Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Proyecto de fin de Grado en Ingeniería Informática

Recogida de datos meteorológicos mediante sensores de bajo coste y posterior análisis de la calidad de estos.

J. Jesús Daryanani Hormiga

Dirigido por: Fernando López Ostenero

Curso 2022/2023, convocatoria Junio

Recogida de datos meteorológicos mediante sensores de bajo coste y posterior análisis de la calidad de estos.

Proyecto de fin de Grado en Ingeniería Informática

de modalidad genérica

Realizado por: J. Jesús Daryanani Hormiga

Dirigido por: Fernando López Ostenero

Fecha de lectura y defensa: 13 Julio 2023

Agradecimientos

A Dios, en sus tres personas, sin ellas todo sería nada [1].

A mis Padres, sin ellos no habría llegado hasta aquí [2].

A Esther, ejemplo a imitar del verdadero amor terrenal [3].

[1] En el periodo de edad comprendido en el intervalo: [0,x]
[2] En el periodo de edad comprendido en el intervalo: [0,y]
[3] En el periodo de edad en el intervalo deseado: [y,x]

Resumen

Este trabajo surge de la necesidad de poder disponer, de una forma económica, de datos meteorológicos obtenido desde diferentes puntos geográficos y centralizados en un único repositorio común. En ese repositorio se podrán consultar tanto los datos actuales como los históricos de cada una de las zonas en estudio. Además, se podrán consultar datos obtenidos de fuentes oficiales, como, por ejemplo, la Agencia Estatal de Meteorología (AEMET OpenData) y realizar comparativas entre los datos oficiales, obtenidos a través de Red Oficial de Estaciones Meteorológicas y los obtenidos con estos sensores de bajo coste, de manera que se pueda obtener una conclusión de la calidad y margen de error de estos.

Abstract

This work arises from the need to be able to have, in an economical way, meteorological data. roles obtained from different geographical points and centralized in a single repository common. In this repository, both current and historical data of each of the study areas. In addition, data obtained from official sources could be consulted. for example, the State Meteorology Agency (AEMET OpenData) and carry out comparisons between the official data, obtained through the Official Network of Meteorological Stations. roles and those obtained with these low-cost sensors, so that a conclusion of the quality and margin of error of these.

Palabras clave

- AEMET: El objeto de la Agencia Estatal de Meteorología es el desarrollo, implantación, y prestación de los servicios meteorológicos de competencia del Estado y el apoyo al ejercicio de otras políticas públicas y actividades privadas, contribuyendo a la seguridad de personas y bienes, y al bienestar y desarrollo sostenible de la sociedad española.

- SOAP: Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

- REST: Es una interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON. El formato más usado en la actualidad es el formato JSON, ya que es más ligero y legible en comparación al formato XML. Elegir uno será cuestión de la lógica y necesidades de cada proyecto.

- API: Es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

- METEOROLOGÍA: La meteorología (del griego 'meteoro' y logos 'conocimiento', 'tratado') es la ciencia atmosférica interdisciplinaria que estudia el estado del tiempo, el medio atmosférico, los fenómenos meteorológicos y las leyes que los rigen con apoyo de disciplinas auxiliares como la física de la atmósfera y la química de la atmósfera.

- SENSOR: Un sensor es todo aquello que tiene una propiedad sensible a una magnitud del medio, y al variar esta magnitud también varía con cierta intensidad la propiedad, es decir, manifiesta la presencia de dicha magnitud, y también su medida.

- HUMEDAD: Es la cantidad de vapor de agua presente en el aire, se puede expresar de forma absoluta mediante la humedad absoluta, o de forma relativa mediante la humedad relativa o grado de humedad. La humedad relativa es la relación porcentual entre la cantidad de vapor de agua real que contiene el aire y la que necesitaría contener para saturarse a idéntica temperatura.

- PRESIÓN: La presión atmosférica es la fuerza por unidad de superficie que ejerce el aire que forma la atmósfera sobre la superficie terrestre. El valor de la presión atmosférica sobre el nivel del mar es de 1013,25 hPa.

- TEMPERATURA: Se llama temperatura atmosférica a uno de los elementos constitutivos del clima que se refiere al grado de calor específico del aire en un lugar y momento determinados así como la evolución temporal y espacial de dicho elemento en las distintas zonas climáticas.

- SERVICIO WEB: Un servicio web (en inglés, web service o web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web.

Keywords

- AEMET: The purpose of the State Meteorological Agency is the development, implementation, and provision of meteorological services under the jurisdiction of the State and support for the exercise of other public policies and private activities, contributing to the safety of people and goods, and the well-being and sustainable development of Spanish society.
- SOAP: It is a standard protocol that defines how two objects in different processes can communicate by means of XML data exchange. This protocol derives from a protocol created by Dave Winer in 1998 called XML-RPC. SOAP was created by Microsoft, IBM, and others. It is currently under the auspices of the W3C. It is one of the protocols used in Web services.
- REST: It is an interface to connect several systems based on the HTTP protocol (one of the oldest protocols) and it is used to obtain and generate data and operations, returning that data in very specific formats, such as XML and JSON. The most widely used format today is the JSON format, since it is lighter and more readable compared to the XML format. Choosing one will be a matter of the logic and needs of each project.
- API: It is an abbreviation for Application Programming Interfaces, which in Spanish means application programming interface. It is a set of definitions and protocols used to develop and integrate application software, allowing communication between two software applications through a set of rules.
- WEATHER: Meteorology (from the Greek 'meteor' and logos 'knowledge', 'treated') is the interdisciplinary atmospheric science that studies the state of the weather, the atmospheric environment, meteorological phenomena and the laws that govern them with the support of auxiliary disciplines such as atmospheric physics and atmospheric chemistry.
- SENSOR: A sensor is anything that has a property that is sensitive to a magnitude of the medium, and when this magnitude varies, the property also varies with a certain intensity, that is, it manifests the presence of said magnitude, and also its measurement.
- HUMIDITY: It is the amount of water vapor present in the air, it can be expressed absolutely by absolute humidity, or relatively by relative humidity or degree of humidity. Relative humidity is the percentage relationship between the actual amount of water vapor contained in the air and the amount it would need to contain to become saturated at the same temperature.
- PRESSURE: La presión atmosférica es la fuerza por unidad de superficie que ejerce el aire que forma la atmósfera sobre la superficie terrestre. El valor de la presión atmosférica sobre el nivel del mar es de 1013,25 hPa.
- TEMPERATURE: Atmospheric temperature is called one of the constitutive elements of the climate that refers to the degree of specific heat of the air at a given place and time, as well as the temporal and spatial evolution of said element in the different climatic zones.

- WEB SERVICE: A web service (in English, web service or web services) is a technology that uses a set of protocols and standards that are used to exchange data between applications. Different software applications developed in different programming languages, and running on any platform, can use web services to exchange data on computer networks such as the Internet. Interoperability is achieved through the adoption of open standards. The OASIS and W3C organizations are the committees responsible for the architecture and regulation of Web services.

Índice

Índice de figuras

Capítulo 1

Introducción

Una de las frases atribuidas al gran físico Albert Einstein es:

"*La única fuente del conocimiento es la experiencia*"

Entonces, si para alcanzar el conocimiento se requiere una buena dosis de experiencia, ¿a qué esperamos para experimentar?

De eso de trata, de jugar y combinar las variables, los números y los datos para armar un puzle, de tal manera que alcancemos un resultado del que seamos capaces de obtener conclusiones. Si esas conclusiones son correctas, mejor dicho, si reiterado el experimento, siguen siendo correctas, estupendo. Hemos logrado un objetivo partiendo de unas premisas y unas observaciones empíricas. Y, ¿Qué ocurre si los resultados obtenidos no son los esperados?

En este punto, la respuesta por parte de Thomas Edison, en el momento de inventar la bombilla después de innumerables intentos, es bastante clara:

"*No fracasé, sólo descubrí 999 maneras de cómo no hacer una bombilla*"

1.1. Meteorología y motivación

La meteorología, del griego "alto en el cielo" (meteoro) y conocimiento (logos), es la ciencia atmosférica interdisciplinar que estudia el estado del tiempo, el medio atmosférico y las leyes que los rigen con apoyo de disciplinas auxiliares como la física de la atmósfera y la química de la atmósfera.

Si nos remontamos a la época de los egipcios cualquier explicación sobre el movimiento de los astros, las estaciones del año o una simple crecida del Rio Nilo, era, sin duda alguna, provocado por el aburrimiento o no, de sus dioses. Si pasamos del mito al logos, es Aristóteles en torno al año 340 AC quien presenta observaciones y especulaciones sobre el origen de los fenómenos atmosféricos y celestes.

Siglos después la invención de otros instrumentos de medida, como por ejemplo el termómetro (Galileo), el Barómetro (Torricelli), el Anemómetro (Hooke) y el Higrómetro (Saussure), ayudaron considerablemente en la recogida de diversas medidas que se intentaban relacionar con los diferentes fenómenos observados. Sin olvidar, como no, el descubrimiento de la dependencia de la presión barométrica con la altitud por parte de Pascal y Descartes.

A finales del siglo XX el avance de la electrónica, la miniaturización de los componentes electrónicos y la aparición del microprocesador impulsa la digitalización de todo lo, hasta ahora, analógico. Como ejemplo de ello podemos citar el sonido, el vídeo, y todas aquellas variables medidas en un espacio continuo de valores, entre ellas, la variables atmosféricas.

Hoy en día es posible la construcción de un sensor, de una manera muy sencilla y económica , que recoja y almacene determinados valores meteorológicos.

La gran pregunta que nos podríamos plantear es: ¿será fiable?

Me quedo con otra frase, esta vez de Lord Kelvin:

> "*Lo que no se define no se puede medir. Lo que no se mide, no se puede mejorar. Lo que no se mejora, se degrada siempre.*"

Por tanto, experimentemos, midamos y mejoremos.

1.2. Objetivos

Este trabajo propone la construcción, por un lado de un determinado número de sensores, denominados "Sensores Zeus" que, con una cadencia establecida, tome los datos ambientales. Estos datos serán transmitidos a una plataforma o repositorio que almacene los datos, a la que hemos llamado "Plataforma Zeus".

Para comprobar la calidad y el margen de error de nuestros sensores, será necesario compararlos con los datos de sensores oficiales situados en las cercanías de los nuestros. Para ello, se usarán los servicios web que la Agencia Estatal de Meteorología (`AEMET`[1]) ofrece a través de su plataforma `AEMET OpenData`[2] y que recogen el "Inventario de estaciones de Valores Climatológicos" ubicadas a lo largo del territorio español.

Por tanto, la "Plataforma Zeus" recogerá los datos tanto de nuestros sensores como de los de la AEMET y los almacenará para su posterior visualización y análisis. Podemos ver un esquema del "Ecosistema Zeus" en la figura 1.1

Por lo tanto, los objetivos concretos de este trabajo son:

1. Diseñar y construir unos sensores de bajo coste que permitan recoger y transmitir datos climatológicos a un repositorio.
2. Estudiar el API de AEMET OpenData para poder recuperar los datos ofrecidos por sus sensores.
3. Diseñar y programar una plataforma interactiva que recoja los datos de nuestros sensores y los proporcionados por la AEMET, de forma que se puedan comparar entre sí.

[1] https://www.aemet.es/es/portada
[2] https://opendata.aemet.es/centrodedescargas/inicio

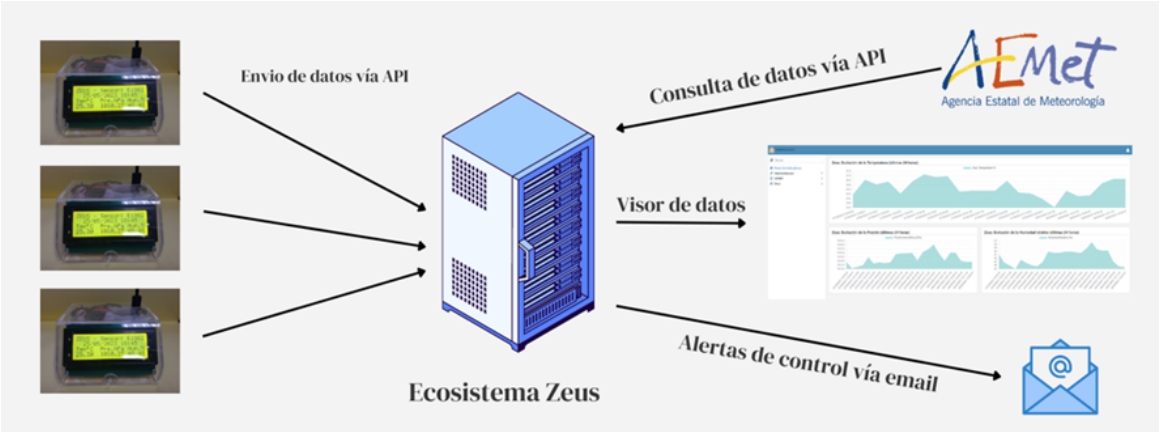


Figura 1.1: Ecosistema Zeus

1.3. Estructura de la memoria

El resto de capítulos de esta memoria se estructura como sigue:

- En el capítulo 2 se habla de las soluciones existentes y disponibles en el mercado actual, sus ventajas e inconvenientes y de los motivos que nos han llevado a la realización de este proyecto.
- En el capítulo 3 se expone la propuesta de nuestra solución para alcanzar el objetivo propuesto.
- En el 4 se detalla cada uno de los elementos que componen el Sensor Zeus, tanto de manera independiente como en su conjunto.
- En el capítulo 5 se explica el proceso general de recopilación de los datos, tanto del Sensor Zeus como de la AEMET.
- En el capítulo 6 se detallan las funcionalidades y cómo se ha implementado toda la parte visual de la Plataforma Zeus.
- En el capítulo 7 se expone toda la fase de prueba que se han realizado, en cada una de las fases, en la ejecución de este proyecto.
- Por último, el capítulo 8 recoge una exposición tanto de los aspectos didácticos como de las conclusiones finales una vez finalizado el proyecto.

Capítulo 2

Estado del arte

En este capítulo se estudian algunos sensores meteorológicos y algunas plataformas encargadas de visualizar distintos tipo de datos, con funciones próximas a las idealizadas y que serán descritas en nuestra propuesta.

Se comienza estudiando algunas estaciones metereológicas existentes en el mercado valorando, siempre a nuestro criterio, tanto sus ventajas como sus incovenientes. A continuación se realiza un análisis similar, esta vez de las plataformas recolectoras y visualizadoras de datos.

Por último, tras estos estudios previos, se realiza una conclusión detallando las motivaciones que nos han guiado en la elaboración de la solución propuesta.

2.1. Revisión de sensores meteorológicos

Es suficiente con introducir las palabras "Estaciones Meteorológicas" en algún buscador de internet para darnos cuenta de la existencia de una amplia diversidad de marcas asociadas a un gran abanico de precios. En la figura 2.1 se muestra, a modo de ejemplo, el resultado de dicha búsqueda.

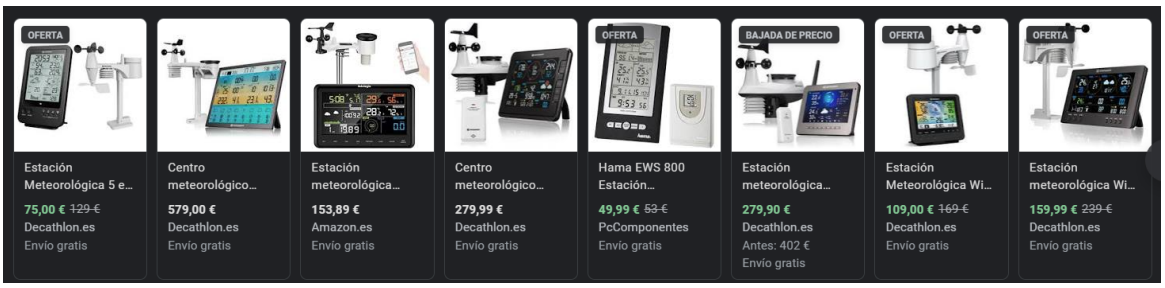


Figura 2.1: Resultado al buscar "Estaciones Meteorológicas" en internet

A continuación se estudian algunos modelos de estaciones meteorológicas.

2.1.1. Estación Meteorológica W-8681 MKII con pantalla LCD táctil

Esta Estación Meteorológica[1], que podemos ver en la figura 2.2, es un sistema de monitorización meteorológica de alta calidad y de fácil uso que mide, muestra y registra los datos climáticos tanto del interior como del exterior.

[1] https://www.astroradio.com/p/estacion-meteorologica-inalambrica-w8681-mkii/

Además de los valores de temperatura, humedad y presión atmosférica en el interior, el sensor externo toma los datos de la temperatura y humedad, además del viento y la lluvia. Estas unidades funcionan por transmisión inalámbrica hacia la estación-base.

Después de instalar el programa "EasyWeather" del CD-ROM su PC puede mostrar todos los datos tanto del interior como los que recibe la estación-base de los sensores situados en el exterior. Para funcionar, simplemente una con el cable USB suministrado la estación-base y el PC. A partir de entonces, se puede seguir la información actual e histórica del tiempo a un toque de sus dedos.

Figura 2.2: Estación W-8681 MKII

Prestaciones:

- Alcance en campo abierto: 105 m
- Frecuencia: 868 MHz (Europa)
- Margen de temperatura: -40 ºC a +65 ºC (muestra OFL fuera de este margen)
- Resolución de temperatura: 0,5 ºC
- Margen de humedad relativa: 1
- Presentación de volumen de lluvia: 0-9999 mm(muestra OFL fuera de este margen)
- Resolución: 0,3 mm (si el volumen de lluvia es menor a 1000 mm), 1 mm (si el volumen de lluvia supera 1000 mm)
- Velocidad del viento: 0-160 KMh (muestra OFL fuera de este margen)
- Intervalo de medida de humedad: 48 seg.
- Nivel de prueba del agua: IPX3[2]

[2] https://es.wikipedia.org/wiki/Grado_de_proteccin_IP

2.1.2. Oregon Scientific WMR300

Este modelo[3], que puede verse en la figura 2.3, recopila registros y carga automáticamente los datos meteorológicos precisos y detallados para facilitar el acceso en el seguimiento de patrones climáticos de mediano y largo plazo. Con una configuración de intervalo de registros personalizable y permite a los usuarios capturar y almacenar información mediante el registro de datos a intervalos pre-seleccionados para aumentar la facilidad de uso en la investigación y el estudio y así obtener resultados de mejor calidad.

Figura 2.3: Estación Oregon Scientific WMR300

Prestaciones:

- Nivel de grabación adaptable permite al usuario especificar la frecuencia con la que graban y almacenan los datos meteorológicos
- Representación gráfica de 24 meses de información meteorológica detallada.
- Previsión de 12 o 24 horas en 30-50 km radio.
- Datos tangenciales como amanecer, puesta del sol, fases lunares
- Estación meteorológica profesional tiene Garantía de 2 años. Reloj radio-controlado.
- Almacenamiento de datos desde 3 semanas hasta 3 años, dependiendo del nivel de grabación seleccionado

[3] https://meteojerezcaballeros.es/producto/oregon-scientific-wmr300/

- Rango de transmisión de los sensores es de 300 m.
- Conexión USB para PC para subir la información

2.1.3. Conclusiones

Como puede observase en las imágenes de ambas estaciones, ver figuras 2.2 y 2.3, las dos disponen tanto de medidores de la velocidad del viento (anemómetros) como de medidores de precipitación (pluviómetros). La medición de éstas variables queda fuera del propósito de éste trabajo donde únicamente quiere medirse la temperatura y humedad del aire así como la presión atmosférica.

Tampoco se requiere una pantalla o display, al menos de dimensiones grandes, puesto que la intención es que los datos se analicen a través de las interfaces que ofrecerá la plataforma Zeus.

Otro de los inconvenientes que se observa es que este tipo de estaciones requiere de una estación de trabajo o PC, donde descargar, mediante el puerto USB, los datos que se han ido recopilando.

Nuestro propósito es construir un sistema Sensor más autónomo y que él mismo sea capaz de remitir las medidas recogidas a un servidor remoto, evitando dependencia de equipos adicionales y que, ante una posible avería, pueda ser sustituido fácilmente.

2.2. AEMET OpenData

La Agencia Estatal de Meteorolgía, partir de ahora AEMET, sucedió ya en 2008 a la entonces Dirección General del Instituto Nacional de Meteorología, con más de 150 años de historia. Actualmente está adscrita al Ministerio para la Transición Ecológica y el Reto Demográfico a través de la Secretaría de Estado de Medio Ambiente.

Como Servicio Meteorológico Nacional y Autoridad Meteorológica del Estado, el objetivo básico de AEMET es contribuir a la protección de vidas y bienes a través de la adecuada predicción y vigilancia de fenómenos meteorológicos adversos y como soporte a las actividades sociales y económicas en España mediante la prestación de servicios meteorológicos de calidad. Se responsabiliza de la planificación, dirección, desarrollo y coordinación de actividades meteorológicas de cualquier naturaleza en el ámbito estatal, así como la representación de éste en organismos y ámbitos internacionales relacionados con la Meteorología.

Una de las fortalezas es la red de observación que actualmente está compuesta por:

- 96 observatorios con personal propio de la Agencia.
- 814 estaciones automáticas de observación.
- Red de 2.335 estaciones pluviométricas y termopluviométricas atendidas por colaboradores altruistas.

- Red de 15 radares meteorológicos con capacidad doppler.
- Red de detección de rayos con 15 equipos detectores en la Península y 5 en las Islas Canarias.
- 7 estaciones de radiosondeo en tierra, 1 en el buque "Espereanza del Mar" y 2 en las oficinas meteorológicas móviles de Defensa.
- 60 estaciones de medida de radiación.
- 6 espectofotómetros [4] Brewer.
- 5 fotómetros [5] Cimel.
- 15 estaciones [6] EMEP [7],VAG [8] y CAMP [9] de medida de la contaminación de fondo.

2.2.1. API

Las mediciones obtenidos por todas las estaciones de observación se ofrecen al público en forma de catálogo de datos y la descarga de datos en formatos reutilizables.

Según documentación extraída de la propia web de la AEMET, AEMET OpenData es una API REST desarrollado por AEMET que permite la difusión y la reutilización de la información meteorológica y climatológica de la Agencia, en el sentido indicado en la Ley 18/2015, de 9 de julio, por la que se modifica la Ley 37/2007, de 16 de noviembre, sobre reutilización de la información del sector público.

AEMET OpenData permite descargar gratuitamente los datos explicitados en el Anexo II de la `resolución de 30 de diciembre de 2015`[10] de AEMET, por la que se establecen los precios públicos que han de regir la prestación de servicios meteorológicos y climatológicos. Esta resolución ha sido publicada en el BOE nº 4 de 5 de enero de 2016. También están disponibles los datos obtenidos por AEMET en el marco del `convenio de colaboración`[11] entre la Agencia Estatal de Investigación y la Agencia Estatal de Meteorología con respecto a las actividades españolas en la Antártida.

En la figura 2.4 podemos apreciar su logotipo y su panel informativo.

[4] Instrumento con el que se apoya la espectrofotometría para medir la cantidad de intensidad de luz absorbida después de pasar a través de una solución muestra.

[5] Aparato para medir la intensidad del sonido.

[6] Estaciones dedicada a la observación de la composición química de la atmósfera a escala regional, lejos de fuentes contaminantes.

[7] Programa concertado de seguimiento y de evaluación del transporte a gran distancia de los contaminantes atmosféricos en Europa.

[8] La Vigilancia Mundial de la Atmósfera, es un proyecto del Programa de Investigación de la Atmósfera y el Medio Ambiente (PIAMA), de la Organización Meteorológica Mundial (OMM, WMO en inglés), organismo de las Naciones Unidas creado tras la firma del Convenio Meteorológico Mundial.

[9] El Programa Integral de Control Atmosférico, tiene por objeto conocer los aportes atmosféricos a la región del Nordeste Atlántico y estudiar sus efectos sobre el medio marino.

[10] https://www.boe.es/boe/dias/2016/01/05/pdfs/BOE-A-2016-111.pdf

[11] https://www.boe.es/boe/dias/2020/04/24/pdfs/BOE-A-2020-4634.pdf

Figura 2.4: AEMET OpenData

El API ofrece múltiple información a través de los diferentes métodos de consulta, tal y como puede observarse en la figura 2.5

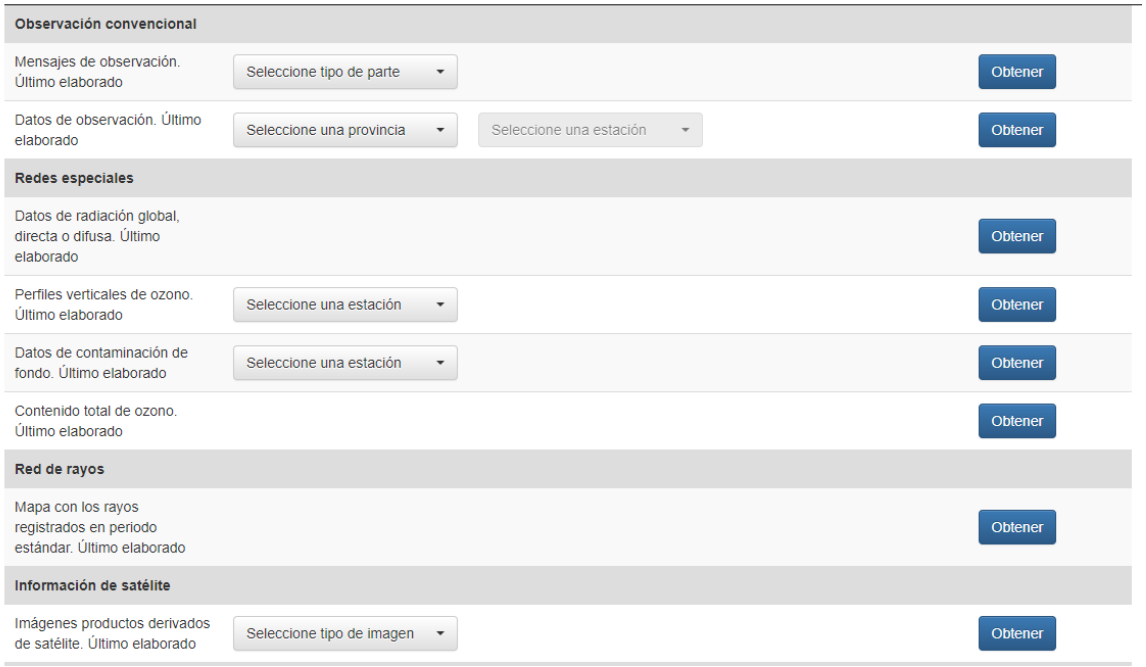


Figura 2.5: Diversos métodos API OpenData

De todos los métodos ofertados y datos que únicamente necesitaremos los detalles de las diversas estaciones así como de los datos recogidos por las mismas, nos parece acertado centrarnos en los métodos que se muestran remarcados en la figura 2.6

A continuación se describe brevemente la funcionalidad de cada uno de ellos, incluyendo la frecuencia con la que actualizan los datos disponibles:

- "Datos de Observación. Tiempo Actual": Se obtienen los datos de observación horarios de las últimas 24 horas de la estación meteorológica que se pasa como parámetro. Frecuencia de actualización: continuamente.
- "Climatologías diarias": Se obtienen los valores climatológicos para el rango de fechas y la estación seleccionada. Frecuencia de actualización: 1 vez al día, desfase 4 días.
- "Inventario de estaciones (valores climatológicos)": Se obtiene el inventario con las características de todas las estaciones climatológicas. Frecuencia de actualización: 1 vez al

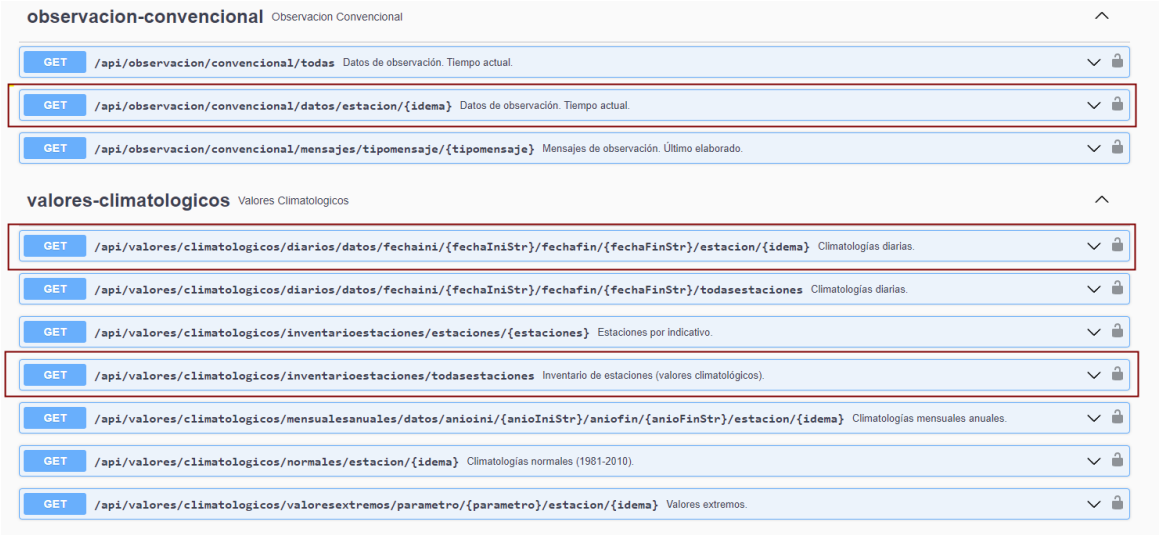


Figura 2.6: Métodos de especial interés para este trabajo

día.

