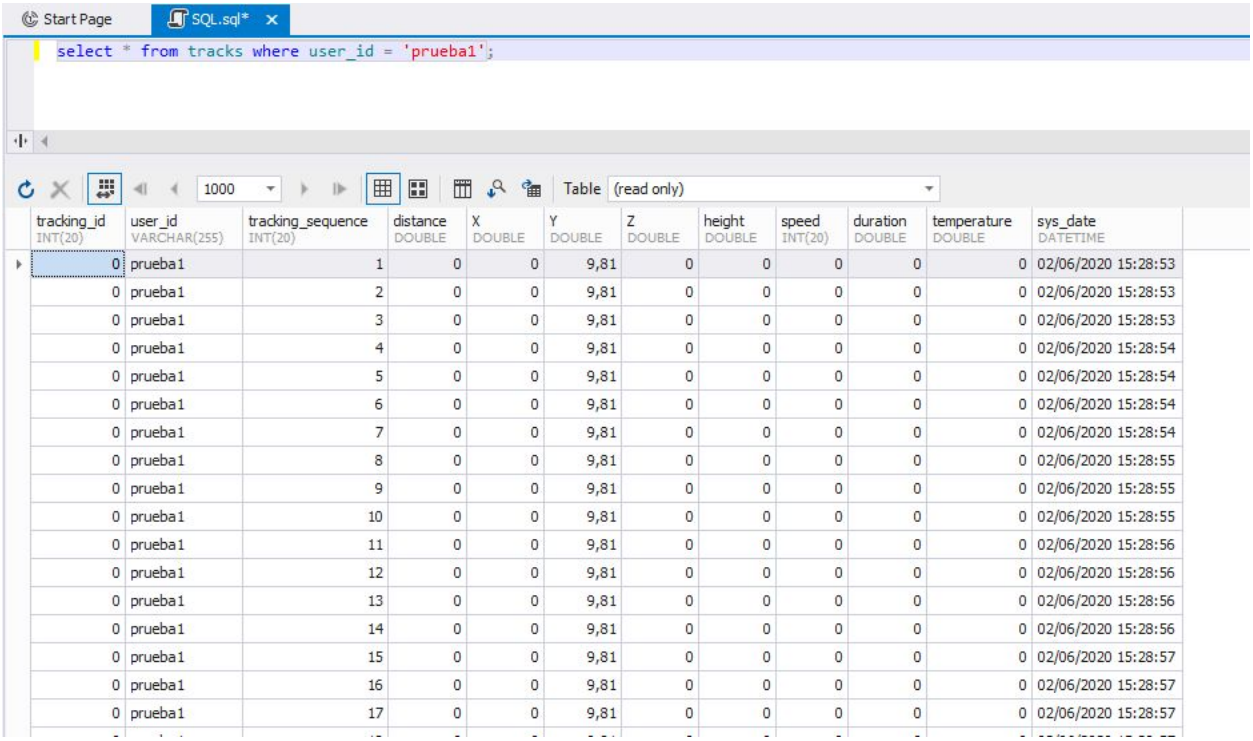
Una vez se ha comprobado que el resto de elementos de la conexión funciona, falta comprobar si las peticiones HTTP se realizan de forma correcta desde la aplicación Android.
Para esto el proceso es similar a las pruebas anteriores, salvo que la aplicación Android deberá estar creando diferentes URLs según la información que le llegue de los distintos sensores y antenas.

Para realizar las peticiones HTTP desde una aplicación Android, como ya se ha comentado anteriormente, se emplea la librería Volley.

```
996         protected void onPostExecute(String resultado){
997             Toast.makeText(context.get(), resultado, Toast.LENGTH_LONG).show();
998         }
```

Para comprobar si realmente se ha ejecutado de forma correcta por parte de la librería Volley se ha implementado este método, el cual solo se emplea para pruebas, con el que se visualiza mediante notificaciones de tipo Toast el resultado de la petición. Este método se ejecuta tras realizar actividades asíncronas mediante el método doInBackground() que es en este caso el encargado de realizar la petición en sí misma.

Tal y como se puede apreciar en la captura superior, la conexión se ha realizado de forma correcta y se ha procesado tal y como debía realizar la librería Volley.
Para asegurar que esto realmente es así, se consulta la base de datos para comprobar que efectivamente se han insertado los datos.



tracking_id INT(20)	user_id VARCHAR(255)	tracking_sequence INT(20)	distance DOUBLE	X DOUBLE	Y DOUBLE	Z DOUBLE	height DOUBLE	speed INT(20)	duration DOUBLE	temperature DOUBLE	sys_date DATETIME
0	prueba1	1	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:53
0	prueba1	2	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:53
0	prueba1	3	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:53
0	prueba1	4	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:54
0	prueba1	5	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:54
0	prueba1	6	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:54
0	prueba1	7	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:54
0	prueba1	8	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:55
0	prueba1	9	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:55
0	prueba1	10	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:55
0	prueba1	11	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:56
0	prueba1	12	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:56
0	prueba1	13	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:56
0	prueba1	14	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:56
0	prueba1	15	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:57
0	prueba1	16	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:57
0	prueba1	17	0	0	9,81	0	0	0	0	0	02/06/2020 15:28:57

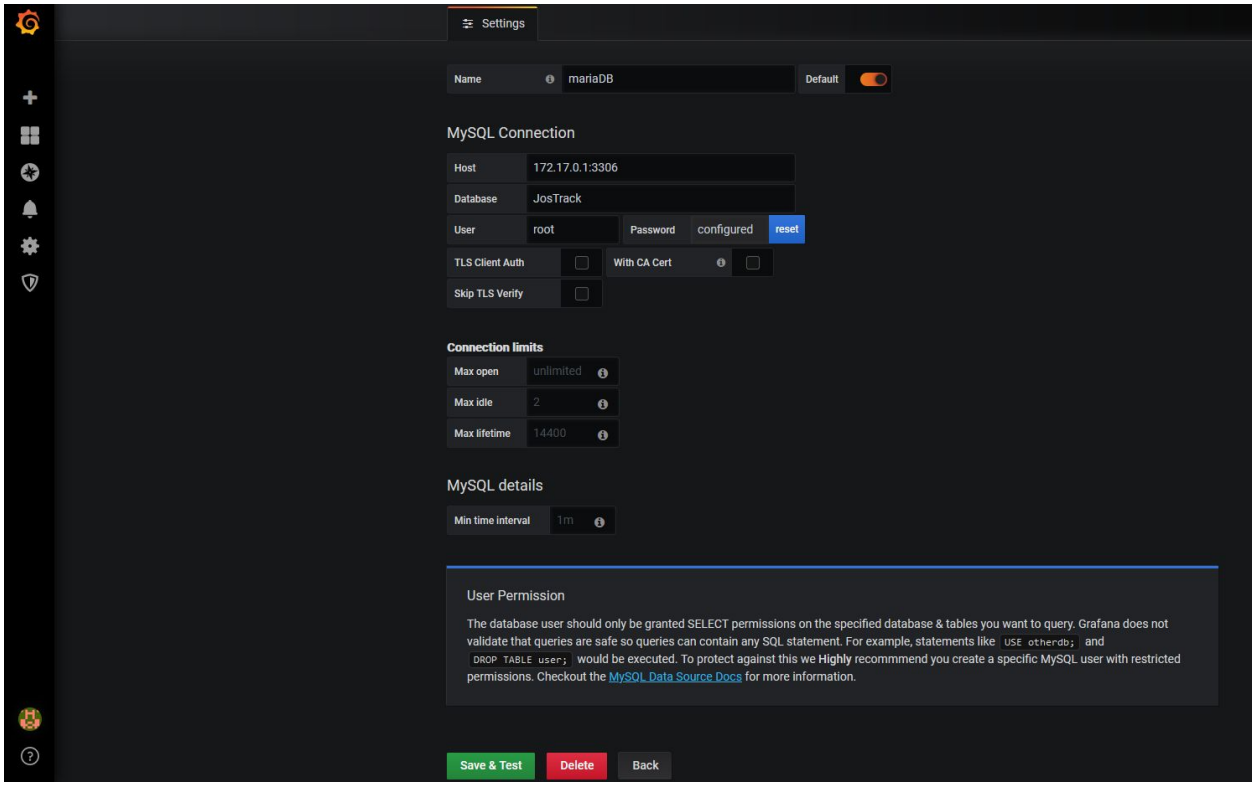
Para comprobar que efectivamente los datos son correctos se adjunta la captura de la aplicación superior para comprobar que efectivamente los datos que se han almacenado corresponden con la primera captura y con la calibración del acelerómetro correspondiente.

Cabe aclarar que esta prueba se ha realizado mediante el propio emulador de Android Studio y es por ello que los valores del acelerómetro son constantes en todos los registros. Con un acelerómetro real y teniendo en cuenta vibraciones del motor dichos valores pueden oscilar levemente. Así como también, si la prueba se realiza sin sujetar el smartphone a una superficie firme, sino que se sujeta con la propia mano, también van variando dichos valores.

Pruebas entre mariaDB y Grafana

Como última prueba, falta por comprobar la integración entre la base de datos y Grafana. Es decir, cómo se integran los datos en los dashboards que se creen.

Para esto, es necesario realizar la conexión con la base de datos:



Una vez se pulsa sobre 'Save & Test' se guarda la conexión y se comprueba que funciona correctamente. A partir de aquí ya se puede emplear para conseguir los datos de los dashboard.

Una vez se crea un dashboard y se añaden todos los elementos que se quieren mostrar junto a sus consultas SQL el resultado puede ser similar a este:

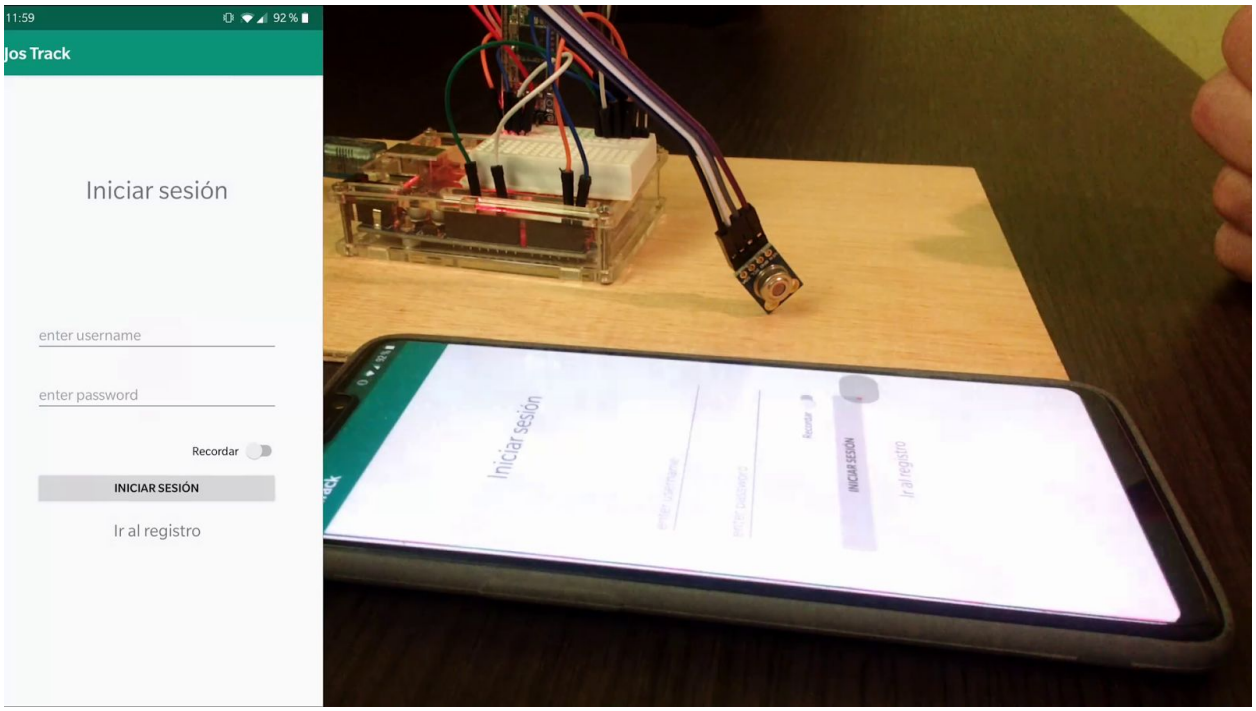
Como punto a destacar, se han añadido variables de Grafana para facilitar el uso de los dashboard. Como se puede apreciar en la parte superior derecha, se dispone de un buscador de usuario con el que introducir el nombre del usuario que se quiere analizar. Una vez se introduce, el desplegable que hay a su derecha lista todos los trackings que tiene realizados dicho usuario pudiendo seleccionar uno o varios para verlos en los gráficos.

Los datos que se muestran en dicho dashboard son los que se han recopilado en la prueba anterior.

3.3.2 De integración de sistemas

Se va a pasar a comprobar que todos los sistemas se integran de forma correcta a la vez, es decir, se va a emular un uso que se podría considerar normal. Esto es debido a que las pruebas anteriores aseguran que los sistemas funcionan correctamente en dichas situaciones concretas. Pero no aseguran que funcionen todos a la vez.

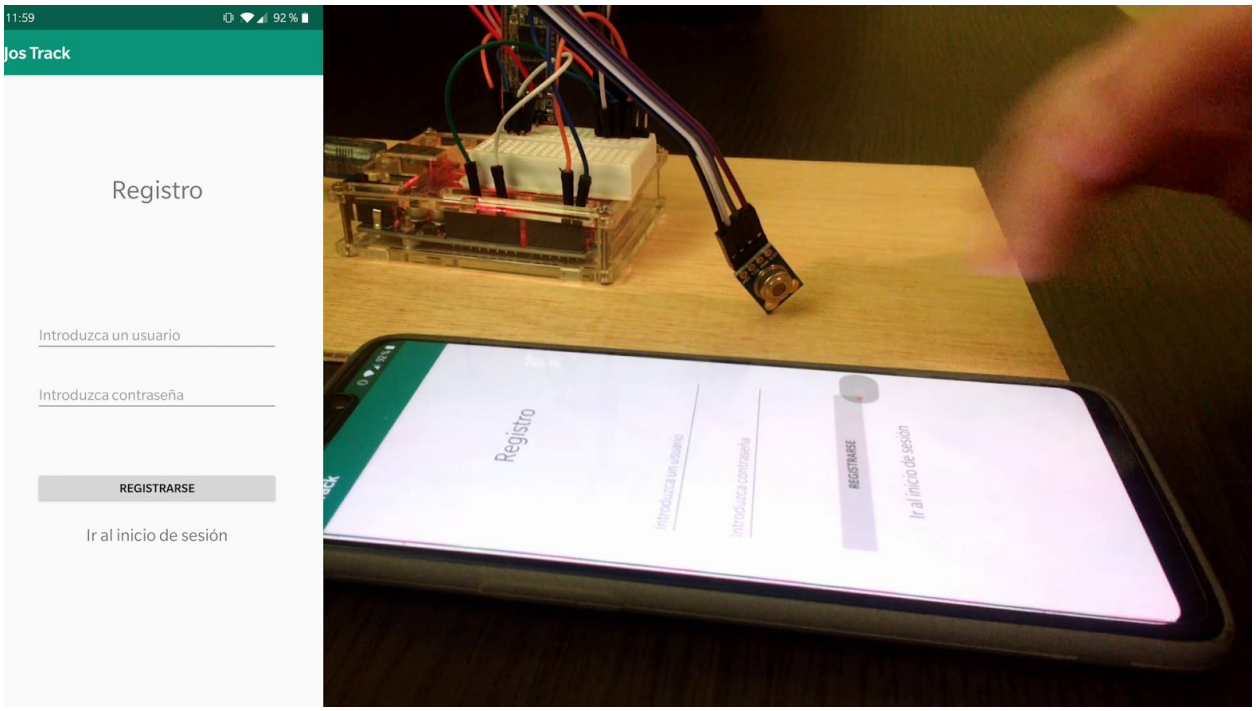
Se ha realizado una primera prueba con todo el sistema montado en un entorno controlado en laboratorio para comprobar que efectivamente todo funciona correctamente. En caso de funcionar todo como es debido se realizará una prueba con un vehículo.



