



HACKATHON
ALLWIZE

2021

WizeTher

Enrique Fernández Sánchez
Lucía Francoso Fernández

Índice

1	Contexto del proyecto	3
1.1	<i>Smart City</i> y IoT	3
1.2	Análisis del reto planteado	3
1.2.1	Retos a conseguir	3
1.2.2	Hardware proporcionado en la hackathon	5
1.3	Identificación de necesidades	6
1.3.1	Ruido	7
1.4	Líneas futuras	7
2	Solución planteada: <i>WizeTher</i>	8
2.1	Qué es	8
2.2	Qué usa	8
2.2.1	Anotaciones sobre sensores	10
2.3	Plataforma web	11
2.3.1	Acceso a la plataforma web	12
2.3.2	Funcionalidades de la plataforma web	13
2.4	Open API	17
3	Resumen	18
4	Anexo. Detalle técnico sobre infraestructura <i>Wizether</i>.	20
4.1	Servidor ranii.pro	21
4.2	Aplicación web Wizether.	21
4.3	Estación Wizether.	22
4.4	Mosquitto + NodeRed	23

1 Contexto del proyecto

1.1 Smart City y IoT

En un mundo cada vez más digitalizado, donde los problemas que genera el aumento de la población mundial son cada vez mayores, resulta de imperiosa necesidad conocer y extender el concepto de *Smart City*. Podemos entender la *Smart City*, o *Ciudad Inteligente*, como una ciudad consciente, que en su toma de decisiones tiene en cuenta tanto factores sociales como medioambientales, sumados a los factores más técnicos que ya se vienen teniendo en cuenta.

Tanto es así que una *Ciudad Inteligente* es capaz de identificar los diferentes problemas, de distintos ámbitos, seccionarlos y abarcárselos, con ayuda de la tecnología, para solventarlos sin olvidar que la solución tendrá un impacto social y medioambiental que tenemos que tener en cuenta. Así surge el concepto de IoT, donde la tecnología se fusiona con *las cosas* a través de Internet (de ahí su nombre, *Internet Of Things*), consiguiendo una poderosa herramienta para monitorizar aquello que sea necesario para la futura solución de un problema, recogiendo datos y subiéndolos a la nube, donde otras poderosísimas herramientas son capaces de realizar análisis, predicciones... de esos datos, reduciendo también el costo en tiempo y humano de toma y tratamiento de datos (o incluso, introduciéndolos, ya que quizás no existía monitorización, aunque fuera manual, de esos parámetros sin la introducción del IoT). Ejemplos de IoT hay muchos, y los encontramos en el sector automovilístico, científico, las telecomunicaciones, doméstico, industrial, etc.

1.2 Análisis del reto planteado

En relación a lo anteriormente comentado, y una vez entendida la importancia de los conceptos expuestos, se presta realizar un análisis del reto planteado en esta hackathon, donde se pone de manifiesto que la solución que debemos aportar debe girar en torno a la calidad ambiental. Por lo tanto, lo primero que hay que hacer es estudiar qué existe en tanto a soluciones que mejoren la calidad ambiental en general y en particular, en la ciudad de Cartagena.

Sea cual sea la solución planteada finalmente, no hay que olvidar que, lo más importante, es tener en claro lo que queremos solucionar, definir bien nuestro objetivo para solventar el problema, ya que cada problema, según la filosofía que impera en la *Smart City*, necesita una solución específica. Asimismo, la solución debe contar con una serie de características para que sea una buena solución (a ser posible, debe procurarse que todas o la mayoría estén contempladas a la hora de desarrollar el prototipo de la solución). Estas suponen un reto en si mismo, ya que definir un producto o una idea es complicado; así pues, las enumeramos en el siguiente apartado.

1.2.1 Retos a conseguir

- **Seguridad.** Cifrado en la comunicación, evitar el problema de *Man in the Middle* (filtraciones o robo de datos por parte de terceras personas).
- **Autonomía.** Equipos donde no se requiera cambiar las baterías, o si se cambian que sea cada mas tiempo. Búsqueda de duración de baterías de 10 años o que hagan harvesting de luz solar.

- **Conectividad.**
- **Interoperabilidad.** Un sistema capaz de comunicarse con otros sistemas (mismas o diferentes tecnologías).
- **Conocimiento y conciencia.** Conocer la naturaleza de los datos que se necesita recabar, recopilar sólo los necesarios y asegurar la privacidad de los mismos.
- ***Big data.*** Recopilación de datos. Como se ha mencionado en el apartado anterior, solo la información necesaria.
- **Modelo de negocio.** Monetización y sostenibilidad de la aplicación.
- **Escalabilidad.** Escalabilidad de la solución. Optimización de la misma.

1.2.2 Hardware proporcionado en la hackathon

Placas controladoras

- AllWize K2. Arduino SAMD.
- Carrier board para K2 (sin batería).
- AllWize K1. Wemos 8266 R1 D2 mini
- x2 Antenas monopolo de cuarto de onda de 168MHz.
- IPEX a SMA
- x2 Cables micro USB
- Cables grove

Sensores incluidos

- Multichannel Gas Sensor v2. I2C. 4 Variables.
- Barometer Sensor (BME280). Presión atmosferica, altitud, temperatura, humedad.
- Dust Sensor. Air quality
- Slide potenciomenter 10k ohms
- RGB Led. WS2813 mini
- OLED display. 0.96”
- Hall sensor. Magnetico.
- Touch Sensor.
- Encoder.
- Loudness sensor.
- Air quality sensor.
- Sound sensor.