2.2.2. Conclusiones

Una vez analizados y teniendo en cuenta que pretendemos cotejar los datos de nuestro sensor con los de la AEMET, son necesarias las siguientes operaciones:

- Recuperar el inventario de las estaciones una primera vez y, posteriormente, una vez al día para mantenerlas actualizadas.
- Recuperar las "Climatologías diarias" de las estaciones en estudio.
- Recuperar los "Datos de Observación" de las últimas 24 horas para las estaciones en estudio.

2.3. Revisión de plataformas de visualización de datos

Existen infinidad de plataformas de recogida de datos y todas ellas con un número considerable de usos, entre ellos:

- Monitor de infraestructuras
- Supervisión de rendimiento de las redes
- Monitor de dispositivos de redes
- Supervisión de base de datos

- Monitor de flujo de datos
- Monitor de usuarios en tiempo real
- Supervisión del rendimiento de aplicaciones
- Indicadores KPI de uso general.

En resumen, todo lo que sea medible a través de un valor es susceptible de ser enviado a través de una red de datos a un repositorio donde se almacenará, se procesará y podrá ser consultado tanto en tiempo real como en diferido.

Y partiendo de la base de que en la vida real cualquier variable puede tener unos umbrales máximos y mínimos, estas plataformas también nos permiten definir determinadas alertas y acciones que se disparan cuando el valor de la variable se encuentra fuera de los límites establecidos. Dichas alertas pueden generar o bien un correo electrónico o bien un mensaje SMS.

2.3.1. DataDog

Un ejemplo de este tipo de plataformas es `DataDog`[12] donde a través del consumo de un API podremos enviar datos y posteriormente crear paneles interactivos en tiempo real (como puede verse en la figura 2.7. Según reza en su página web:

Cree paneles interactivos en tiempo real:

- Más que paneles de resumen, Datadog ofrece todas las métricas y eventos de alta resolución para manipulación y gráficos.
- Ver gráficos de fuentes en tiempo real
- Dividir datos por host, dispositivo o cualquier otra etiqueta
- Calcular tasas, ratios, promedios o integrales
- Personalice fácilmente las vistas, de forma interactiva o en código

2.3.2. Ubidots

Otra de las plataformas estudiadas es `Ubidots`[13] autodefinida como "Herramientas de IoT y nube para construir su negocio".

Su esquema funcional es el mostrado en la figura 2.8.

[12] https://www.datadoghq.com/
[13] https://es.ubidots.com/

Figura 2.7: Panel de control de DataDog

Esta plataforma nos ofrece un gran abanico de opciones para conectar nuestros dispositivos hardware a dicha nube usando API y SDK a través de diversos protocolos, como, por ejemplo: HTTP, TCP, UDP o MQTT.

Dispone de librerías y tutoriales de gran ayuda al desarrollador de tal manera que la integración es muy asequible y cómoda. Su filosofía puede verse en la figura 2.9.

2.3.3. Conclusiones

Las dos plataformas que hemos revisado son muy completas amén de los fácil que resulta agregar un dispositivo y comenzar a recibir, almacenar y visualizar sus datos.

No obstante, desde mi punto de vista, hay dos inconvenientes: el primero, el económico. Si bien es cierto que se ofrecen diversas tarifas dependiendo tanto del número de dispositivos a conectar, del número de variables a medir o métricas y del periodo de retención, hasta que no se tenga un proyecto final totalmente instalado con un determinado número de sensores, no es posible cuantificar si el precio de esta plataforma es realmente "asumible".

El segundo inconveniente, aunque no hay nada del todo malo en ello, de hecho, la tendencia empresarial es decantarse claramente por los sistemas de infraestructuras en la nube, es no tener un control total de nuestros datos al estar almacenados en un servidor remoto, en una plataforma ajena y a la que sólo accedemos a través de las API suministradas.

En resumen:

Figura 2.8: Diagrama de la plataforma UbiDots

- Ubidots podría ser una buena elección a no ser porque preferimos disponer de acceso a la base de datos puesto que esto permitiría, en una futura mejora, relacionarla con valores de un CRM [14] o similar ya preexistente.
- Las pruebas con ambas plataformas se realizaron usando el protocolo `MQTT`[15].
- Si bien las representaciones gráficas de ambas plataformas son atractivas, a la vez que disponen de una gran usabilidad, se opta por algo más sencillo y "manejable" desde el punto de vista de la programación, aunque sea sacrificando una parte de la estética.

[14]CRM: Customer Relationship Management. Es un software que permite a las empresas rastrear cada interacción con los usuarios y clientes actuales.

[15]MQTT: "Message Queue Server Telemetry Transport". Es un protocolo ligero de mensajería abierta que se desarrolló para entornos restringidos como M2M (máquina a máquina) e IoT (Internet de las cosas), donde se requiere una pequeña huella de código.) que si bien es muy sencillo de implementar se prefiere que los 'sensores' consuman otro tipo de servicios web soap o rest por compatibilidad con otras aplicaciones o servicios ya existentes.

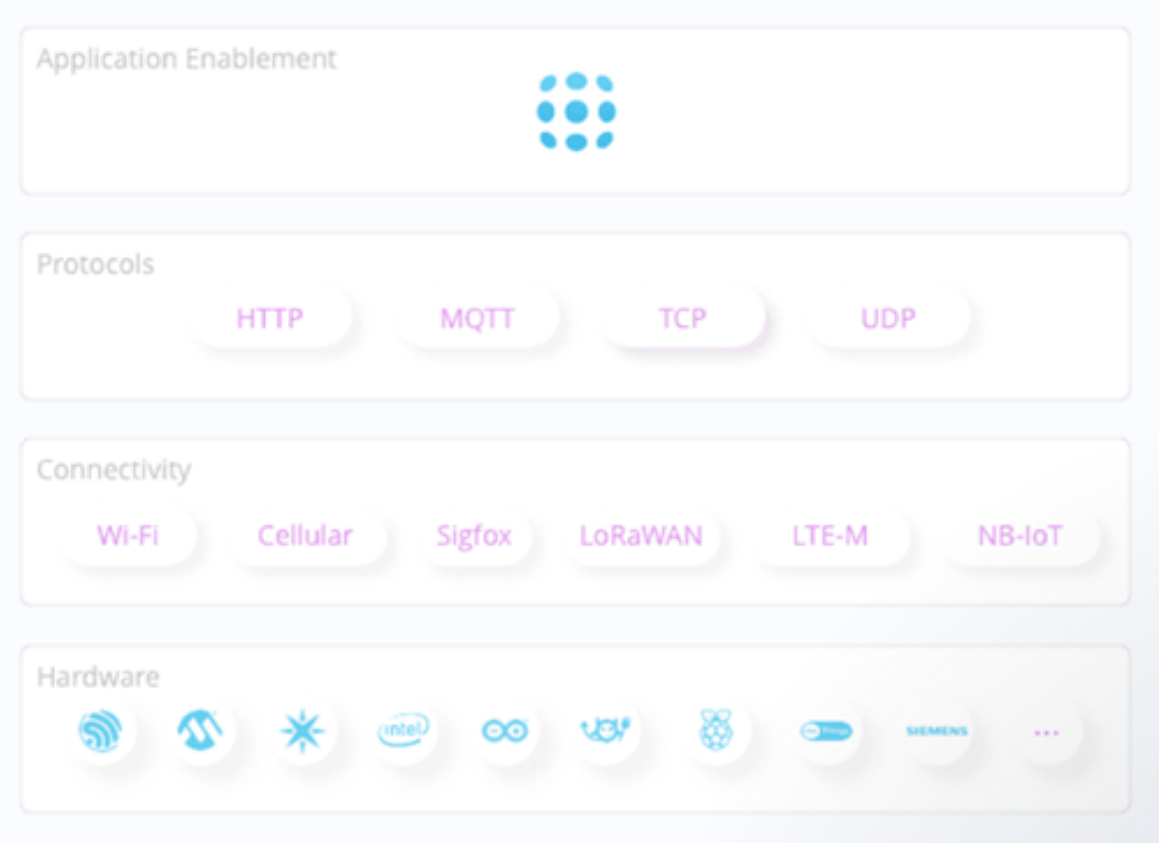


Figura 2.9: Filosofía de la plataforma UbiDots

Capítulo 3

Propuesta

Hasta ahora hemos analizado los elementos, tanto estaciones como plataformas donde monitorizar y explotar los datos recogidos, existentes en el mercado con sus ventajas e inconvenientes.

Sabemos lo "que tenemos", pero cabe preguntarse, ¿realmente es lo que queremos?

3.1. Sensor

Deseamos construir un "Sensor" que recoja determinadas variables climatológicas y las remita, mediante el consumo de un servicio web, a un servidor remoto.

Este sensor se conectará a una red WiFi existente en la ubicación escogida y, tras conectarse a dicha red, recogerá cada quince minutos las medidas deseadas, las mostrará en un display durante cinco segundos y, por último, las remitirá al servidor.

En caso de error en algunos de sus procesos, éste se mostrará en el display.

En la figura 3.1 puede apreciarse lo comentado en los puntos anteriores.

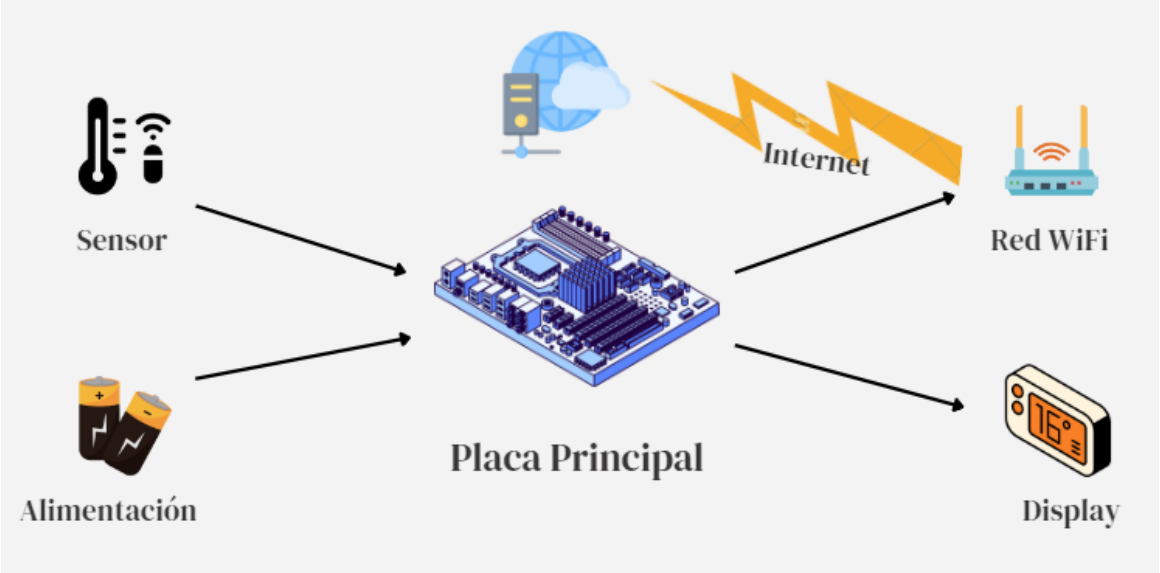


Figura 3.1: Esquema modular del Sensor

Se ha pretendido dotar al conjunto del sensor junto con el resto de los elementos que lo componen, de dos factores claves: será robusto y compacto.

Por ello se diseñará una caja de metacrilato manteniendo las siguientes premisas:

- Que sea capaz de alojar comodamente todos los elementos evitando posibles calentamientos asi como contactos indeseados internos.
- Que permita ubicar el sensor propiamente dicho en una zona en la que no sufra interferencias térmicas o lúminicas de otros elementos.
- Que pueda alimentarse de manera autónoma mediante baterías recargables.
- Que sea cómoda la lectura de la información mostrada en el Display.
- Que ante un despliegue masivo de sensores mantenga la filosofía `Plug and Play`[1]'

3.2. Obtención de datos de la AEMET

El proceso de recuperación de registros de la AEMET debe ser fiable y, en cualquier caso, se deben evitar perder datos de algunas fechas. Los registros que se vayan recuperando han de ser continuos evitando saltos que mermarán la fiabilidad de los análisis posteriores.

Por tanto, el proceso de recuperación de datos se plantea funcionalmente como un proceso o servicio que se ejecutará tres veces al día, y que realizará la siguientes tareas:

- Recuperará el inventario de estaciones de manera que se puedan actualizar los datos existentes en la base de datos.
- Recuperará los datos diarios del periodo comprendido entre la fecha de la última fecha registrada y el último día del año en curso.
- Recuperará los datos horarios y almacenarlos o actualizarlos en la base de datos.

3.3. Plataforma de Visualización

Una vez que los sensores han cumplido su misión y sus datos (y los de la AEMET) ya se encuentran en nuestra base de datos, será necesaria una plataforma o conjunto de interfaces donde el usuario pueda consultar la información deseada.

Se plantea la construcción de una `Plataforma Web`[2] puesto que tanto sus ventajas como sus incovenientes se adaptan perfectamente al objetivo que se persige.

Sus ventajas son:

[1] https://es.wikipedia.org/wiki/Plug_and_play

[2] Las plataformas digitales o plataformas virtuales, son espacios en Internet que permiten la ejecución de diversas aplicaciones o programas en un mismo lugar para satisfacer distintas necesidades.

- Se puede usar desde cualquier lugar siempre y cuando se disponga de conectividad a internet.
- No requiere hacer actualizaciones en los usuarios o clientes consumidores.
- No hay problemas de incompatibilidad entre versiones, porque todos trabajan con la misma.
- Los trabajos de respaldo de copias de seguridad se centralizan.
- No necesita instalar nada en el cliente. Un navegador apuntando a la dirección de la plataforma es suficiente para acceder a la misma.
- No se obliga a usar determinado sistema operativo.

Y sus desventajas:

- Requiere conexión a la red.
- Toma mas tiempo de desarrollo haciéndola compatible con los distinto navegadores, no obstante los entorno de desarrollo ayudan a minimizar estos problemas.
- Su tiempo de respuesta es más lento, aunque esto ha mejorado usando tecnologías como AJAX haciéndolas más rápidas.

Las funcionalidades deseadas para esta plataforma serían:

- Panel de control con diversas gráficas.
- Gestión de las Estaciones de la AEMET.
- Gestión de los Datos Diarios y Horarios de la AEMET.
- Registro de las llamadas al API de la AEMET.
- Gestión de los Sensores Zeus y sus datos.
- Análisis comparativo datos AEMET versus Zeus.

Sin olvidar algo necesario en toda plataforma de gestión, el control de los usuarios y permisos, por lo que el sistema deberá realizar una validación por parte del usuario.

3.4. Estimación de Coste

El coste del análisis, desarrollo e implantación de cada uno de los procesos descritos anteriormente se detalla en la figura 3.2 que estimado a un coste de 30 euros por hora, asciende a un total de 28.560,00 euros.

A modo de resumen tenemos:

- Construcción del Sensor: 30 jornadas / 7.200 euros.
- Despliegue Servidor Cloud: 10 jornadas / 2.400 euros.
- Desarrollo API Zeus: 17 jornadas / 4.080 euros.
- Desarrollo ImportaAemet: 23 jornadas / 5.520 euros.
- implementar Plataforma Zeus: 39 jornadas / 9.360 euros.

1	CONSTRUCCIÓN DEL SENSOR	INICIO	FIN	JORNADAS	HORAS	IMPORTE
	Análisis previo	7-11-22	10-11-22	3,00		
	Construcción del Sensor	14-11-22	28-11-22	14,00		
	Fase de Pruebas	29-11-22	9-12-22	10,00		
	Ensamblado	12-12-22	15-12-22	3,00		
	Pruebas finales y análisis de datos	15-5-23	1-6-23	0,00		
	SUBTOTAL:			30,00	240	7.200,00 €
2	DESPLIEGUE SERVIDOR CLOUD	INICIO	FIN	JORNADAS	HORAS	IMPORTE
	Análisis previo	9-1-23	12-1-23	3,00		
	Instalación y configuracion SQLServer	16-1-23	18-1-23	2,00		
	Configuración del IIS	18-1-23	20-1-23	2,00		
	Fase de Pruebas	23-1-23	26-1-23	3,00		
	Pruebas finales y análisis de datos	15-5-23	1-6-23	0,00		
	SUBTOTAL:			10,00	80	2.400,00 €
3	DESARROLLO API ZEUS	INICIO	FIN	JORNADAS	HORAS	IMPORTE
	Analisis previo	6-2-23	10-2-23	4,00		
	Desarrollo y pruebas	13-2-23	21-2-23	8,00		
	Despliegue en Servidor Cloud	22-2-23	23-2-23	1,00		
	Pruebas y evaluación resultados	24-2-23	28-2-23	4,00		
	Pruebas finales y análisis de datos	15-5-23	1-6-23	0,00		
	SUBTOTAL:			17,00	136	4.080,00 €
4	Desarrollo Importa AEMET	INICIO	FIN	JORNADAS	HORAS	
	Análisis previo	1-3-23	6-3-23	5,00		
	Desarrollo Servicio importaAemet	7-3-23	22-3-23	15,00		
	Despliegue en Servidor Cloud	23-3-23	24-3-23	1,00		
	Pruebas y evaluación resultados	27-3-23	29-3-23	2,00		
	Pruebas finales y análisis de datos	15-5-23	1-6-23	0,00		
	SUBTOTAL:			23,00	184	5.520,00 €
5	Implementar Plataforma Zeus	INICIO	FIN	JORNADAS	HORAS	IMPORTE
	Análisis previo	3-4-23	6-4-23	3,00		
	Desarrollo Plataforma Zeus	10-4-23	24-4-23	14,00		
	Despliegue en Servidor Cloud	25-4-23	27-4-23	2,00		
	Pruebas y evaluación resultados	27-4-23	30-4-23	3,00		
	Pruebas finales y análisis de datos	15-5-23	1-6-23	17,00		
	SUBTOTAL:			39,00	312	9.360,00 €
	TOTAL:			119,00	952	28.560,00 €

Figura 3.2: Detalle del coste de cada fase del desarrollo

Capítulo 4

Sensor Zeus

Ante el escenario en el que nos encontramos, nuestro sensor debe poder recopilar datos de sensores meteorológicos, mostrarlos, y, sin asistencia de elementos adicionales, enviarlos al servidor.

4.1. Elementos hardware

En esta sección analizaremos cada uno de los elementos que conforman el sensor Zeus, los cuales se pueden ver en el esquema general de la figura 4.1, ya adelantada en el capítulo anterior.

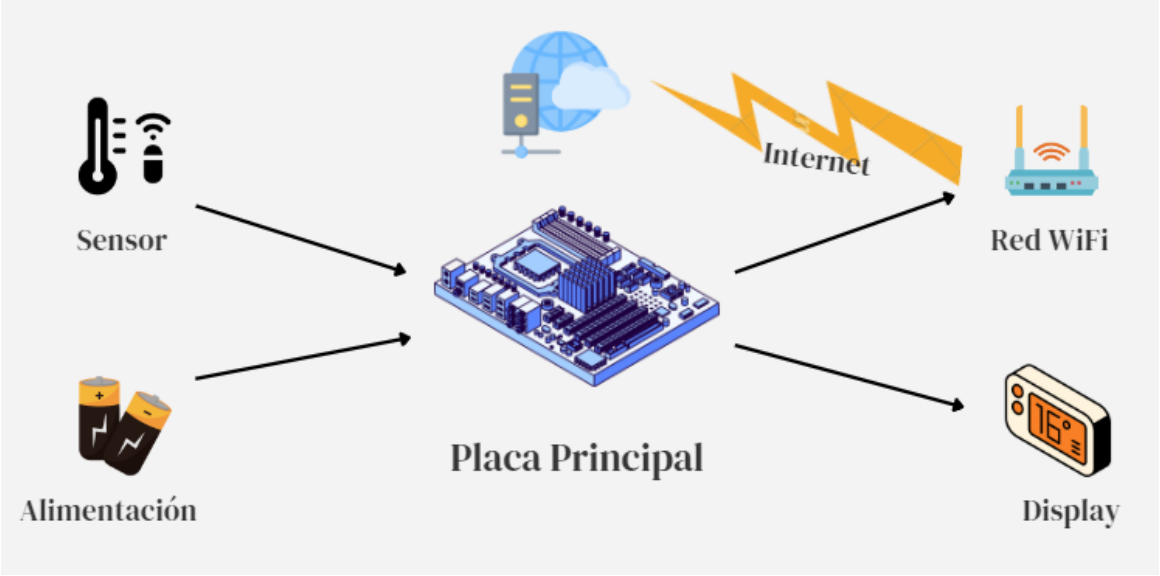


Figura 4.1: Esquema modular del Sensor Zeus

4.1.1. Placa principal

En un principio se valoraron diferentes opciones para una placa base que gestione la lógica de recopilación de datos y envío de los mismos.

- Raspberry Pi[1]. Son placas SoC[2] de propósito general. Presentan un bajo consumo y prestaciones suficientes para nuestro proyecto, especialmente en lo concerniente a la conectividad WiFi.

[1] https://www.raspberrypi.com/
[2] https://es.wikipedia.org/wiki/SOC

Sin embargo, el hecho de depender de un Sistema Operativo dificulta su mantenimiento, que incluiría actualizaciones periódicas de todo el software. Además de que podría introducir errores ajenos a nuestra aplicación, lo que restaría robustez al sensor. Por todo ello, se descarta su uso.

- Placas Arduino[3]. Estas placas son muy populares entre la comunidad de desarrolladores de hardware por su bajo coste y consumo y su alta versatilidad.

  Si bien nos elimina el problema del sistema operativo que presenta la Raspberry Pi, las placas Arduino no ofrecen conectividad WiFi o Bluetooth de manera integrada. Esto provoca la necesidad de hardware adicional, lo que repercute directamente en el coste y el consumo. Por lo tanto, se descarta su uso.

- La familia de chips `ESP32`[4] son una buena alternativa a valorar. El bajo coste y consumo de energía así como su integración con tecnología WiFi y Bluetooth son sus mayores atractivos.

  Se programan de manera similar a las placas Arduino, sin necesitar un Sistema Operativo, e incluyen la conectividad WiFi necesaria en nuestro proyecto. Por lo tanto, esta es la opción elegida.

En vista de lo anteriormente expuesto , un `SoC` ideal para nuestro proyecto es el circuito basado en `ESP32-DevKitC`[5], conocido como ESP32-WROOM-32D.

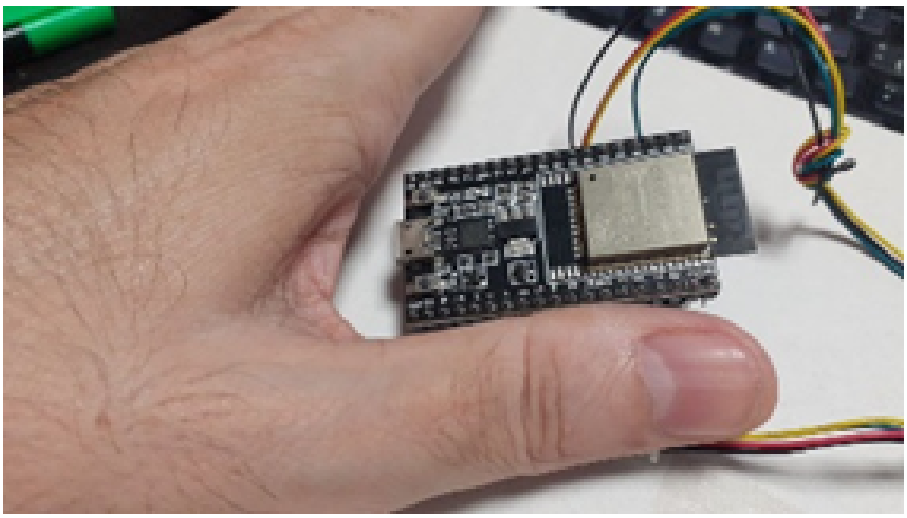


Figura 4.2: Circuito ESP32-WROOM-32D mostrado a su tamaño natural

Una variante de la misma, dotada de más funcionalidades como pueden ser los relés para ejecutar acciones de potencia y basado en el mismo chip es la mostrada en la figura 4.3

Las características de esta placa son las siguientes:

- Módulo de ESP32-WROOM-32E estable, Flash Byte de 4M de gran capacidad.

[3] https://www.arduino.cc/
[4] https://es.wikipedia.org/wiki/ESP32
[5] https://www.espressif.com/en/products/devkits/esp32-devkitc

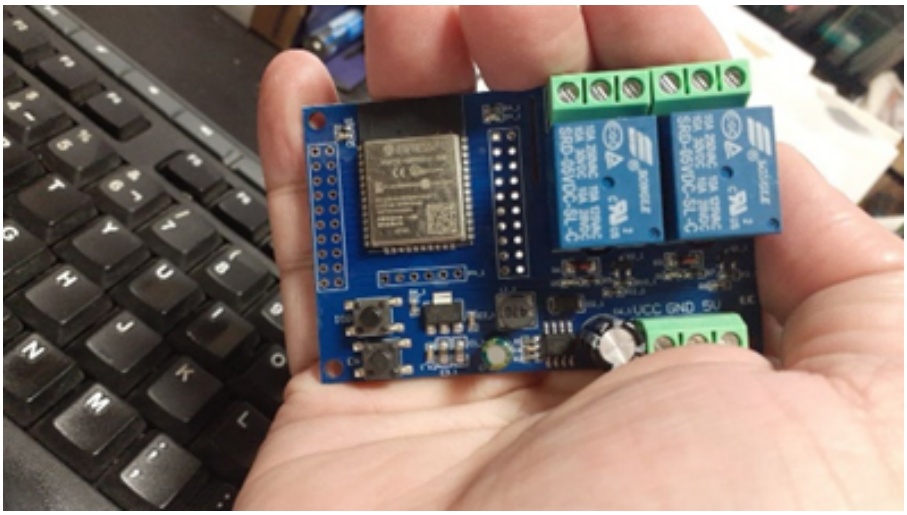


Figura 4.3: Placa de desarrollo basada en ESP32-WROOM, con Wifi y dos relés.

- El puerto de E/S y el puerto de descarga del programa UART del módulo ESP32 tienen salida, lo cual es conveniente para el desarrollo secundario.
- Módulo ESP32 integrado IO0, botón programable y botón de Reinicio.
- Admite el uso de herramientas de desarrollo para Arduino IDE y proporciona programas de referencia en el entorno de desarrollo para Arduino.
- Relé bidireccional de 5V incluido, señal de interruptor de salida, adecuado para controlar la carga cuyo voltaje de trabajo está dentro de AC250V/DC30V.
- LED programable, indicador led y dós relés integrados en placa.

4.1.2. Sensor

El otro elemento principal del circuito es el sensor propiamente dicho, es decir, el hardware que es capaz de transformar las variables atmosféricas a medir en señales que luego serán interpretadas por el procesador de la placa principal.

Después de estudiar diferentes sensores, entre otros, los de la marca Adafruit[6] y descartando disponer de un sensor independiente para cada medida, se optó por una solución más compacta, usando para ello un componente trifuncional.

El Taidacent HTU21D BMP180 BH1750FVI[7] es un módulo sensor de temperatura, humedad, presión y luz. Destaca su reducido tamaño (20mm x 8mm) que puede apreciarse en la figura 4.4, donde se compara con las dimensiones de una tarjeta SIM de las empleadas habitualmente en telefonía móvil.