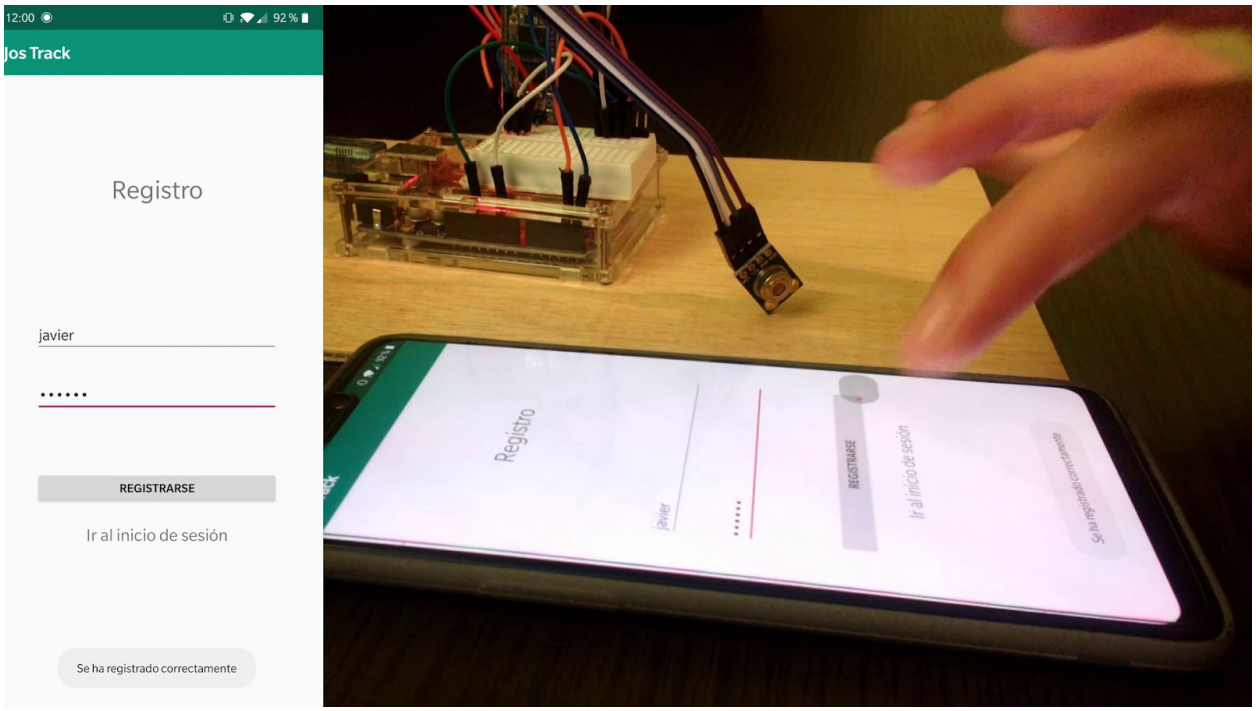
Al abrir la aplicación se abre, una vez cargada, la pantalla de inicio de sesión. Por su parte, el dispositivo Arduino, una vez se enciende realiza la configuración inicial. La configuración inicial termina una vez se queda encendido el LEDpin o LED asociado al pin 13 de Arduino. También parpadea el LED integrado en el módulo Bluetooth indicando que está listo para ser emparejado.
Nota: Hace falta vincular el dispositivo Arduino con el smartphone antes de acceder a la aplicación.

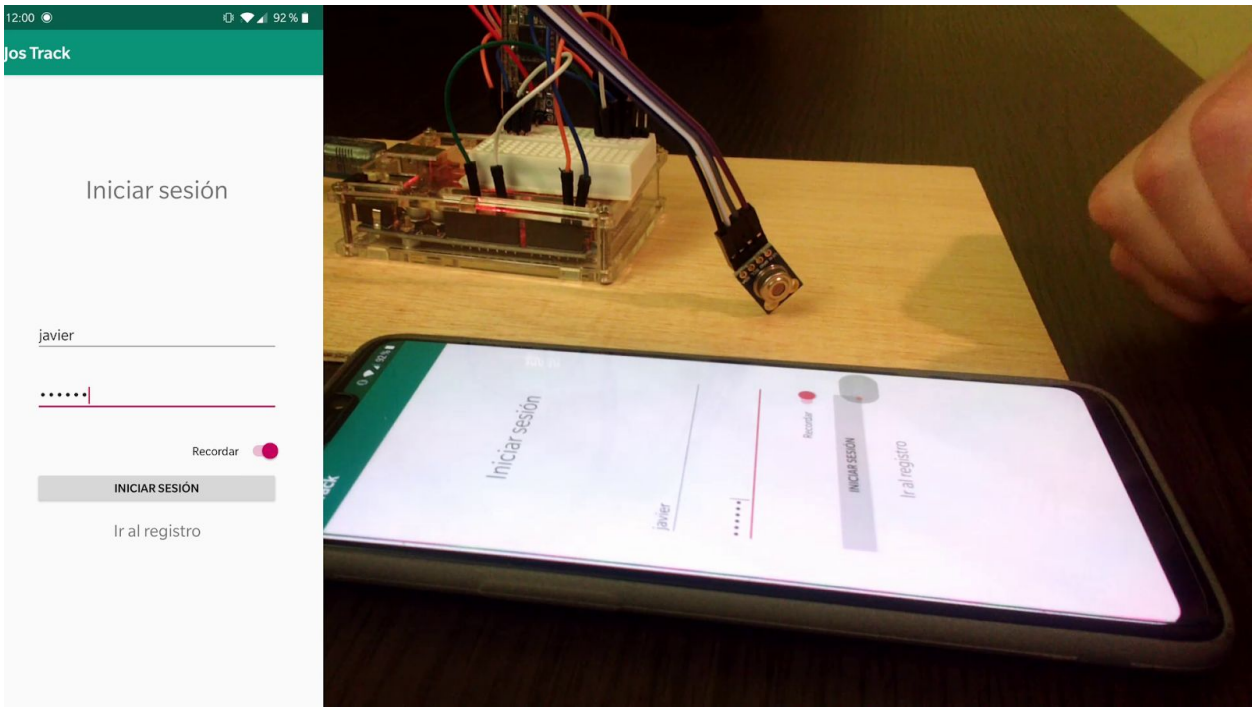
Al pulsar sobre el botón 'Ir al registro' la aplicación cambia a la pantalla donde rellenar el formulario de registro. En este es donde el usuario puede crear su cuenta, necesaria para iniciar sesión.

Una vez se haya registrado puede volver al inicio de sesión e iniciar sesión con las credenciales anteriormente introducidas.

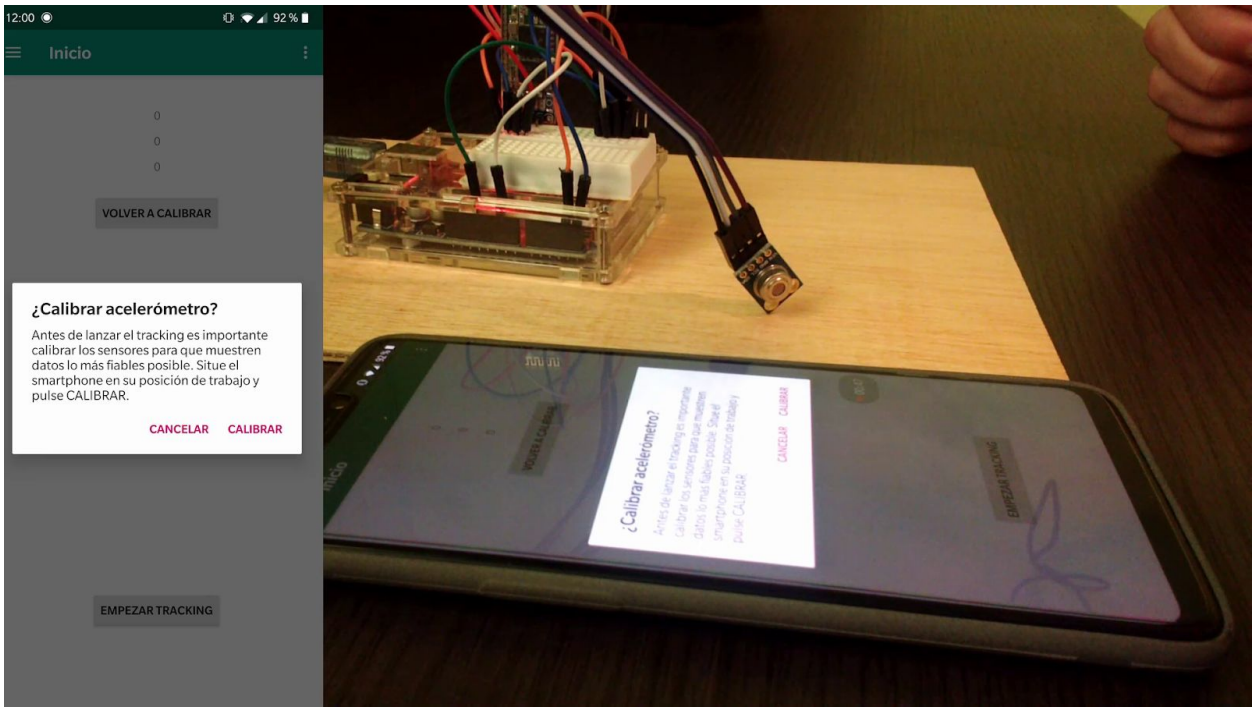


Una vez abierta la pantalla de registro el usuario puede introducir un nombre de usuario y una contraseña. Dicho nombre de usuario debe tener al menos cinco caracteres y no estar ya registrado. En caso de escoger un nombre ya registrado, la aplicación notificará que dicho nombre ya está registrado y no avanzará en el registro. Una vez se introduce un nombre de usuario válido y se trata de registrar pulsando el botón 'REGISTRARSE' el sistema notifica que se ha registrado correctamente y ya puede volver al inicio de sesión mediante el botón 'Ir al inicio de sesión'.

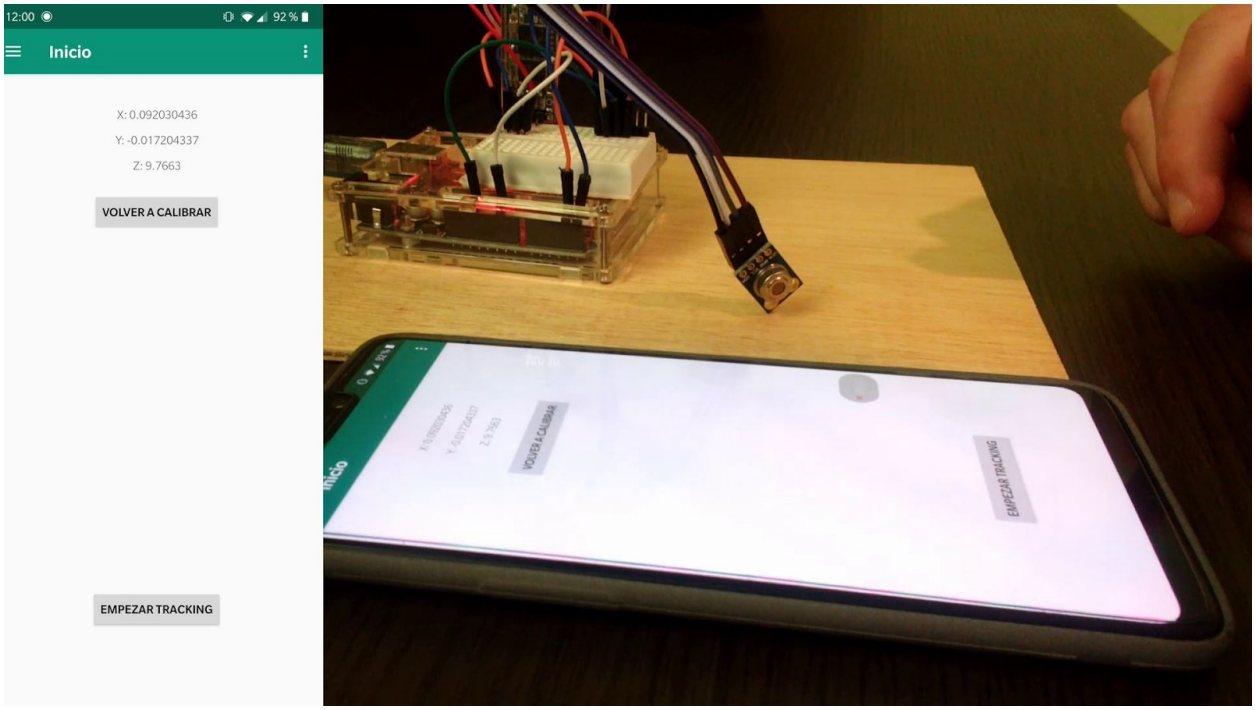


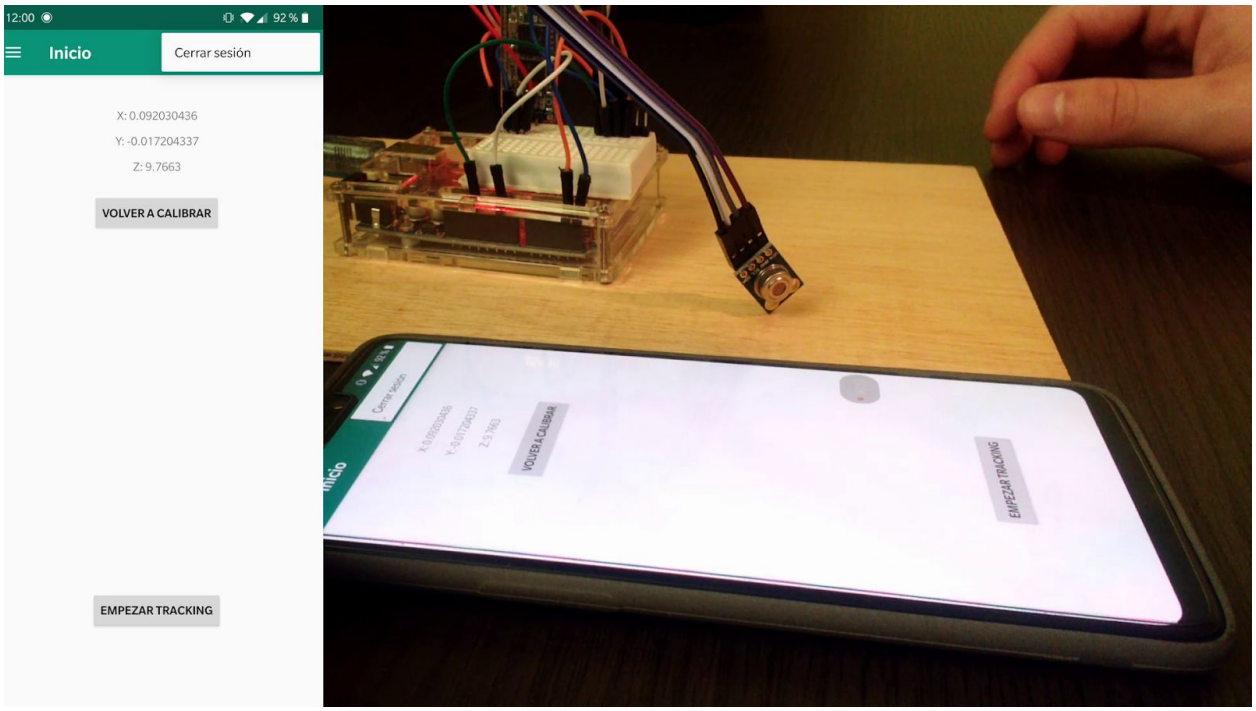


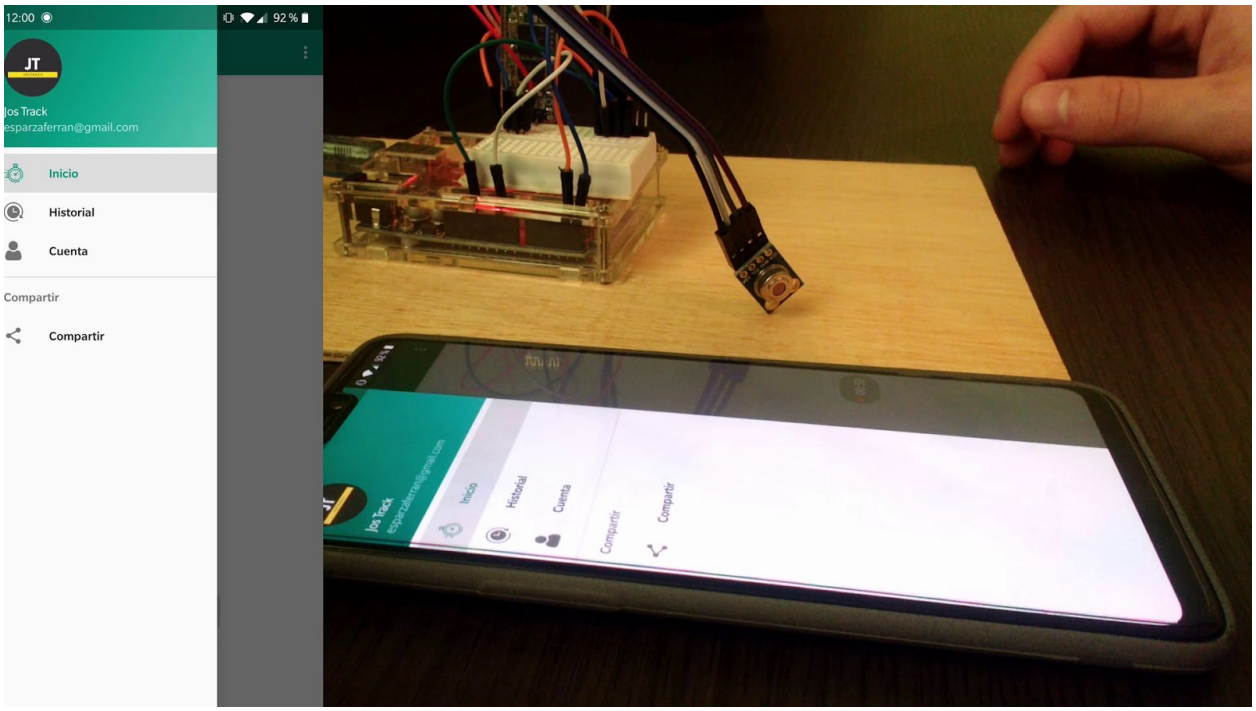
Una vez se cuenta ya con las credenciales correctamente registradas se puede proceder a iniciar sesión marcando el slider 'Recordar' en caso de querer que la aplicación inicie sesión de forma automática, sin necesidad de volver a introducir las credenciales.