1.3 Identificación de necesidades

Una vez analizado el material que se nos ha proporcionado, y reflexionado acerca de la tecnología a emplear (en este caso, Wize), el siguiente paso fue hacer una investigación acerca del OpenData que ofrece el Ayuntamiento de Cartagena, así como de posibles líneas futuras que ha manifestado un interés en abarcar en cuanto al empleo de soluciones IoT con el objetivo de conseguir suficientes datos para toma de decisiones que conlleven una mejora en la calidad ambiental.

Dentro del [Portal de Transparencia del Ayto de Cartagena \(Ciudad Sostenible\)](#), en tanto a calidad ambiental, encontramos:

- El apartado *Infraestructuras*, dentro del cual están los subapartados: agua potable, agua reciclada y reutilizada, e impactos ambientales.
- El apartado *Desarrollo sostenible*, dentro del cual están los subapartados: gases de efecto invernadero, niveles de calidad del aire, medio natural y mapa de ruidos.

Visualizando el contenido de dichos apartados, nos dimos cuenta de que sólo el subapartado [*Calidad del Aire \(Campo de Cartagena\)*](#) está actualizándose todos los días. En [*Mapa de ruidos*](#) no conseguimos visualizar fechas en los documentos generados. Destacamos un documentos PDF en el subapartado *Gases de efecto invernadero*, llamado “*Inventario de emisiones y plan de acción para la energía sostenible del municipio de Cartagena*”, que propone una serie de objetivos para el año 2020 de cara a reducir la huella de carbono en la ciudad de Cartagena; entendemos que es una preocupación por parte del Ayuntamiento el poner medios para dicha mejora.

Centrándonos en el subapartado de *Calidad del aire*, encontramos un dashboard donde se reflejan los datos recopilados por diferentes estaciones, dentro de la zona del campo de Cartagena, relacionados con la calidad del aire. Dichas estaciones son, o se encuentran en:

- Estación de Alumbres
- Estación de Escombreras
- Estación de la Aljorra
- Estación de Monpeán

Por las gráficas que se muestran, podemos observar que cada estación realiza varias mediciones al día de diferentes gases contaminantes; a la vez que dicha gráfica con datos variantes a lo largo del día (de ayer, hoy o previsión para mañana, según seleccionemos), se muestra una tabla, donde intuimos que se muestra el valor medio de concentración de dicho gas contaminante en el aire. Sin embargo, no existe (o al menos, no hemos encontrado) un portal con *OpenData* donde descargar en archivo csv/excel estos datos, ni un histórico con datos recopilados en días anteriores al de la consulta (como decimos, sólo existe un dashboard con datos del día anterior, actual y previsión para el siguiente).

1.3.1 Ruido

Como hemos comentado previamente, desconocemos la fecha de actualización de dichos datos. Además, no existe un *OpenData* donde descargar históricos de datos en un archivo CSV o Excel; se nos proporcionan unos mapas de calor marcando los sitios donde hay más ruido a lo largo del día. Están disponibles a través de [*Acceso a mapas de ruido*](#).

1.4 Líneas futuras

En cuanto a proyectos futuros en relación a la calidad ambiental en la ciudad de Cartagena, hemos podido leer que el ayuntamiento de Cartagena ha mostrado interés, desde la Concejalía de Ciudad Sostenible y Proyectos Europeos, por la monitorización de la calidad del aire, ruido y aforos peatonales ([*Monitorización calidad aire, ruido y aforos \(Ayto Cartagena\)*](#))

2 Solución planteada: *WizeTher*

Una vez hemos visto qué existe en la ciudad de Cartagena en tanto a calidad ambiental, y entendida la tecnología que debemos usar para esta hackathon, definimos *WizeTher*. El nombre es el resultado de la combinación de las palabras *Wize* y *Weather*, la tecnología radio que usa y el resumen de su funcionalidad principal.

2.1 Qué es

WizeTher es, por un lado, una estación meteorológica capaz de recoger datos sobre:

- Temperatura
- Humedad
- Presión atmosférica
- Ruido
- Calidad del aire

Por otro lado, *WizeTher* es una plataforma web donde el usuario podrá dar de alta su estación meteorológica y visualizar datos, además de descargar en formato CSV los que sean de su interés.

Esta solución se ha pensado con el objetivo concreto de añadir valor y servir de apoyo al despliegue que ya tiene hecho el Ayuntamiento de Cartagena para medir los niveles de calidad de aire y ruido, principalmente. Desglosado por funcionalidades, la solución recogerá los datos sobre los parámetros arriba enumerados, los subirá a la nube y en su plataforma web los mostrará en gráficas, a la vez que permitirá su consulta y descarga de manera totalmente gratuita.