Sus especificaciones técnicas son las siguientes:

[6] https://www.adafruit.com/category/35

[7] https://www.amazon.com/-/es/Rakstore-BH1750FVI-temperatura-estacin-meteorolgica/dp/B0B1PQJX5H

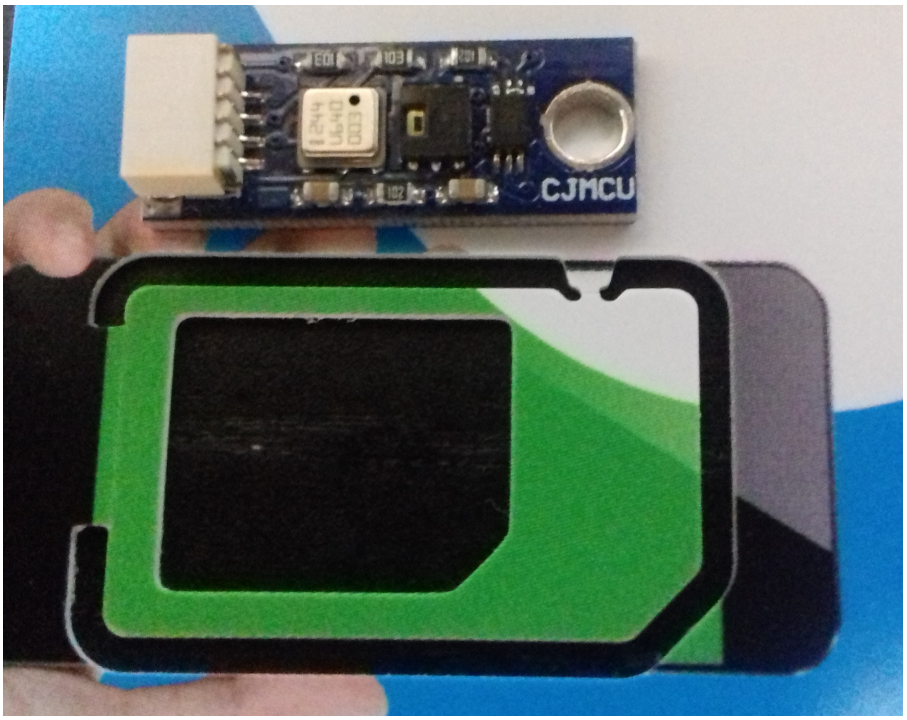


Figura 4.4: Sensor trifuncional comparado con una tarjeta SIM

- HTU21D está hecho en base al sensor de humedad de alto rendimiento de Humirel (Francia), y el sensor emite el formato estándar I^2C. Al mismo tiempo, tiene alta precisión de temperatura y precisión de humedad. HTU21 está especialmente diseñado para aplicaciones de bajo consumo de energía y pequeño volumen, con velocidad de respuesta rápida, fuerte capacidad anti-interferencia y consumo de energía extremadamente bajo.

  Parámetros del producto:

  - Voltaje de la fuente de alimentación: 1,5 V 3,6 V
  - Consumo de energía: menor o igual a 2,7uw
  - Método de comunicación: I^2C
  - Histéresis de humedad: ± 1
  - Tiempo de medición: 50ms
  - Deriva anual: -0. 5
  - Tiempo de respuesta de la humedad: 5 s
  - Tiempo de respuesta de temperatura: 10 s
  - Rango de medición de humedad: 0-100 % RH
  - Rango de medición de temperatura: -40^{o}C a 125 °C ± 0,3 °C
  - Rango de precisión de la humedad (10 % RH a 95 % RH): HTU21D ± 2 % RH

- BMP180 es un sensor de presión de alta precisión, tamaño pequeño y consumo energético ultra bajo. Integra sensores piezoresistivos, ADC, controlador EEPRM y controlador IIC. La precisión absoluta puede alcanzar hasta 0.03 hPa, y el consumo de electrodos es bajo. 3uA, se puede conectar directamente a varios microprocesadores a través del bus I^2C. Puede medir la presión y la altitud del aire.

Parámetros del producto:

- Tensión de alimentación: 1. 8V - 3,6 V
- Temperatura de trabajo: -40 °C - 85 °C
- Método de comunicación: IIC hasta 3,4 MHz
- Rango de presión: 300 - 1100hPa

- 750FVI es un sensor de intensidad de luz ambiental de tipo de salida digital, convertidor AD integrado de 16 bits, características de sensibilidad espectral cerca de la sensibilidad visual, poca influencia de la fuente de luz (lámpara incandescente, lámpara fluorescente, lámpara halógena, LED blanco, lámpara fluorescente), Se puede utilizar para una amplia gama de brillo. Se puede realizar una medición de alta precisión de 1 lux y ajustar el brillo de la retroiluminación del LCD o del teclado de acuerdo con los datos de intensidad de luz recopilados.

  Parámetros del producto:

  - Tensión de alimentación: 3,0 V-3,6 V
  - Temperatura de trabajo: -40 °C 85 °C
  - Rango de iluminación: 0-65535 lx
  - Método de comunicación: I^2C
  - Cambio mínimo de error: ± 20

  En las figuras 4.5 y 4.6 se puede apreciar dos imágenes que corresponden al sensor seleccionado:

Figura 4.5: Sensor tri-funcional, según foto del catálogo del fabricante

4.1.3. Display

Con el fin de obtener una lectura cómoda se seleccionó un Display LCD de 20 caracteres x 4 líneas, alimentado con 5 voltios y de unas dimensiones aproximadas de 146 mm x62.5 mm. Dispone de una placa adaptadora interfaz I^2C lo que facilita tanto la programación como el cableado interno.

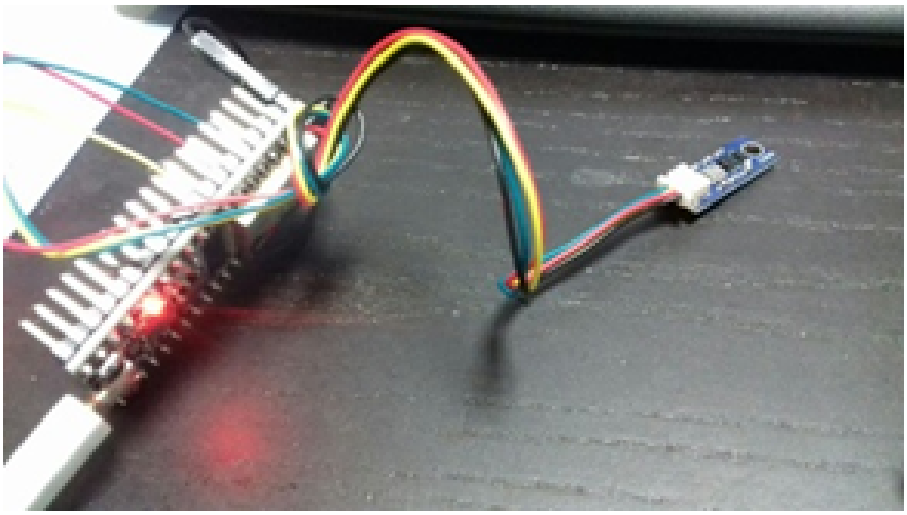


Figura 4.6: Sensor tri-funcional conectado a la placa ESP32-WROOM-32D

En la imagen 4.7 se pueden apreciar dos unidades del Display con sus placas interfaces I^2C correspondientes.

Figura 4.7: Pareja de display 20 caracteres x 4 líneas con interfaces I^2C

4.1.4. Alimentación

Ya en la etapa muy inicial del proyecto se tenía claro que el sensor debía de disponer de una cierta autonomía en cuanto a la alimentación eléctrica. Si hay fallos de alimentación, el sensor no recoge ni envía datos y por tanto los análisis posteriores no serán lo fiable que deberían ser.

Por ello se consideró muy adecuada la siguiente tarjeta que recarga y mantiene dos baterías de litio-ion y ofrece dos líneas de salida de voltaje, a 3 y a 5 voltios respectivamente, ideal para

alimentar tanto a nuestra placa principal y Display (alimentados ambos a 5V) como al sensor propiamente dicho (alimentado a 3V). Esto nos evita la necesidad de hardware adicional de regulación de voltajes.

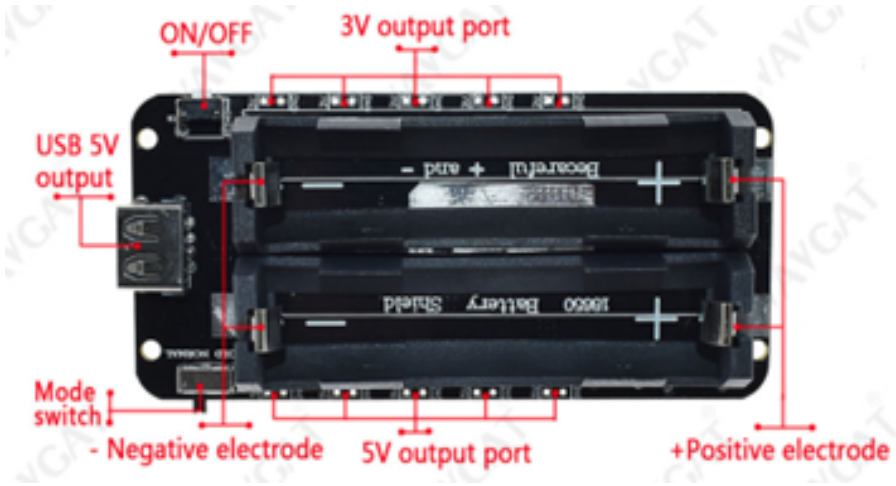


Figura 4.8: Alimentador/Cargador de Baterías de Litio-Ion

En la imagen 4.8 se muestra una foto del circuito alimentador con las leyendas correspondientes a cada entrada, salida o función.

4.1.5. Conexión entre los elementos hardware

Hasta este punto hemos ido comentando acerca de los diferentes elementos que componen el sensor objeto de este proyecto. Ahora bien, esos elementos son como los integrantes de una orquesta: cada uno por separado realiza su función. Cada uno toca su instrumento y además de sonar, suena bien aunque a destiempo. En nuestro caso todos conocen su trabajo: el minúsculo sensor trifuncional "sabe" obtener las mediciones, el display "sabe" mostrarlas, la placa principal "sabe" enviar el dato y por último la batería "sabe" que tiene que alimentarlos a todos.

Ahora nos preguntamos, ¿que ocurriría si conectamos todo y ponemos un poco de orden? En la imagen 4.9 podemos observar los modos de interconexión entre los elementos.

A modo de resumen tenemos:

- Sensor se conecta a la placa principal usando el protocolo I^2C[8]
- La placa principal también utiliza el mismo protocolo para comunicarse con el Display, por tanto el Sensor y el Display comparten un mismo bus de datos. El protocolo utilizado ofrece una gran escalabilidad a la hora de ir incorporando sensores adicionales que se conectarán en paralelo con los ya existentes.

[8]Este protocolo se utiliza principalmente internamente para la comunicación entre diferentes partes de un circuito,por ejemplo, entre un controlador y circuitos periféricos integrados.

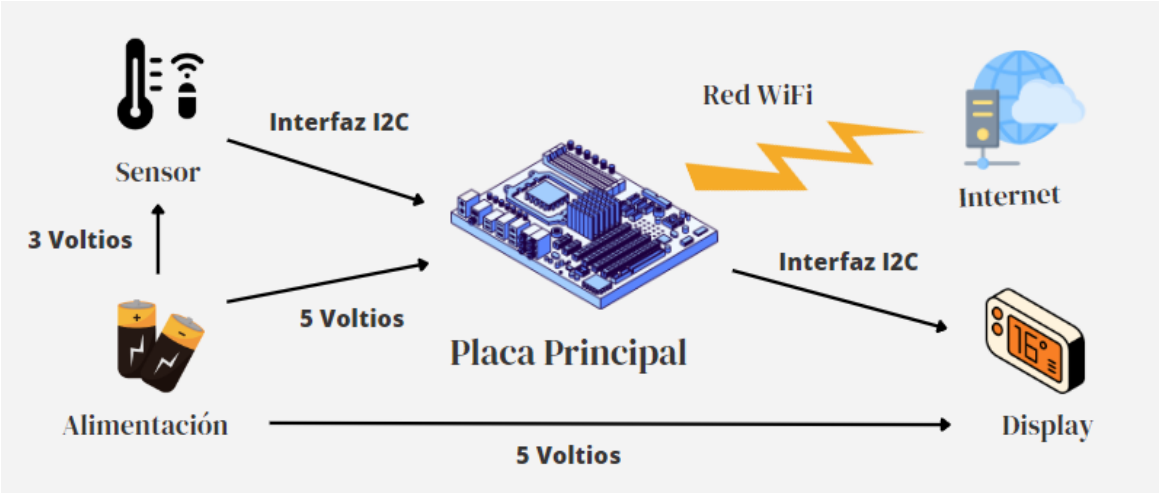


Figura 4.9: Elementos del Sensor y su modo de interconexión

- Alimentación es capaz de ofrecer tensión tanto a 5 Voltios para la placa principal como a 3 Voltios para el Sensor. Debe extremarse la precaución de cara a la incorporación de más sensores.

4.1.6. Estimación de coste

A continuación se relaciona el inventario de los elementos que componen el sensor así como el coste de los mismos en el momento de su adquisición:

- Placa principal: 1 Unidad ESP32-WROOM + Wifi + Módulo de relé 2 canales: 8,24 euros.
- Sensor: 1 Unidad Sensor trifuncional Taidacent HTU21D BMP180 BH1750FVI: 27,62 euros.
- Display: 1 Unidad Módulo LCD 2004 20x4 IIC I^2C, pantalla blanca de caracteres con salida de 3,3V. y 5V: 7,50 euros.
- Alimentación: 1 Unidad Protector de batería de litio 18650. Placa de expansión de energía móvil 5V/3A 3V/1A: 5,00 euros.
- Baterías: 2 Unidades Baterías Litio Ión 18650 3.7V 2200mAh: 12,00 euros.
- Otros: Cableado y otro pequeño material: 10,00 euros
- Caja: 1 Unidad Caja Metacrilato: 40,00 euros.

Lo que arroja un importe total de 110,36 euros para la construcción de un Sensor Zeus.

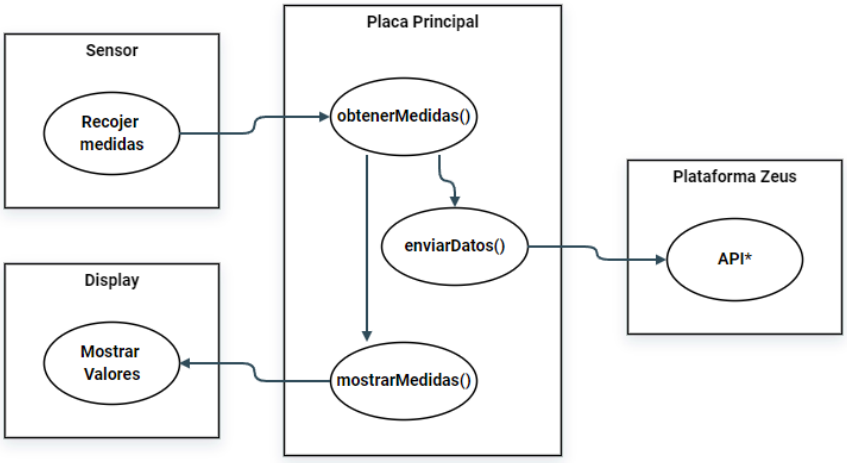


Figura 4.10: Casos de uso del Sensor Zeus

4.2. Firmware

En la imagen 4.10 se muestran los casos de uso que se precisan para obtener el objetivo pactado.

Dichos casos son:

- CU.S1: Obtener las medidas del sensor hardware.
- CU.S2: Remitir las medidas al Servidor Remoto.
- CU.S3: Mostrar a través del Display diversos mensajes.

Antes de comenzar a explicar el esquema del código implementado se realizará una pequeña introducción teórica al funcionamiento de los programas para placas ESP32 (similar a las placas Arduino).

Hay dos funciones de vital importancia: `setup()` y `loop()`. Comenzando por la primera, la función setup() es llamada de forma automática en el momento del inicio del dispositivo. Es el lugar donde se deberán situar los comandos necesarios para inicializar nuestro programa:

```
1 void setup() {
2     Comando 1
3     Comando 2
4     ...
5     Comando N}
```

La función loop() es el bucle principal de nuestro programa. Es aquí donde tenemos que ubicar los comandos que se ejecutarán mientras la placa esté habilitada. Comenzando con el primer comando, el microcontrolador irá ejecutando secuencialmente el resto de ellos hasta el final, desde donde saltará inmediatamente al primer comando, repitiendo la misma secuencia de manera indefinida (siempre y cuando la placa tenga suministro eléctrico). En el siguiente ejemplo

se ejecutarían indefinidamente los comandos 1 y 2 en ese orden y con una pausa de 1 segundo (1000 milisegundos) entre cada uno de ellos:

```
void loop() {
    comando 1;              // Se ejecuta el comando 1
    delay (1000);           // Se espera 1 segundo
    comando 2;              // Se ejecuta el comando 2
    delay (1000);           // Se espera 1 segundo
}
```

Una vez comentado el comportamiento de estas funciones, se muestra a continuación el contenido de la función setup() de nuestro proyecto:

```
void setup() {
    //Ajustamos el puerto RELAY1 como de salida.
    pinMode(RELAY1, OUTPUT);
    //Puerto serial a 115200bps
    Serial.begin(115200);
    //Inicializamos el Display LCD
    lcd.init();
    lcd.clear();
    lcd.backlight();
    //Inicializamos la red wifi con los datos nuestra red wifi
    WiFi.begin(WIFISSID, PASSWORD);
    // Mensaje de control: conectando a la wifi...
    Serial.println();
    Serial.print("Esperando conexion WiFi...");
    // Bucle de control para controlar el acceso a la red WiFi
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    //Mensajes de Control sobre el acceso a la WiFi
    Serial.println("");
    Serial.println("WiFi Connectado");
    Serial.println("Direccion: ");
    Serial.println(WiFi.localIP());
    macSensor=WiFi.macAddress();
    Serial.println(macSensor);
    //Control de acceso al sensor
    if (!bmp.begin()) {
        Serial.println("No se pudo encontrar un sensor BMP085 válido,
           verifique el cableado!");
        muestraError("Err02: No sensor",10000);
        while (1) {}
    }
    //Activamos el sensor.
    LightSensor.begin();
    myHumidity.begin();
}
```

A continuación mostramos una breve explicación del código anterior:

- El pin RELAY1 se usa para controlar el encendido del display, por lo que se configura como pin de salida.
- Se activa el puerto serie a modo de depuración.
- Se inicializa el display.
- Se inicia la conexión a la red WiFi existente, configurada de antemano con las variables WIFISSID y PASSWORD que contienen, respectivamente el identificador de la red y su contraseña.
- Cada medio segundo se comprueba el estado de la conexión con la red WiFi, mostrándose en la consola de depuración el estado de conexión.
- Una vez conectado a la WiFi e comprueba el acceso al sensor y que este responda adecuadamente, bloqueando el programa en caso contrario (mostrando un error en el display).
- Por último, se inicializa el sensor.

De manera análoga se muestra el contenido de la función loop():

```
1  void loop() {
2      // Desactivamos relé para apagar el LCD
3      digitalWrite(RELAY1,LOW);
4      // Obtenemos la fecha/hora desde pool.ntp.org
5      getFechaHora();
6      // Leemos los datos del sensor
7      obtenerMedidas();
8      // Activamos el relé para encender el LCD
9      digitalWrite(RELAY1,HIGH);
10     // Mostramos los datos en el LCD
11     mostrarMedidas();
12     // Envio los datos al Host remoto
13     enviarDatos(Linea0);
14     // Desactivo relé para apagar el LCD
15     digitalWrite(RELAY1,LOW);
16     // Esperamos 15 min para repetir bucle
17     delay(900000);
18     // Limpio el LCD antes de volver a imprimir
19     lcd.clear();
20 }
```

Destacamos principalmente las líneas 7, 11 y 13 donde se muestran las funciones claves obtenerMedidas(), mostrarMedidas() y enviarDatos() respectivamente.

Como hemos podido observar, la función loop() repite de manera cíclica llamadas a las siguientes funciones auxiliares:

- digitalWrite(RELAY1, LOW): Se desactiva el relé de la placa con el fin de apagar la iluminación trasera del Display con esto se ahorra en consumo de las baterías de alimentación.
- getFechaHora(): Se obtiene la fecha y hora a través de un servidor de hora de internet, en este caso pool.ntp.org. Al carecer la placa ESP32 de reloj interno es necesario recurrir a un recurso externo donde obtener la fecha/hora que se desea mostrar en el Display.
- obtenerMedidas(): Esta función se encarga de recoger del sensor las medidas de la temperatura, presión, altitud, luminosidad y humedad relativa y con ellas crear las cadenas de texto tanto para visualizar en el Display como para enviar a través del servicio web.
- digitalWrite(RELAY1, HIGH): Se activa el relé iluminando la parte trasera del Display facilitando de esta manera la lectura de este.
- mostrarMedidas(): función que limpia el Display y muestra las líneas de datos obtenidas en la función descrita anteriormente.
- enviarDatos(Linea0): función que envía el contenido de la variable "Linea0" al servidor remoto. Para ello realiza una invocación al método "putMedidasSensores1" del servicio web "ZeusService.asmx". Cuando termina hace una pausa de dos segundos.

  Nota: Es importante destacar que en el momento de llamar a la función enviarDatos() sabemos exactamente el paquete de datos que queremos enviar al servidor remoto aunque aún no se haya definido el API al que estamos llamando. En este momento vemos la necesidad de que en dicha API exista un método llamado "putMedidasSensores1" que es invocado en esta función.
- digitalWrite(RELAY1, LOW): Se vuelve a desactivar el relé y por tanto la retroiluminación del Display vuelve a apagarse.
- delay(900000): Se hace una pausa de 15 minutos (900.000 segundos) antes de volver a iniciar el bucle de funciones.
- Lcd.clear(): Se limpia en Display antes de volver a imprimir datos. Hasta este momento permanecen y son visibles los últimos datos medidos y enviados, aunque el Display permanezca no iluminado.

El Display además nos muestra determinados mensajes de error, entre ellos, los siguientes:

- Err01: pool.ntp.org: No se puede acceder al servidor horario establecido.
- Err02: No sensor: No se pudo inicializar el sensor BMP085.
- Err03: Host OFF: No se puede conectar al servidor remoto ZeusService.asmx
- Err04: Server OUT: El servidor remoto ha abortado la conexión
- Err05: Cliente OUT: No se encuentra el método putMedidasSensores1
- Err07: Red Wifi ON: Red Wifi conectada
- Err08: Red Wifi OFF: No es posible conectarse a la red Wifi

En el anexo 1 se muestra la totalidad del código fuente del firmware del Sensor Zeus.

Capítulo 5

Recopilación de datos

Como ya comentamos en el Capítulo 3, compararemos los datos recogidos por nuestro sensor con los datos de una estación de la AEMET lo más cercana posible. Para ello, necesitamos un servicio automático que recoja los datos tanto de nuestro sensor (construido según lo explicado en el Capítulo 4), como de las estaciones de la AEMET involucradas en el estudio comparativo. En este capítulo desarrollaremos dicho servicio automático.

5.1. Casos de uso

Los casos de uso identificados para esta parte del proyecto son los siguientes:

- Casos de uso del servicio "Zeus.WebService.asmx":
  - CU.RS1: El servicio web "escucha" las peticiones de los sensores.
  - CU.RS2: El servicio web "almacena" los valores recogidos.
  - CU.RS3: El servicio web envía alertas por correo electrónico en caso de operaciones exitosas.
  - CU.RS4: El servicio web envía alertas por correo electrónico en caso de operaciones fallidas.
- Casos de uso del servicio o proceso "ImportaAEMET":
  - CU.RA1: Este servicio consulta y recupera el listado de las estaciones censadas en el catálogo de la AEMET.
  - CU.RA2: Este servicio consulta los indicativos de las estaciones para las que debe recuperar medidas.
  - CU.RA3: El servicio invoca al API de la AEMET.
  - CU.RA4: El servicio almacena los datos obtenidos.
  - CU.RA5: El servicio actualiza la fecha de ultima sincronización para la estación sincronizada.
  - CU.RA6: El servicio envía alertas por correo electrónico en caso de operaciones exitosas.
  - CU.RA7: El servicio envía alertas por correo electrónico en caso de operaciones fallidas.
  - CU.RA8: El servicio almacena los resultados de las llamadas al API a modo de log.

5.1.1. Recogida de datos del Sensor Zeus

Una vez definidos los casos de uso, se hace necesario definir que datos se recuperan del sensor y que, por tanto, se deben almacenar.

Las medidas recogidas a través del sensor trifuncional detallado en capítulos anteriores y que son objeto de nuestro estudio, son las siguientes:

- La temperatura en grados Celsius (ºC).
- La humedad relativa expresada en valores porcentuales (%).
- La presión atmosférica expresada en hectopascales o milibares (hPa/mb).
- La luminosidad expresada en lúmenes (lx).
- La altidud expresada en metros (m).

A las medidas anteriores se añaden como datos complementarios el nombre y el identificador del sensor (éste último valor no es más que la dirección `MAC`[1] asignada al conectarse a la red WiFi).

Como consecuencia de lo expuesto se considera necesario la creación de los métodos mostrados en la figura 5.1 y que componen el servicio web ZeusService.asmx:

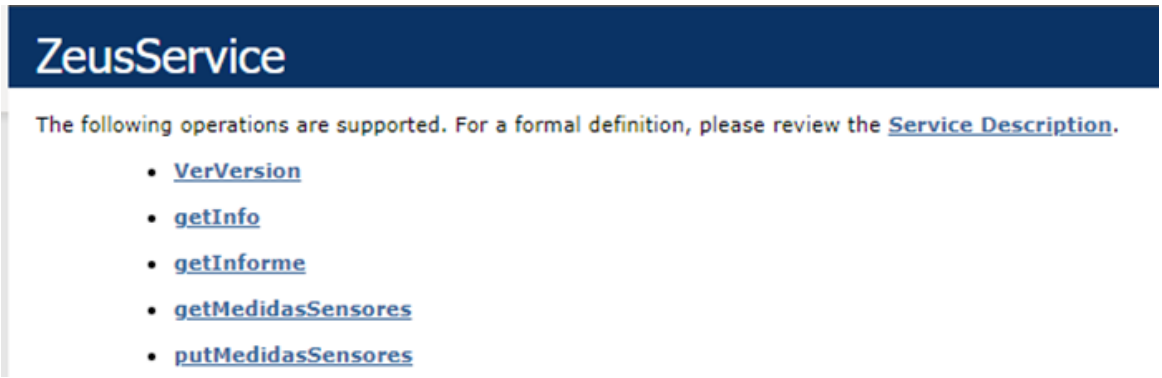


Figura 5.1: Métodos que componen el servicio web ZeusService.asmx

Se comentan a continuación con más detalle:

- verVersion(): Devuelve la versión de los servicios publicados.
- getInfo(): En desuso, solo para fines de depuración.
- getInforme(): En desuso, solo para fines de depuración.

[1]En las redes de computadoras, la dirección MAC es un identificador de 48 bits (6 bloques de dos caracteres hexadecimales [8 bits]) que corresponde de forma única a una tarjeta o dispositivo de red. Se la conoce también como dirección física, y es única para cada dispositivo.

- putMedidaSensores(): Se envían al servidor los valores de las siguientes variables: pIdSensor, pTemperatura, pPresion, pHumedad y pLuminosidad.
- putMedidasSensores1(pInfo): Evolución del método anterior donde se envían los mismos valores que en el método anterior, pero esta concatenados en una única cadena de caracteres denominada "pInfo" y según se muestra en la figura 5.2.

```
(nameSensor + "#" + temperatura + "#" + humedad + "#" + presion + "#" + lux+ "#" + altitud + "#" + macSensor)
```

Figura 5.2: Modo de concatenación de las medidas obtenidas del sensor

- sendEmail: en desuso, solo para fines de depuración.
- getMedidasSensores(pIdSensor): Devuelve una cadena json con los últimos 24 registros del sensor pasado como parámetro. Solo para fines de depuración.

En el apéndice 3 se exponen el código fuente de las funciones principales que ejecutan los métodos descritos.

5.1.2. Recogida de datos de la AEMET

De manera similar a los datos enviados desde los sensores Zeus, a continuación se muestran que datos necesitamos obtener de la AEMET y que procedemos a clasificar según el siguiente criterio:

- Datos relativos a la estaciones meteorológicas: identificador (o indicativo según la AEMET), nombre, provincia, altitud sobre el nivel del mar y coordenadas geográficas (latitud y longitud).
- Datos relativos a los registros diarios: fecha, indicativo de la estación, las horas en las que se han registrados las temperaturas y presiones mínimas y máximas y sus valores, valor de la precipitación, horas de sol, velocidad media del viento, y por último, la dirección y velocidad de la racha de viento.
- Datos relativos a los registros horarios: fecha y hora del registro, indicativo de la estación, temperatura del aire, temperatura máxima y mínima de esa franja horaria, presión atmosférica, humedad relativa, velocidad máxima del viento, precipitación e índice de radiación solar.

Todos los datos descritos anteriores deberán ser almacenados en una base de datos que describiremos en el siguiente apartado.

Es importante destacar que en el momento de almacenar los citados datos, además se actualiza un registro denominado "Últ. Actualización"donde se guarda la fecha y hora del sistema en el momento de insertar o modificar dicho registro en la base de datos.

En el caso de los registros de las estaciones de la AEMET hay otro campo adicional denominado "Activo", que siempre por defecto tiene el valor "NO". Una estación activa es aquella para que el proceso "ImportaAmet" recuperará datos y los almacenará para su estudio posterior.