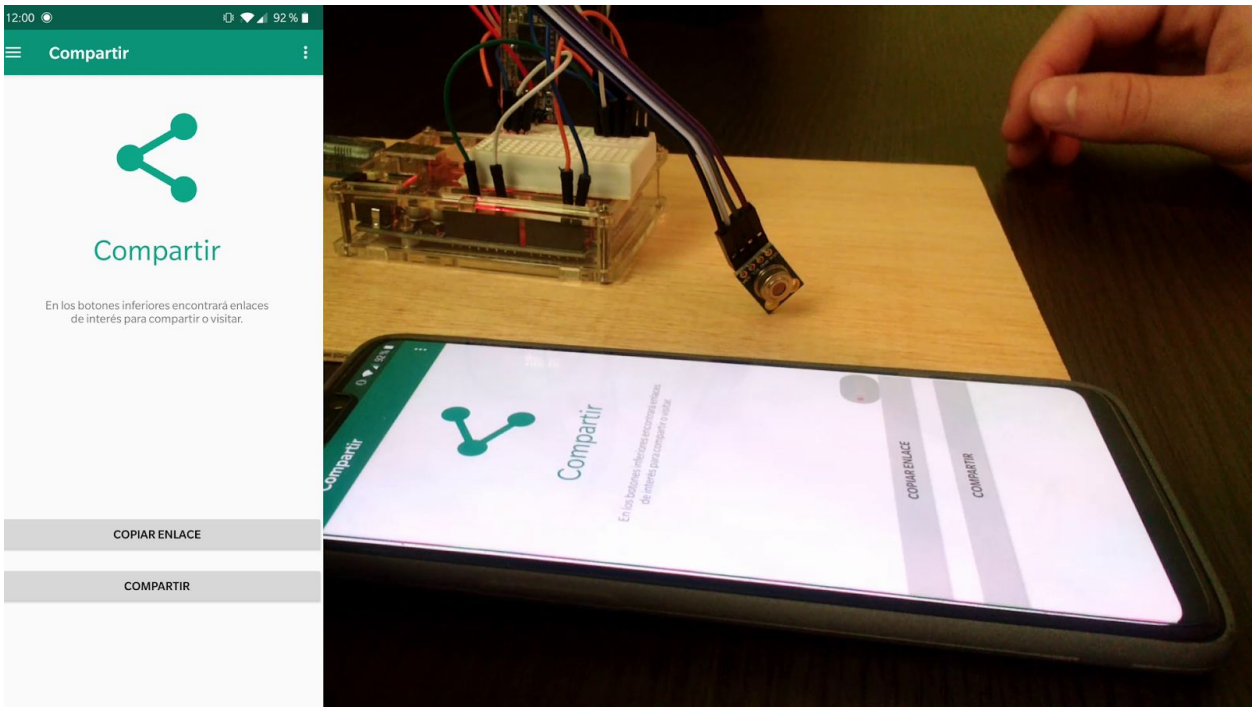
Una vez se ha iniciado sesión, lo primero que realiza la aplicación al acceder a la pantalla principal es calibrar el acelerómetro. Esto debe realizarse en la posición que ocupará el smartphone una vez colocado en el vehículo, es decir en su respectivo soporte.
Desde esta pantalla de inicio se puede navegar al resto de pantallas como el tracking, cerrar sesión, etc.



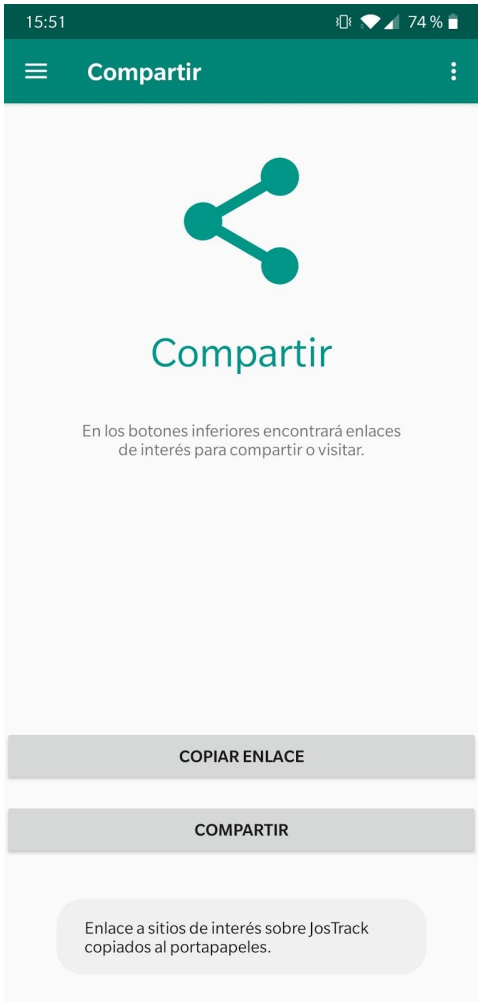


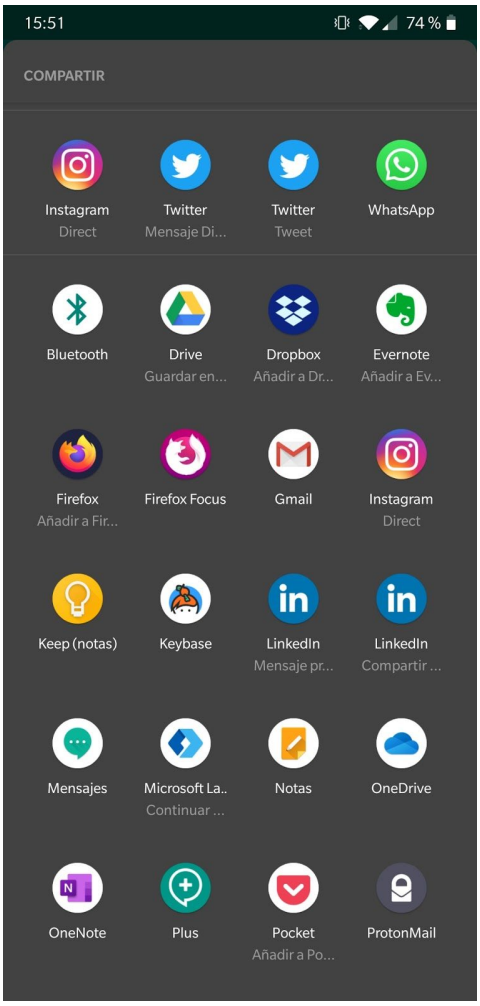


Se dispone de un menú lateral con el que navegar entre distintas pantallas las cuales se va a entrar más al detalle.

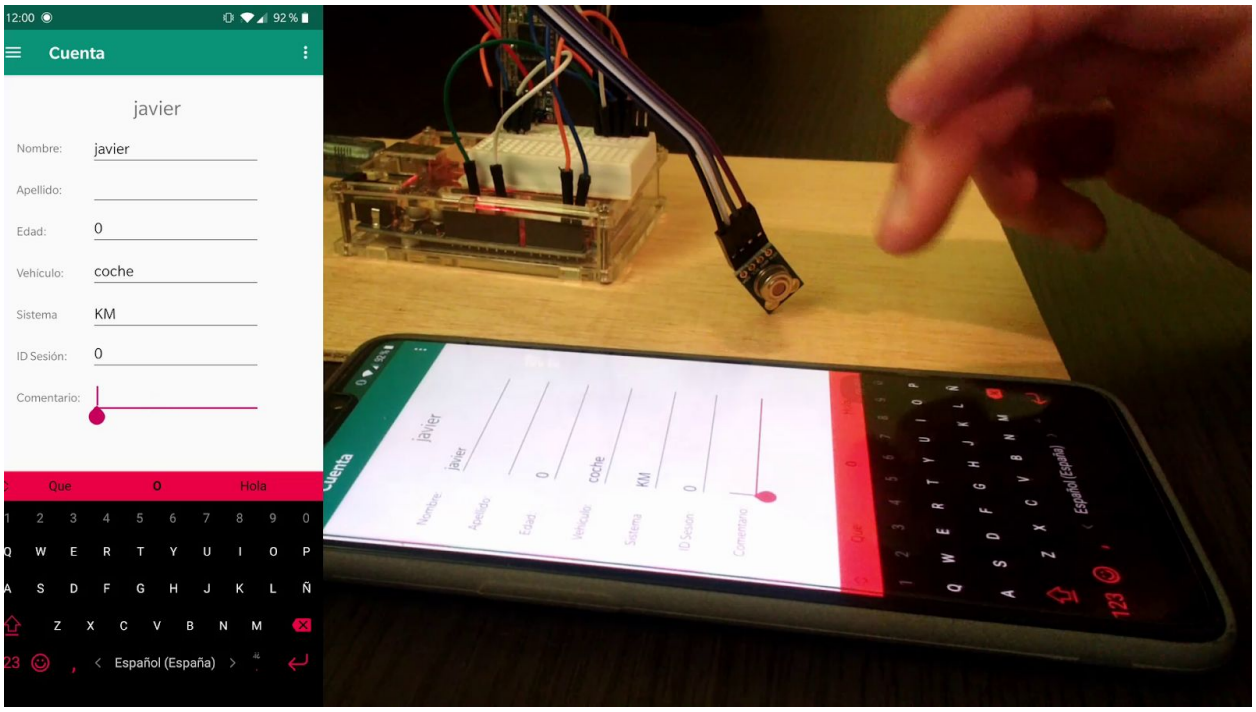


En la pantalla compartir se puede copiar al portapapeles un enlace donde se agrupan distintos enlaces de interés. Una vez copiado, la aplicación lanza una notificación para indicar que se ha copiado dicho link al portapapeles. Para ello simplemente es necesario pulsar el primer botón.

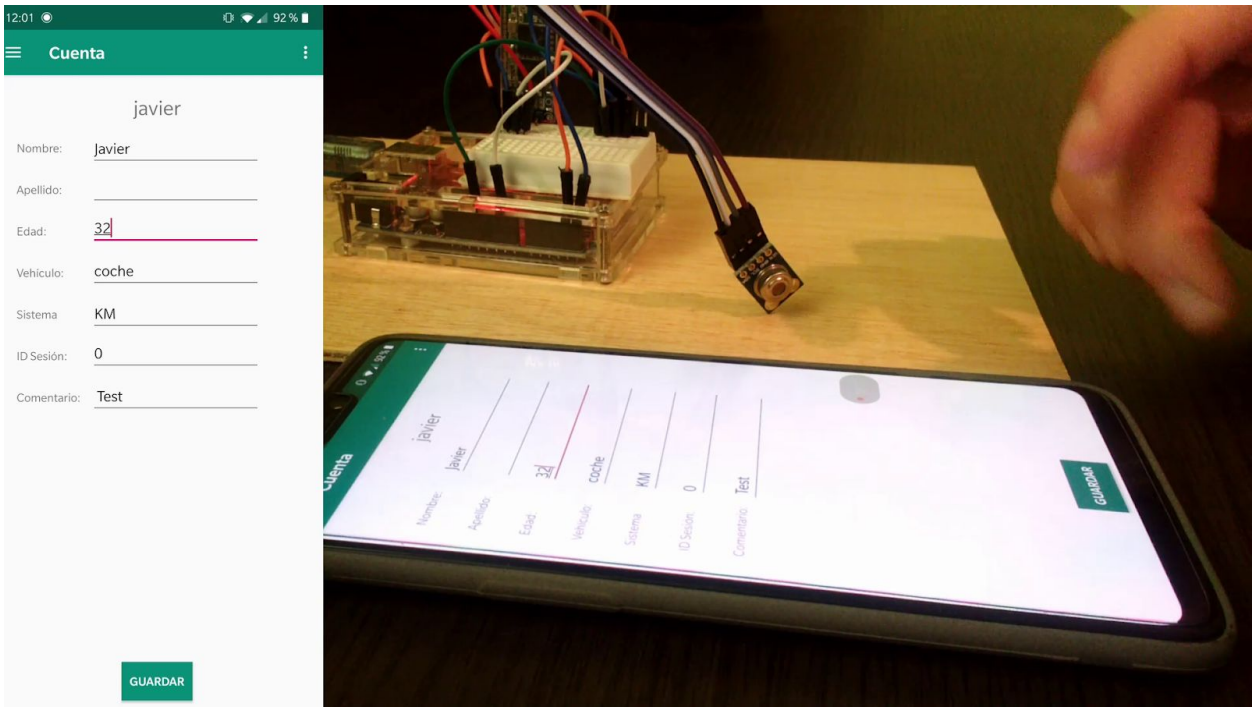


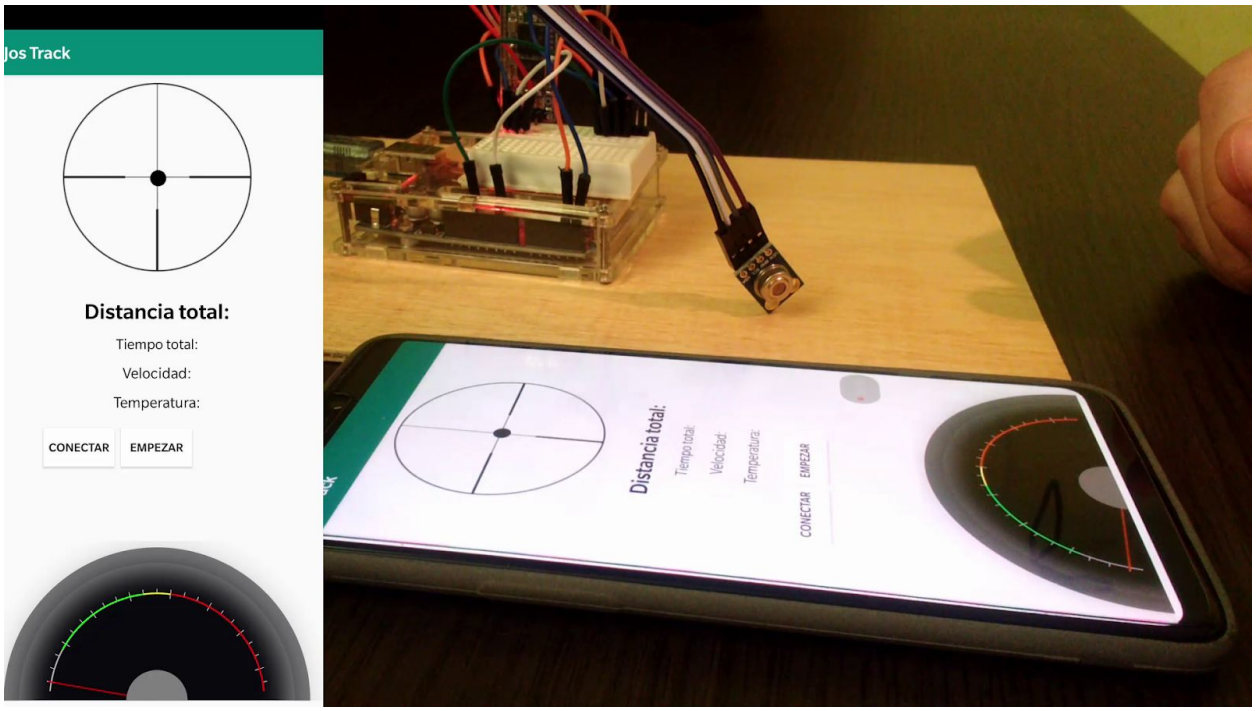


En cambio, si se pulsa el botón 'COMPARTIR', el sistema lanza un menú desplegable para seleccionar dónde se quiere compartir dicho enlace, como por ejemplo por Whatsapp o Telegram.

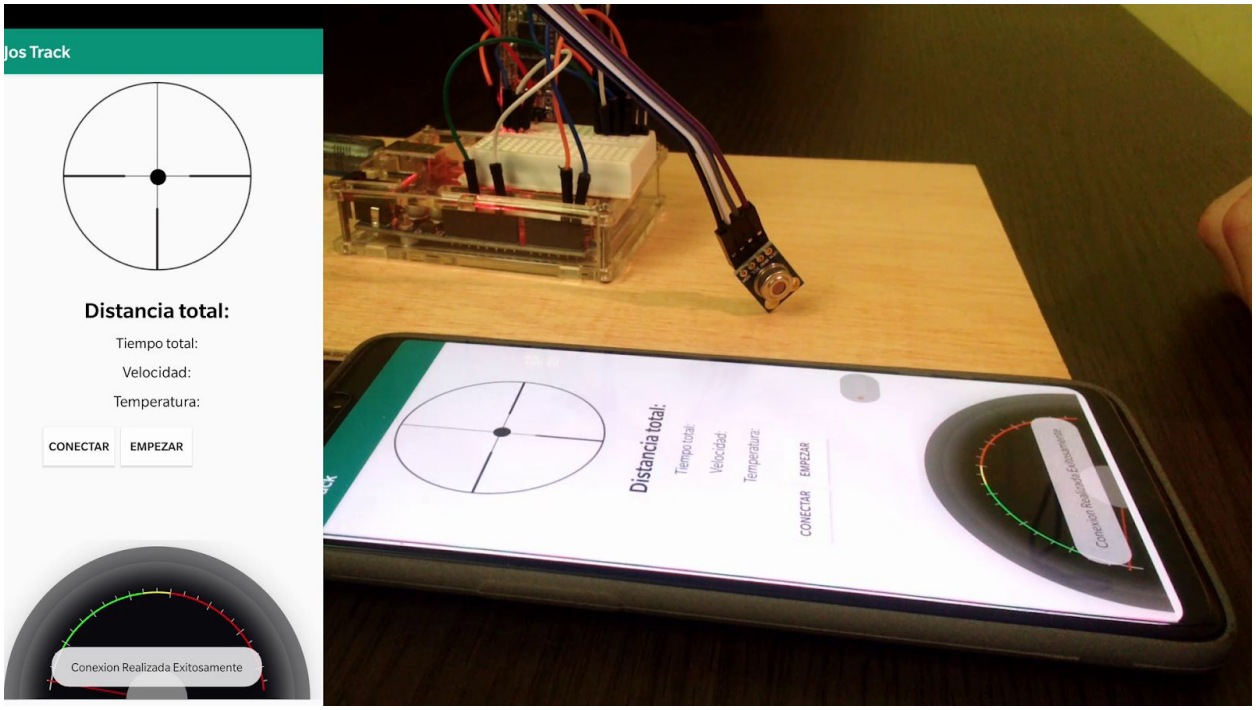


La pantalla de Cuenta, como se puede apreciar muestra los datos por defecto de una cuenta recien creada los cuales pueden ser actualizados. La siguiente vez que se inicie sesión se puede comprobar que efectivamente los datos se han guardado correctamente o se puede comprobar desde la propia base de datos de Firebase como ya se hizo en las pruebas Pruebas de Sistema.