2.2 Qué usa

La estación meteorológica *WizeTher* hace uso de componentes incluidos en el kit que se nos proporcionó; en concreto, hace uso de:

- *AllWize K2*
- Carrier board para K2
- Antena monopolo lambda cuartos de 168 MHz
- Sensor barométrico BME280 (*Barometer Sensor*). Mide presión atmosférica, temperatura y humedad
- Sensor de ruido (*Loudness Sensor*)
- Sensor de calidad de aire (*Air Quality Sensor v1.3*)

- Pantalla OLED 0.96" (*OLED Display 0.96 inch*)
- Sensor táctil (*Touch sensor*)

Los últimos dos componentes enumerados se han incorporado para mejorar la interacción entre el usuario y la estación, de manera que la pantalla es capaz de mostrar los últimos valores medidos por los sensores BME280, ruido y calidad de aire, y el usuario a través del sensor táctil puede navegar por la pantalla; esto es, cada vez que pulsa el sensor táctil, visualiza los datos más recientes de un sensor en concreto.

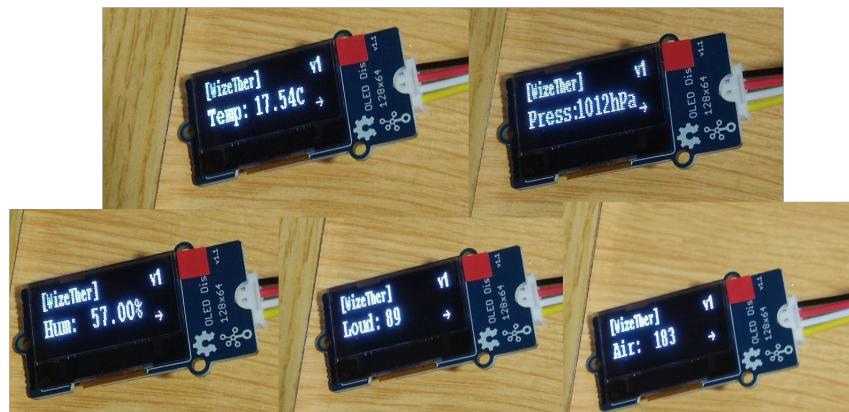


Figura 1: Vistas de las diferentes pantallas *Wizether*

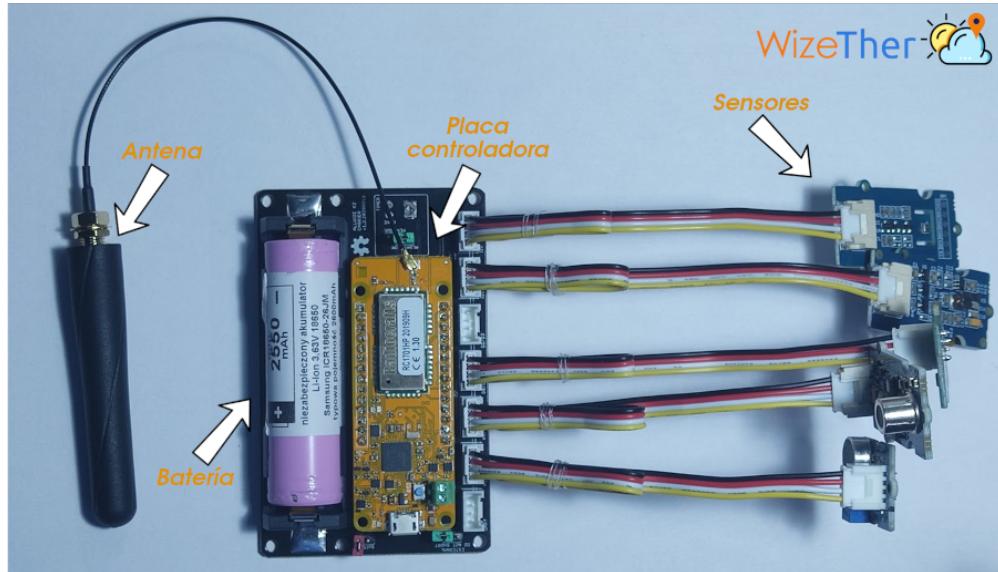


Figura 2: Infografía de una estación *Wizether* (versión 1)

A medio y largo plazo, se requeriría de un despliegue de una red más amplia de estaciones *WizeTher*, y por consecuente de gateways *Wize*, para crear una red *Wize* en la ciudad de Cartagena. De esta manera, el volumen de datos será mayor y más significativo, de cara a la toma de decisiones.



Figura 3: Fotografía de un gateway *Wize*

Cabe anotar que este elemento, el gateway (ajeno a la estación pero necesario para la creación de una red), ejerce la función de recibir los mensajes de una estación y subirlos a la nube a través de MQTT. Además, cada 5 minutos envía un mensaje MQTT al servidor indicando que el gateway sigue activo (detectando así posibles caídas del servicio desde el otro extremo).

2.2.1 Anotaciones sobre sensores

Los sensores de calidad de aire y de ruido miden valores cualitativos, es decir, no proporcionan un valor exacto pero nos aportan valores relativos a la medida tomada. En el código que le cargamos a la placa controladora establecemos los márgenes cualitativos (son personalizables), que actualmente son:

- Para el **sensor de calidad de aire**: *Fresh air* (0 a 180), *Low pollution* (180 a 300) y *High pollution* (más de 300, máximo de 400).
- Para el **sensor de ruido**: *Good* (0 a 150), *Acceptable* (150 a 250) y *Bad* (250 a 350).

2.3 Plataforma web

Pensamos que para una mejor integración de la solución era necesaria la creación de una plataforma web, a la cual podemos acceder a través de *Página web con acceso a plataforma web de WizeTher*.