En el apéndice 2 se muestra con todo detalle las funciones principales que ejecutan los procesos descritos hasta este momento.

5.2. Base de Datos Zeus

Llegados a esta sección, disponemos por un lado de un sensor que recoge medidas, un servicio web que "escucha" esas medias y un proceso que "llama" a la AEMET y recupera determinados datos. En resumen, datos, datos y datos, que deben estar clasificados y ordenados adecuadamente para sus consultas y análisis posteriores. Y para ello no hay nada mejor que el uso de una `Base de Datos`[2]

5.2.1. Diseño de la Base de Datos

Antes de proceder con el diseño y posterior construcción de la base de datos, es necesario analizar que funciones debe realizar la misma, para ello se estudian y se proponen los siguientes casos de uso:

- CU.BD1: Almacenar y actualizar, si procede, los datos de las estaciones meteorológicas. Se actualizan todos los datos de la estación. El campo "Activo" por defecto toma el valor "NO" y el campo "Últ. Actualicación" queda sin valor.
- CU.BD2: Almacena los datos diarios: Se registran todos los datos diarios de las estaciones marcadas como activas y en el momento de la inserción el campo "Últ. Actualicación" toma el valor de la fecha/hora del sistema en ese momento.
- CU.BD3: Almacena los datos horarios: Similar lo descrito en el punto anterior. En este caso se trata de los datos horarios.
- CU.BD4: Almacena las llamadas al API de la AEMET, el método consultado y el resultado obtenido.
- CU.BD5: Almacen los datos enviados por los sensores Zeus. No existe un campo "Últ. Actualicación" pero en cambio el campo "Fecha Registro" nos indica el momento de la inserción del registro.

Trás un análisis de los datos que debemos analizar, se procede a crear el modelo de datos expuesto en la figura 5.3 que servirá como modelo incial para la creación de la base de datos sobre la que realizan todas las consultas para la visualización y exportación de registros.

[2]Sistema capaz de almacenar gran cantidad de datos, relacionados y estructurados, para su posterior consulta, modificación o nuevo ingreso de datos de manera rápida y simple, concentrando toda la información en un único lugar.

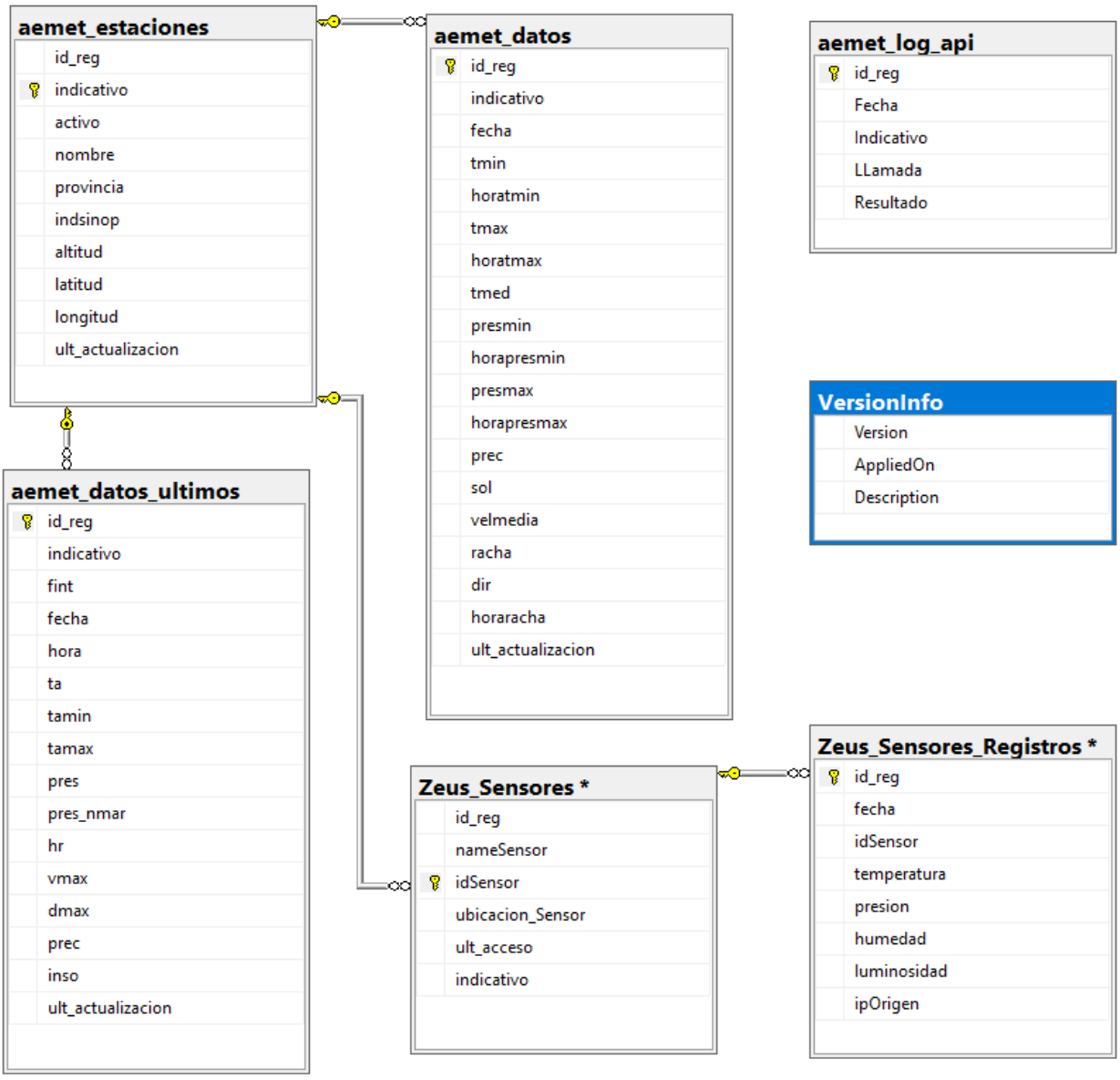


Figura 5.3: Modelo de la Base de Datos Zeus

A continuación describimos cada una de ellas:

- "aemet-datos": Donde se almacenan los datos diarios obtenidos desde la AMET.
- "aemet-datos-ultimos": En esta tabla se almacenan los datos horarios obtenidos desde la AEMET.
- "aemet-estaciones": Es en esta tabla donde se almacenan las estaciones de observación registradas en la AMET.
- "aemet-log-api": Aquí se almacena el registro de las llamadas a la API Aemet con el resultado obtenido y indicador de éxito o fallo.
- "Zeus-Log": En desuso, sólo para fines de depuración.

- "Zeus-Sensores": Donde se almacenan y gestionan los sensores Zeus.
- "Zeus-Sensores-Registros": Donde se almacenan los datos que son enviados por los sensores Zeus.

A la hora de buscar un identificador para nombrar las tablas, hemos optado por identificarlas con un prefijo que indique que sistema se encarga de alimentarla. De esta manera todas aquellas tablas cuyo nombre comience por "aemet" se nutren con datos que provienen de la AEMET en cambio aquellas que comiencen por "Zeus" son los sensores Zeus quien las alimenta.

En el apéndice 4 se expone la estructura completa de la base de datos Zeus con el detalle de los campos y los tipos, de cada una de las tablas que la componen.

5.2.2. Selección del motor de base de datos

Ahora bien, nos encontramos en una situación donde sabemos que datos necesitamos almacenar y que base de datos necesitamos para ello. Como en etapas anteriores de este proyecto, cabe preguntarse ¿Que motor de base de datos es el más adecuado para este proyecto? Y antes de seguir, ¿que es un motor de base de datos?

Un motor de base de datos es un elemento existente bajo el sistema de una base de datos que se utiliza para su funcionamiento. Éstos construyen los bloques sobre los que el resto de elementos de la bases de datos van a sostenerse y desarrollarse. Además es un elemento que está caracterizado por todos los componentes del sistema, los cuales se encargan de almacenar y de recuperar datos.

Los motores de bases de datos más utilizados en la actualidad son los que se relacionan a continuación:

- `Microsoft SQL Server`[3] es una herramienta creada por `Microsoft`[4] que se ha establecido para controlar y desarrollar bases de datos relacionales. Esta herramienta está acentuada por el lenguaje de desarrollo Transact – SQL.
- `MySQL`[5] es uno de los motores de bases de datos más empleados actualmente. El crecimiento de esta herramienta es cada vez mayor en comparación con los competidores del mercado. Esta herramienta se creo para controlar bases de datos relacionales.
- `SQLite`[6], al igual que muchas otras, es una herramienta del Big Data que se creó para desarrollar y controlar bases de datos relacionales. SQLite está complementada por ACID.
- `Oracle database`[7] es un sistema que gestiona bases de datos de objetos y relacionales.

[3] https://www.microsoft.com/es-es/sql-server/sql-server-2022
[4] https://www.microsoft.com/es-es/
[5] https://www.mysql.com/
[6] https://www.sqlite.org/index.html
[7] https://www.oracle.com/es/database/

- ODBC[8] (Open Database Connectivity) es una herramienta que permite el acceso a bases de datos y fue desarrollada por SQL Access Group.

- PostgreSQL[9] es una herramienta open source que está muy optimizada para guardar datos geográficos y, de esta manera, para localizar personas.

Las ventajas e inconvenientes de cada uno de los motores citados no afecta a la elección del mismo para la ejecución de este proyecto. Si es de especial interés que tanto los motores SQL Server y Oracle son propietarios de las marcas correspondientes y por tanto son productos para los que debe adquirirse una licencia con el costo asociado correspondiente. El resto de las soluciones planteadas corresponden a gestores de código abierto[10]

A pesar de ello, entre las diversas versiones del producto SQL Server que la empresa Microsoft comercializa, existe una versión denominada "Express Edition" sin costo por licencia y con ciertas limitaciones. Según la información extraída de la web[11] del fabricante:

"Express edition: La edición de SQL Server Express es una base de datos gratuita para principiantes y es ideal para aprender a compilar pequeñas aplicaciones de servidor y de escritorio orientadas a datos. Es la mejor opción para los fabricantes de software independientes, los desarrolladores y los aficionados que compilan aplicaciones cliente. Si necesita características de base de datos más avanzadas, SQL Server Express se puede actualizar sin problemas a otras versiones superiores de SQL Server. SQL Server Express LocalDB es una versión ligera de la edición Express que tiene todas sus características de capacidad de programación, se ejecuta en modo usuario y presenta una instalación rápida y sin configuración, y una lista reducida de requisitos previos."

Además de otras limitaciones técnicas, el tamaño de la base de datos permitido para este tipo de licenciamiento, sin costo, es de 10 Gb, tamaño más que suficiente para el alcance de este proyecto. No obstante se podría haber utilizado sin mayor problema cualquiera de los otros motores open source, como por ejemplo, PostgreSQL. El principal motivo a la hora de seleccionar este sistema es que ya disponíamos de él preinstalado en el servidor utilizado como repositorio del ecosistema Zeus.

5.2.3. Copia de Seguridad de la Base de Datos

A pesar de tratarse de un prototipo experimental no se deben descuidar algunas cuestiones que sí son vitales en un entorno en producción. Entre ellas están las copias de seguridad y garantizar que los servicios esenciales del sistema operativo sean y estén iniciados sin errores.

[8] https://es.wikipedia.org/wiki/Open_Database_Connectivity

[9] https://www.postgresql.org/

[10] El código abierto (en inglés: open source) es un modelo de desarrollo de software basado en la colaboración abierta.Se enfoca en los beneficios prácticos (acceso al código fuente) y en cuestiones éticas o de libertad que tanto se destacan en el software libre.Para muchos el término "libre"hace referencia al hecho de adquirir un software de manera gratuita. Sin embargo, de lo que se trata es de abaratar los costos y ampliar la participación; que sea libre no necesariamente implica que sea gratuito, lo importante sigue siendo ampliar la participación y extender libertades.

[11] https://learn.microsoft.com/es-es/sql/sql-server/editions-and-components-of-sql-server-2022?view=sql-server-ver16#sql-server-editions

Para las copias de seguridad, en esta primera fase, se ha optado por crear un fichero batch (archivo de texto plano, que contiene códigos o instrucciones escritos en el lenguaje de MSDOS) que se encarga de las siguientes tareas:

- Asigna a la variable "fecha" el valor de la fecha actual.
- Comprime utilizando la utilidad 7z el contenido del directorio "Zeus.Web" y lo copia a la carpeta "Backup_SQL" con el nombre "Zeus.Web.Fecha.7z".
- Comprime utilizando la utilidad 7z el contenido del directorio Zeus y lo copia a la carpeta "Backup_SQL" con el nombre "Zeus.Fecha.7z".
- Comprime utilizando la utilidad 7z el contenido del directorio Serene y lo copia a la carpeta "Backup_SQL" con el nombre "Serene.Fecha.7z".
- Invoca al proceso "Copia_SQL_Zeus.bat" que realiza las siguientes tareas:
  - Copia de la base de datos "Zeus" y la copia en la carpeta "Backup_SQL" con el nombre "Zeus-n.bak".
  - Copia de la base de datos "Zeus_Admin" y la copia en la carpeta "Backup_SQL" con el nombre "Zeus_Admin_n.bak" (Donde n indica el número de la copia. Se mantienen las copias de los últimos 7 días).

El contenido de la carpeta "Backp_SQL" debería estar asegurado en una ubicación externa al servidor. Para ello se ha optado por utilizar "Google Drive", que en la modalidad ofertada para uso personal (sin coste), ofrece un espacio de almacenamiento seguro en la nube de 15Gb. De esta manera tenemos en la nube los elementos esenciales para garantizar la continuidad de los servicios ofrecidos por "Zeus".

En caso de algún problema en la máquina virtual, sólo sería necesario levantar otra similar y volver a copiar las aplicaciones y restaurar la base de datos. Como es obvio, en un sistema en producción sería más ágil y ofrecería un menor tiempo de inactividad disponer de una copia de la máquina virtual (snapshot), además de las correspondientes copias de seguridad.

Otra de las cuestiones a tener presente es la disponibilidad del servicio IIS [12] (Internet Information Services), cuya misión es ofrecer servicios Web [13], FTP [14] y SMTP [15] entre otros. A veces este servicio puede llegar a pararse dejando de publicar los servicios ofertados. Por experiencia en otros proyectos y siguiendo la filosofía del mal menor, es preferible forzar un reinicio del

[12]Servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Este servicio convierte a un PC en un servidor web para Internet o una intranet, es decir que en los ordenadores que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente.

[13]Tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos.

[14]Protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

[15]Protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos

servicio varias veces al día, aunque en ese momento pueda coincidir con una conexión entrante y rechazarla.

Todas estas acciones que se repiten periódicamente son gestionadas por el propio "Programador de Tareas" que acompaña a todos los sistemas operativos, en este caso a Windows Server 2022.

En la imagen 5.4 se pueden las tres siguientes tareas programadas:

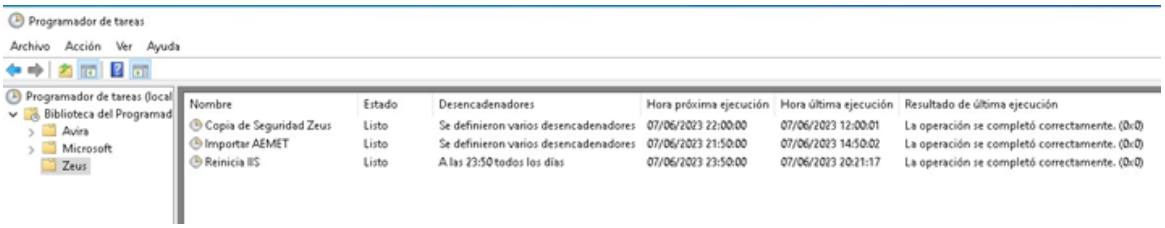


Figura 5.4: Panel denominado "Programador de Tareas" en un Windows Server 2022

- Copia de Seguridad Zeus: ejecuta el fichero de comandos "bat" comentando anteriormente.
- Importar AEMET: invoca al proceso "ImportaAemet" que llamando al API de la AEMET recupera datos y los almacena.
- Reinicia IIS: ejecuta el fichero de comandos "bat" que lanza el comando para reiniciar el servicio IIS.

5.3. Desarrollo del servicio o proceso "importaAemet"

Es el proceso que llama al Api de la AEMET y recupera, para aquellas estaciones marcadas como "activa" los datos diarios desde la última fecha registrada y los datos de las últimas 24 horas y los almacena en la base de datos. Además recupera el inventario de las estaciones de la AEMET.

Funcionalmente se realizan las siguientes subtareas:

- Obtiene la lista de las estaciones de la AEMET.
- Se obtiene la lista de los indicativos de las estaciones para las que se debe recuperar sus datos.
- Para cada una de ellas se consulta la fecha del último registro, última actualización, que obra en nuestra base de datos.
- Se llama al método del API de AEMET para recuperar los datos diarios desde la fecha del último registro hasta el último día del año en curso.
- Se llama al método del API de AEMET para recuperar los valores de las medidas de las últimas 24 horas.

En general, este proceso dispone de las siguientes variables de configuración:

- "cfg-apiKeyAemet": Llave (`key`[16]) facilitada por la AEMET cuando nos registramos como consumidores de su API.
- "cfg-SendEmailForError"(true/false): Si el valor es "true" remite un correo de alerta si alguna de las funciones internas produce una excepción de error.
- "cfg-EmailForErrorReport"(string): Dirección de correo electrónico donde remitir los correos de alerta en caso de fallo.
- "cfg-SendEmailForSuccess"(true/false): Si el valor es "true" remite un correo de alerta si alguna todas las funciones internas se han realizado con éxito.
- "cfg-EmailForSuccessReport"(string): Dirección de correo electrónica donde remitir los correos de alerta en caso de éxito en la ejecución de todas las funciones internas.
- "cfg-SQL-DataBase"(string): Nombre de la base de datos, por defecto "Zeus".
- "cfg-SQL-DataBase-User"(string): usuario de la base de datos SQL Server.
- "cfg-SQL-Server"(string): Nombre del Servidor SQL Server.
- "cfg-SQL-DataBase-Pass"(string): Contraseña del usuario con acceso de escritura para la base de datos.

En el diseño de este proceso se ha tenido en cuenta la importancia de mantener la base de datos lo más actualizada posible, de ahí que hayamos creado un sistema de avisos por correo electrónico ante un posible fallo tanto en las llamadas al api como en el almacenamiento de los datos recuperados. Obviamente estos correos pueden ser recibidos no sólo por el administrador del sistema sino por terceras personas previa configuración de los parámetros "cfg-SendEmailForError" y "cfg-EmailForErrorReport" explicados anteriormente.

No obstante, se ha añadido un plus de control almacenando el histórico de las llamadas efectuadas a la AEMET, así como el resultado, exitoso o no, de la misma. En figura 5.5 podemos ver el contenido de la tabla "aemet_log_api".

En las dos primeras líneas (06/06/2023 14:50h) se puede observar que en la primera llamada (Datos horarios) se han recuperado las medidas de los últimos 24 registros en cambio en la segunda llamada se han solicitado los Datos Diarios desde el día 03/06/2023 hasta el 31/12/2023 y no ha habido respuesta.

Por otro lado, si analizamos la última línea de la foto podemos deducir que el día 05/06/2023 a las 07:50h se intentó recuperar los datos diarios desde el día 24/05/2023 hasta fin de año sin que la AEMET devolviera dato alguno. En la llamada efectuada el 05/06/2023 a las 21:50h (línea número 8) vemos que en ese momento para la misma llamada anterior la AEMET devolvió 10 registros.

[16] https://opendata.aemet.es/centrodedescargas/inicio

SQLQuery1.sql - ZE...Administrator (54))*

```
/****** Script para el comando SelectTopNRows de SSMS  ******/
SELECT *
  FROM [Zeus].[dbo].[aemet_log_api]
  where id_reg<23
  order by fecha desc
```

100 %

Resultados | Mensajes

	id_reg	Fecha	Indicativo	LLamada	Resultado
1	22	2023-06-06 14:50:03.817	C449C	getDatosHorariosFromAemet(C449C) -> Recuperados: 24 Registros.	OK
2	21	2023-06-06 14:50:03.520	C449C	getDatosDiariosFromAemet(C449C , 2023-06-03 , 2023-12-31) -> No hay datos que satisfagan esos criterios	NO
3	20	2023-06-06 07:50:03.673	C449C	getDatosHorariosFromAemet(C449C) -> Recuperados: 24 Registros.	OK
4	19	2023-06-06 07:50:02.547	C449C	getDatosDiariosFromAemet(C449C , 2023-06-03 , 2023-12-31) -> No hay datos que satisfagan esos criterios	NO
5	18	2023-06-05 22:55:03.147	C449C	getDatosHorariosFromAemet(C449C) -> Recuperados: 24 Registros.	OK
6	17	2023-06-05 22:55:02.693	C449C	getDatosDiariosFromAemet(C449C , 2023-06-03 , 2023-12-31) -> No hay datos que satisfagan esos criterios	NO
7	16	2023-06-05 21:50:06.353	C449C	getDatosHorariosFromAemet(C449C) -> Recuperados: 24 Registros.	OK
8	15	2023-06-05 21:50:02.637	C449C	getDatosDiariosFromAemet(C449C , 2023-05-24 , 2023-12-31) -> Recuperados: 10 Registros.	OK
9	14	2023-06-05 14:50:05.607	C449C	getDatosHorariosFromAemet(C449C) -> Recuperados: 24 Registros.	OK
10	13	2023-06-05 14:50:03.387	C449C	getDatosDiariosFromAemet(C449C , 2023-05-24 , 2023-12-31) -> No hay datos que satisfagan esos criterios	NO
11	12	2023-06-05 07:50:03.567	C449C	getDatosHorariosFromAemet(C449C) -> Recuperados: 24 Registros.	OK
12	11	2023-06-05 07:50:02.630	C449C	getDatosDiariosFromAemet(C449C , 2023-05-24 , 2023-12-31) -> No hay datos que satisfagan esos criterios	NO

Figura 5.5: Registro de los accesos de los usuarios a modo de Log

Esto es debido a una actualización no diaria por parte de la AEMET de ahí que siempre se tenga en cuenta el último valor registrado a la hora de hacer la siguiente llamada.

Esto último puede apreciarse en la llamada efectuada el mismo día a las 22:55h donde ya comenzamos a pedir datos desde el día 02/06/2023 puesto que los anteriores han sido recuperados en la llamada anterior.

Hasta el día 23/05/2023 los datos se iban recuperando diariamente con un retraso de 4 días tal y como se explica en la propia web de la AEMET, en cambio a partir del día 24/05/2023 no se han actualizado los datos diariamente hasta el día 05/06/2023, momento en el que se han recuperado los valores que faltaban.

Capítulo 6

Plataforma Zeus

Llegados a este punto, no sólo tenemos una infinidad de datos, además los tenemos almacenados de manera organizada en forma de tablas, las cuales pueden relacionarse entre si creando consultas (querys). Éstas serán de gran utilidad cuando un usuario desee realizar consultas a o sobre la base de datos.

Es evidente que los resultados de esas consultas deberán ser mostrados en algún tipo de interfaz práctica y que disponga de una gran `usabilidad`[1] para el usuario que interactúa con ella.

6.1. Casos de uso

A continuación se describen los casos de uso para ese conjunto de interfaces que componen la llamada Plataforma Zeus:

- CA.PZ1: Interfaz con diversos indicadores y gráficas al estilo de cuadro de mando.
- CA.PZ2: Interfaz para consultar y gestionar las estaciones de la AEMET.
- CA.PZ3: Interfaz para consultar y gestionar los datos diarios de aquellas estaciones seleccionadas como "Activas".
- CA.PZ4: Interfaz similar a la descrita anteriormente salvo que los datos que se gestionan en ella son los datos horarios.
- CA.PZ5: Interfaz donde se pueden consultar los resultados de las llamadas al API de la AEMET.
- CA.PZ6: Interfaz donde se gestiona el parque de sensores Zeus.
- CA.PZ7: Interfaz donde se muestran los datos enviados por los sensores Zeus.
- CA.PZ8: Interfaz que mostrará las comparativas entre los datos obtenidos por los sensores Zeus y los sensores de la AEMET de aquellas estaciones en estudio.
- CA.PZ9: Interfaz inicial que validará las credenciales del usuario.

Además existirán otros casos de uso orientado a funciones administrativas y de control y destinadas a un usuario del tipo administrador o gestor de la aplicación.

[1] Cualidad de la página web o del programa informático que son sencillos de usar porque facilitan la lectura de los textos, descargan rápidamente la información y presentan funciones y menús sencillos, por lo que el usuario encuentra satisfechas sus consultas y cómodo su uso.

- CA.PA1: Interfaz donde gestionar los idiomas en los que queremos se muestren los textos y leyendas de nuestra plataforma.
- CA.PA2: Interfaz donde gestionar las traducciones de los mensajes permitiendo tener de una plataforma multi idioma.
- CA.PA3: Interfaz donde crear y gestionar los distintos roles de los usuarios.
- CA.PA4: Interfaz donde gestionar los usuarios y sus permidos de nuestra aplicación.
- CA.PA5: Interfaz donde se mostrará el registro de los accesos de los usuarios.

Todas las interfaces enumeradas anteriormente dispondrán además de dos funcionalidades a través de las cuáles se permite exportar los datos mostrados o bien a un fichero con formato Excel o bien a informe en formato PDF.

En algunas de ellas, y de modo experimental, se ha incorporado un sistema de conversión de "texto a voz" que se encarga de generar un audio, donde una voz femenina, lee el contenido del registro seleccionado.

6.2. Plataforma Zeus

Analizados los casos de uso y sin perder de vista lo descrito en la propuesta (ver capítulo 3), se considera la necesidad de construir una plataforma o portal web que contemple las funcionalidades deseadas.

6.2.1. Implementación de la Plataforma Zeus

Para la realización de esta tarea y dada la poca experiencia del autor con las herramientas típicas empleadas en el desarrollo web, se optó por utilizar la plantilla `Serene`[2] basado en la plataforma `Serenity`[3], de aplicaciones ASP.NET Core/TypeScript que se basa en tecnologías de código abierto. Su objetivo es facilitar el desarrollo y reducir los costos de mantenimiento al evitar el código repetitivo, reducir el tiempo dedicado a tareas repetitivas y aplicar las mejores prácticas de diseño de software.

Como entorno de desarrollo para trabajar con esta plantilla se ha utilizado la herramienta `Microsoft Visual Studio Comunity`[4]. Además `Microsoft Team Explorer`[5] sirvió como software de control de versiones. Ambos productos, al igual que ocurre con la base de datos SQL Server, se ofrecen bajo un tipo de versión denominado Comunity, suficiente para nuestros propósitos evitando costes de licencia.

[2] https://marketplace.visualstudio.com/items?itemName=VolkanCeylan.SereneSerenityApplicationTemplate
[3] https://serenity.is/
[4] https://visualstudio.microsoft.com/es/vs/community/
[5] https://learn.microsoft.com/es-es/azure/devops/repos/tfvc/what-is-tfvc?view=azure-devops

6.2.2. Interfaz Principal: Cuadro de Indicadores

Nuestra interfaz principal nos ofrece determinados indicadores de interés para el usuario así como una gráficas con la evolución de determinadas variables climatológicas.