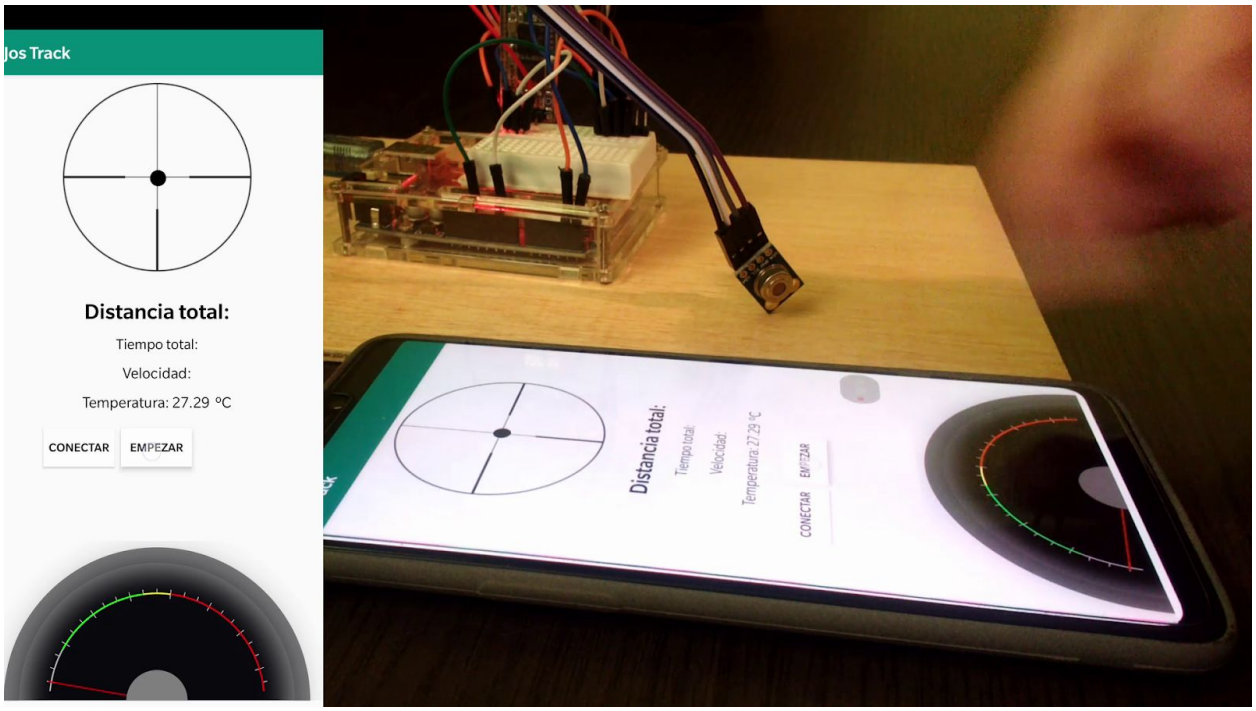




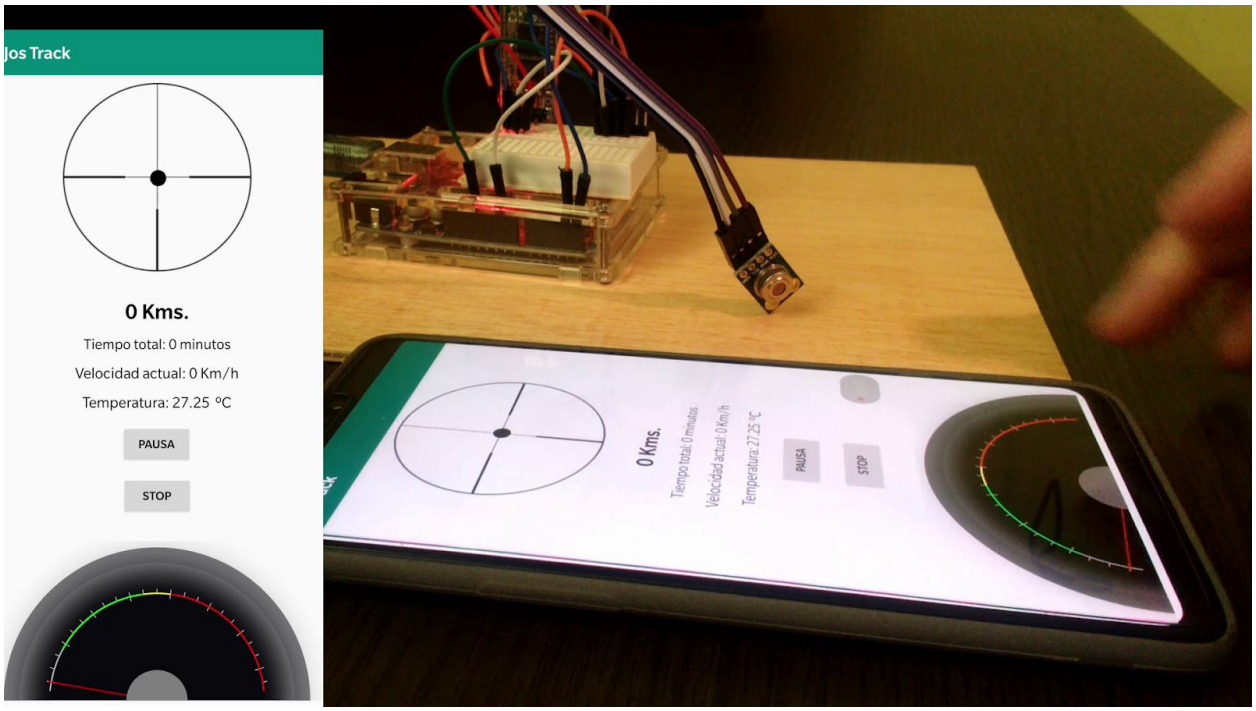
Una vez en la pantalla Tracking es hora de comprobar cómo se integra tanto el dispositivo Android como el dispositivo Arduino, es decir si estos se comunican de forma correcta. Por otra parte, también se comprueba que el acelerómetro se comporta correctamente y el GPS ofrece información correcta. Como cualquier dispositivo con GPS y más haciendo pruebas dentro de un edificio, es importante recordar que puede haber cierto margen de error dependiendo de qué smartphone se esté usando y la señal GPS que se disponga en ese mismo lugar y momento.

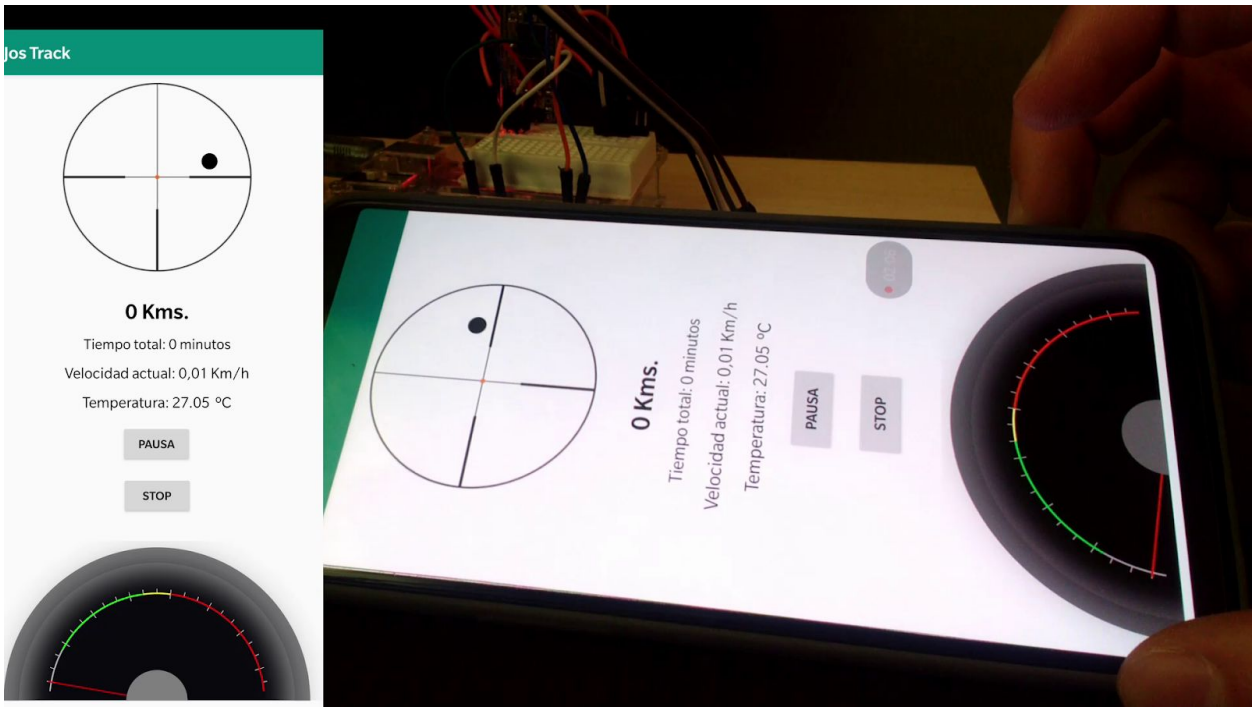


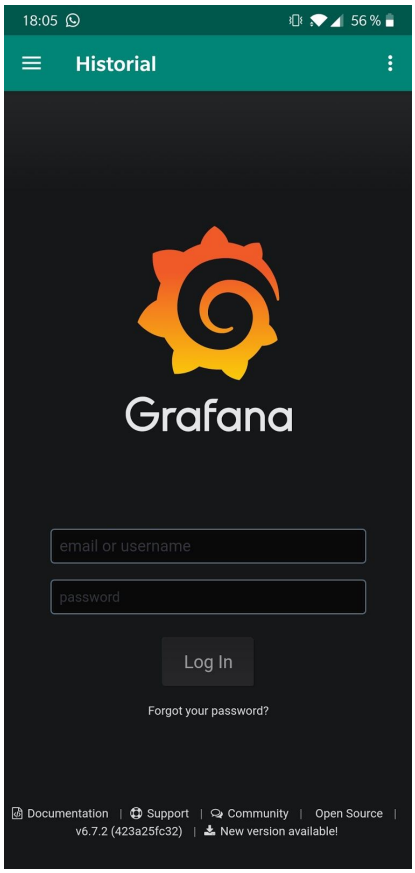
Como se puede apreciar en la captura superior, una vez se pulsa sobre el botón 'CONECTAR' la aplicación trata de conectar con el dispositivo Arduino. Si esto se realiza de forma correcta, el led del módulo Bluetooth HC-06 deja de parpadear y se queda encendido de forma fija y la aplicación muestra un mensaje indicando que la conexión se ha realizado de forma correcta. Acto seguido los datos de temperatura empiezan a mostrarse en pantalla.



Por último solo queda comprobar que se realiza correctamente el seguimiento de altura y velocidad y que el acelerómetro funciona correctamente.

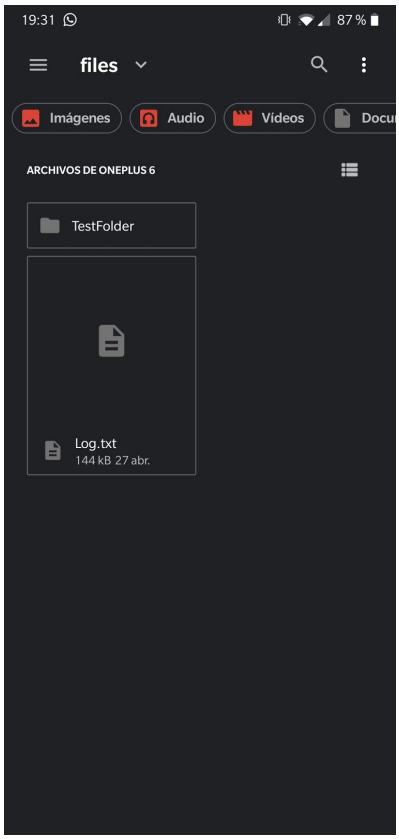

Jos Track
0 Kms.
Tiempo total: 0 minutos
Velocidad actual: 0 Km/h
Temperatura: 27.25 ºC
PAUSA
STOP
0 Kms.
Tiempo total: 0 minutos
Velocidad actual: 0 Km/h
Temperatura: 27.25 ºC
PAUSA
STOP


Jos Track
0 Kms.
Tiempo total: 0 minutos
Velocidad actual: 0,01 Km/h
Temperatura: 27.05 ºC
PAUSA
STOP
0 Kms.
Tiempo total: 0 minutos
Velocidad actual: 0,01 Km/h
Temperatura: 27.05 ºC
PAUSA
STOP



En cuanto a la pantalla Historial, se puede comprobar que accede correctamente a Grafana para poder visualizar los gráficos creados con los datos anteriormente obtenidos.

En este punto sería necesario iniciar sesión en una de las distintas cuentas que se disponga para poder manejar los distintos dashboards que se hayan creado previamente o crear nuevos en caso de ser necesario.
Si bien es posible gestionar los dashboards desde la propia aplicación Android, Grafana está mejor aprovechado en una tablet o monitor de ordenador donde poder observar de forma más cómoda el tablero entero.



Por otra parte, también se puede apreciar que el archivo Log.txt se encuentra creado y dentro incluye todos los datos que se pueden encontrar en la base de datos mariaDB en forma de texto plano.

Link al video demostración donde se incluye el procedimiento anterior (link).

Para realizar las pruebas de integración por parte del Cloud, es necesario tener en cuenta la parte anterior. Es decir, que se dispone de un sistema de obtención de métricas y de enviar dichas métricas a la nube. Teniendo esto ya se puede comprobar los datos que hay en la nube.

También se han realizado pruebas con vehículos para comprobar que los sistemas actúan de forma correcta en un entorno lo más real posible.

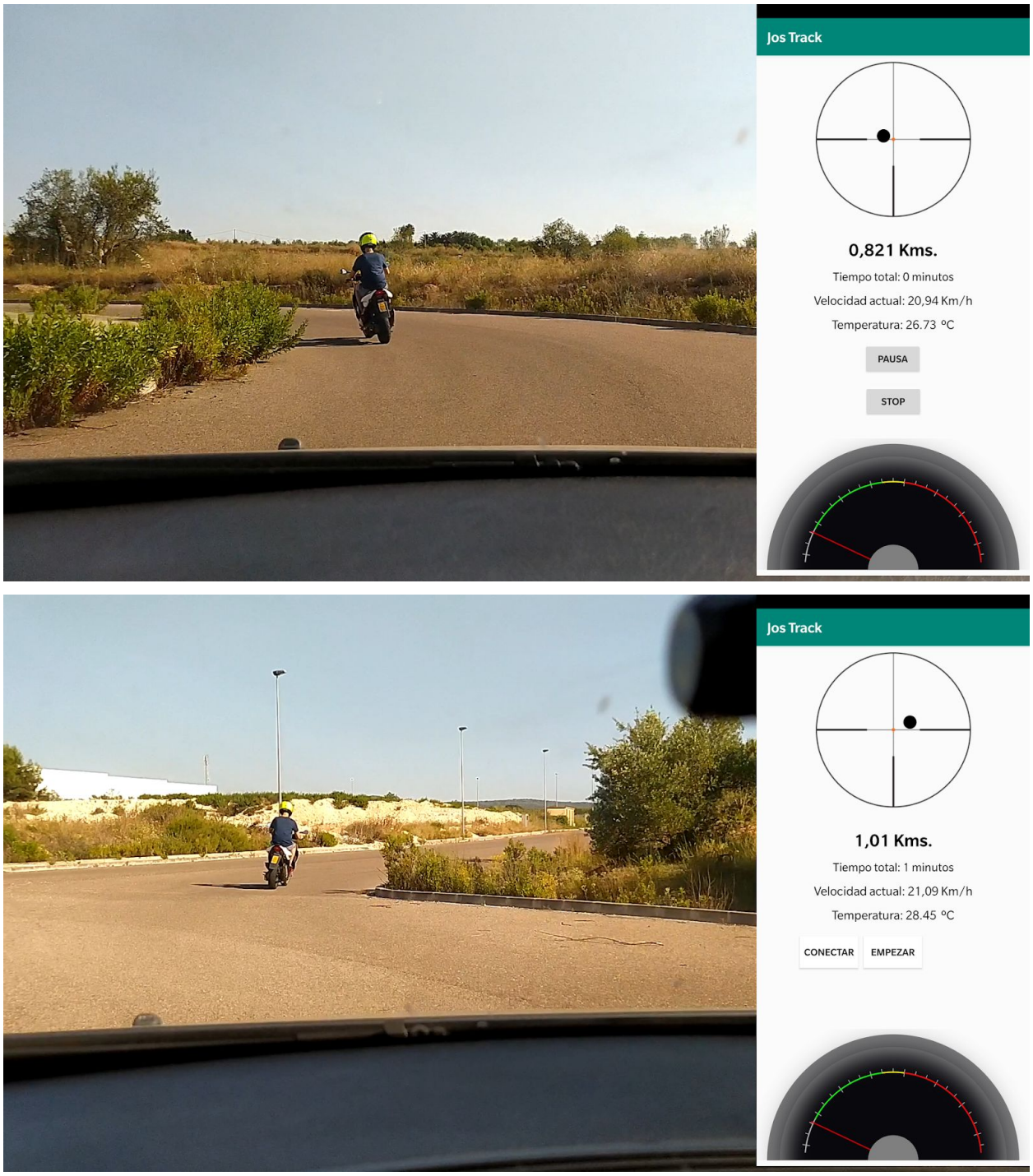
Para realizar las pruebas se ha empleado en primer lugar un automóvil en el que se ha aprovechado un soporte que se disponía para sujetar el smartphone al alcance del conductor y el dispositivo Arduino se ha colocado de forma que pudiera registrar la temperatura de uno de los neumáticos.