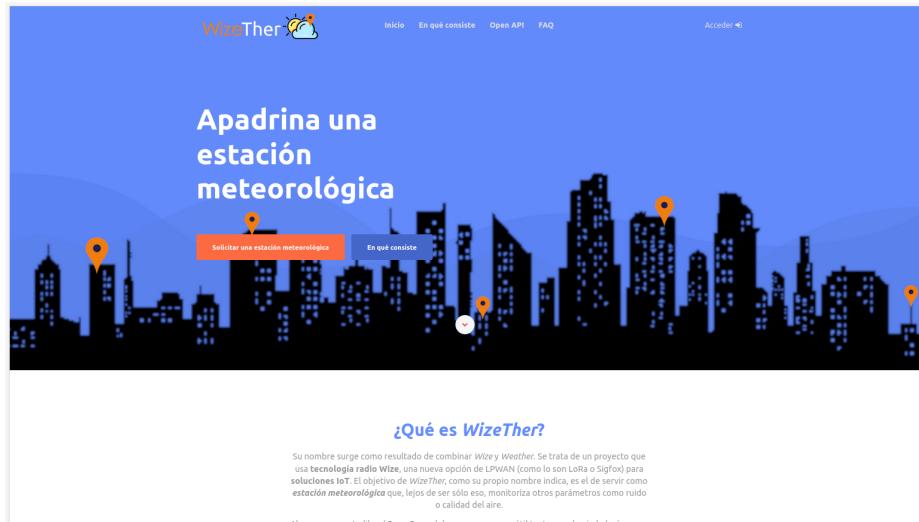


Figura 4: Captura web WizeTher (inicio)

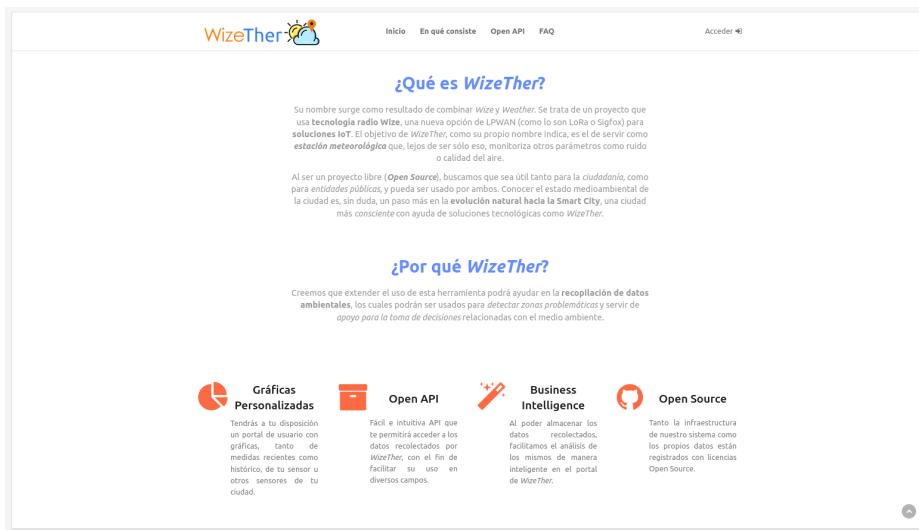


Figura 5: Captura web WizeTher (Sobre WizeTher)

2.3.1 Acceso a la plataforma web

Se ha habilitado el acceso a la plataforma a través de un usuario de prueba (clicando sobre *Acceder* en la esquina superior derecha de la página web):

- Usuario: wizether@ranii.pro
- Contraseña: wizether2021

Por motivos de seguridad, se ha deshabilitado temporalmente el registro de usuarios (sólo los administradores de la plataforma pueden crear nuevos usuarios).



Figura 6: Captura inicio sesión en *WizeTher*



Figura 7: Captura registro de nuevo usuario en *WizeTher*

2.3.2 Funcionalidades de la plataforma web

Lo primero que aparece al iniciar sesión en la plataforma son una serie de gráficas donde se pueden visualizar datos recogidos por sensores de estaciones *WizeTher* durante las últimas 24 horas. Además, se irán actualizando conforme el servidor vaya recibiendo datos nuevos.



Figura 8: Captura bienvenida portal en *WizeTher*

En este apartado aparecerán las diferentes estaciones que se hayan dado de alta en el servidor, junto con su ubicación (longitud, latitud e interior/exterior) y su estado (apagada, activa o durmiendo).

The screenshot shows the WizeTher web interface. On the left is a sidebar with navigation links: Portal, Inicio, ESTACIONES (with 'Mis estaciones' selected), DATOS, and USUARIO. The main content area has a header 'Show 10 entries' and a search bar 'Search:'. A table displays station information with columns: Número de serie, Latitud, Longitud, Localización, and Versión. One entry is shown: 20212230, 37, -0.9, in-door, v1. Below the table, it says 'Showing 1 to 1 of 1 entries'. At the bottom right are 'Previous' and 'Next' buttons.

Número de serie	Latitud	Longitud	Localización	Versión
20212230	37	-0.9	in-door	v1

Figura 9: Captura *Mis estaciones* en *WizeTher*

En el apartado *Nueva estación* podemos dar de alta una estación en el servidor, introduciendo los datos *Número de serie*, *Longitud* y *Latitud*, pudiéndonos ayudar para ubicar la estación del mapa que aparece a la derecha (haciendo clic sobre él, se autocompletan los datos de *Longitud* y *Latitud*). Además, se requiere seleccionar dónde está ubicada la estación, en interior o exterior.