Su aspecto puede verse con detalle en la figuras 6.1 y 6.2

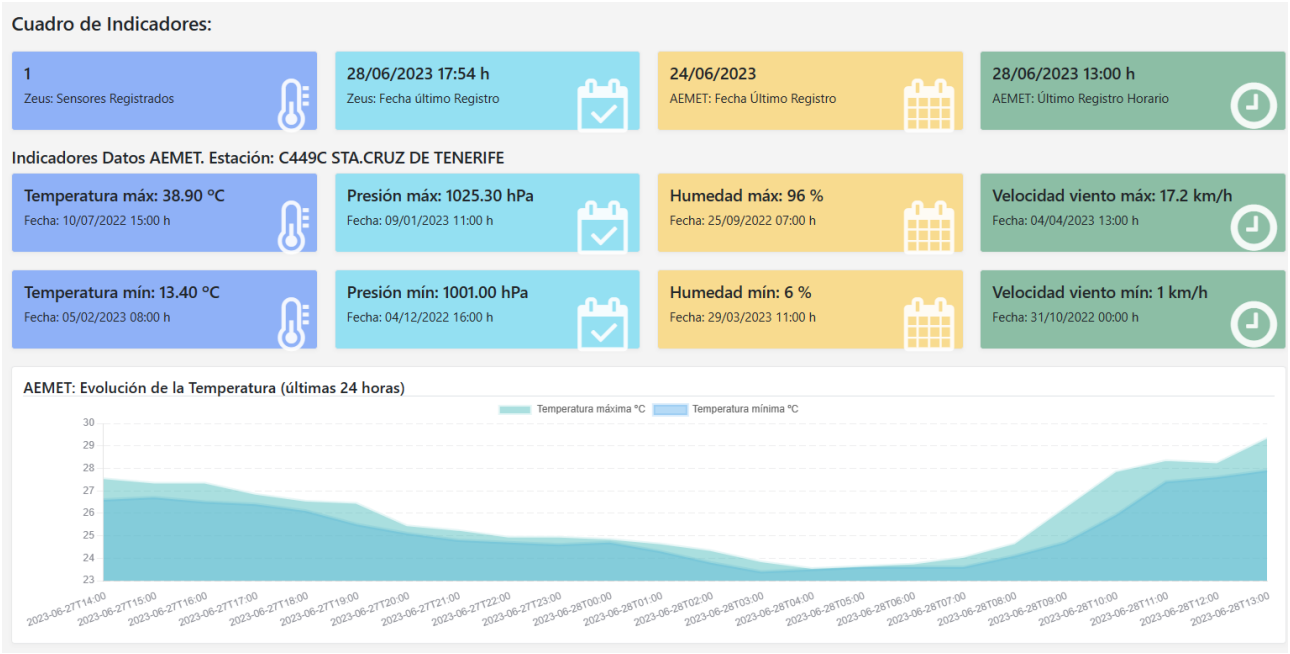


Figura 6.1: Interfaz principal Plataforma Zeus: Panel de indicadores

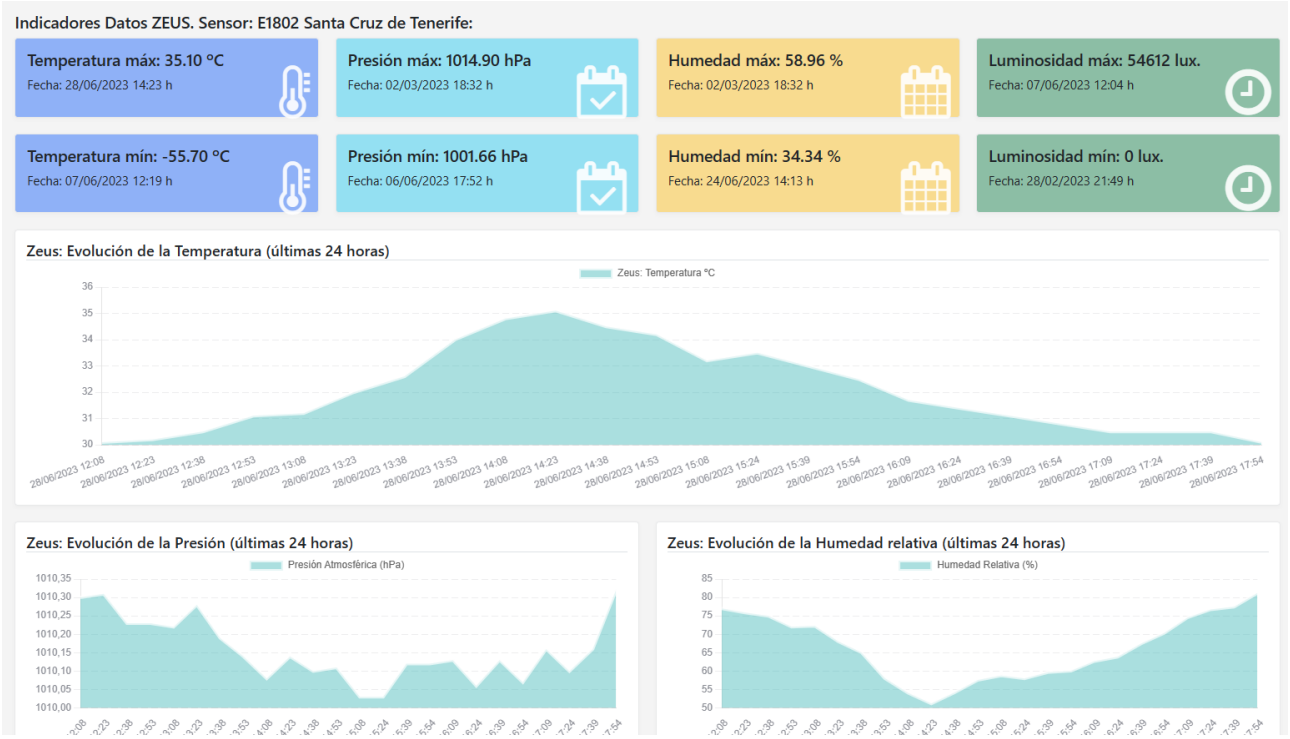


Figura 6.2: Interfaz principal Plataforma Zeus: Panel de indicadores (continuación)

6.2.3. Interfaz Principal: Menú izquierdo de opciones

Todas las opciones que se ofrecen en la Plataforma Zeus están accesibles desde un menú que se muestra en la margen izquierdo de la interfaz principal. En la figura 6.3 puede apreciarse con todas las opciones.

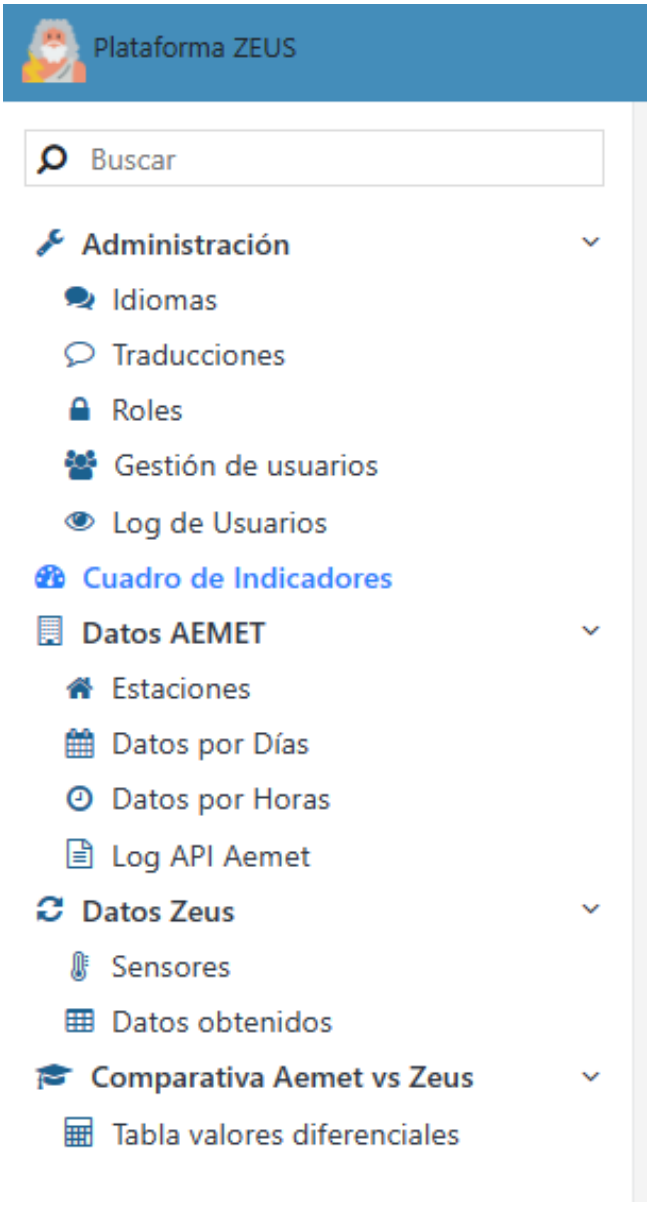


Figura 6.3: Interfaz de la Plataforma Zeus: Menú lateral izquierdo de opciones

6.2.4. Interfaz Datos AEMET: Estaciones, Datos Diarios y Horarios

Son las encargadas de mostrar los datos de las estaciones de la AEMET, sus valores diarios y sus valores horarios, ver figuras 6.4 y 6.5 y respectivamente (por ser similar a Datos Diarios no se muestra la correspondiente a Datos Horarios).

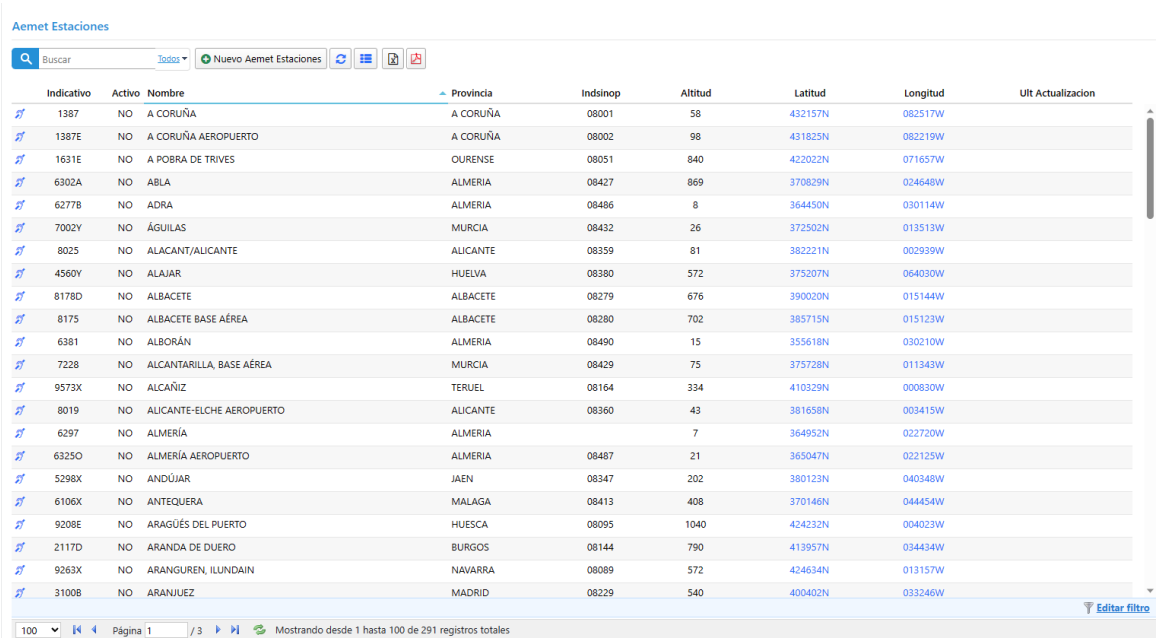


Figura 6.4: Interfaz de la Plataforma Zeus: Datos AEMET - Estaciones

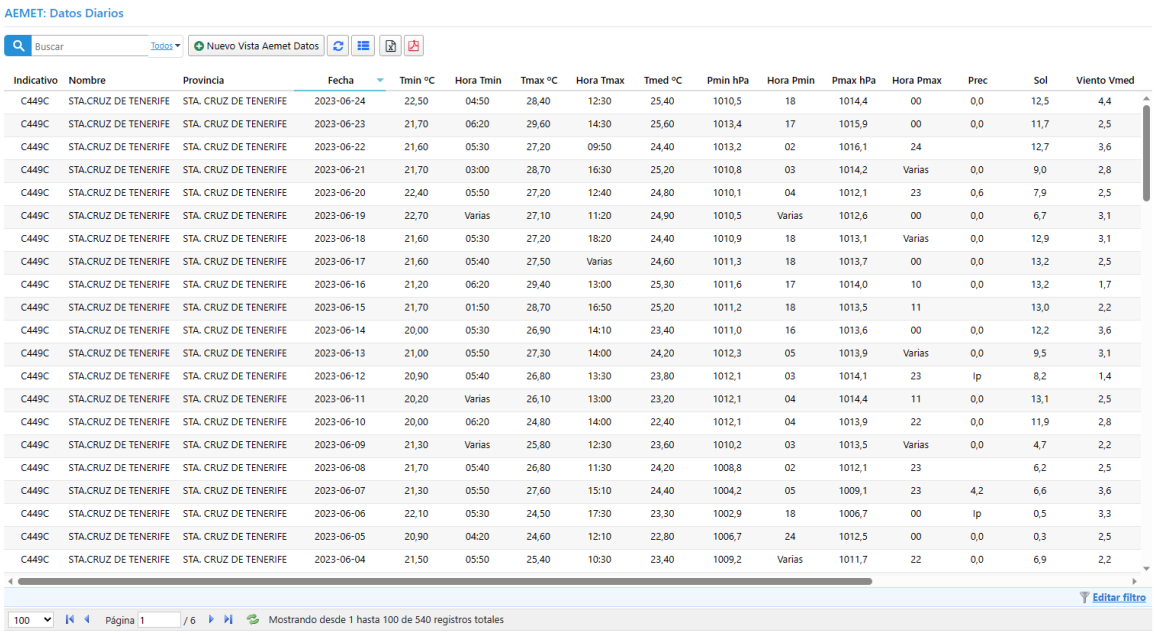


Figura 6.5: Interfaz de la Plataforma Zeus: Datos AEMET - Datos por Dias

En la visualización de los datos de las estaciones se ofrece, además, la posibilidad de visualizar la localización de cada estación mediante Google Maps, simplemente pinchando sobre uno de los campos de latitud y longitud.

6.2.5. Interfaz Datos Zeus: Sensores y Datos Obtenidos

Esta interfaz se divide en dos diferentes vistas. En la primera se muestran los sensores Zeus que hemos desplegado 6.6 y en la segunda los valores recogidos por dichos sensores 6.7.

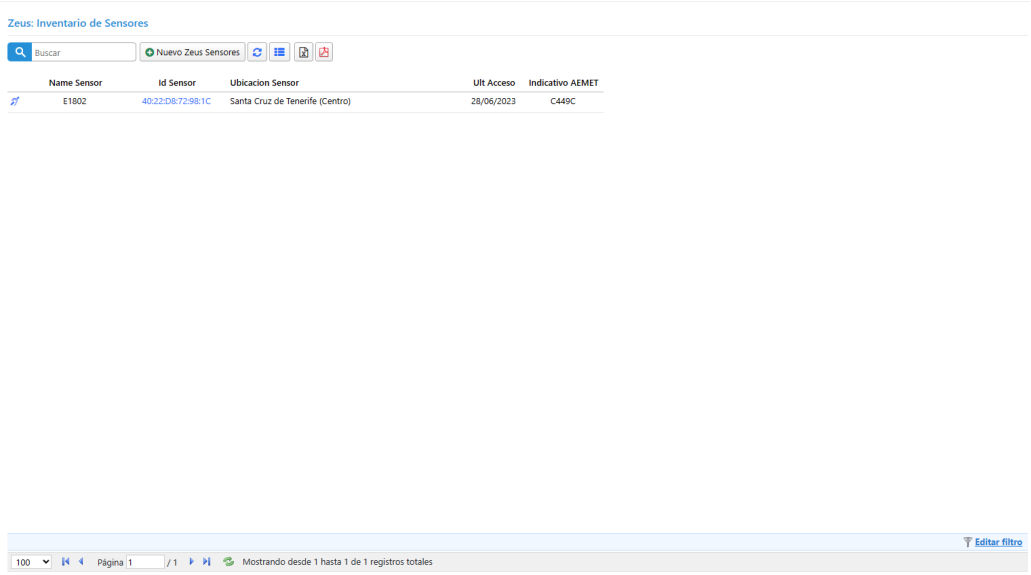


Figura 6.6: Interfaz de la Plataforma Zeus: Datos Zeus - Sensores

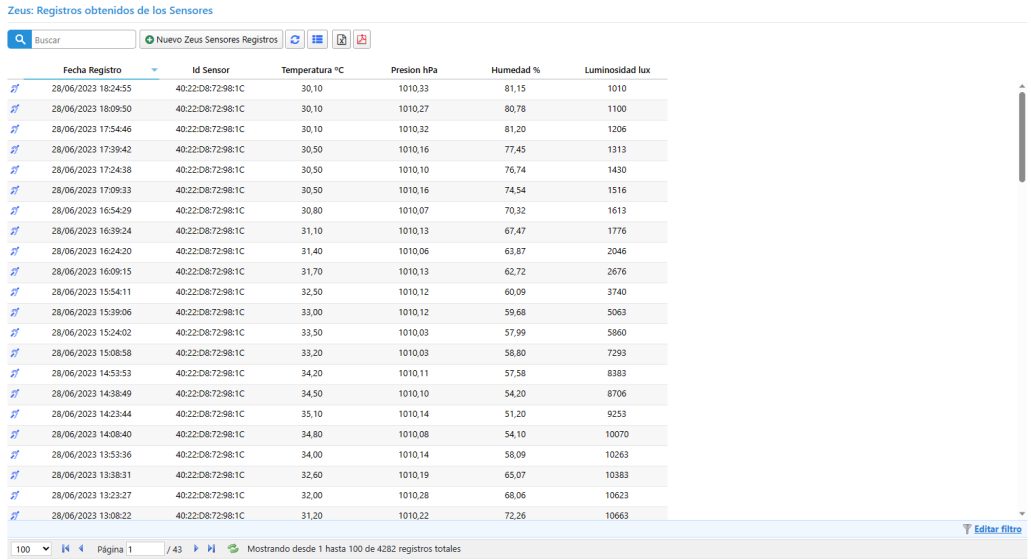
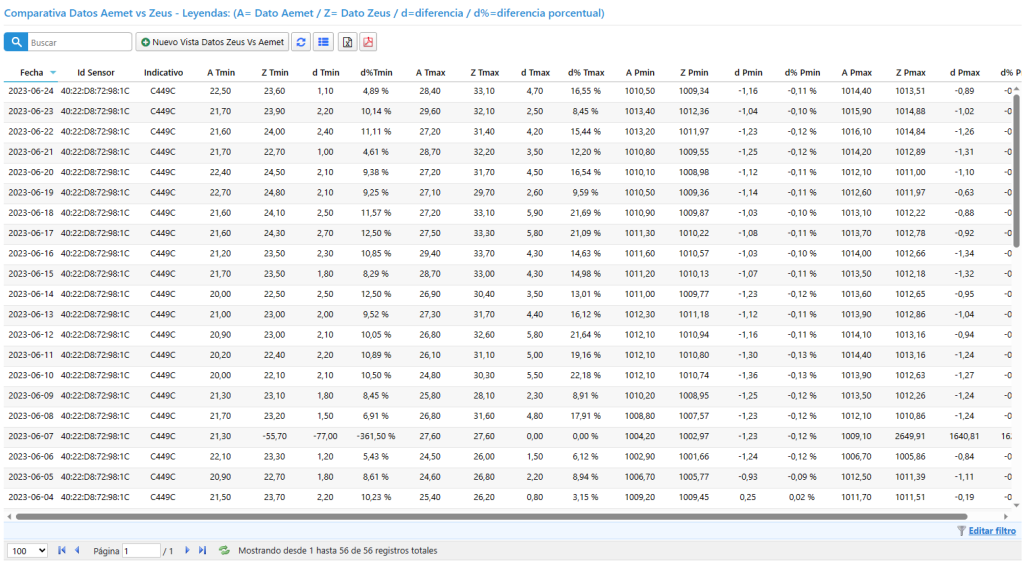


Figura 6.7: Interfaz de la Plataforma Zeus: Datos Zeus - Datos Obtenidos

6.2.6. Interfaz Comparativa Aemet vs Zeus: Tabla valores diferenciales

Esta interfaz muestra el objetivo real de nuestro proyecto. Se trata de poder mostrar, comparar y poder operar con los datos obtenidos desde los dos diferentes orígenes: la AEMET y Zeus.

En la figura 6.6 podemos apreciar la riqueza de los datos que en ella se muestran:



Fecha	Id Sensor	Indicativo	A Tmin	Z Tmin	d Tmin	d%Tmin	A Tmax	Z Tmax	d Tmax	d% Tmax	A Pmin	Z Pmin	d Pmin	d% Pmin	A Pmax	Z Pmax	d Pmax	d% P
2023-06-24	40:22:D8:72:98:1C	C449C	22,50	23,60	1,10	4,89 %	28,40	33,10	4,70	16,55 %	1010,50	1009,34	-1,16	-0,11 %	1014,40	1013,51	-0,89	-0
2023-06-23	40:22:D8:72:98:1C	C449C	21,70	23,90	2,20	10,14 %	29,60	32,10	2,50	8,45 %	1013,40	1012,36	-1,04	-0,10 %	1015,90	1014,88	-1,02	-0
2023-06-22	40:22:D8:72:98:1C	C449C	21,60	24,00	2,40	11,11 %	27,20	31,40	4,20	15,44 %	1013,20	1011,97	-1,23	-0,12 %	1016,10	1014,84	-1,26	-0
2023-06-21	40:22:D8:72:98:1C	C449C	21,70	22,70	1,00	4,61 %	28,70	32,20	3,50	12,20 %	1010,80	1009,55	-1,25	-0,12 %	1014,20	1012,89	-1,31	-0
2023-06-20	40:22:D8:72:98:1C	C449C	22,40	24,50	2,10	9,38 %	27,20	31,70	4,50	16,54 %	1010,10	1008,98	-1,12	-0,11 %	1012,10	1011,00	-1,10	-0
2023-06-19	40:22:D8:72:98:1C	C449C	22,70	24,80	2,10	9,25 %	27,10	29,70	2,60	9,59 %	1010,50	1009,36	-1,14	-0,11 %	1012,60	1011,97	-0,63	-0
2023-06-18	40:22:D8:72:98:1C	C449C	21,60	24,10	2,50	11,57 %	27,20	33,10	5,90	21,69 %	1010,90	1009,87	-1,03	-0,10 %	1013,10	1012,22	-0,88	-0
2023-06-17	40:22:D8:72:98:1C	C449C	21,60	24,30	2,70	12,50 %	27,50	33,30	5,80	21,09 %	1011,30	1010,22	-1,08	-0,11 %	1013,70	1012,78	-0,92	-0
2023-06-16	40:22:D8:72:98:1C	C449C	21,20	23,50	2,30	10,85 %	29,40	33,70	4,30	14,63 %	1011,60	1010,57	-1,03	-0,10 %	1014,00	1012,66	-1,34	-0
2023-06-15	40:22:D8:72:98:1C	C449C	21,70	23,50	1,80	8,29 %	28,70	33,00	4,30	14,98 %	1011,20	1010,13	-1,07	-0,11 %	1013,50	1012,18	-1,32	-0
2023-06-14	40:22:D8:72:98:1C	C449C	20,00	22,50	2,50	12,50 %	26,90	30,40	3,50	13,01 %	1011,00	1009,77	-1,23	-0,12 %	1013,60	1012,65	-0,95	-0
2023-06-13	40:22:D8:72:98:1C	C449C	21,00	23,00	2,00	9,52 %	27,30	31,70	4,40	16,12 %	1012,30	1011,18	-1,12	-0,11 %	1013,90	1012,86	-1,04	-0
2023-06-12	40:22:D8:72:98:1C	C449C	20,90	23,00	2,10	10,05 %	26,80	32,60	5,80	21,64 %	1012,10	1010,94	-1,16	-0,11 %	1014,10	1013,16	-0,94	-0
2023-06-11	40:22:D8:72:98:1C	C449C	20,20	22,40	2,20	10,89 %	26,10	31,10	5,00	19,16 %	1012,10	1010,80	-1,30	-0,13 %	1014,40	1013,16	-1,24	-0
2023-06-10	40:22:D8:72:98:1C	C449C	20,00	22,10	2,10	10,50 %	24,80	30,30	5,50	22,18 %	1012,10	1010,74	-1,36	-0,13 %	1013,90	1012,63	-1,27	-0
2023-06-09	40:22:D8:72:98:1C	C449C	21,30	23,10	1,80	8,45 %	25,80	28,10	2,30	8,91 %	1010,20	1008,95	-1,25	-0,12 %	1013,50	1012,26	-1,24	-0
2023-06-08	40:22:D8:72:98:1C	C449C	21,70	23,20	1,50	6,91 %	26,80	31,60	4,80	17,91 %	1008,80	1007,57	-1,23	-0,12 %	1012,10	1010,86	-1,24	-0
2023-06-07	40:22:D8:72:98:1C	C449C	21,30	-55,70	-77,00	-361,50 %	27,60	27,60	0,00	0,00 %	1004,20	1002,97	-1,23	-0,12 %	1009,10	2649,91	1640,81	16
2023-06-06	40:22:D8:72:98:1C	C449C	22,10	23,30	1,20	5,43 %	24,50	26,00	1,50	6,12 %	1002,90	1001,66	-1,24	-0,12 %	1006,70	1005,86	-0,84	-0
2023-06-05	40:22:D8:72:98:1C	C449C	20,90	22,70	1,80	8,61 %	24,60	26,80	2,20	8,94 %	1006,70	1005,77	-0,93	-0,09 %	1012,50	1011,39	-1,11	-0
2023-06-04	40:22:D8:72:98:1C	C449C	21,50	23,70	2,20	10,23 %	25,40	26,20	0,80	3,15 %	1009,20	1009,45	0,25	0,02 %	1011,70	1011,51	-0,19	-0

Figura 6.8: Interfaz de la Plataforma Zeus: Comparativo AEMET versus ZEUS

6.3. Base de Datos Zeus.Admin

Los casos de uso que hemos propuesto son los siguientes:

- CU.BDA1: Almacenar y actualizar, si procede, los idiomas.
- CU.BDA2: Almacena las traducciones de los mensajes y leyendas al resto de los idiomas disponibles.
- CU.BDA3: Almacenar los roles y preferencias de los usuarios.
- CU.BDA4: Almacena y gestiona los usuarios del sistema y sus permisos.
- CU.BDA5: Almacena el registro de los accesos de los usuarios.

En este caso la plantilla "Serene" nos ofrece una gran ayuda pues ya incluye una base de datos con la que poder realizar las funciones anteriormente descritas. El modelo de dicha base de datos se muestra en la figura 6.9.

Tras un análisis de la misma se añadió la tabla "UsersLog" que consideramos con el fin de almacenar en ella los accesos de los usuarios a la plataforma a modo de auditoría.

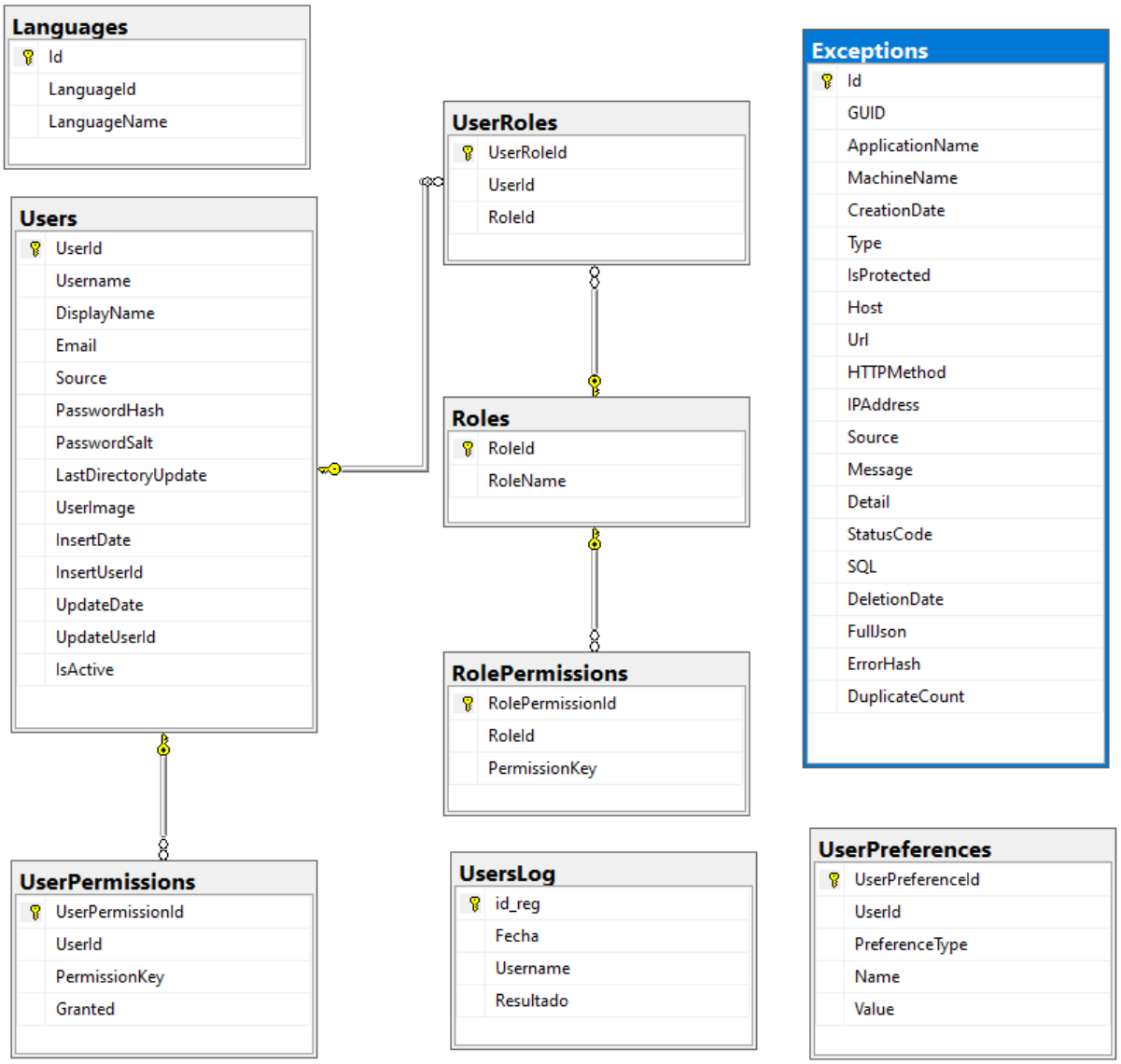


Figura 6.9: Modelo de la Base de Datos Zeus_Admin

6.3.1. Diseño de la Base de Datos

La base de datos Zeus.Admin está compuesta por las siguientes tablas que describimos a continuación:

- "Exceptions": Donde se almacenan las excepciones ocurridas en tiempo de ejecución.
- "Languajes": En esta tabla se almacenan los idiomas que queremos tener disponibles.
- "RolesPermissions": Tabla que relaciona los permisos asignados a un determinado rol de usuarios.
- "Roles": Aquí se gestionan los roles de los usuarios que se quieran definir.
- "UserPermissions": Aqui se almacenan los permisos concedidos y denegados a un determinado usuario.
- "UserPreferences": Donde se almacenan y gestionan las prefenrecias de los usuarios (no se utiliza en este proyecto).
- "UserRoles": Donde se almacenan los usuarios que perteneces a un determiando rol de usuarios.
- "Users": Donde se gestionan los usuarios del sistema.
- "UsersLog": Donde se almacenan los accesos al sistema por parte de los usuarios.
- "VersionInfo": Donde se almacena versiones de las migraciones de la base de datos (sólo para usos de desarrollo).