La placa arduino junto a otros componentes se protegió durante las pruebas con una bolsa hermética para evitar posibles daños. Por otra parte, todos los elementos fueron fijados empleando cinta de doble cara.

Link al vídeo con la prueba del sistema con un coche (link).

Como el proyecto no va destinado únicamente a automóviles también se realizaron algunas pruebas con un ciclomotor.

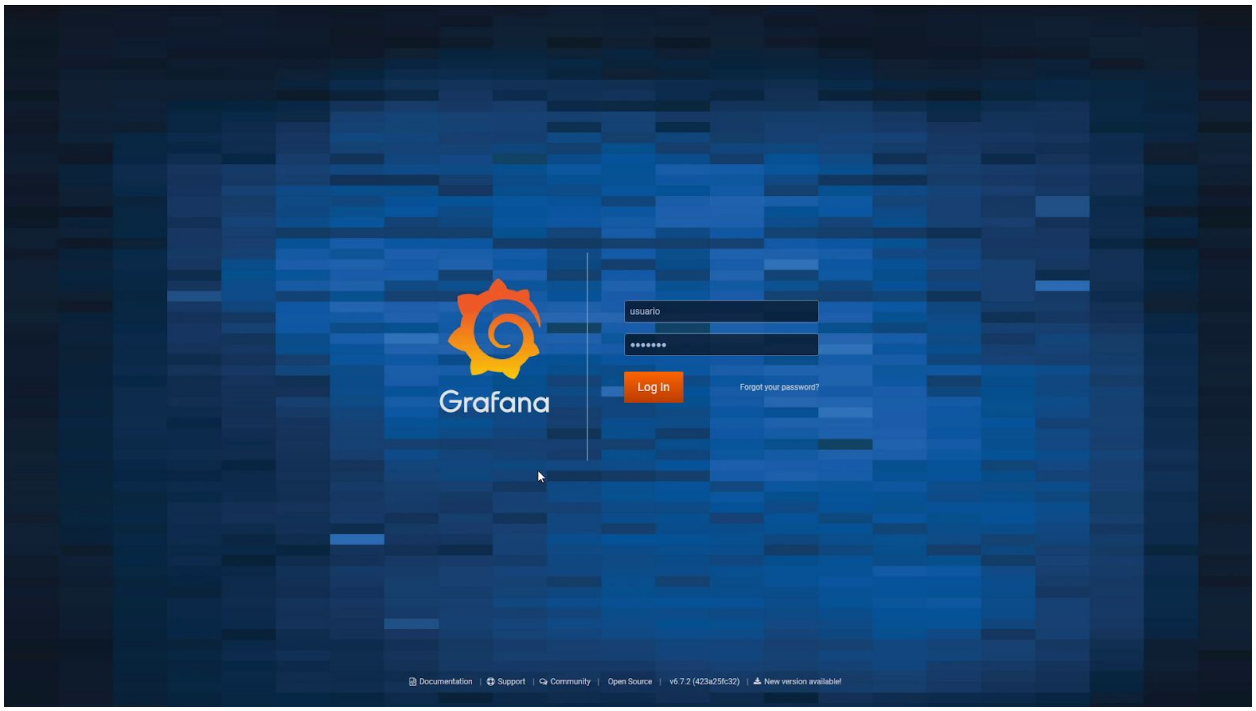
Para ello se fijó, al igual que con el automóvil, el dispositivo Arduino de forma que el sensor de temperatura pudiera apuntar a la rueda motriz. Por otra parte, al no disponer de un soporte donde colocar el smartphone a la vista del conductor, se optó por colocarlo dentro del cofre portaequipajes fijado dentro de dicho compartimento.



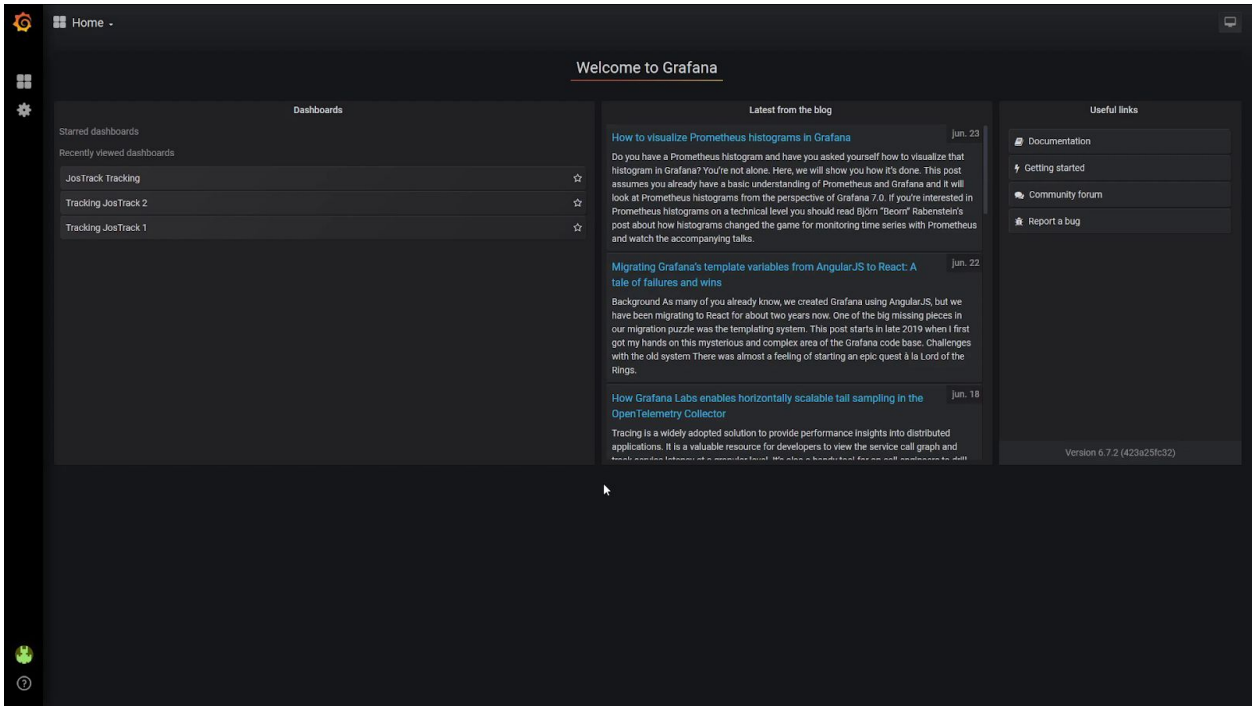
Link al vídeo con la prueba del sistema con un ciclomotor (link).

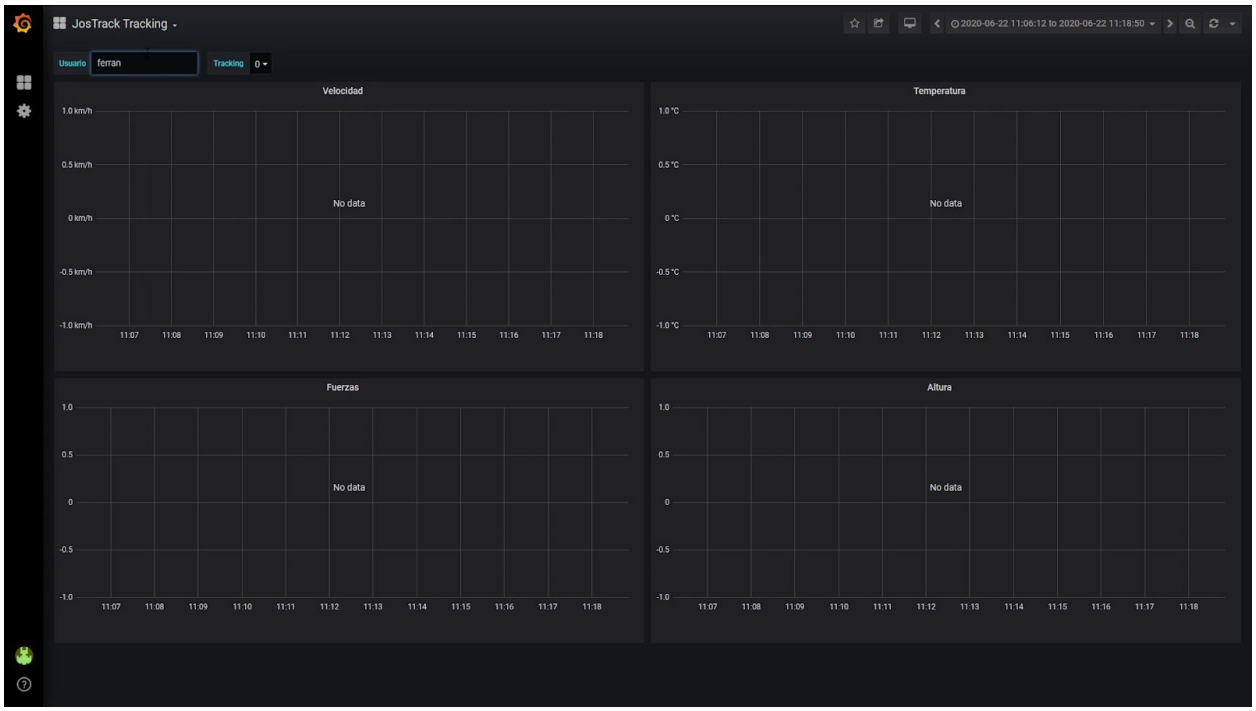
Finalmente, una vez se han realizado pruebas con toda la parte del sistema que se encarga de recoger y enviar los datos a la nube, queda por comprobar si dichos datos han llegado correctamente a la nube y cómo se ven dichos datos en alguno de los dashboards que se han

creado empleando Grafana. Para poder visualizar dichos datos, se ha creado una cuenta llamada '*usuario*' para poder visualizar mediante gráficos todos los datos.



Una vez iniciada la sesión se tiene acceso al menú lateral para navegar por Grafana así como un listado de los dashboards empleados anteriormente, un apartado de noticias y novedades sobre Grafana y links de interés sobre el propio Grafana.





Una vez se accede a uno de los dashboard, tal y como se han creado para adaptarse al proyecto, lo más normal es que no muestren datos. Es necesario introducir el nombre de usuario del que se quieren ver los datos en la caja de texto superior derecha. En caso de estar ya introducido, es necesario simplemente pinchar en dicha caja de texto y pulsar el botón Enter para que Grafana busque los datos.

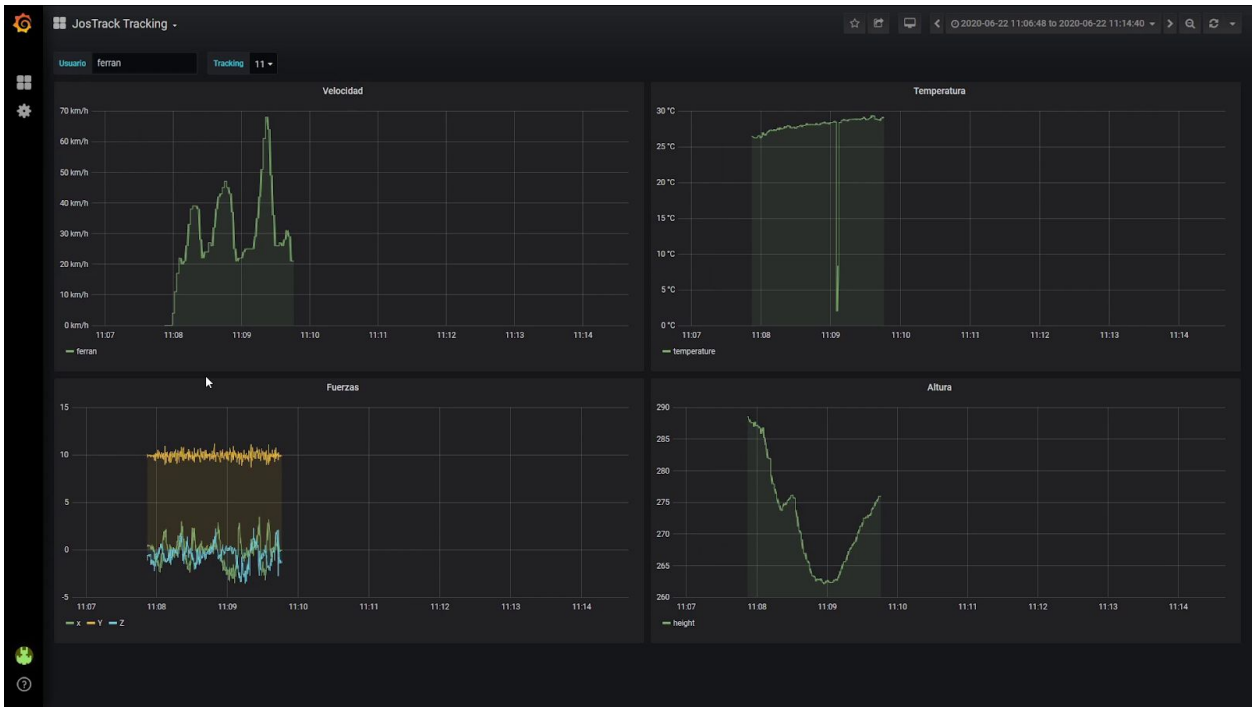
Una vez introducido el usuario, Grafana mostrará los datos de todos los registros que haya en la base de datos mariaDB relacionados con dicho usuario. Ver todos los registros de golpe puede ser bastante confuso y es por ello que se ha habilitado, justo al lado de la caja de texto destinada a encontrar el usuario deseado, un menú desplegable que se encarga de listar los registros que tiene el usuario introducido anteriormente.

En este listado se pueden seleccionar los registros que se crean oportunos, así como listarlos de uno en uno para comprobar los datos de dicho registro.

Como es puede apreciar en la captura inferior, para el usuario 'ferran' se ha seleccionado el track '11' para poder visualizar dichos datos.

En caso de querer ver los datos ampliados se puede seleccionar el rango deseado arrastrando con el ratón en uno de los gráficos. Al soltar el botón izquierdo del ratón se ampliarán los gráficos de todo el dashboard al rango de tiempo deseado.

En caso de que el rango de tiempo estuviese demasiado ampliado se puede seleccionar uno más elevado en el desplegable superior derecho.



Link al video demostración Grafana (link).

4 Resultados

4.1 Migración al entorno de producción

En primer lugar es necesario desplegar el entorno de producción, para ello, una vez contratado el Servidor Cloud en Neodigit, la instalación del sistema operativo Ubuntu 18.04 LTS se realiza de forma automática por parte de la propia compañía.
Una vez el servidor está disponible, es necesario conectarse por SSH para poder gestionarlo a distancia.