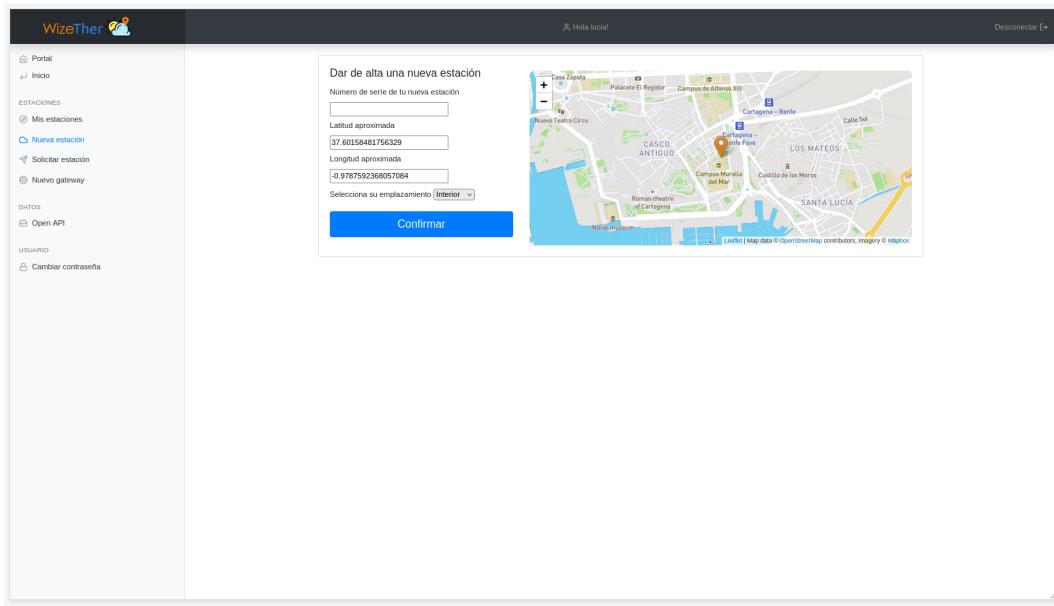


Figura 10: Captura *Nueva estación* portal en *WizeTher*

En el apartado *Nuevo gateway* podemos añadir un nuevo gateway completando los mismos parámetros que en el apartado *Nueva estación*, y además nos encontramos con un nuevo campo (*Potencia de transmisión*) que nos permite modificar el alcance visual de nuestro gateway (se observa en el mapa).

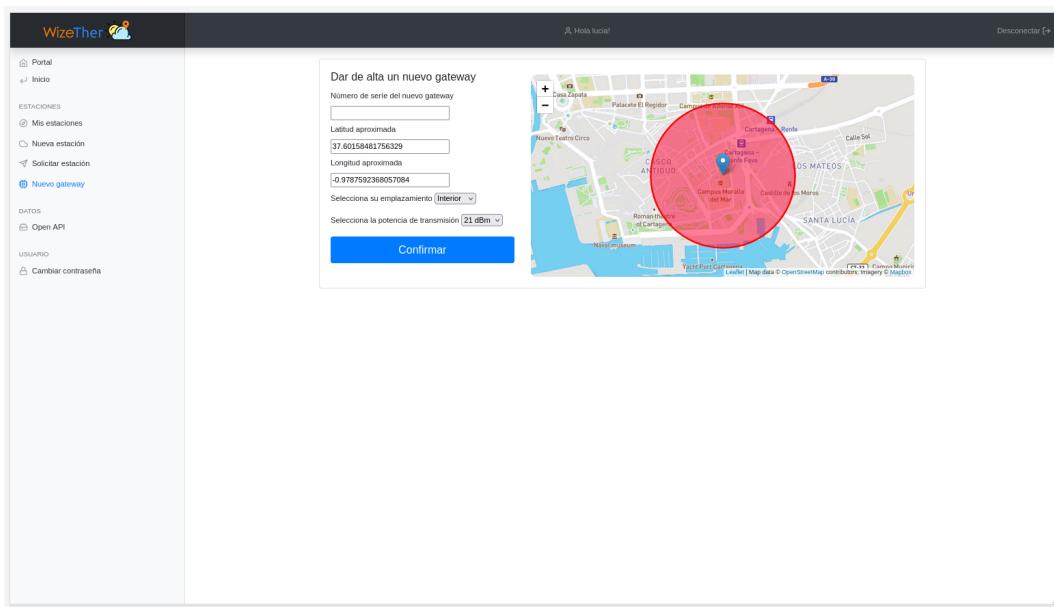


Figura 11: Captura *Nuevo gateway* portal en *WizeTher*

2.4 Open API

Como funcionalidad extra que vimos necesaria para complementar el estudio medioambiental, está la *Open API*. Una API es una interfaz de programación que nos permite facilitar el acceso a la información de nuestra base de datos; en este caso, la salida de la API es en formato CSV. A través de ella podemos realizar una serie de consultas que nos devuelven los datos relacionados que nos encontramos en la base de datos.