En el apéndice 5 se expone la estructura completa de la base de datos Zeus_Admin, con el detalle de los campos y los tipos de cada una de las tablas que la componen.

Capítulo 7

Pruebas

A lo largo de este capítulo se irá describiendo toda la colección de pruebas modulares que se han ido realizando a lo largo de la construcción de este proyecto.

Las pruebas modulares son un enfoque para testear el software que se basa en probar cada módulo individualmente antes de integrarlos en el sistema completo. Cada módulo es evaluado de forma independiente para detectar posibles errores y problemas antes de ser combinado con otros módulos en el sistema completo. Para realizar pruebas modulares es necesario seguir un proceso en el que se divide el software en módulos individuales y se realiza una prueba exhaustiva de cada uno de estos módulos. Esto se hace para asegurarse de que cada módulo cumple con los requisitos y funciona correctamente antes de integrarlo con otros módulos.

Siguiendo este mismo criterio se describen y realizan las pruebas descritas en los siguientes subcapítulos.

7.1. Pruebas del sensor y toma de datos

El objetivo de esta fase de pruebas consitió en comprobar que la programación de la placa principal ESP32 fuera la correcta y que se pudieran obtener los valores necesarios del sensor.

Para ello, en una versión reducida de la placa ESP32, se cargó el código y se analizaron los resultados obtenidos a través del puerto serie. Así, mientras se ejecutaba el código se podía monitorizar los valores leídos además de los contenidos de otras variables del código.

En la figura 7.1 puede observarse el sensor conectado a la placa principal y en la figura 7.2 el resultado obtenido a traves del monitor del puerto serie.

7.2. Pruebas del Servicio web

Una vez desarrollado e implantado sobre el IIS (Internet Information Services) el servicio web ZeusService.asmx (que es el llamado por el sensor) se procedió a realizar la prueba del mismo.

Se utilizó para ello un código simple realizado en C# que generaba dos números aleatorios y los usaba como parámetros en la llamada al método provisional putRegistros (p1, p2). Fruto de esas pruebas se generan las siguientes inserciones en la base de datos.

Validando así el funcionamiento del servicio web y su correcta publicación.

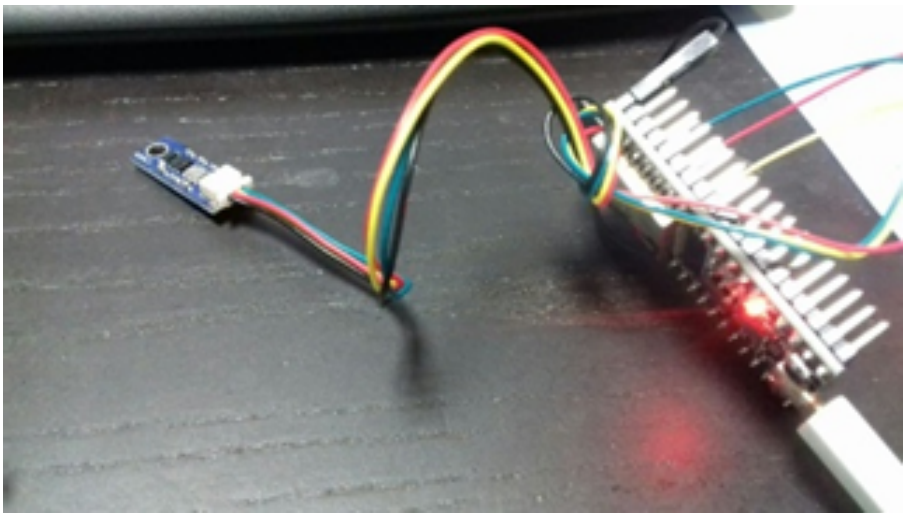


Figura 7.1: Sensor conectado a la placa ESP32-WROOM

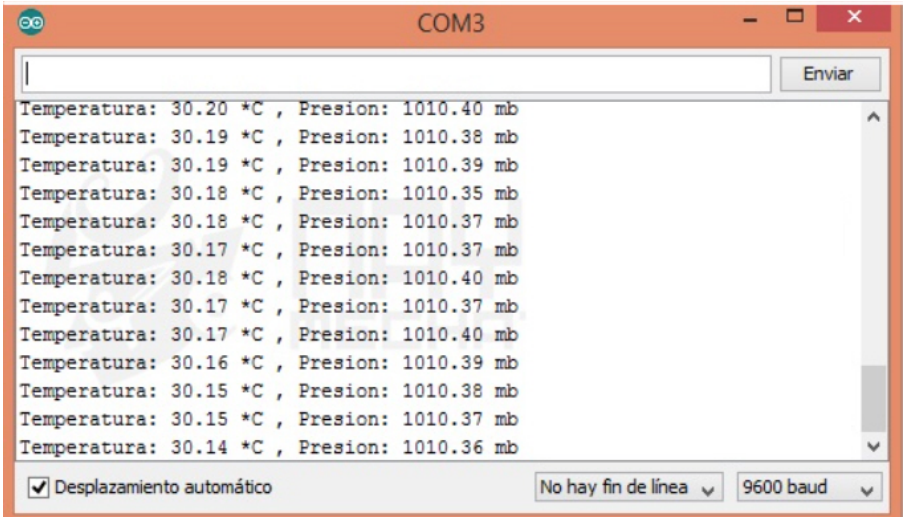


Figura 7.2: Resultados obtenidos a través del puerto serie

Los registros almacenados en la base de datos pueden verse en la figura 7.3

7.3. Pruebas de envio de datos del sensor al servidor ZeusService.asmx

Una vez teniendo verificado el sensor y las medidas obtenidas, y habiendo comprobado el correcto funcionamiento del servicio web, se procedió a testear la parte del código de la placa ESP32 que realiza la llamada al servicio web ZeusService.asmx y a través del cual se remiten los datos medidos.

Para tal fin se utilizó un código más simple que no toma los datos del sensor, simplemente genera números aleatorios y los remite al servidor remoto.

Básicamente es una prueba similar a la ejecutada en el paso anterior, sólo que esta vez ejecutada

```
select * from registros order by id_reg desc
```

100 %

Resultados | Mensajes

	id_reg	p1	p2	plp	ult_actualizacion
1	304	01	84	192.168.1.76	NULL
2	303	15	06	192.168.1.76	NULL
3	302	18	02	192.168.1.76	NULL
4	301	16	33	192.168.1.76	NULL
5	300	98	75	192.168.1.76	NULL
6	299	14	18	192.168.1.76	NULL
7	298	78	32	192.168.1.76	NULL
8	297	20	64	192.168.1.76	NULL
9	296	47	86	192.168.1.76	NULL

Figura 7.3: Registros aleatorios almacenados en la base de datos

desde la propia placa ESP32. En la figura 7.4 puede apreciarse la placa realizando esta prueba, cuyo resultado fue que estos valores se enviaron y almacenaron de manera correcta en la base de datos.

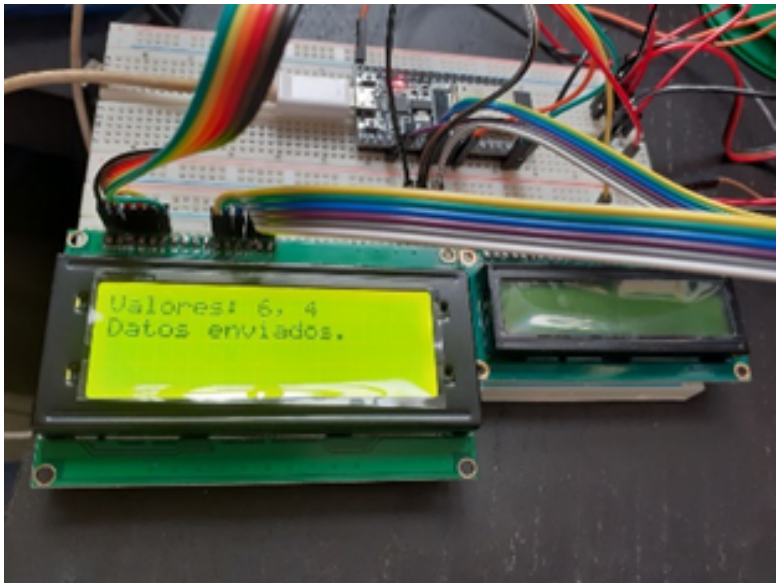


Figura 7.4: Placa ESP32 generando números para remitir al servidor remoto

7.4. Pruebas finales con el sensor con el código ESP32 unificado

Una vez pasadas con éxito las pruebas anteriores, sólo restaba unificar las dos ramas del código ESP32, la que de toma las medidas del sensor y, la que se encarga de remitirlos al servidor y ponerlo en marcha.

Fue una vez el circuito alimentado cuando se comprobó que su comportamiento era el esperado, tal y como se puede observar en la figura 7.5 que muestra una medición con el display encendido.

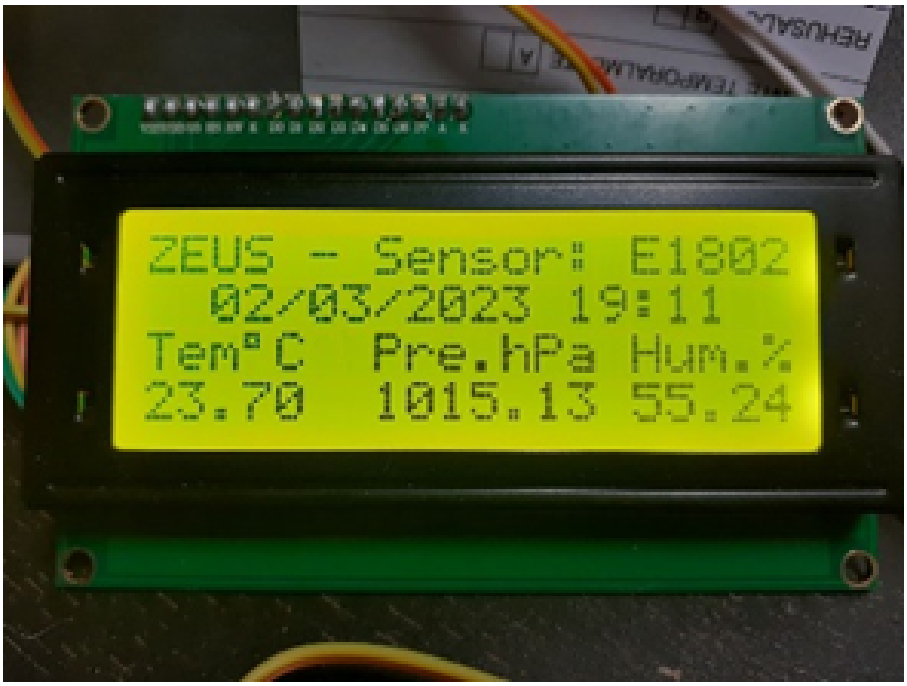


Figura 7.5: Zeus mostrando datos en un Display iluminado

Las medidas recogidas fueron almacenadas en la base de datos. Un extracto de las mismas puede verse en la imagen figura 7.6, en el cual se puede apreciar los registros correspondientes a la imagen anterior (figura 7.5) en la primera fila de la tabla.

Resultados Mensajes

	id_reg	fecha	idSensor	temperatura	presion	humedad	luminosidad
1	176	2023-03-02 19:11:44.510	40:22:D8:72:98:1C	23,7	1015,13	55,24	193
2	175	2023-03-02 19:11:32.213	40:22:D8:72:98:1C	23,7	1015,09	55,09	196
3	174	2023-03-02 19:10:04.650	40:22:D8:72:98:1C	23,7	1015,08	55,15	193
4	173	2023-03-02 19:09:35.777	40:22:D8:72:98:1C	23,7	1015,06	55,13	196
5	172	2023-03-02 18:59:34.883	40:22:D8:72:98:1C	23,6	1015,07	56,16	196
6	171	2023-03-02 18:44:31.007	40:22:D8:72:98:1C	23,6	1014,95	57,77	200
7	170	2023-03-02 18:42:09.583	40:22:D8:72:98:1C	23,6	1014,97	57,28	196

Figura 7.6: Registros de los datos enviados por el Sensor Zeus

Una vez todos los componentes ensamblados y colocados en el interior de una caja de metacrilato diseñada a medida para este propósito tal y como se aprecia en la figura 7.7

Figura 7.7: Aspecto final del Sensor Zeus ensamblado en su caja de Metacrilato

7.5. Pruebas en las llamadas a la API de la AEMET

Si bien en un principio se utilizó la herramienta "SopaUI" (para las pruebas de los servicios web de la AEMET), finalmente por practicidad y puesto que dichos servicios devuelven ficheros "json" perfectamente legibles para los desarrolladores, se optó por usar directamente el propio entorno de desarrollo Visual Studio.

En el código, justo después de hacer la llamada, se añadió un punto de interrupción que nos permitió analizar la variable con el resultado devuelvo por la AEMET. Si no hay error y además, el resultado es una cadena con formato "json", es cuando se analizó el contenido de los valores para comprobar su coherencia.

En la figura 7.8 puede observase con detalle el ejemplo de las pruebas al recuperar los datos diarios de la AEMET, de la estación con indicativo "C449C", ubicada en Santa Cruz de Tenerife.

Además se procedió a relizar otra prueba, esta vez recuperando los datos horarios de la misma estación. El resultado se muestra la figura 7.9.

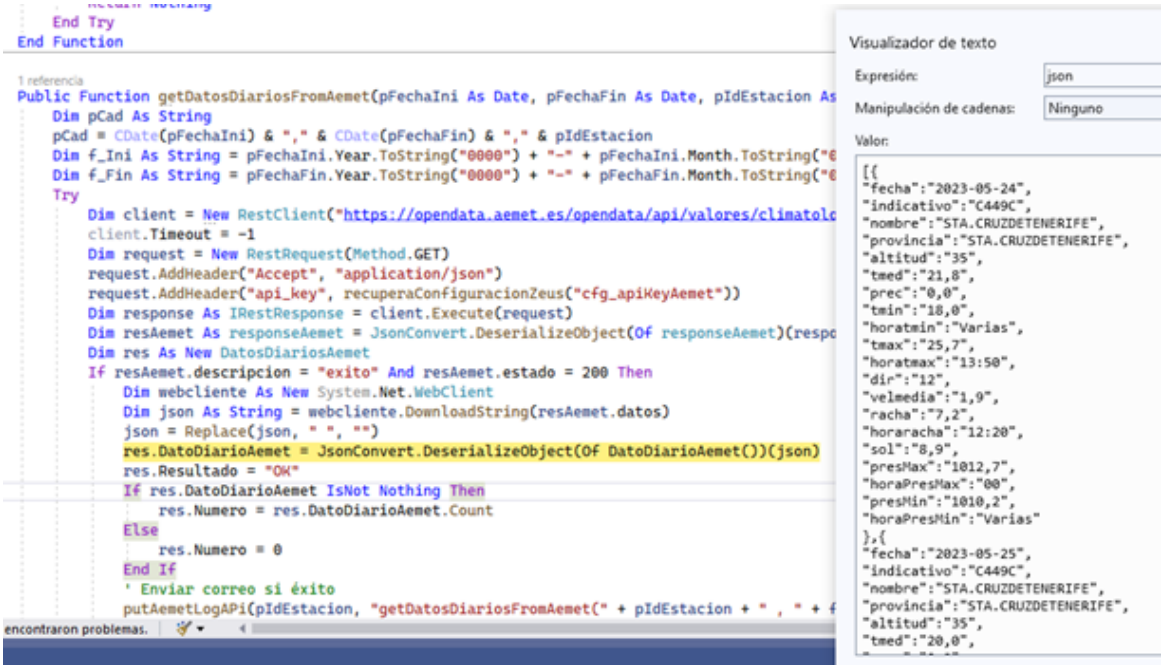


Figura 7.8: Consultado los resultados devueltos por la AEMET. Datos diarios.

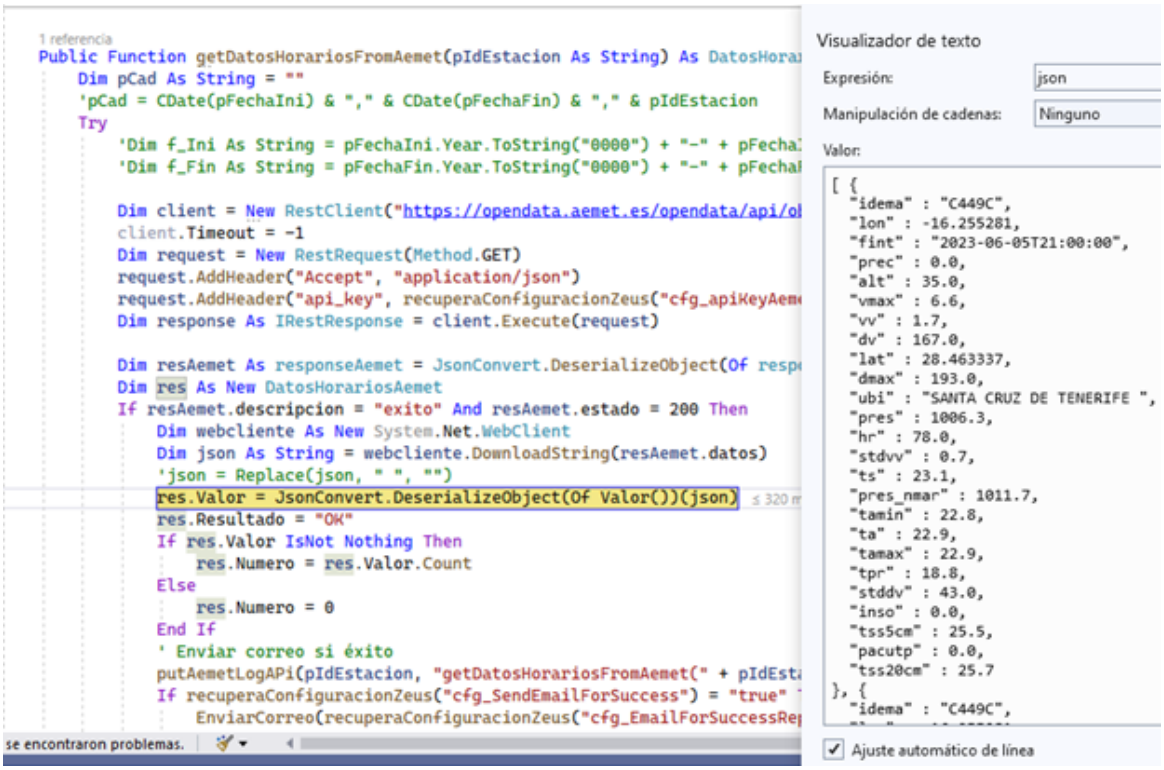


Figura 7.9: Consultado los resultados devueltos por la AEMET. Datos horarios.

7.6. Pruebas de la plataforma de visualización "Zeus.Web"

Como se ha indicando anteriormente, esta herramienta es la encargada de ofrecer una serie de interfaces al usuario donde poder consultar los datos recopilados, tanto de la AEMET como de los sensores Zeus, así como una serie de gráficos con la evolución de determinadas variables atmosféricas.

Al acceder al enlace de la plataforma, en primer lugar, se procedió a la comprobación de la autenticación del usuario. Ver figura 7.10.

Si las credenciales introducidas son correctas accederemos a la interfaz principal (figura 7.11) que está compuesta por un menú de opciones en el lateral izquierdo de la misma y por un panel donde se indican valores máximos y mínimos de algunas variables y algunas gráficas con la evolución de dichos valores en las últimas 24 horas.

Con el fin de comprobar que los valores máximos y mínimos que se muestran son los correctos se realiza la consulta indicada en la figura 7.12 directamente a la base de datos obteniendo los resultados esperados.

Se procede de igual manera para confirmar que los registros que se ofrecen en el resto de las interfaces también son los correctos, a modo de ejemplo se consulta los "Datos por Días" para la estación "C449C" ubicada en Santa Cruz de Tenerife. En las figuras 7.13 y figura 7.14 podemos ver, respectivamente, la interfaz presentando los datos y la consulta correspondiente a la base de datos. La prueba mostró que los datos coincidían.

La opción "Comparativa Aemet vs Zeus" es la que podría presentar algún error pues cruza los datos de varias tablas usando para ello los campos comunes a ambas, que son "indicativo" y "fecha". La consulta se muestra en la figura 7.15 y el resultado obtenido en la figura 7.16 y se puede ver que coinciden.

Por último, quedaría validar algunas de las opciones existentes en las rejillas de datos. Para ello se toma como ejemplo la opción "Datos Aemet - Estaciones", figura 7.17.

- Exportación a PDF: Genera un fichero con formato PDF con los registros que se muestran en ese momento en la rejilla (figura 7.18).
- Exportación a Excel: Genera un fichero con formato Excel con los registros que se muestran en ese momento en la rejilla (figura 7.19).
- Filtrado de registros: Permite filtrar los datos mostrados mediante los campos "Indicativo", "Nombre", "Provincia", "Activo" (figura 7.20).
- Filtro personalizado "Editar filtro": Nos permite con cierto grado de libertad elaborar un filtro personalizado. Se prueba con la opción del menú "Zeus - Datos Obtenidos" (figura 7.21).

  Queremos obtener todos aquellos registros cuya temperatura sea mayor o igual a 27ºC y cuya humedad relativa sea mayor o igual al 70 % (figura 7.22). Filtrando por estas condiciones, obtenemos lo mostrado en la figura 7.23. Se debe tener en cuenta que se

Figura 7.10: Interfaz de acceso a Zeus, donde se valida el usuario

pueden ir añadiendo condiciones de filtrado con ambos operadores lógicos: “AND” y “OR”.

- Filtro personalizado “Acciones asociadas a un determinado registro: En este caso concreto

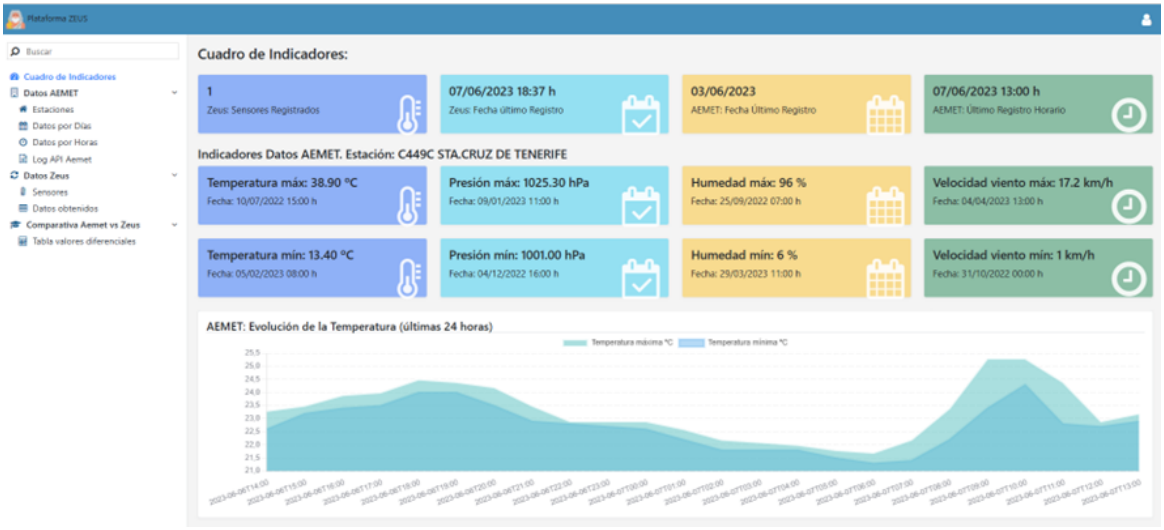


Figura 7.11: Interfaz sobre se muestran algunos indicadores y gráficas

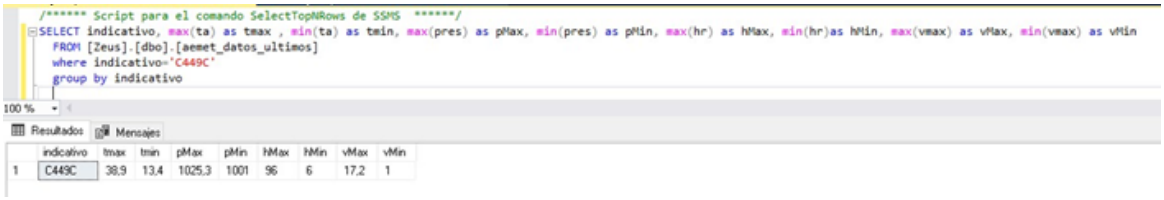


Figura 7.12: Consulta de valores realizada sobre la base de datos Zeus

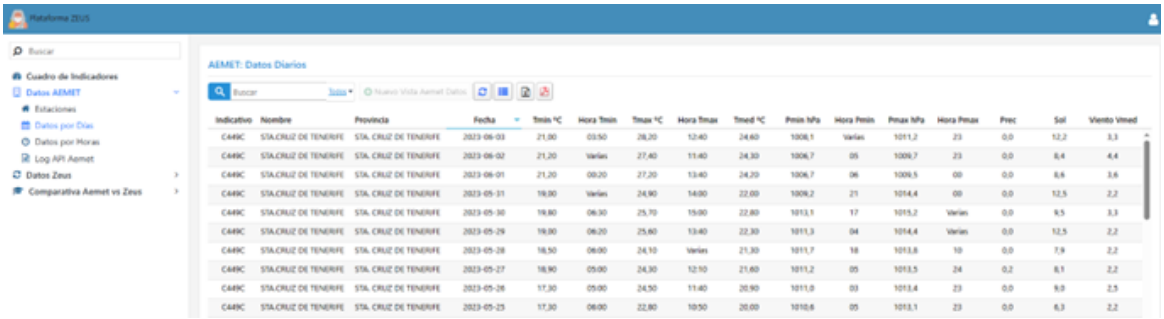


Figura 7.13: Interfaz de Zeus que muestra los datos diarios de una estación de la AEMET

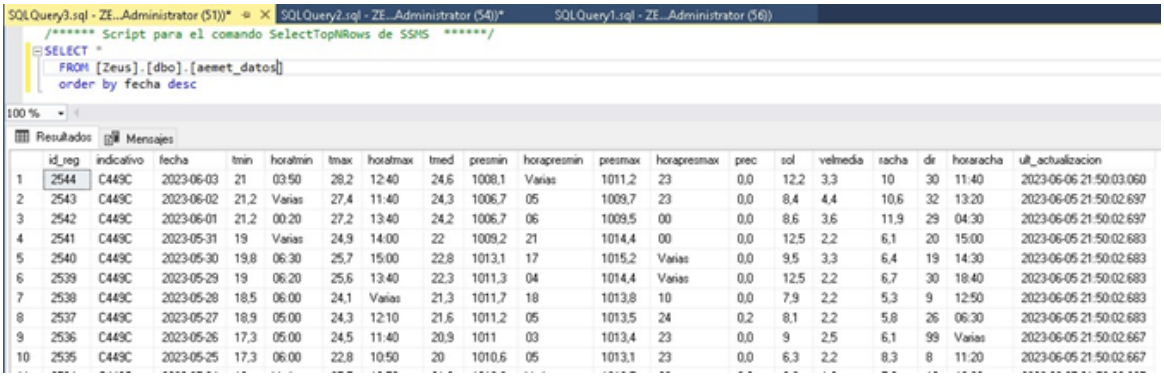


Figura 7.14: Consulta realizada sobre la base de datos Zeus

al pulsar sobre las coordenadas de una determinada estación, nos ofrece su ubicación