Desde la terminal, se instala en primer lugar Docker. Para ello primero actualizaremos todos los repositorios con:
sudo apt-get update
sudo apt-get upgrade

Se eliminan posibles versiones que pudieran haber anteriores:
sudo apt-get remove docker docker-engine docker.io

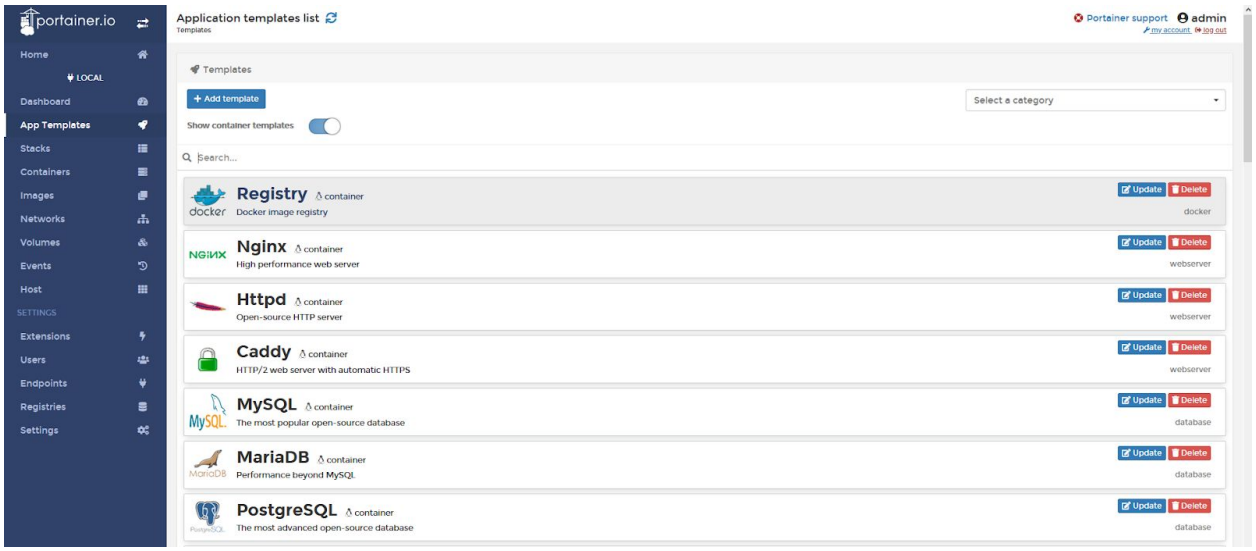
Y finalmente se instala Docker:
sudo apt install docker.io

También se debe configurar para que se inicie cada vez que se arranque el servidor:
sudo systemctl start docker
sudo systemctl enable docker

Una vez se tiene Docker, el servicio Portainer puede ser bastante útil para facilitar la instalación de los contenedores. Se requerirá un contenedor de MySQL o mariaDB y otro para Grafana. Para ello primero se creará un volumen para almacenar datos:
docker volume create portainer_data

Para luego poder instalar Portainer:
docker run -d \
--name portainer \
--restart=always \
-p 9000:9000 \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data portainer/portainer

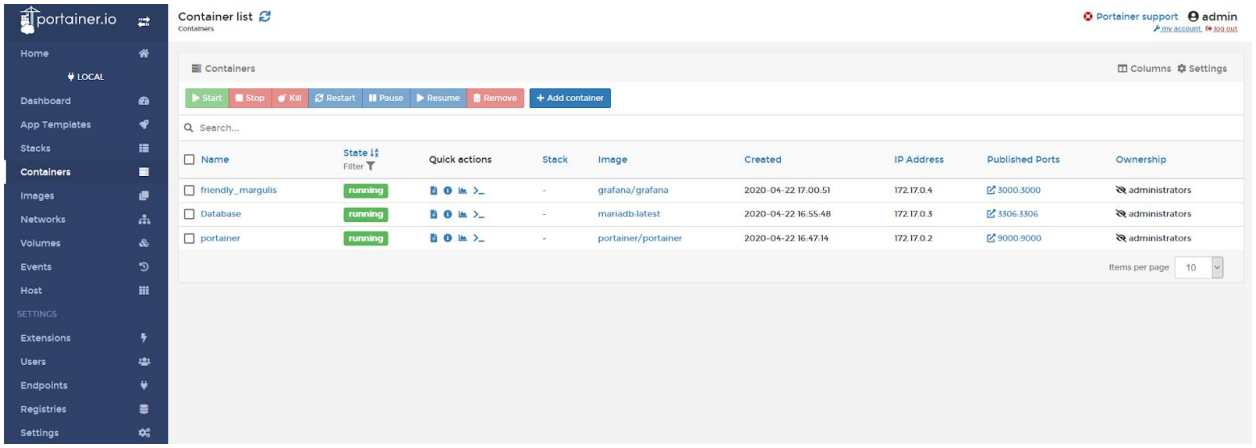
Una vez finalice ya se puede acceder desde el navegador con la dirección de la máquina+el puerto (en este caso le hemos asignado el puerto 9000). La primera vez que se acceda requerirá una contraseña para la cuenta de administrador y ya se podrán gestionar los contenedores.



Mediante Portainer simplemente es necesario acceder a las plantillas y desplegar un contenedor con la plantilla que se requiera. Una de esas plantillas que se emplean es la de mariaDB para desplegar la base de datos donde se guarda todos los datos del proyecto.

Pero sigue siendo posible emplear Docker mediante terminal si no se quiere o no se puede emplear una plantilla para desplegar un contenedor. Para desplegar Grafana es tan sencillo como lanzar en la terminal:
docker run -d -p 3000:3000 grafana/grafana

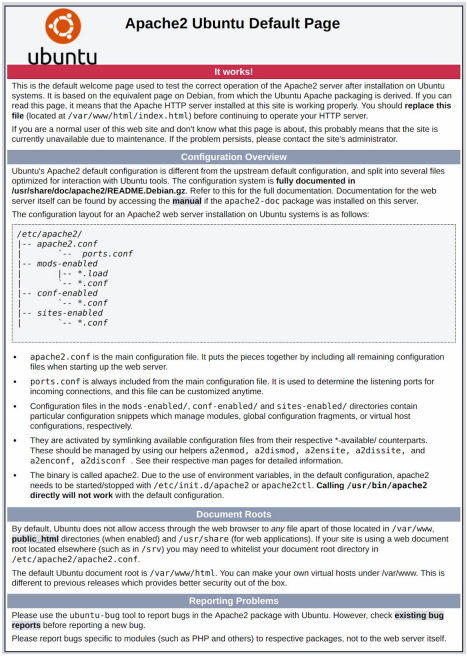
Como detalle, los contenedores desplegados por terminal directamente con Docker, pueden verse y manipularse mediante Portainer:



Para la conexión con el servidor de base de datos se emplea DB Forge Studio Express. Una vez lanzado el programa, se crea la conexión con el servidor de base de datos y se abre automáticamente una pestaña donde poder trabajar con las consultas SQL.

En esta pestaña es donde se ejecutará el código que se ha diseñado para generar la base de datos y la tabla Tracks. Dentro de dicha tabla será donde se almacenen todos los datos de las sesiones al volante.

Por otra parte, se requiere un servidor web capaz de ejecutar código PHP. Para ello se instala Apache y PHP. Posteriormente se subirán los scripts que se requieren para la plataforma.
En primer lugar, para instalar apache simplemente se requiere un comando en la terminal:
sudo apt install apache2
Una vez termina el proceso de instalación se puede acceder a la dirección IP mediante el navegador para comprobar que realmente se ha instalado y funciona correctamente.



Pero este servidor todavía requiere de PHP para poder ejecutar archivos PHP:
sudo apt install php libapache2-mod-php php-mysql

A partir de aquí ya se pueden colocar archivos PHP o con código PHP en el servidor web.
Para ello, en la página por defecto de Apache, que ha mostrado antes se puede ver que las páginas html u otros archivos deben ir en la ruta **/var/www/html/** como el propio index.html. Aquí es donde deben colocarse tanto el *db_conection.php* como el script *asd.php* para realizar los inserts. El archivo asd realiza los insert basándose en la conexión que tiene establecida el *db_conection*. Siguiendo este esquema se podrían crear otros scripts basándose en el mismo archivo para la conexión según se requiera.

El último punto para finalizar la migración de forma correcta es redirigir la aplicación Android al servidor de producción. Para poder hacer esto, es necesario cambiar la dirección IP que hay en la variable '*registrar_url*' dentro del archivo **MainActivity.java** por la IP del servidor de producción. En caso de querer volver al servidor de pruebas se comenta y descomenta la linea que corresponda y la aplicación vuelve a apuntar donde se requiera.

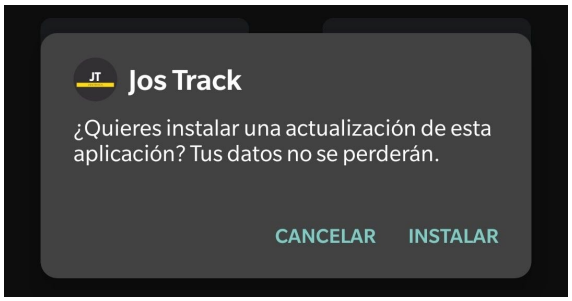
```
905         @Override
906         protected String doInBackground(String... strings) {
907             //Servidor de pruebas
908             //String registrar_url = "http://raspberryserver.hopto.org/asd.php";
909             //Servidor de pruebas en local
910             //String registrar_url = "http://192.168.1.250/asd.php";
911             //Servidor de produccion en Cloud
912             String registrar_url = "http://31.47.76.61/asd.php";
913 
914             String resultado = null;
```

4.2 Manual del usuario

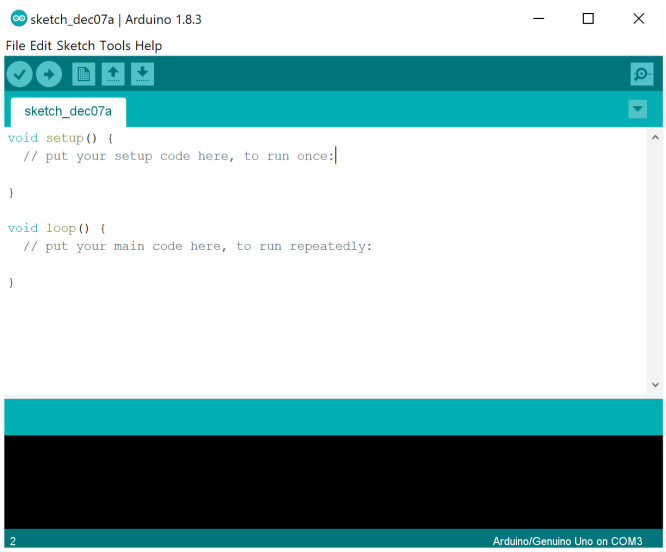
Preparar todo el sistema incluye la instalación tanto de la aplicación en el dispositivo Android como del script para el dispositivo Arduino.
Posteriormente se debe instalar cada dispositivo en el sitio oportuno del vehículo.

4.2.1 Instalación de la aplicación

La aplicación se puede instalar visitando este enlace y pulsando sobre 'Descarga de la app'. El enlace llevará a la descarga del APK desde Google Drive.



Una vez descargado, es necesario abrir el APK y pulsar sobre instalar para que se instale en el terminal.
En caso de bloquear la instalación, es necesario habilitar la opción 'Orígenes desconocidos' en los ajustes del smartphone.

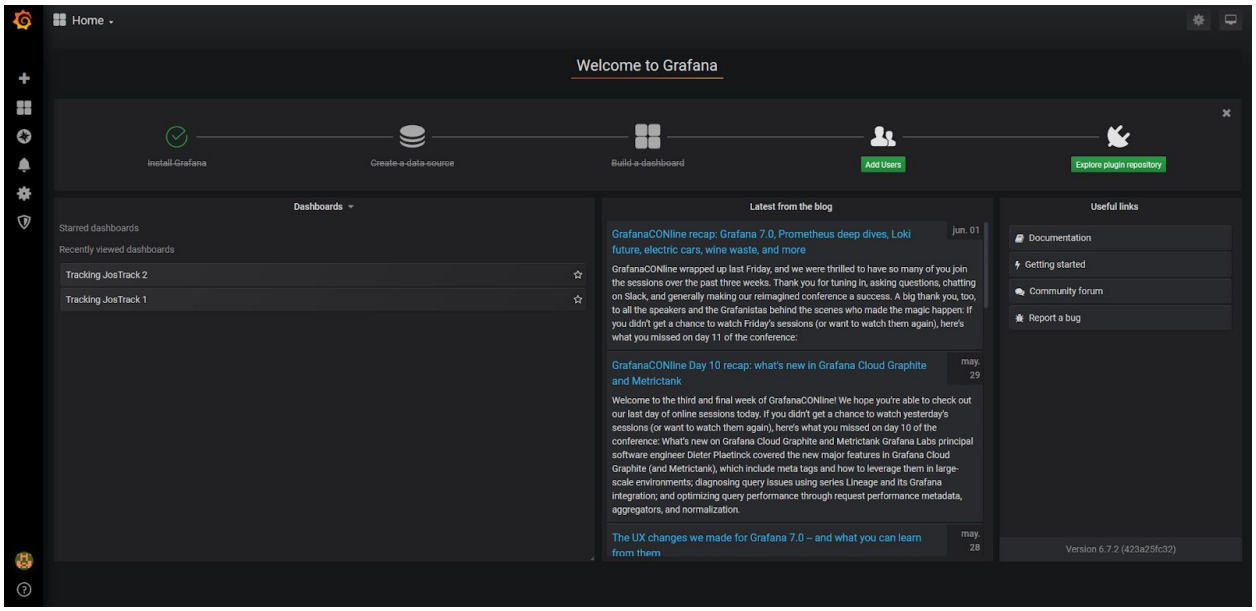


Para cargar el programa que debe llevar la placa Arduino simplemente es necesario conectarla al ordenador mediante un cable USB, verificar que el Arduino IDE la detecta, cargar el archivo en el editor y subirlo a la placa para que disponga de él en memoria.

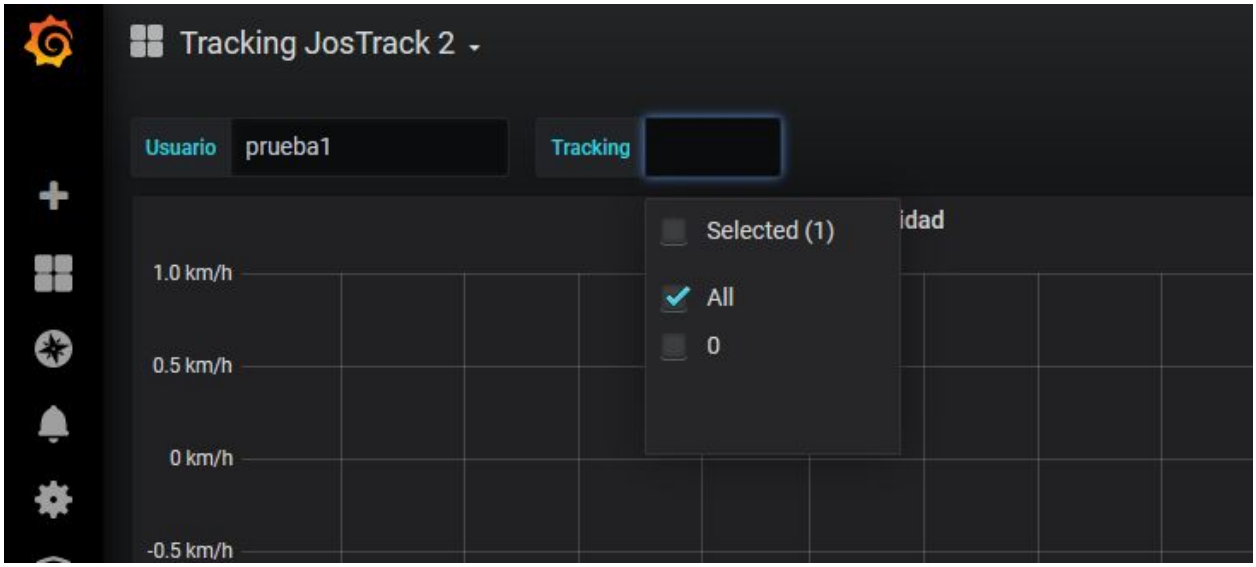
4.2.2 Uso de Grafana

Una vez se han recopilado los datos, estos pueden ser analizados mediante dashboards en Grafana.

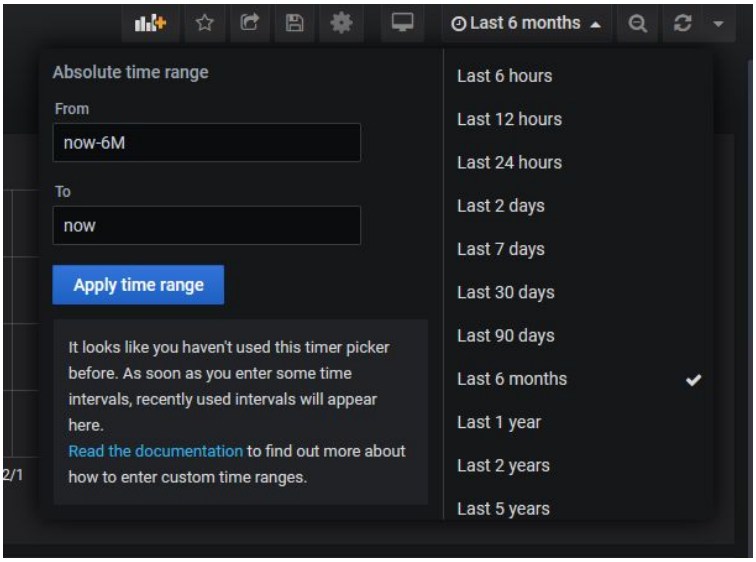
Para poder iniciar sesión se ha habilitado la cuenta usuario/usuario para poder visualizar los distintos dashboards.



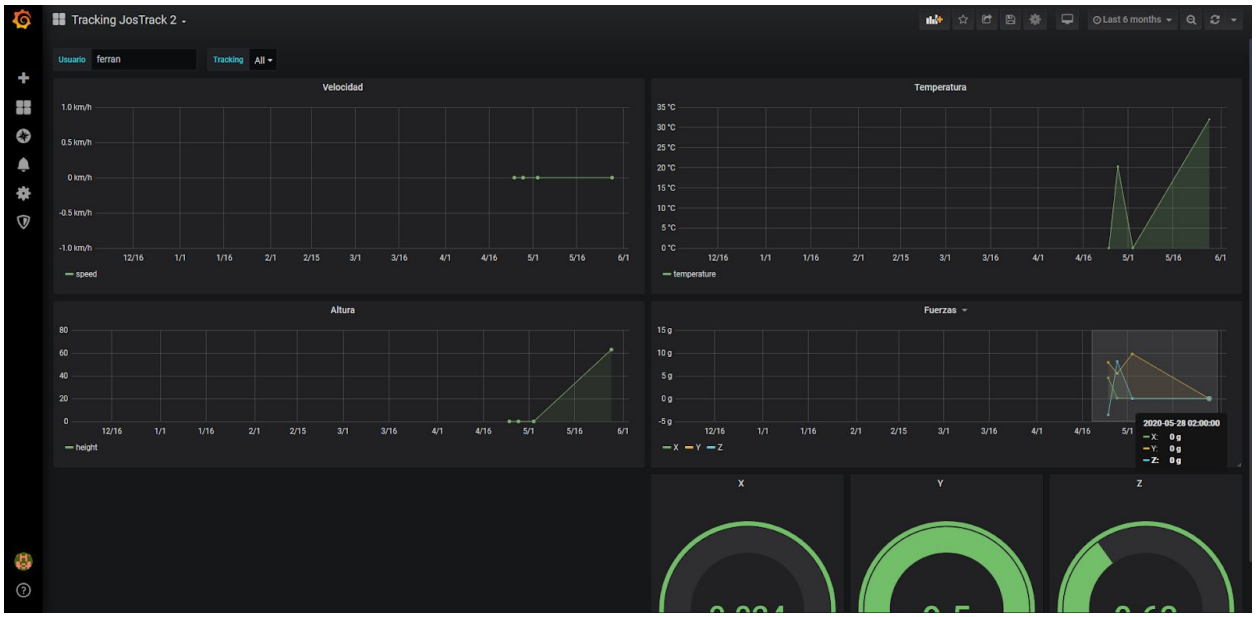
Para poder ver los datos deseados, se dispone de un buscador de usuario donde buscar el nombre de usuario correspondiente. Una vez Grafana encuentra el usuario genera un listado de forma automática con todos los trackings disponibles para dicho usuario en un desplegable a la derecha del buscador de usuario.