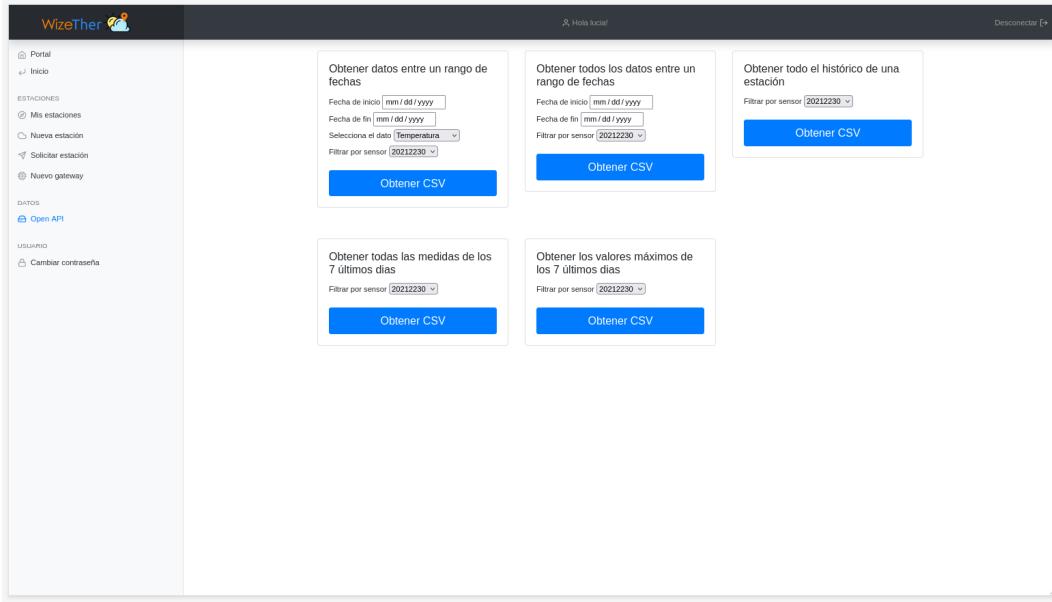


Figura 12: Captura *Open API* en *WizeTher*

Por último, anotar que el nombre de *Open API* viene de una de las funcionalidades propias de *WizeTher*, que es la de permitir a los usuarios acceder a los datos de las estaciones registradas en el servidor; además, también recuerda a *Open Source*, una característica del proyecto, el ser ofrecido bajo esa filosofía, por lo que los datos que se descargan a través de la API son *open*.

3 Resumen

Nuestro objetivo antes de empezar esta hackathon fue el de intentar proponer una solución sencilla pero completa; en tanto a *sencilla*, que el concepto fuera sencillo, que supiéramos en todo momento cuál sería la aplicación directa del producto creado. Así fue concebido *WizeTher*, una idea sencilla pero completa, capaz de explotar los puntos fuertes de la tecnología *Wize*, sin descuidar otros apartados técnicos (no sólo sensores o gateway, sino también la web o el propio servidor que aloja bastante porcentaje del servicio). Un producto *completo*, que fuera escalable y realizable en Cartagena, sin mucho coste de implantación; y, por supuesto, que añadiera valor a lo ya existente, y en línea a lo que ya está previsto para la ciudad de Cartagena en materia de calidad ambiental (sobretodo por parte del Ayuntamiento y empresas colaboradoras o que ejercen en la ciudad).

La resolución del reto ha seguido unas fases bien diferenciadas, que pueden resumirse en:

- **Fase 1: análisis e identificación de necesidades**
- **Fase 2: definición del producto**
- **Fase 3: programación de la placa controladora y gateway.** Programación en *Arduino* para que la placa controladora de la estación lea la información de los sensores y construya el mensaje para enviarlo vía radio al gateway. A su vez, programamos la placa controladora del gateway para que reciba el mensaje y sepa enviarlo por MQTT posteriormente.
- **Fase 4: despliegue de contenedores en servidor *ranii.pro*.** Se requiere para alojar las diferentes aplicaciones y funcionalidades de las que consta *WizeTher*.
- **Fase 5: programación MQTT y NodeRed.** Esta fase es necesaria para subir los datos a la nube, es decir, almacenarlos en la base de datos del servidor.
- **Fase 6: creación plataforma web.** Esta tarea comienza en este punto pero continua durante las siguientes fases. La plataforma web se ha programado en *Python*, y contiene las funcionalidades descritas en la sección 2, apartado *Plataforma web*.
- **Fase 7: subida de datos al servidor.** Aquí comprobamos que las tareas asociadas a las fases anteriores se han realizado con éxito, recolectando datos, es decir, dejando encendida la estación *WizeTher* con los sensores y subiendo cada 15 minutos esos datos al servidor a través de MQTT. Como anotación, decir que tanto estación como gateway como servidor permanecen encendidos en todo momento.
- **Fase 8: visualización de los datos.** Por un lado, para esta tarea se estudió cómo visualizar los datos almacenados en la base de datos del servidor (allí están los datos de los sensores de las estaciones dadas de alta) en *Grafana*. Por otro lado, se estudió cómo implementar las gráficas generadas en la plataforma web ya creada en la *Fase 6*.
- **Fase 9: creación de la *OpenAPI*.** Implementación en la plataforma web de la funcionalidad *OpenAPI*, la cual consiste en actuar como asistente de consultas a la base de datos del servidor a partir de los parámetros introducidos por el usuario (rango de fechas, sensor o estación en específico, etc). Además, durante esta fase, se añadió mapas interactivos creados

con *Leaflet* y *Mapbox*, que aparecen en los apartados de la plataforma web llamados *Nueva estación*, *Nuevo gateway* y en la página web de inicio, apartado *FAQ*.