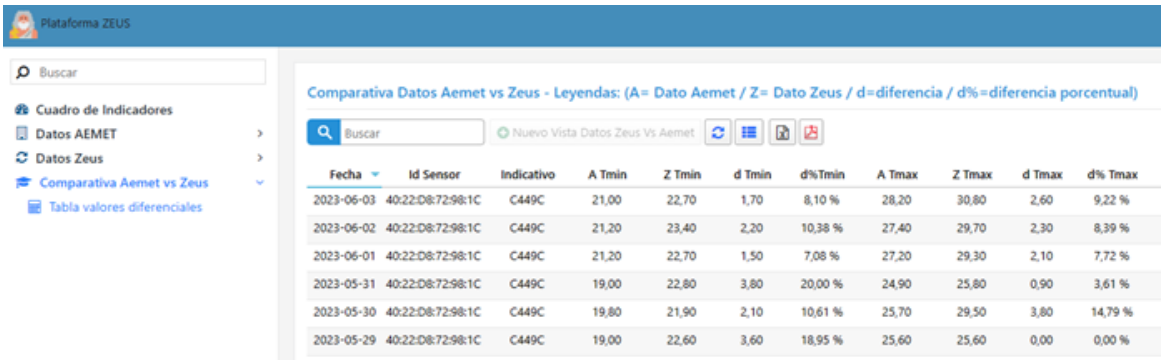


Figura 7.15: Datos mostrados en Zeus: Comparativa AEMET vs Zeus

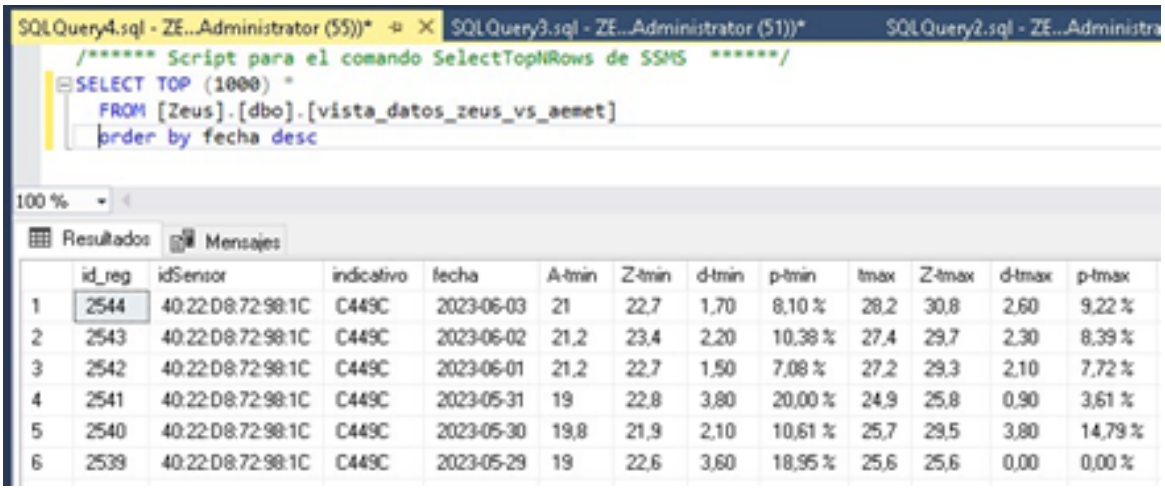


Figura 7.16: Consulta a la base de datos de datos AEMET vs Zeus

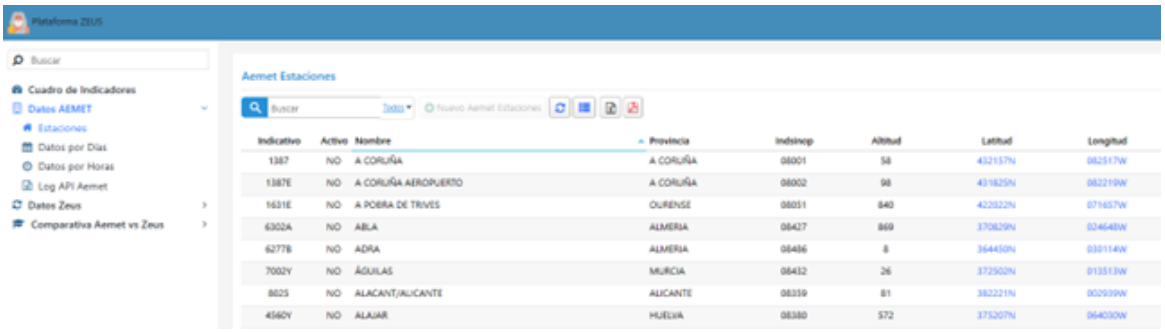
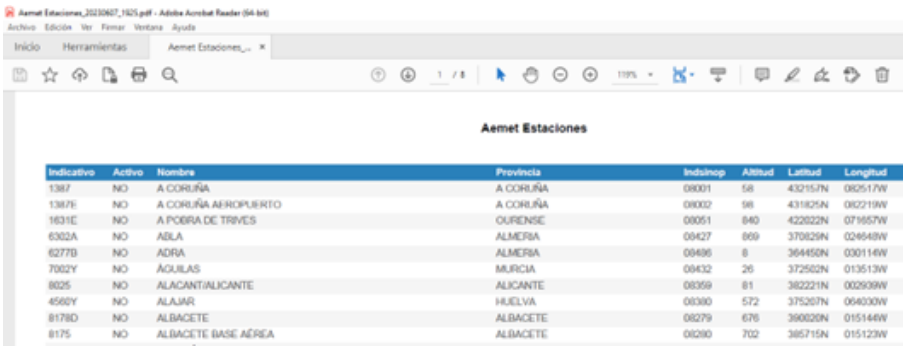


Figura 7.17: Intefaz que muestra las estaciones de la AEMET de nuestra base de datos

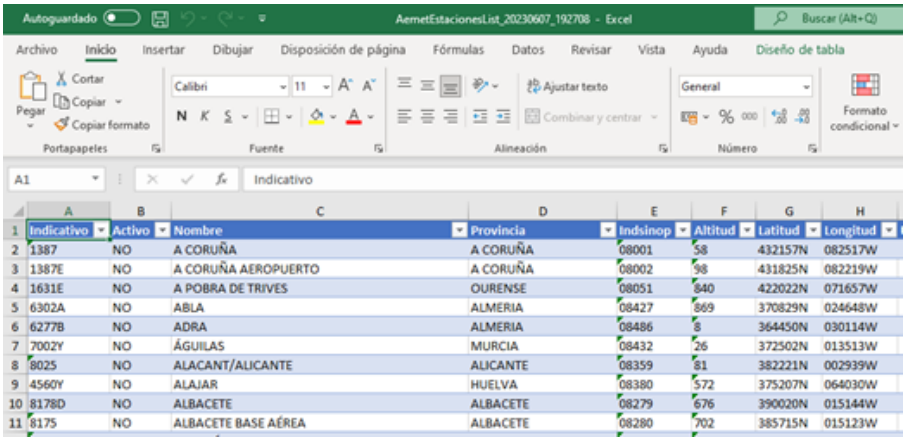
geográfica a través de “Google Maps”, para la estación ubicada en el Aeropuerto de la isla de La Palma (figura 7.24).



Aemet Estaciones

Indicativo	Activo	Nombre	Provincia	Indsinop	Altitud	Latitud	Longitud
1387	NO	A CORUÑA	A CORUÑA	08001	58	432157N	082517W
1387E	NO	A CORUÑA AEROPUERTO	A CORUÑA	08002	98	431825N	082219W
1631E	NO	A POBRA DE TRIVES	OURENSE	08051	840	422022N	071657W
6302A	NO	ABLA	ALMERIA	08427	869	370829N	024648W
6277B	NO	ADRA	ALMERIA	08486	8	364450N	030114W
7002Y	NO	ÁGUILAS	MURCIA	08432	26	372502N	013513W
8025	NO	ALACANT/ALICANTE	ALICANTE	08359	81	382221N	002939W
4560Y	NO	ALAJAR	HUELVA	08380	572	375207N	064030W
8178D	NO	ALBACETE	ALBACETE	08279	676	390020N	015144W
8175	NO	ALBACETE BASE AÉREA	ALBACETE	08280	702	385715N	015123W

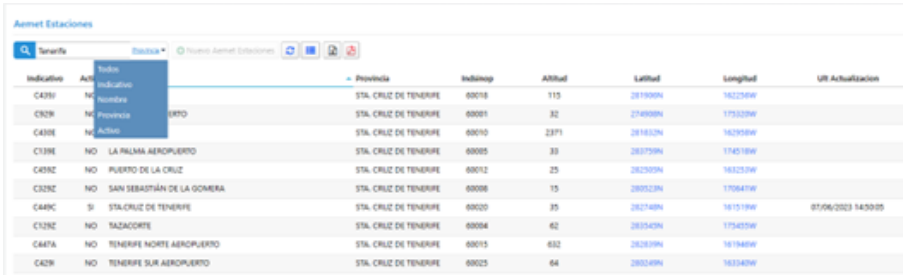
Figura 7.18: Ejemplo de fichero PDF generado desde la Plataforma Zeus



AemetEstacionesList_20230607_192708 - Excel

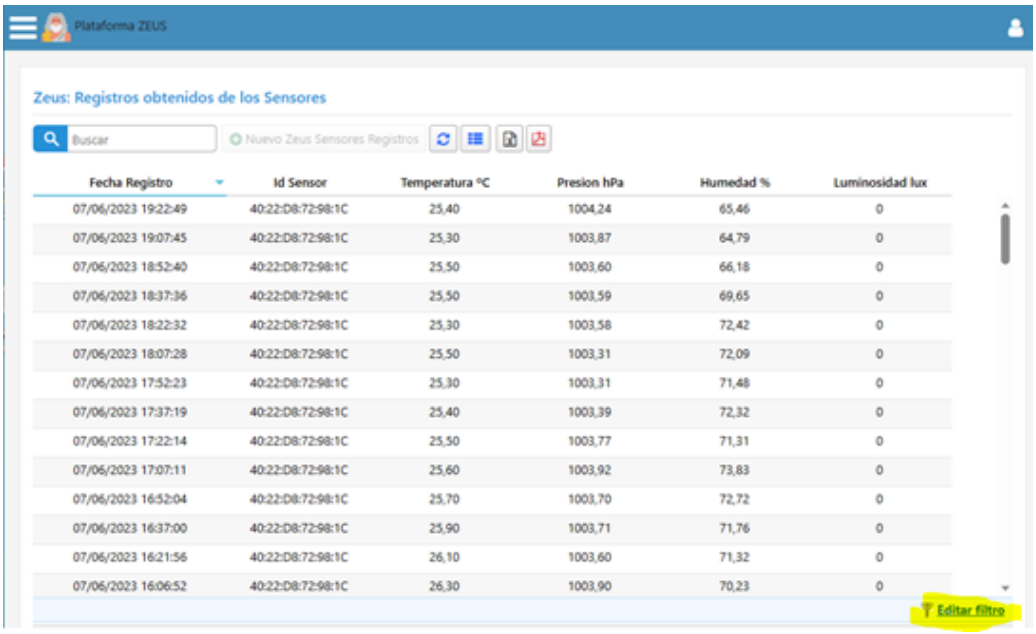
Indicativo	Activo	Nombre	Provincia	Indsinop	Altitud	Latitud	Longitud
1387	NO	A CORUÑA	A CORUÑA	08001	58	432157N	082517W
1387E	NO	A CORUÑA AEROPUERTO	A CORUÑA	08002	98	431825N	082219W
1631E	NO	A POBRA DE TRIVES	OURENSE	08051	840	422022N	071657W
6302A	NO	ABLA	ALMERIA	08427	869	370829N	024648W
6277B	NO	ADRA	ALMERIA	08486	8	364450N	030114W
7002Y	NO	ÁGUILAS	MURCIA	08432	26	372502N	013513W
8025	NO	ALACANT/ALICANTE	ALICANTE	08359	81	382221N	002939W
4560Y	NO	ALAJAR	HUELVA	08380	572	375207N	064030W
8178D	NO	ALBACETE	ALBACETE	08279	676	390020N	015144W
8175	NO	ALBACETE BASE AÉREA	ALBACETE	08280	702	385715N	015123W

Figura 7.19: Ejemplo de fichero Excel generado desde la Plataforma Zeus



Aemet Estaciones

Figura 7.20: Ejemplo de un filtrado usando campos previamente establecidos.



Fecha Registro	Id Sensor	Temperatura °C	Presion hPa	Humedad %	Luminosidad lux
07/06/2023 19:22:49	40:22:D8:72:98:1C	25,40	1004,24	65,46	0
07/06/2023 19:07:45	40:22:D8:72:98:1C	25,30	1003,87	64,79	0
07/06/2023 18:52:40	40:22:D8:72:98:1C	25,50	1003,60	66,18	0
07/06/2023 18:37:36	40:22:D8:72:98:1C	25,50	1003,59	69,65	0
07/06/2023 18:22:32	40:22:D8:72:98:1C	25,30	1003,58	72,42	0
07/06/2023 18:07:28	40:22:D8:72:98:1C	25,50	1003,31	72,09	0
07/06/2023 17:52:23	40:22:D8:72:98:1C	25,30	1003,31	71,48	0
07/06/2023 17:37:19	40:22:D8:72:98:1C	25,40	1003,39	72,32	0
07/06/2023 17:22:14	40:22:D8:72:98:1C	25,50	1003,77	71,31	0
07/06/2023 17:07:11	40:22:D8:72:98:1C	25,60	1003,92	73,83	0
07/06/2023 16:52:04	40:22:D8:72:98:1C	25,70	1003,70	72,72	0
07/06/2023 16:37:00	40:22:D8:72:98:1C	25,90	1003,71	71,76	0
07/06/2023 16:21:56	40:22:D8:72:98:1C	26,10	1003,60	71,32	0
07/06/2023 16:06:52	40:22:D8:72:98:1C	26,30	1003,90	70,23	0

Figura 7.21: Interfaz donde se ha remarcado la opción de “Editar filtro”

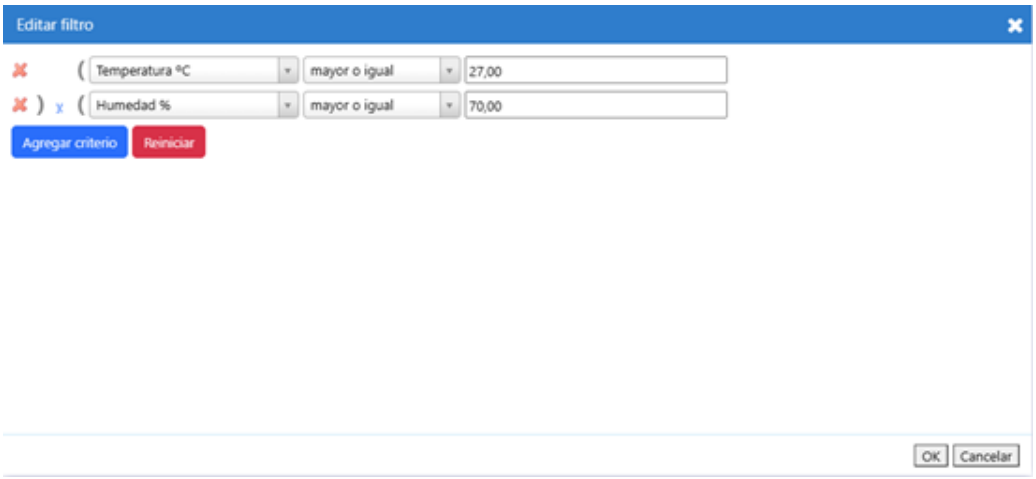


Figura 7.22: Detalle del formulario donde de edita el filtro personalizado

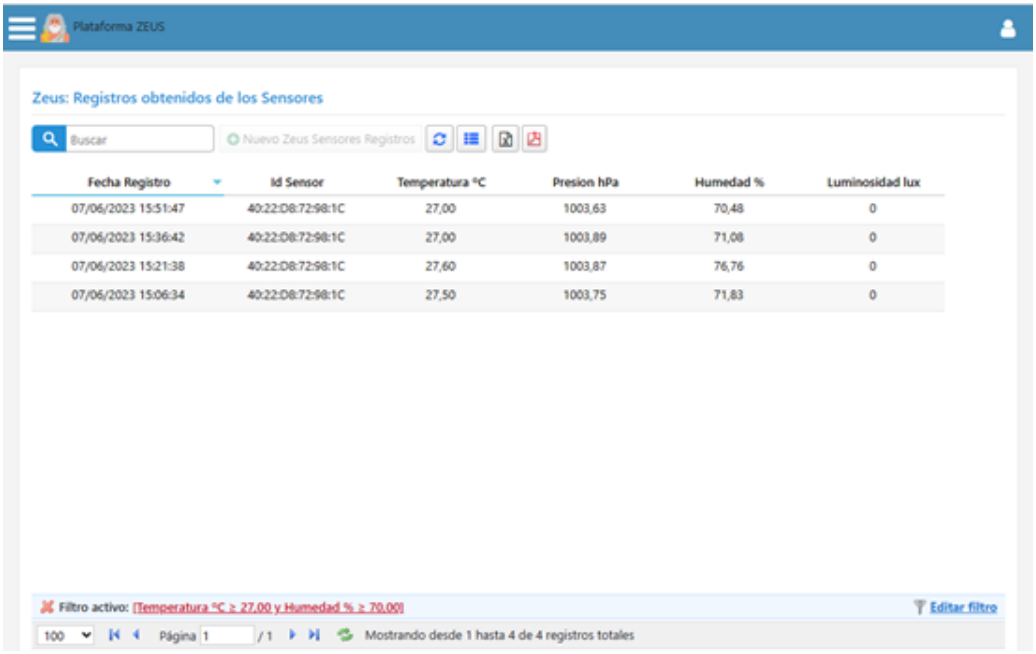


Figura 7.23: Resultado del filtrado personalizado

Figura 7.24: Estación de la AEMET en el Aeropuerto de la Isla de la Palma según Google Maps

7.7. Análisis y comparativa de los datos recopilados

Una vez comprobadas todas las fases anteriores del proyecto se procedió a instalar el sensor en una ubicación adecuada y a alimentarlo de manera apropiada. La idea era clara. Era el momento en que el sensor hiciera el trabajo para el que fue diseñado, es decir, recopilar datos y enviarlos al servidor con la frecuencia establecida (15 minutos).

Por otro lado, se seleccionó de entre todas las estaciones de la AEMET la más próxima a la ubicación del sensor Zeus y se escogió la estación de la AEMET con indicativo "C499C" que está instalada en el Centro Meteorológico Territorial de Canarias. Su ubicación puede apreciarse en la figura 7.25.

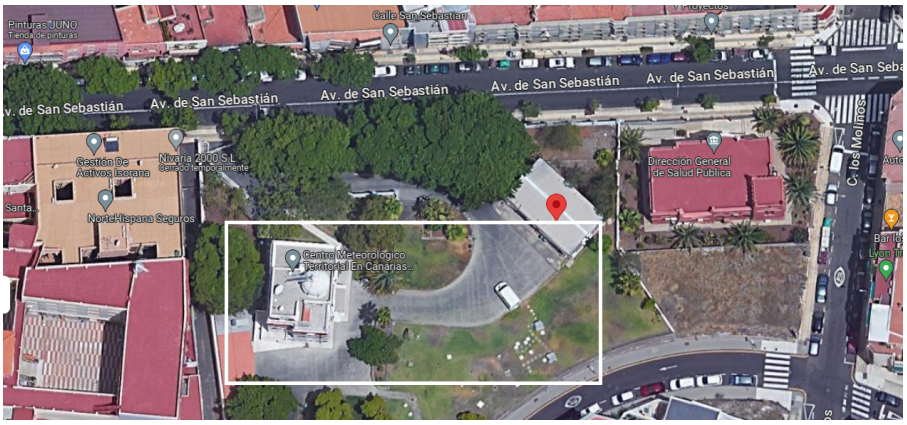


Figura 7.25: Centro Meteorológico Territorial de Canarias según Google Maps

Si ampliamos el mapa con el fin de poder ver las dos ubicaciones e incluso la distancia, en línea recta, que las separa podemos ver que dicha distancia es de 695 metros, tal y como se muestra en la figura 7.26 que ha sido obtenida a través de la herramienta `Google Maps`[1].

Figura 7.26: Distancia en línea recta que separa las ubicaciones en estudio

Utilizando la misma herramienta, podemos calcular la diferencia de altitud de ambas ubicaciones, que es de 36 metros, pues el sensor Zeus se encuentra a 61 metros sobre el nivel del mar y

[1] https://mapsplatform.google.com/intl/es/

la estación de la AEMET a 25 metros (siempre según información obtenida de Google Maps, según la propia AEMET la estación se encuentra a 35 metros sobre el nivel del mar). Esto se observa en la figura 7.27

Figura 7.27: Detalle del recorrido a pie entre las dos ubicaciones en estudio

Con fecha 19 de junio se consulta la opción "Tabla valores diferenciales" de Zeus y se obtiene la tabla mostrada en la figura 7.28.

Comparativa Datos Aemet vs Zeus - Leyendas: (A= Dato Aemet / Z= Dato Zeus / d=diferencia / d%=diferencia porcentual)

Fecha	Id Sensor	Indicativo	A Tmin	Z Tmin	d Tmin	d%Tmin	A Tmax	Z Tmax	d Tmax	d% Tmax	A Pmin	Z Pmin	d Pmin	d% Pmin	A Pmax	Z Pmax	d Pmax	d% Pmax
2023-06-19	40:22:D8:72:98:1C	C449C	22,70	24,80	2,10	9,25 %	27,10	29,70	2,60	9,59 %	1010,50	1009,36	-1,14	-0,11 %	1012,60	1011,97	-0,63	-0,06 %
2023-06-18	40:22:D8:72:98:1C	C449C	21,60	24,10	2,50	11,57 %	27,20	33,10	5,90	21,69 %	1010,90	1009,87	-1,03	-0,10 %	1013,10	1012,22	-0,88	-0,09 %
2023-06-17	40:22:D8:72:98:1C	C449C	21,60	24,30	2,70	12,50 %	27,50	33,30	5,80	21,09 %	1011,30	1010,22	-1,08	-0,11 %	1013,70	1012,78	-0,92	-0,09 %
2023-06-16	40:22:D8:72:98:1C	C449C	21,20	23,50	2,30	10,85 %	29,40	33,70	4,30	14,63 %	1011,60	1010,57	-1,03	-0,10 %	1014,00	1012,66	-1,34	-0,13 %
2023-06-15	40:22:D8:72:98:1C	C449C	21,70	23,50	1,80	8,29 %	28,70	33,00	4,30	14,98 %	1011,20	1010,13	-1,07	-0,11 %	1013,50	1012,18	-1,32	-0,13 %
2023-06-14	40:22:D8:72:98:1C	C449C	20,00	22,50	2,50	12,50 %	26,90	30,40	3,50	13,01 %	1011,00	1009,77	-1,23	-0,12 %	1013,60	1012,65	-0,95	-0,09 %
2023-06-13	40:22:D8:72:98:1C	C449C	21,00	23,00	2,00	9,52 %	27,30	31,70	4,40	16,12 %	1012,30	1011,18	-1,12	-0,11 %	1013,90	1012,86	-1,04	-0,10 %
2023-06-12	40:22:D8:72:98:1C	C449C	20,90	23,00	2,10	10,05 %	26,80	32,60	5,80	21,64 %	1012,10	1010,94	-1,16	-0,11 %	1014,10	1013,16	-0,94	-0,09 %
2023-06-11	40:22:D8:72:98:1C	C449C	20,20	22,40	2,20	10,89 %	26,10	31,10	5,00	19,16 %	1012,10	1010,80	-1,30	-0,13 %	1014,40	1013,16	-1,24	-0,12 %
2023-06-10	40:22:D8:72:98:1C	C449C	20,00	22,10	2,10	10,50 %	24,80	30,30	5,50	22,18 %	1012,10	1010,74	-1,36	-0,13 %	1013,90	1012,63	-1,27	-0,13 %
2023-06-09	40:22:D8:72:98:1C	C449C	21,30	23,10	1,80	8,45 %	25,80	28,10	2,30	8,91 %	1010,20	1008,95	-1,25	-0,12 %	1013,50	1012,26	-1,24	-0,12 %
2023-06-08	40:22:D8:72:98:1C	C449C	21,70	23,20	1,50	6,91 %	26,80	31,60	4,80	17,91 %	1008,80	1007,57	-1,23	-0,12 %	1012,10	1010,86	-1,24	-0,12 %
2023-06-07	40:22:D8:72:98:1C	C449C	21,30	-55,70	-77,00	-361,50 %	27,60	27,60	0,00	0,00 %	1004,20	1002,97	-1,23	-0,12 %	1009,10	2649,91	1640,81	162,60 %
2023-06-06	40:22:D8:72:98:1C	C449C	22,10	23,30	1,20	5,43 %	24,50	26,00	1,50	6,12 %	1002,90	1001,66	-1,24	-0,12 %	1006,70	1005,86	-0,84	-0,08 %
2023-06-05	40:22:D8:72:98:1C	C449C	20,90	22,70	1,80	8,61 %	24,60	26,80	2,20	8,94 %	1006,70	1005,77	-0,93	-0,09 %	1012,50	1011,39	-1,11	-0,11 %
2023-06-04	40:22:D8:72:98:1C	C449C	21,50	23,70	2,20	10,23 %	25,40	26,20	0,80	3,15 %	1009,20	1009,45	0,25	0,02 %	1011,70	1011,51	-0,19	-0,02 %
2023-06-03	40:22:D8:72:98:1C	C449C	21,00	22,70	1,70	8,10 %	28,20	30,80	2,60	9,22 %	1008,10	1007,82	-0,28	-0,03 %	1011,20	1009,58	-1,62	-0,16 %
2023-06-02	40:22:D8:72:98:1C	C449C	21,20	23,40	2,20	10,38 %	27,40	29,70	2,30	8,39 %	1006,70	1006,46	-0,24	-0,02 %	1009,70	1009,45	-0,25	-0,02 %
2023-06-01	40:22:D8:72:98:1C	C449C	21,20	22,70	1,50	7,08 %	27,20	29,30	2,10	7,72 %	1006,70	1006,45	-0,25	-0,02 %	1009,50	1009,58	0,08	0,01 %
2023-05-31	40:22:D8:72:98:1C	C449C	19,00	22,80	3,80	20,00 %	24,90	25,80	0,90	3,61 %	1009,20	1009,01	-0,19	-0,02 %	1014,40	1014,45	0,05	0,00 %
2023-05-30	40:22:D8:72:98:1C	C449C	19,80	21,90	2,10	10,61 %	25,70	29,50	3,80	14,79 %	1013,10	1012,97	-0,13	-0,01 %	1015,20	1014,80	-0,40	-0,04 %

Figura 7.28: Tabla con valores y diferencias entre datos de Zeus vs AEMET a 20 Junio 2023

Donde:

- A Tmin es la temperatura mínima según la AEMET.
- Z Tmin es la temperatura mínima según el sensor Zeus.
- d Tmin es el valor diferencial entre las medidas anteriores.
- d %Tmin es el valor diferencial porcentual entre las medidas anteriores.

Nota: La misma definición anterior se aplica a Tmax (temperatura máxima), Pmax (Presión máxima) y Pmin (Presión mínima).

A continuación nos centramos en analizar las diferencias observadas en cada una de las variables que hemos considerado comparar en este estudio.

7.7.1. Análisis de las diferencias: Temperatura Mínima

La diferencia media de la temperatura mínima es de +2.08ºC (Z Tmin-A Tmin).

Si procedemos a comparar sobre una gráfica los valores de Zeus y de la AEMET podemos apreciar que la diferencia entre estos valores se mantiene estable.

En la figura 7.29 podemos analizarlo con más detalle.