Se puede seleccionar uno o varios según se estime oportuno. Una vez que se seleccione los datos aparecerán en pantalla representados en los gráficos disponibles. Es posible que dependiendo de cuando se hayan registrado dichos datos sea necesario desplazarse de forma temporal por los gráficos. Para ello, la opción más recomendable es introducir un intervalo elevado de tiempo, por ejemplo 30 días y una vez se visualicen los datos acercar la vista a donde se encuentran dichos datos.



De forma que se pase de ver a un nivel de detalle mayor. Para ello simplemente es necesario seleccionar mediante un clic y arrastrando para seleccionar la franja de tiempo deseada.



Como se puede apreciar en la captura superior, en el segundo gráfico a la derecha se ha seleccionado un intervalo de tiempo donde se incluyen datos.

Al soltar el clic del ratón se amplían los datos de forma automática pudiendo ver mejor las curvas de los distintos gráficos de todo el tablero.

4.2.3 Explotación

Este proyecto se ha llevado a cabo como una prueba de concepto con la que comprobar la viabilidad de la idea y cómo podría ser llevada a cabo.

Contemplando la suposición de lanzar dicho proyecto al mercado, personalmente contemplo tres opciones:

- La primera: Ponerse a la venta al público bajo suscripción mensual. Esta idea se basaría en lanzar el sistema al mercado por cuenta propia ya sea asumiendo los costes y/o buscando inversiones de terceros.
- La segunda: Lanzar el proyecto con la intención de venderlo a una empresa. Se trataría pues, de pulir el proyecto y tratar de venderlo posteriormente,
- La tercera: Intentar un híbrido de las dos opciones anteriores lanzando al mercado el producto en forma de fase beta con una suscripción agresiva con la que captar usuarios que prueben el sistema mientras se consigue pulir y conseguir inversión por parte de empresas que pudieran estar interesadas en su compra o colaboración en el proyecto.

Personalmente me decanto más por la tercera opción pero al incluir partes de las dos anteriores se podría considerar una evolución de las dos anteriores añadiendo como detalle:

- Lanzar en Google Play Store una versión beta con la que captar los primeros usuarios. Que estos aporten su experiencia con distintos smartphone e ir puliendo errores, añadiendo funcionalidad y seguridad.

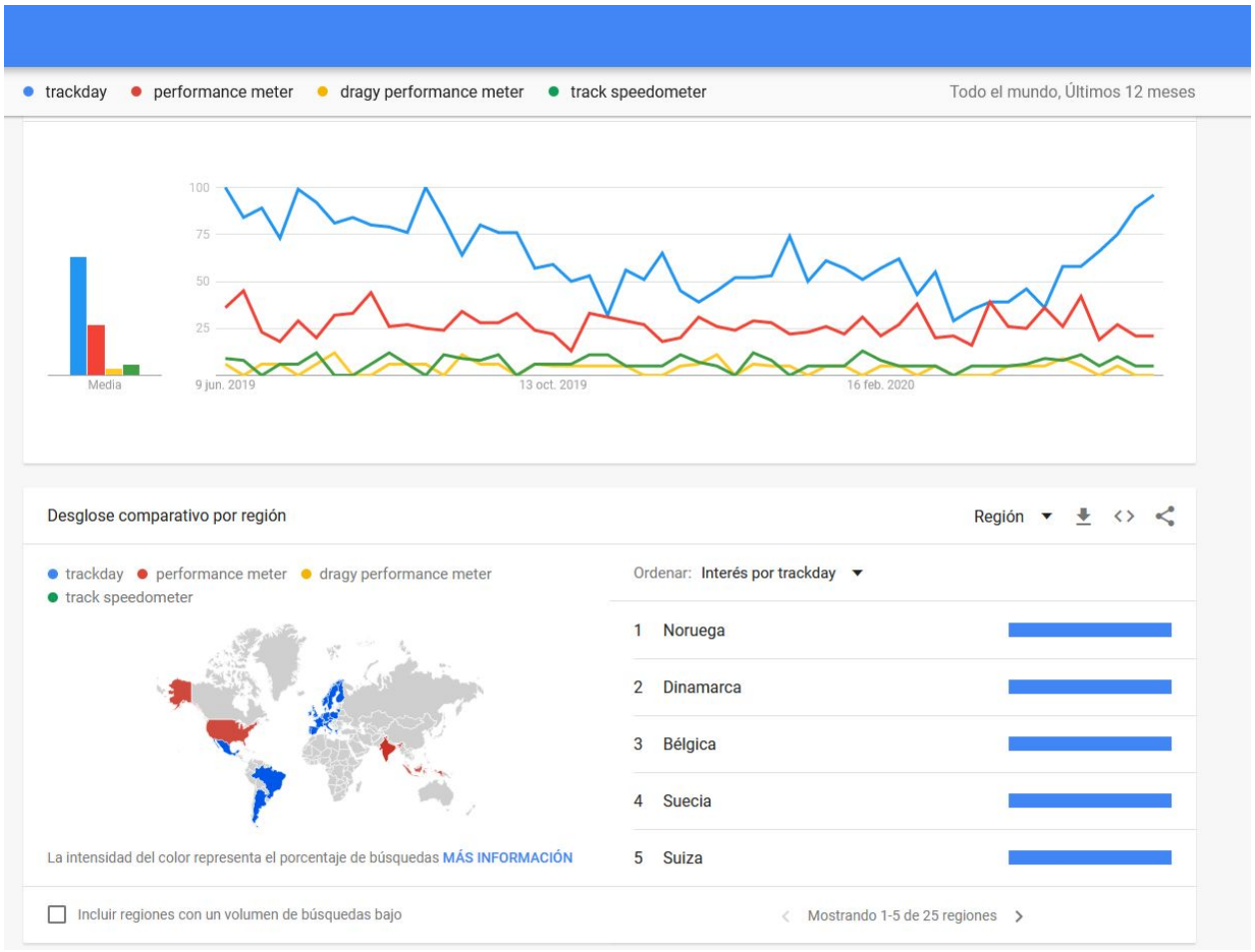
- Mientras se va puliendo todo el sistema ir buscando tanto posibles inversores como empresas que pudiera interesarles colaborar en el proyecto de forma que se beneficien ambas partes de poder sacar al mercado un sistema de telemetría para vehículos y dicha empresa pueda emplearlo en sus vehículos o aprovecharlo para generar nuevas oportunidades de negocio al impulsar posibles competiciones.

- Mediante campañas de Facebook Ads y contactando con gente influyente en las redes sociales relacionadas al mundo motor validar si gusta o si puede ser útil a un nicho lo suficientemente grande.

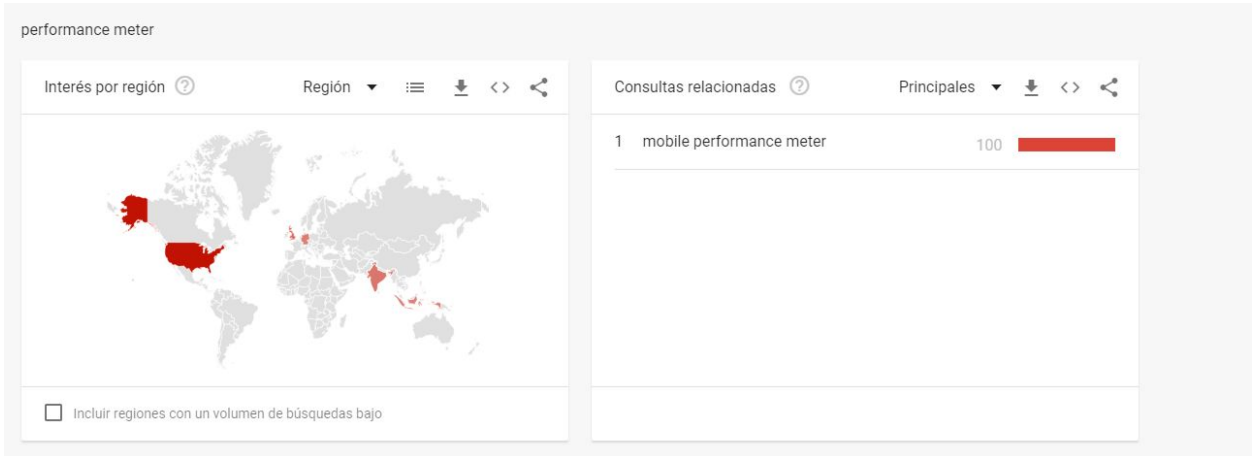
4.3 Estadísticas de explotación

4.3.1 Estudio del mercado

Como punto inicial para valorar el interés del mercado en el proyecto es interesante emplear Google Trends para analizar el volumen de búsquedas que hay en Google sobre distintas palabras clave y poder realizar una comparativa de su evolución a lo largo del tiempo.

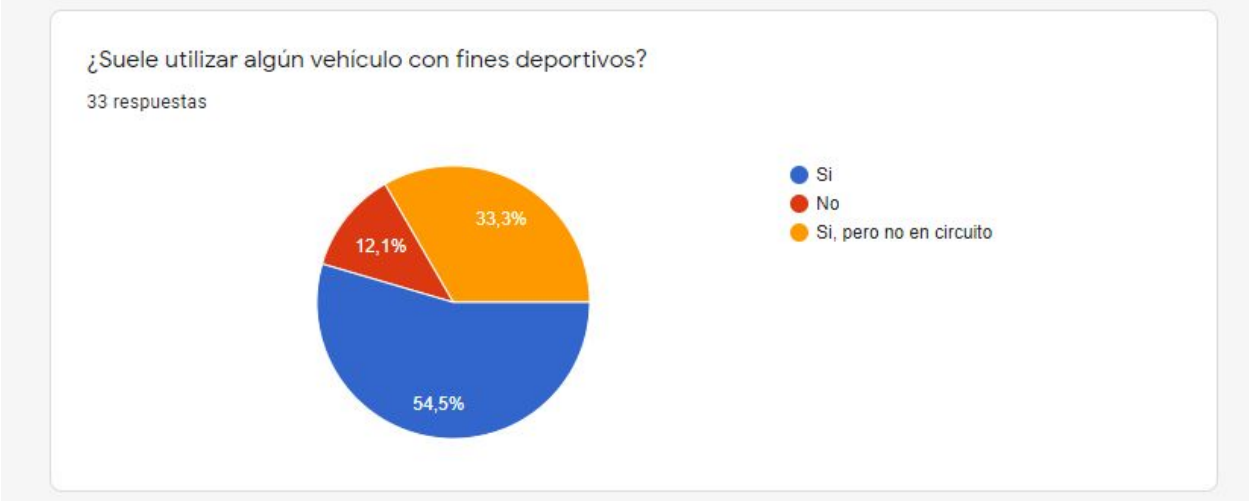


Como se puede apreciar en la captura superior. Se ha establecido como referencia la búsqueda de 'trackday' como referencia para el resto de palabras clave. Se ha añadido '*performance meter*' y '*track speedometer*' como palabras clave sobre las que basarse y 'dragy' como aplicación de referencia que hay en el mercado.

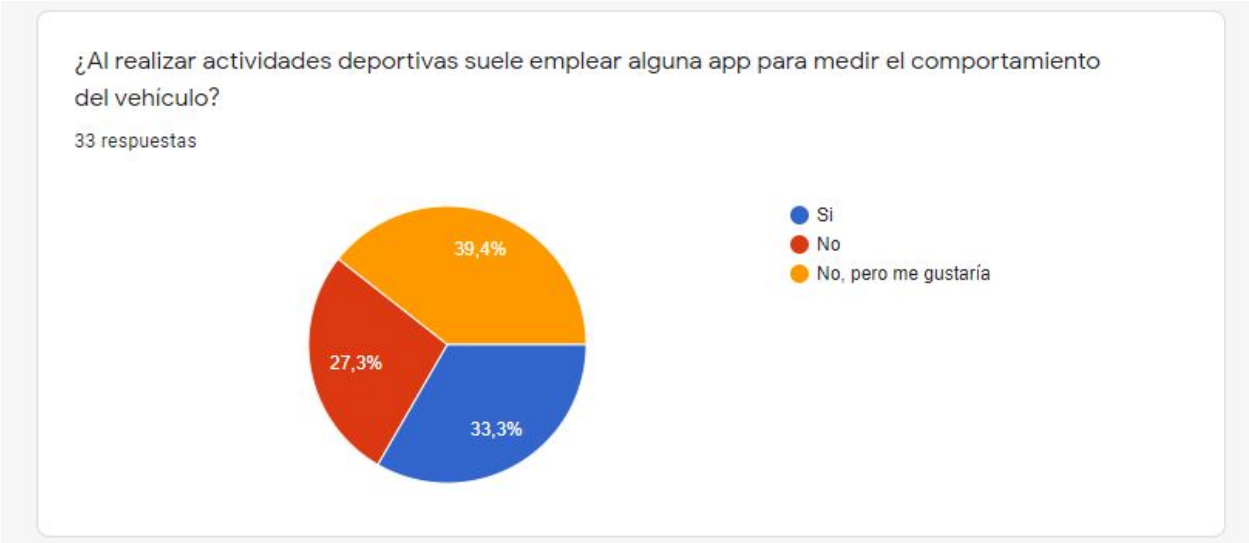


Cómo búsquedas relacionadas a '*performance meter*' sale esta relacionada con los smartphone lo cual puede ser un buen indicio de que puede haber interés en un sistema que trabaje con el smartphone pero todavía no hay una aplicación de referencia para desempeñar dicho trabajo de conseguir datos de vehículos.

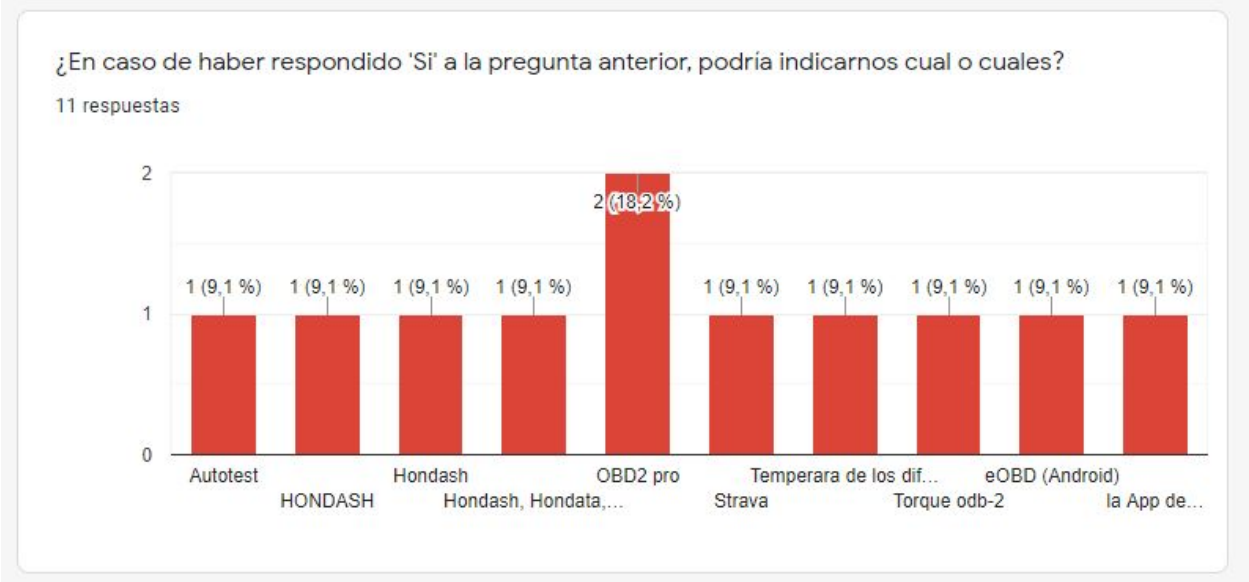
También se realizó una encuesta para valorar el interés real que tienen los posibles usuarios de un sistema como el que se ha construido en este proyecto. Ver qué alternativas emplean o conocen y qué les gustaría que tuvieran. La encuesta se compartió entre algunos grupos de mensajería instantánea destinados al mundo motor recibiendo 33 respuestas. Estas 33 respuestas han estado condicionadas por el límite de tiempo que ha tenido el proyecto para poder ser finalizado y entregado.



Como punto de partida se preguntó si los encuestados empleaban algún vehículo con fines deportivos tanto dentro como fuera de un circuito. Esto puede ayudar a saber cuánta gente estaría interesada tanto para destinarlo a un circuito como fuera de él.
Un 54,5% dispone de un vehículo para emplear en circuito, pero un 33,33% lo emplea fuera de un circuito y debería ser tomada en cuenta la proporción.