- **Fase 10: redacción de memoria y verificación del producto.** La redacción de la memoria se ha realizado usando *LaTeX*. Durante la redacción de la misma, se ha ido revisando los pasos previos realizados y corrigiendo posibles *bugs* detectados.

Por último, y retomando la lista de retos a conseguir con la introducción de una solución IoT para un determinado problema, hemos analizado si *WizeTher* iría bien encaminado hacia la consecución de dichos retos:

-

4 Anexo. Detalle técnico sobre infraestructura *Wizether*.

Actualmente, la infraestructura que da soporte al ecosistema *Wizether*, esta compuesta por tres partes.

1. La primera parte corresponde con la captación de datos, dichos datos son detectados por las estaciones, las cuales los transmitirán a los gateway utilizando la tecnología radio *Wize*. Una vez los datos han llegado al gateway, son encapsulados en un mensaje MQTT (tipo publish) que se subirán a un topic específico en el servidor destino.
2. En la segunda parte del sistema, correspondiente con el procesado de los datos en el servidor. Nuestro broker de MQTT, mosquitto, recepciona los paquetes MQTT entrantes y actualiza a todos los clientes que previamente se han suscrito a estos topics. En nuestro caso, utilizamos NodeRed para transformar los mensajes MQTT en datos aptos para guardarlos en una base de datos de tipo Time Series, en este caso, InfluxDB (en su versión 2.0, incluyendo ventajas como son el lenguaje Flux).
3. La tercera parte de nuestro sistema, corresponde con la representación de los datos. En este apartado encontramos la aplicación Grafana, encargada de representar los datos que tenemos almacenados en InfluxDB. Por otro lado, encontramos *Wizether_webapp*, que tiene por función aportar la lógica característica a lo que sería *Wizether* (OpenAPI, portal de usuarios, registro de estaciones...)

A modo resumen, un esquemático de la infraestructura que estamos utilizando, podría ser el siguiente:

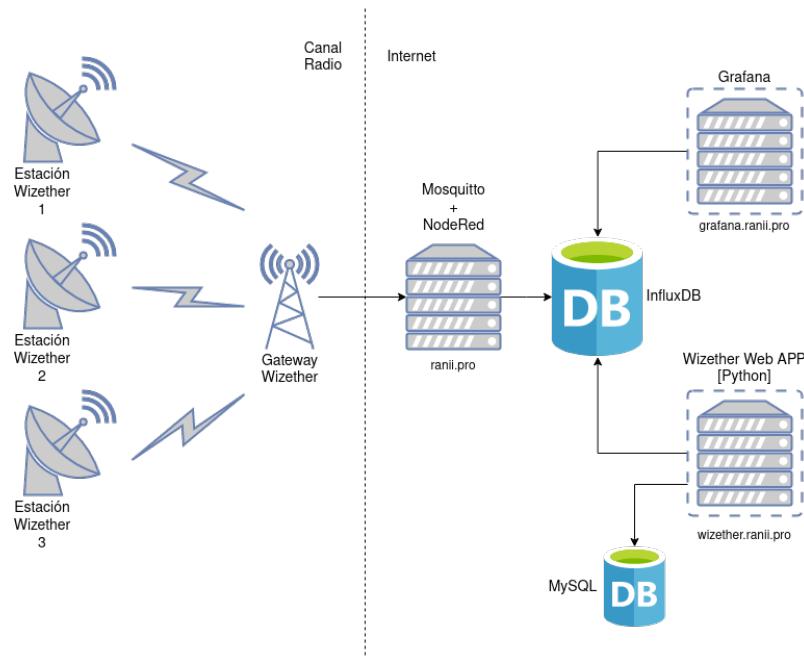


Figura 13: Esquema infraestructura *Wizether*

4.1 Servidor ranii.pro

Utilizamos un servidor dedicado para hacer de host de nuestros servicios. Para levantar cada una de las diferentes aplicaciones, hemos utilizado *Docker*, con ayuda del asistente *docker-compose*. Además, como proxy de entrada HTTP, nos hemos favorecido de [Traefik](#) (versión 2.0), facilitándonos la creación de subdominios específicos para los contenedores y aportando los certificados necesarios para establecer comunicaciones seguras entre ellos. Hemos desplegado los siguientes subdominios:

- <https://nodered.wize.ranii.pro>
- <https://grafana.wize.ranii.pro>
- <https://influxdb.wize.ranii.pro>
- <https://wizether.ranii.pro>

Además, de manera complementaria, se ha utilizado la plataforma de integración continua [Drone.CI](#) para desplegar de manera rápida y segura los servicios de *Wizether_webapp*. Esta aplicación se encuentra alojada en: <https://drone.ranii.pro>.

Disclaimer. Muchas de las páginas referenciadas en este anexo, por no decir la mayoría, tienen un login asociado. Sin embargo, creemos que es importante mencionarlas ya que sin ellas no habría sido posible el desarrollo de *Wizether*.