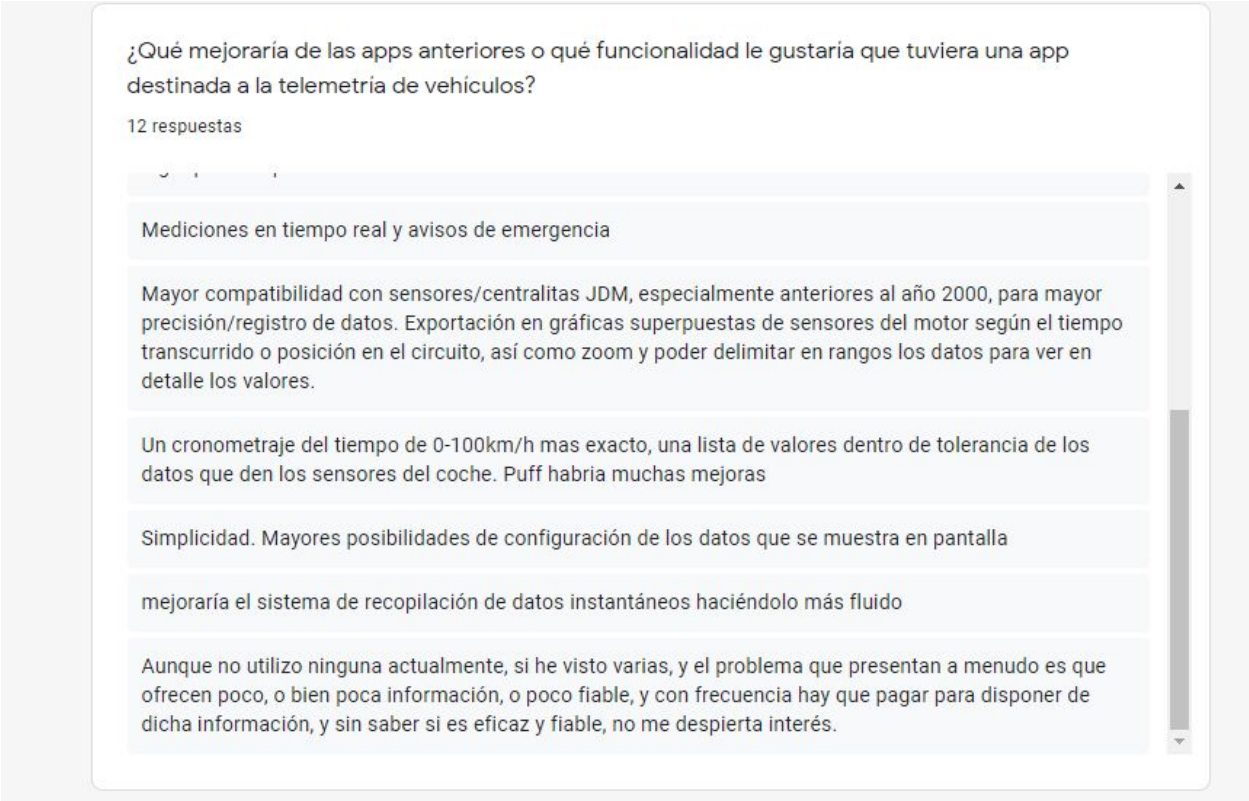


Las respuestas a la segunda pregunta revelan que una gran parte (39,4%) no emplean todavía algún tipo de aplicación para medir el comportamiento del vehículo pero les gustaría. Podría ser buen indicio de que todavía no se ha extendido el uso de este tipo de aplicaciones y que hay gente buscando una que les guste.



En la tercera pregunta, la mayoría de respuestas se enfocan en alguna aplicación o dispositivo enfocado a aprovechar la tecnología OBD de los automóviles. Entre ellas se encuentra la aplicación 'Torque' la cual se ha contemplado en este proyecto. Como curiosidad algún usuario emplea Strava, una aplicación enfocada a corredores y ciclistas para hacer un seguimiento de sus actividades con el vehículo. Esto puede ser un indicio de lo que la gente quiere y que busca cierto componente social. También se encuentra 'la App de Garmin', en Google Play se puede

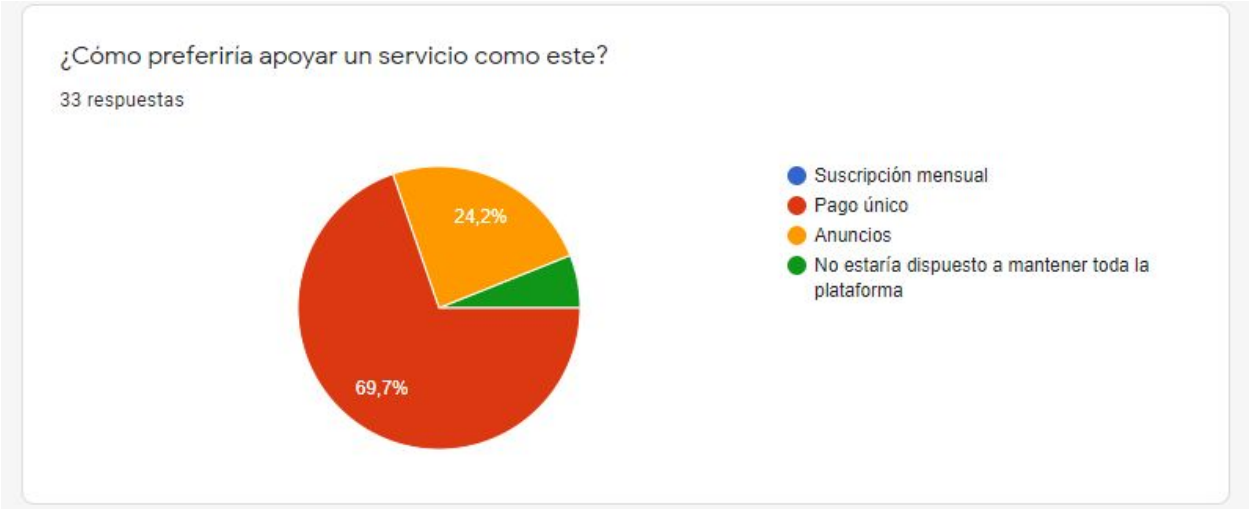
encontrar como Garmin Connect. Es una aplicación parecida a Strava pero centrada en los wearables de su propia marca.



En la cuarta pregunta, donde se pide facilitar qué mejorarían de las aplicaciones que ya utilizan la mayoría se centra en explotar más la tecnología OBD de los vehículos. Entre algunas respuestas, mejorar la precisión y poder recibir más parámetros de los que reciben la mayoría de aplicaciones.

Como quinta pregunta se quería saber la importancia de un apartado 'social' en un sistema como este. El factor social puede ser un factor clave a la hora de despertar el interés de posibles usuarios y como desvelan las respuestas podría haber un gran interés en que se incluyera un apartado social.

Otro de los puntos que se consideran clave para que tenga éxito es el precio de dicho sistema y como refleja la siguiente pregunta es un factor muy importante para que el público se decida a adoptar una solución de este estilo para sus vehículos.



Finalmente, también se preguntó de qué forma preferirían apoyar al proyecto quedando como claro ganador el pago único (69,7%) y por detrás con una clara diferencia los anuncios (24,2%).

Como conclusión de la encuesta se podría sacar en claro que puede haber todavía cierto interés por un sistema así ya que las aplicaciones que ya hay en el mercado o no son conocidas o no terminan de gustar lo cual podría ser clave estudiar la tecnología OBD para ampliar la funcionalidad del sistema ya creado. Por otra parte no hay que olvidar que el factor social puede ser clave y algunas de las aplicaciones analizadas no incluyen nada al respecto. También se tendría que valorar el factor de cómo conseguir apoyo en caso de lanzarlo a un público masivo y analizar con más detalle si el pago único y/o los anuncios podrían ser una buena fuente de ingresos con la que mantener toda la infraestructura.

4.3.2 Estudio comparativo

Aquí se va a ampliar información sobre las aplicaciones del estado del arte con el objetivo de ver el volumen de descargas o ventas que manejan.

Track Addict: Su última actualización es del 30 de Octubre de 2019 y en total lleva acumuladas más de 100.000 descargas con distintos pagos 'in-app' (haciendo uso de la propia aplicación).

Torque Pro: Si bien se trata de una aplicación de pago (su coste es de 3,55€) cuenta con más de un millón de descargas. Su última actualización es del 15 de noviembre de 2019 y no cuenta con pagos extra. Como detalle importante, para su funcionamiento requiere un dispositivo OBD que se conecte al vehículo y que transmita mediante WiFi o Bluetooth los datos.

PACE Car: Se trata de una aplicación gratuita que cuenta con más de 10.000 descargas. Su última actualización es del 14 de mayo de 2020. Como punto importante, se requiere un dispositivo de la propia compañía valorado en 119€, el PACE Link el cual es un dispositivo OBD que se conecta al vehículo pero desarrollado por la propia compañía.

GPS Race Timer: Esta aplicación cuenta con más de 50.000 descargas. No cuenta con ningún tipo de pago ni para su compra ni una vez se esté usando la aplicación. Su última actualización es del 17 de mayo de 2020. Contiene anuncios.

RaceChrono: Esta aplicación cuenta con más de 100.000 descargas. Dispone de productos disponibles para su compra dentro de la aplicación con precios entre los 4,19€ y los 19,99€. Su última versión data del 8 de junio de 2020.

5 Conclusiones

5.1 Conclusiones personales

La aplicación JosTrack y todo el sistema que lo compone se ha creado con la intención de ayudar a todo aquel que requiera poder conseguir métricas de su vehículo a un bajo coste y poder analizar a posteriori dichas métricas.

Este sistema ha sido desarrollado con la intención de acercar las nuevas tecnologías al mundo motor el cual, fuera de las grandes competiciones se encuentra todavía un paso por detrás.

Tras meses de trabajo, se ha llegado a un sistema bastante fiable para comprobar si el concepto inicial era válido y asentar las bases para un posible nuevo servicio o producto.

Personalmente, este proyecto me ha servido para afianzar los conocimientos adquiridos a lo largo de toda la carrera y descubrir nuevas tecnologías con las que trabajar, por ejemplo Android y el desarrollo de aplicaciones o trabajar con Docker. Son dos tecnologías muy interesantes que he ido aprendiendo durante el desarrollo de este proyecto y que tengo previsto seguir perfeccionando al finalizar la carrera.

Por otra parte y siguiendo con el punto anterior, algunos conocimientos no he conseguido perfeccionarlos como me hubiese gustado o haberlos desarrollado de otra forma. Si se volviese a desarrollar un sistema como este de cero seguramente cambiaría el enfoque de desarrollo planteado inicialmente implicando mucho más tecnologías de la nube como Firebase e integrando un cloud más robusto e inteligente aprovechando tecnologías como puede ser Node.js, por ejemplo.

Realizar un proyecto como este me ha hecho reflexionar acerca de muchas asignaturas que se imparten a lo largo de toda la carrera y que al haber realizado un proyecto de este calibre me he dado cuenta de la importancia de estas. Por ejemplo la asignatura de Ingeniería del Software.

Este proyecto me ha motivado a seguir aprendiendo nuevas tecnologías y sentar las bases sobre qué campos he considerado estudiar antes para poder seguir avanzando como profesional.

5.2 Futuras líneas de desarrollo

- Añadir certificado SSL al servidor Apache y Grafana para poder realizar conexiones seguras.
- Ajuste gráfico del velocímetro según vehículo del usuario introducido en el perfil.
- Añadir sensor de calidad del aire para medir la calidad del aire ambiente. Puede ser interesante en vehículos de motor de combustión interna.
- Publicar la aplicación Android en Google Play Store.
- Añadir soporte opcional para la tecnología OBD disponible en muchos automóviles.
- Sensorizar vehículo impulsado por etanol desarrollado en 'Generación Espontánea'.

6 Bibliografía

Ntoskrnl. *Android Widgets. SpeedometerGauge*. <https://github.com/ntoskrnl/AndroidWidgets> [Consulta: 22 de Febrero de 2020]

Manish Kumar. *Android SpeedoMeter Tutorial using Google Location Service.* <https://www.simplifiedcoding.net/android-speedometer-tutorial/> [Consulta: 4 de Febrero de 2020]

Android Developers. *Location*. <https://developer.android.com/reference/android/location/Location> [Consulta: 1 de Febrero de 2020]

Android Developers. *Diálogos.* <https://developer.android.com/guide/topics/ui/dialogs?hl=es-419> [Consulta: 15 de Febrero de 2020]

Alex Céspedes. *Cómo obtener la altitud latitud y longitud del gps*. <https://www.androfast.com/2018/07/como-obtener-la-altitud-latitud-y-logintud-del-gps.html> [Consulta: 5 de Febrero de 2020]

Adrián Catalán. *Curso Android: Trabajar con el acelerómetro.* <http://www.maestrosdelweb.com/curso-android-sensores-trabajar-con-acelerometro/> [Consulta: 1 de Febrero de 2020]

Naylampmechatronics. *Configuración del módulo bluetooth HC-06 usando comandos AT.* <https://naylampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html> [Consulta: 23 de Febrero de 2020]

Adafruit. *Adafruit-MLX90614-Library.* <https://github.com/adafruit/Adafruit-MLX90614-Library> [Consulta: 23 de Febrero de 2020]

Luís Llamas. *ARDUINO Y EL TERMÓMETRO INFRARROJO A DISTANCIA MLX90614.* <https://www.luisllamas.es/arduino-y-el-termometro-infrarrojo-a-distancia-mlx90614/> [Consulta: 23 de Febrero de 2020]

Diego. *Conectar Con Un Dispositivo Bluetooth Vinculado.* <https://es.stackoverflow.com/questions/166466/conectar-con-un-dispositivo-bluetooth-vinculado> [Consulta: 1 de marzo de 2020]

Ugeek. Mariadb.
<https://hub.docker.com/r/ugeek/mariadb> [Consulta: 5 de Abril de 2020]

Shariar *Shovon. How to Install Docker on Raspbian OS.*
<https://linuxhint.com/install_docker_on_raspbian_os/ > [Consulta: 2 de Abril de 2020]

Rodrigo Tomé Nieto. *Docker 2, guía de instalación de Portainer.*
<https://www.midomotica.com/instalar-portainer/ > [Consulta: 2 de Abril de 2020]

Toni Miquel Llull. *Configurar no-ip para Raspberry Pi y de paso, qué es no-ip.*
<https://www.realdroid.es/2016/10/29/configurar-no-ip-para-raspberry-pi-y-de-paso-que-es-no-ip/> [Consulta: 8 de Abril de 2020]

Grafana Docs. Run Grafana Docker image.
<https://grafana.com/docs/grafana/latest/installation/docker/> [Consulta: 8 de Abril de 2020]

Belal Khan. *Android Volley Tutorial – User Registration and Login.*
<https://www.simplifiedcoding.net/android-volley-tutorial/> [Consulta: 14 de Abril de 2020]

COMO SE HACE. "Guardar datos en una base de datos online MySQL desde una aplicación Android" en YouTube.
<https://www.youtube.com/watch?time_continue=3&v=_YjNXTybueY&feature=emb_logo> [Consulta: 13 de Abril de 2020]

Mark Drake. *Cómo instalar en Ubuntu 18.04 la pila LAMP — Linux, Apache, MySQL y PHP.*
<https://www.digitalocean.com/community/tutorials/como-instalar-en-ubuntu-18-04-la-pila-lamp-linux-apache-mysql-y-php-es> [Consulta: 22 de Abril de 2020]

Sofija Simic. How To Install Docker On Ubuntu 18.04 Bionic Beaver.
<https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04> [Consulta: 22 de Abril de 2020]

Eduardo Medina. *Kotlin superaría a Java como lenguaje más usado en Android a finales de 2018.*
<https://www.muylinux.com/2017/10/17/kotlin-superar-java-android-finales-2018/> [Consulta: 16 de Junio de 2020]

Samuel Fernández. *Ni Java ni C++, Kotlin pasa a ser el lenguaje preferido por Google para desarrollar apps en Android.*
<https://www.xatakandroid.com/programacion-android/no-hara-falta-aprender-java-para-programar-android-kotlin-pasa-a-ser-preferido-google> [Consulta: 16 Junio de 2020]

7 Acrónimos

- SBC: Single Board Computer
- USB: Universal Serial Bus
- WiFi: Wireless Fidelity
- GSM: Global System for Mobile communications
- 4G: Cuarta generación de tecnologías de telefonía móvil
- AWS: Amazon Web Services
- OBD: On Board Diagnostics
- GPRS: General Packet Radio Service
- GPS: Global Positioning System
- PCB: Printed Circuit Board
- LoRa: Long Range Modulation
- WAN: Wide Area Network
- 3G: Tercera generación de tecnologías de telefonía móvil
- XML: eXtensible Markup Language
- PHP: Hypertext Pre-Processor
- JS: JavaScript
- IDE: Integrated Development Environment
- CSS: Cascading Style Sheet
- SQL: Structured Query Language
- DB: Data Base
- HTTP: HyperText Transfer Protocol
- IP: Internet Protocol. *Muchas veces empleado para referirse a dirección IP*
- SSH: Secure SHell
- ID: IDentification
- Apk: Android Application Package
- API: Application Programming Interface
- JSON: JavaScript Object Notation
- PIN: Personal Identification Number
- ISO: International Organization for Standardization.
- SD: Secure Digital
- APT: Advanced Packaging Tool
- URL: Uniform Resource Locator
- LTS: Long-Term Support
- HTML: HyperText Markup Language

8 Anexos

8.1 Codigos

Repositorio en GitHub: https://github.com/f3rran/jos_track