4.2 Aplicación web Wizether.

Para la plataforma web de [Wizether](#), hemos utilizado como lenguaje de programación Python complementándolo con el framework de desarrollo web [Flask](#). Algunas de las características importantes que dispone nuestro sistema son las siguientes:

- Página de bienvenida.
- Portal de usuarios.
- Portal de administración.
- Sistema de acceso, registro y cambio de contraseña.
- Base de datos dedicada (MySQL).
- Formulario protegidos contra CSRF.
- Tareas asíncronas utilizando redis.
- Control de acceso dependiendo del rol del usuario.
- Conexión a InfluxDB. Consultas utilizando el lenguaje Flux.
- Descarga de datos tipo Time Series en formato CSV.

- Acceso e interacción con mapas geográficos.

Todo el código de la aplicación esta público en el repositorio de GitHub: [Wizether_webapp](#).

4.3 Estación Wizether.

Para programar la estación y el *gateway*, hemos utilizado los kits proporcionados por la empresa *AllWize*, siendo estos programadas con *Arduino*. Algunas funcionalidades extras que hemos añadido a la estación han sido:

- Sensor BM280 (temperatura, humedad, presión)
- Sensor de calidad de aire
- Sensor de ruido
- Pantalla OLED que permite cambiar entre diferentes vistas
- Sensor táctil que activa el cambio entre vistas de la pantalla.
- Mandar información vía radio Wize cada 15 minutos.
- Autonomía sin cables gracias a una pila Samsung 18650 de 2600mAh.
- Monitorización del estado de la batería.

En el caso del *gateway*, hemos incorporado las siguientes funcionalidades:

- Procesado de los datos de los sensores recibidos.
- Paquete *MQTT* con toda la información medida.
- Paquete *MQTT* tipo *PING* para que el servidor pueda detectar caídas de *gateways*.

Todo el código relacionado con la estación *Wizether* y su *gateway*, se encuentra en el siguiente repositorio público de GitHub: [WizeTher código estación](#).

4.4 Mosquitto + NodeRed

En el servidor hemos configurado el *broker MQTT mosquitto*, que recepcionará los paquetes entrantes y se encargará de mandar la información a todos los clientes suscritos. En este caso, se ha incorporado *NodeRed* como cliente suscrito a los *topics* de datos y *ping*. La función principal será pasar esta información recibida por *MQTT* a una base de datos de tipo Time Series, como es *InfluxDB*. Nuestro *canvas nodered* sería tal que así:

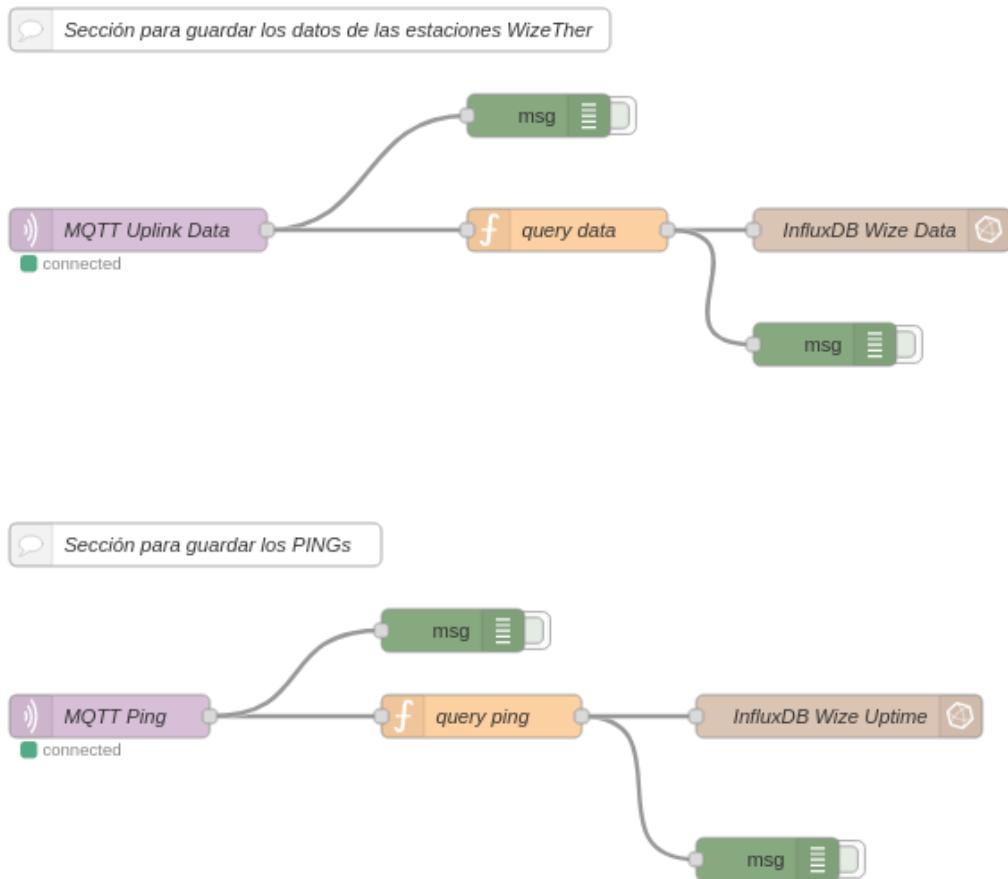


Figura 14: Canvas nodered: MQTT a InfluxDB