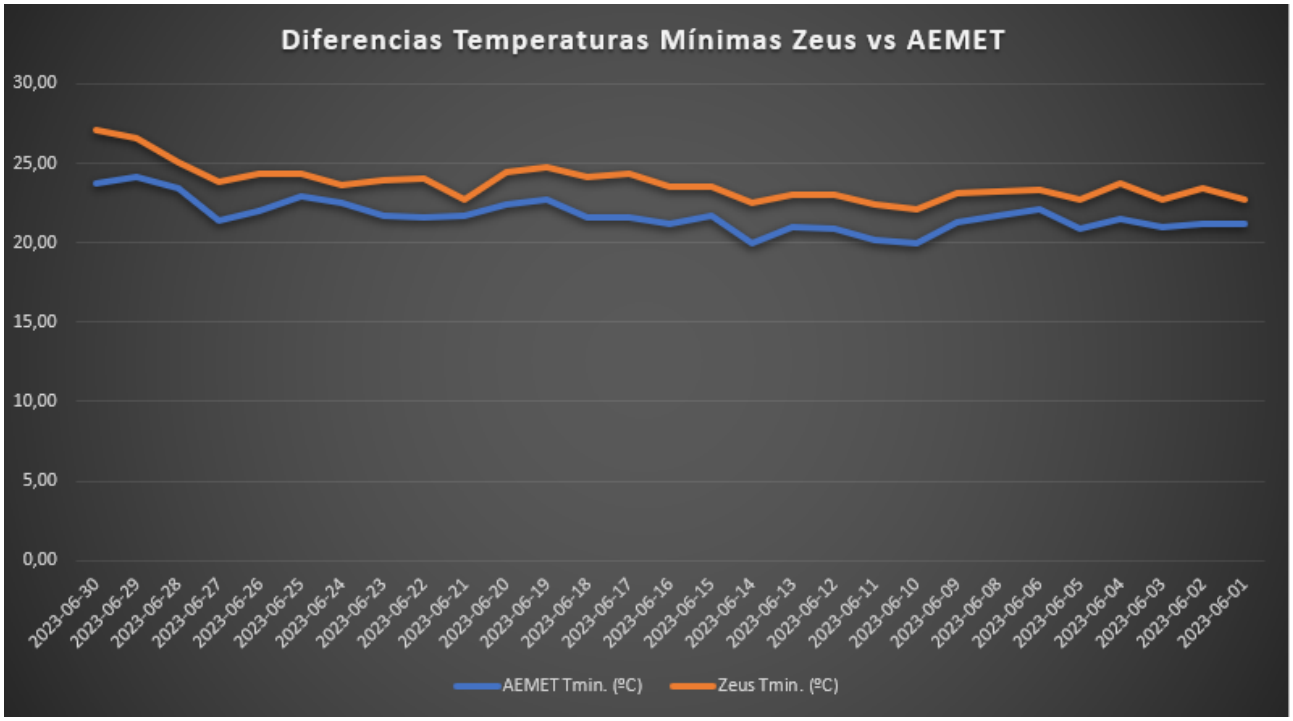


Figura 7.29: Gráfica de las diferencias sobre las Temperaturas Mínimas Zeus vs AEMET

7.7.2. Análisis de las diferencias: Temperatura Máxima

La diferencia media de la temperatura máxima es de +3.60ºC (Z Tmax-A Tmax).

De manera análoga a la variable anterior, si procedemos a comparar sobre una gráfica los valores de Zeus y de la AEMET podemos apreciar que, también en este caso, la diferencia entre estos valores se mantiene estable.

En la figura 7.30 podemos analizarlo con más detalle.

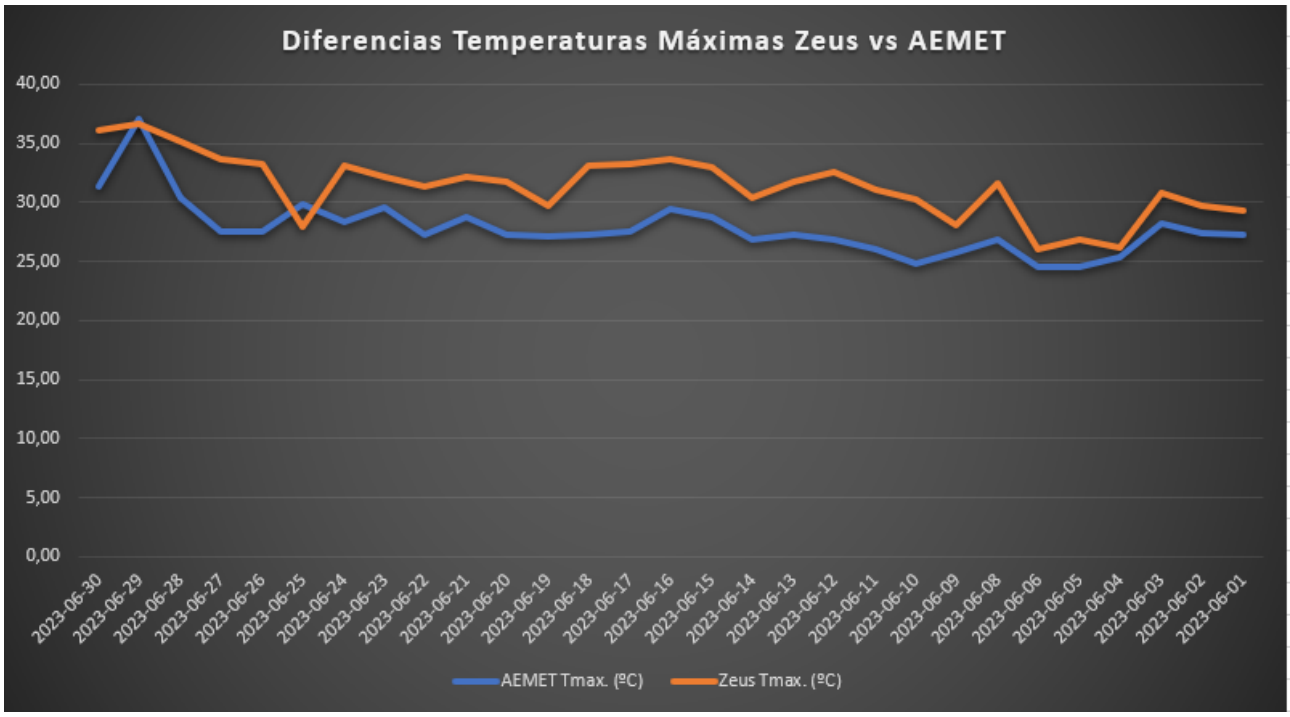


Figura 7.30: Gráfica de las diferencias sobre las Temperaturas Máximas Zeus vs AEMET

7.7.3. Análisis de las diferencias: Presión Mínima

En el caso de la presiones en general las diferencias entre Zeus y Aemet se disminuyen drásticamente.

En este caso, la diferencia media de la presión mínima es de -1.12 hPa ó mb.

Graficando nuevamente ambos conjuntos de datos obtenemos la gráfica mostrada en la figura figura 7.31.

7.7.4. Análisis de las diferencias: Presión Máxima

Si estudiamos ahora la diferencia media de la presión máxima concluimos que es similar la de la presión mínima, en este caso es de -1.05 hPa o mb.

El comportamiento de la gráfica también es similar tal y como puede apreciarse en la figura 7.32.

7.7.5. Conclusiones

Destacamos que en general las temperaturas medidas por Zeus son ligeramente mas altas que las medidas por la AEMET, en cambio las presiones medidas por Zeus son más bajas que las medidas por la AEMET.

Este hecho podría explicarse mediante los siguientes argumentos:

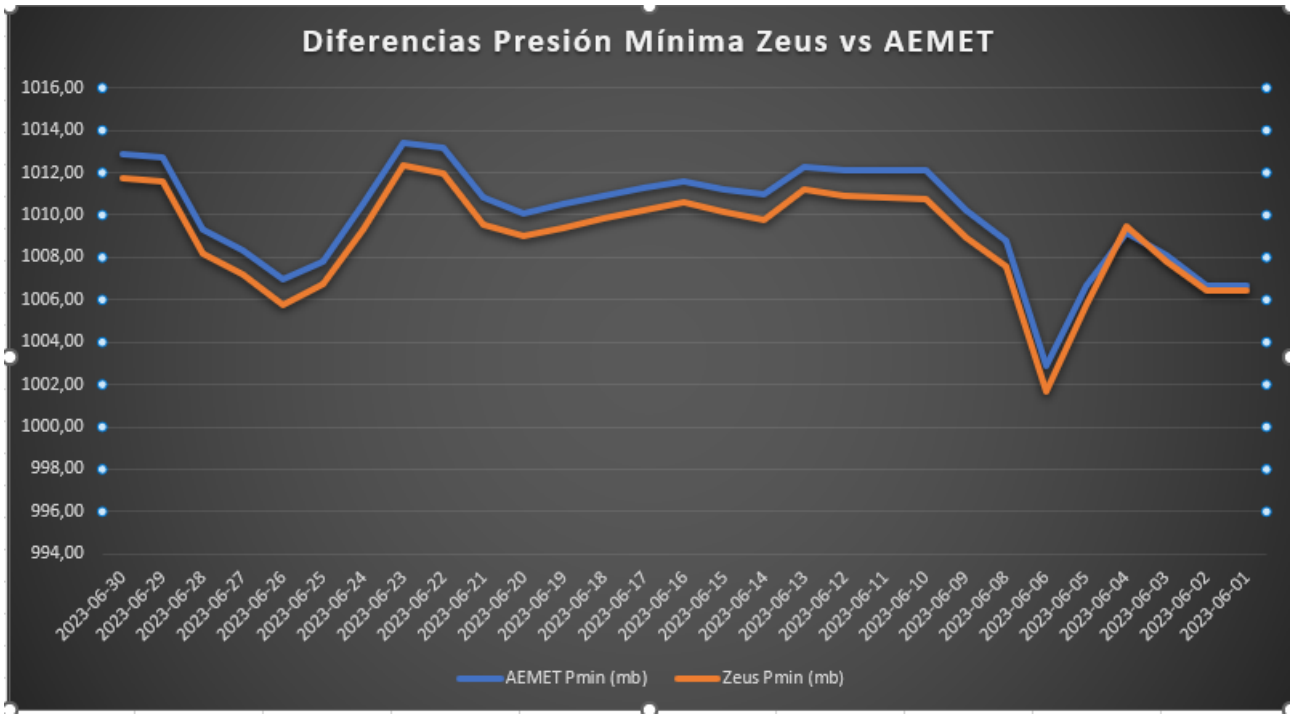


Figura 7.31: Gráfica de las diferencias sobre las Presiones Mínimas Zeus vs AEMET

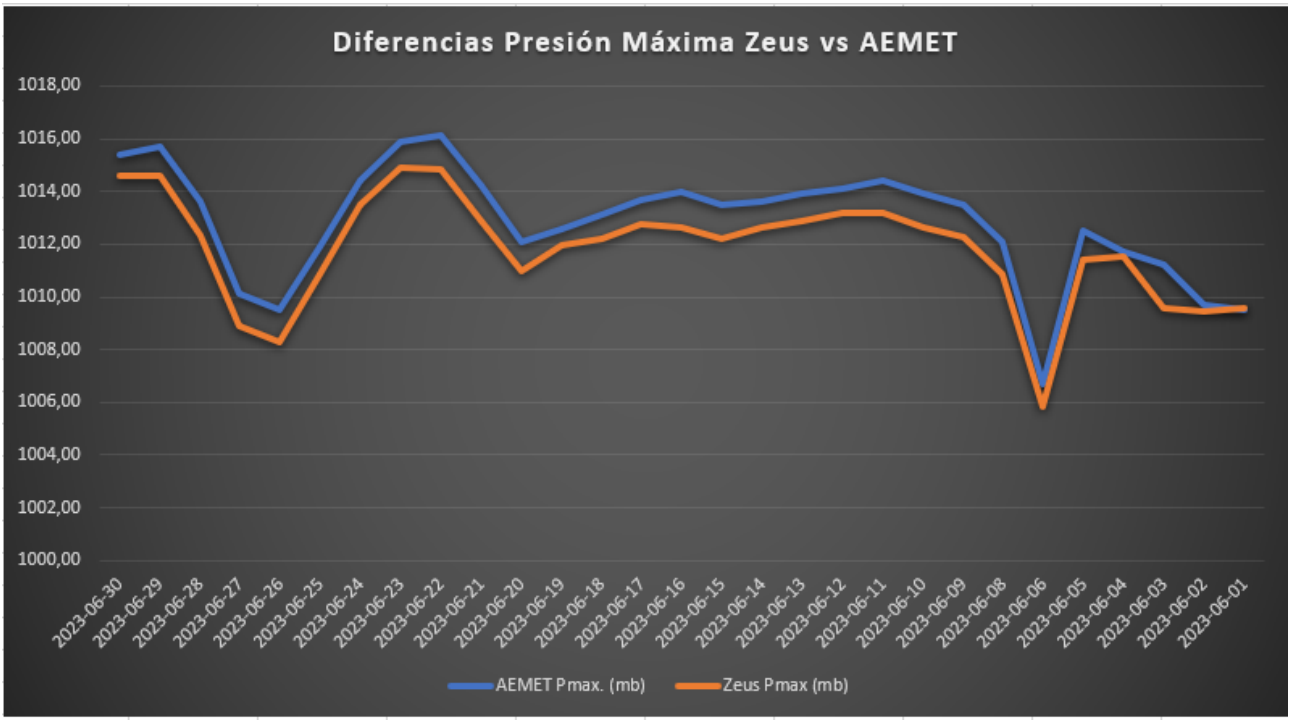


Figura 7.32: Gráfica de las diferencias sobre las Presiones Máximas Zeus vs AEMET

- Temperatura: el sensor de Zeus se encuentra ubicado en el patio interior de una vivienda con un patio de una vivienda colindante en cuyo bajo hay un negocio de restauración. Es lógico pensar que se generen flujos de aire caliente provenientes de los sistemas de aire acondicionado. A esto hay que sumarle que la estación de la AEMET se encuentra en un recinto abierto sin elementos perturbadores a su alrededor.

- Presión: la estación de la AEMET se encuentra, según la documentación oficial de la propia AEMET, a unos 25 metros sobre el nivel del mar mientras que la ubicación del sensor Zeus está a 65 metros sobre el nivel del mar. Teniendo en cuenta que la presión disminuye con la altura es lógico pensar que la diferencia encontrada pueda deberse a la diferencia de altitud entre ambos puntos de referencia.

Si bien las temperaturas obtenidas por Zeus superan en aproximadamente 2 ºC[2] (las mínimas, 3.6ºC las máximas) a los de la AEMET, esta diferencia se mantiene muy constante a lo largo de los días. Si en un futuro disponemos de un conjunto mayor de datos, podríamos volver a calcular los promedios de las diferencias y así poder determinar un factor de corrección (Ct) que se aplicaría a los valores de los sensores Zeus, aproximando así sus medidas a las de la AEMET.

En el estudio realizado con las diferencias de las presiones se observa algo similar, aunque en este caso el valor diferencial medio es de aproximadamente 0.10 hPa[3] o mb[4], realmente casi un uno por mil. Para un valor de presión media de 1010 mb, implicaría una medida de Zeus comprendida en el intervalo [1009 mb,1011 mb]. Realmente es una diferencia inapreciable y que perfectamente es provocada por la diferencia de la altitud sobre el nivel del mar de las estaciones objeto del estudio.

El análisis anterior nos lleva a pensar que se puede estimar un factor de corrección para las medidas obtenidas por el sensor Zeus. Para ello, se ha realizado un estudio con las mediciones obtenidas entre los días 1 y 17 del mes de Junio, obteniendo unos factores de corrección que se aplicarán a las medidas obtenidas por Zeus a partir del día 18 de Junio en adelante.

Para cada una de las medidas del estudio se muestra la hoja de calculo con los datos registrados, los valores estimados aplicados el factor de corrección correspondiente y el margen de error antes y después de aplicar dicho factor.

7.7.5.1. Estimación del factor de corrección para la temperatura mínima obtenida por Zeus

En la figura 7.33 se puede ver el estudio del factor de corrección para la temperatura mínima, observándose que el margen de error de la medida de Zeus una vez aplicado el factor de corrección (-1.98ºC) ha pasado del 9.29 % al 0.47 %.

7.7.5.2. Estimación del factor de corrección para la temperatura máxima obtenida por Zeus

Del mismo modo, en la figura 7.34 procedemos a estimar el factor de corrección para la temperatura máxima, reduciendo el margen de error del 12.78 % al 0.44 %.

[2]Grados Celsius https://es.wikipedia.org/wiki/Grado_Celsius
[3]HectoPascal https://www.convertworld.com/es/presion/hectopascal.html
[4]Milibar https://es.wikipedia.org/wiki/Milibar

Fecha	Id Sensor	Indicativo	AEMET Tmin. (ºC)	Zeus Tmin. (ºC)	d Tmin	d%Tmin	Est. Tmin (ºC)	Est. d Tmin	Est. d%Tmin	
2023-06-17	40:22:D8:72:98:1C	C449C	21,60	24,30	2,70	12,50%				
2023-06-16	40:22:D8:72:98:1C	C449C	21,20	23,50	2,30	10,85%				
2023-06-15	40:22:D8:72:98:1C	C449C	21,70	23,50	1,80	8,29%				
2023-06-14	40:22:D8:72:98:1C	C449C	20,00	22,50	2,50	12,50%				
2023-06-13	40:22:D8:72:98:1C	C449C	21,00	23,00	2,00	9,52%				
2023-06-12	40:22:D8:72:98:1C	C449C	20,90	23,00	2,10	10,05%				
2023-06-11	40:22:D8:72:98:1C	C449C	20,20	22,40	2,20	10,89%				
2023-06-10	40:22:D8:72:98:1C	C449C	20,00	22,10	2,10	10,50%				
2023-06-09	40:22:D8:72:98:1C	C449C	21,30	23,10	1,80	8,45%				
2023-06-08	40:22:D8:72:98:1C	C449C	21,70	23,20	1,50	6,91%				
2023-06-06	40:22:D8:72:98:1C	C449C	22,10	23,30	1,20	5,43%				
2023-06-05	40:22:D8:72:98:1C	C449C	20,90	22,70	1,80	8,61%				
2023-06-04	40:22:D8:72:98:1C	C449C	21,50	23,70	2,20	10,23%				
2023-06-03	40:22:D8:72:98:1C	C449C	21,00	22,70	1,70	8,10%				
2023-06-02	40:22:D8:72:98:1C	C449C	21,20	23,40	2,20	10,38%				
2023-06-01	40:22:D8:72:98:1C	C449C	21,20	22,70	1,50	7,08%				
	Medias:		21,09	23,07	1,98	9,39%				
18/06/2023		C449C	21,60	24,10	2,50	11,57%	22,13	0,52	2,43%	
19/06/2023		C449C	22,70	24,80	2,10	9,25%	22,83	0,13	0,55%	1,49%
20/06/2023		C449C	22,40	24,50	2,10	9,38%	22,53	0,13	0,56%	1,18%
21/06/2023		C449C	21,70	22,70	1,00	4,61%	20,73	-0,98	-4,49%	-0,24%
22/06/2023		C449C	21,60	24,00	2,40	11,11%	22,03	0,42	1,97%	0,20%
23/06/2023		C449C	21,70	23,90	2,20	10,14%	21,93	0,22	1,04%	0,34%
24/06/2023		C449C	22,50	23,60	1,10	4,89%	21,63	-0,88	-3,89%	-0,26%
25/06/2023			22,90	24,30	1,40	6,11%	22,33	-0,57	-2,51%	-0,54%
26/06/2023			22,00	24,30	2,30	10,45%	22,33	0,32	1,48%	-0,32%
27/06/2023			21,40	23,80	2,40	11,21%	21,83	0,43	1,99%	-0,09%
28/06/2023			23,40	25,10	1,70	7,26%	23,13	-0,27	-1,18%	-0,19%
29/06/2023			24,10	26,60	2,50	10,37%	24,63	0,52	2,18%	0,01%
30/06/2023			23,70	27,10	3,40	14,35%	25,13	1,43	6,01%	0,47%
	Factor Correctivo	1,98			%Error:	9,29%		Nuevo % Error:	0,47%	

Figura 7.33: Hoja de cálculo con las conclusiones para la Temperatura mínima

Fecha	Id Sensor	Indicativo	AEMET Tmax. (ºC)	Zeus Tmax. (ºC)	d Tmax	d% Tmax	Est. Tmax (ºC)	Est. d Tmax	Est. d%Tmax	
2023-06-17	40:22:D8:72:98:1C	C449C	27,50	33,30	5,80	21,09%				
2023-06-16	40:22:D8:72:98:1C	C449C	29,40	33,70	4,30	14,63%				
2023-06-15	40:22:D8:72:98:1C	C449C	28,70	33,00	4,30	14,98%				
2023-06-14	40:22:D8:72:98:1C	C449C	26,90	30,40	3,50	13,01%				
2023-06-13	40:22:D8:72:98:1C	C449C	27,30	31,70	4,40	16,12%				
2023-06-12	40:22:D8:72:98:1C	C449C	26,80	32,60	5,80	21,64%				
2023-06-11	40:22:D8:72:98:1C	C449C	26,10	31,10	5,00	19,16%				
2023-06-10	40:22:D8:72:98:1C	C449C	24,80	30,30	5,50	22,18%				
2023-06-09	40:22:D8:72:98:1C	C449C	25,80	28,10	2,30	8,91%				
2023-06-08	40:22:D8:72:98:1C	C449C	26,80	31,60	4,80	17,91%				
2023-06-06	40:22:D8:72:98:1C	C449C	24,50	26,00	1,50	6,12%				
2023-06-05	40:22:D8:72:98:1C	C449C	24,60	26,80	2,20	8,94%				
2023-06-04	40:22:D8:72:98:1C	C449C	25,40	26,20	0,80	3,15%				
2023-06-03	40:22:D8:72:98:1C	C449C	28,20	30,80	2,60	9,22%				
2023-06-02	40:22:D8:72:98:1C	C449C	27,40	29,70	2,30	8,39%				
2023-06-01	40:22:D8:72:98:1C	C449C	27,20	29,30	2,10	7,72%				
	Medias:		26,71	30,29	3,58	13,32%				
18/06/2023		C449C	27,20	33,10	5,90	21,69%	29,53	2,33	8,55%	
19/06/2023		C449C	27,10	29,70	2,60	9,59%	26,13	-0,98	-3,60%	2,48%
20/06/2023		C449C	27,20	31,70	4,50	16,54%	28,13	0,93	3,40%	2,78%
21/06/2023		C449C	28,70	32,20	3,50	12,20%	28,63	-0,07	-0,26%	2,02%
22/06/2023		C449C	27,20	31,40	4,20	15,44%	27,83	0,63	2,30%	2,08%
23/06/2023		C449C	29,60	32,10	2,50	8,45%	28,53	-1,08	-3,63%	1,13%
24/06/2023		C449C	28,40	33,10	4,70	16,55%	29,53	1,13	3,96%	1,53%
25/06/2023			29,80	28,00	-1,80	-6,04%	24,43	-5,38	-18,04%	-0,92%
26/06/2023			27,60	33,30	5,70	20,65%	29,73	2,13	7,70%	0,04%
27/06/2023			27,60	33,60	6,00	21,74%	30,03	2,43	8,79%	0,92%
28/06/2023			30,40	35,10	4,70	15,46%	31,53	1,13	3,70%	1,17%
29/06/2023			37,00	36,60	-0,40	-1,08%	33,03	-3,98	-10,74%	0,18%
30/06/2023			31,40	36,10	4,70	14,97%	32,53	1,13	3,58%	0,44%
	Factor Correctivo	3,58			%Error:	12,78%		Nuevo % Error:	0,44%	

Figura 7.34: Hoja de cálculo con las conclusiones para la Temperatura máxima

7.7.5.3. Estimación del factor de corrección para la presión mínima obtenida por Zeus

En la figura 7.35 se observa que al aplicar el mismo método, el margen de error de la medida de Zeus una vez aplicado el factor de corrección (-0.91hPa) ha pasado del -0.11 % al -0.02 %,

prácticamente anulando el margen de error.

	A	B	C	D	E	F	G	H	I	J	K
1	Fecha	Id Sensor	Indicativo	AEMET Pmin (mb)	Zeus Pmin (mb)	d Pmin	d% Pmin	Est. Pmin (ºC)	Est. d Pmin	Est. d%Pmin	
2	2023-06-17	40:22:D8:72:98:1C	C449C	1011,30	1010,22	-1,08	-0,11%				
3	2023-06-16	40:22:D8:72:98:1C	C449C	1011,60	1010,57	-1,03	-0,10%				
4	2023-06-15	40:22:D8:72:98:1C	C449C	1011,20	1010,13	-1,07	-0,11%				
5	2023-06-14	40:22:D8:72:98:1C	C449C	1011,00	1009,77	-1,23	-0,12%				
6	2023-06-13	40:22:D8:72:98:1C	C449C	1012,30	1011,18	-1,12	-0,11%				
7	2023-06-12	40:22:D8:72:98:1C	C449C	1012,10	1010,94	-1,16	-0,11%				
8	2023-06-11	40:22:D8:72:98:1C	C449C	1012,10	1010,80	-1,30	-0,13%				
9	2023-06-10	40:22:D8:72:98:1C	C449C	1012,10	1010,74	-1,36	-0,13%				
10	2023-06-09	40:22:D8:72:98:1C	C449C	1010,20	1008,95	-1,25	-0,12%				
11	2023-06-08	40:22:D8:72:98:1C	C449C	1008,80	1007,57	-1,23	-0,12%				
12	2023-06-06	40:22:D8:72:98:1C	C449C	1002,90	1001,66	-1,24	-0,12%				
13	2023-06-05	40:22:D8:72:98:1C	C449C	1006,70	1005,77	-0,93	-0,09%				
14	2023-06-04	40:22:D8:72:98:1C	C449C	1009,20	1009,45	0,25	0,02%				
15	2023-06-03	40:22:D8:72:98:1C	C449C	1008,10	1007,82	-0,28	-0,03%				
16	2023-06-02	40:22:D8:72:98:1C	C449C	1006,70	1006,46	-0,24	-0,02%				
17	2023-06-01	40:22:D8:72:98:1C	C449C	1006,70	1006,45	-0,25	-0,02%				
18		Medias:		1009,56	1008,66	-0,91	-0,09%				
19											
20	18/06/2023		C449C	1010,90	1009,87	-1,03	-0,10%	1010,78	-0,12	-0,01%	
21	19/06/2023		C449C	1010,50	1009,36	-1,14	-0,11%	1010,27	-0,23	-0,02%	-0,02%
22	20/06/2023		C449C	1010,10	1008,98	-1,12	-0,11%	1009,89	-0,21	-0,02%	-0,02%
23	21/06/2023		C449C	1010,80	1009,55	-1,25	-0,12%	1010,46	-0,34	-0,03%	-0,02%
24	22/06/2023		C449C	1013,20	1011,97	-1,23	-0,12%	1012,88	-0,32	-0,03%	-0,02%
25	23/06/2023		C449C	1013,40	1012,36	-1,04	-0,10%	1013,27	-0,13	-0,01%	-0,02%
26	24/06/2023		C449C	1010,50	1009,34	-1,16	-0,11%	1010,25	-0,25	-0,02%	-0,02%
27	25/06/2023			1007,80	1006,73	-1,07	-0,11%	1007,64	-0,16	-0,02%	-0,02%
28	26/06/2023			1007,00	1005,77	-1,23	-0,12%	1006,68	-0,32	-0,03%	-0,02%
29	27/06/2023			1008,30	1007,19	-1,11	-0,11%	1008,10	-0,20	-0,02%	-0,02%
30	28/06/2023			1009,30	1008,19	-1,11	-0,11%	1009,10	-0,20	-0,02%	-0,02%
31	29/06/2023			1012,70	1011,58	-1,12	-0,11%	1012,49	-0,21	-0,02%	-0,02%
32	30/06/2023			1012,90	1011,75	-1,15	-0,11%	1012,66	-0,24	-0,02%	-0,02%
33		Factor Correctivo	-0,91			%Error:	-0,11%		Nuevo % Error:	-0,02%	
34											

Figura 7.35: Hoja de cálculo con las conclusiones para la Presión mínima

7.7.5.4. Estimación del factor de corrección para la presión máxima obtenida por Zeus

Por último, en la figura 7.36 se puede ver que el margen de error una vez aplicado el factor de corrección (-0.96hPa) ha pasado del -0.10 % al 0.01 %, prácticamente anulando nuevamente el margen de error.

	A	B	C	D	E	F	G	H	I	J	K
1	Fecha	Id Sensor	Indicativo	AEMET Pmax. (mb)	Zeus Pmax (mb)	d Pmax	d% Pmax	Est. TPmax (ºC)	Est. d Pmax	Est. d%Pmax	
2	2023-06-17	40:22:D8:72:98:1C	C449C	1013,70	1012,78	-0,92	-0,09%				
3	2023-06-16	40:22:D8:72:98:1C	C449C	1014,00	1012,66	-1,34	-0,13%				
4	2023-06-15	40:22:D8:72:98:1C	C449C	1013,50	1012,18	-1,32	-0,13%				
5	2023-06-14	40:22:D8:72:98:1C	C449C	1013,60	1012,65	-0,95	-0,09%				
6	2023-06-13	40:22:D8:72:98:1C	C449C	1013,90	1012,86	-1,04	-0,10%				
7	2023-06-12	40:22:D8:72:98:1C	C449C	1014,10	1013,16	-0,94	-0,09%				
8	2023-06-11	40:22:D8:72:98:1C	C449C	1014,40	1013,16	-1,24	-0,12%				
9	2023-06-10	40:22:D8:72:98:1C	C449C	1013,90	1012,63	-1,27	-0,13%				
10	2023-06-09	40:22:D8:72:98:1C	C449C	1013,50	1012,26	-1,24	-0,12%				
11	2023-06-08	40:22:D8:72:98:1C	C449C	1012,10	1010,86	-1,24	-0,12%				
12	2023-06-06	40:22:D8:72:98:1C	C449C	1006,70	1005,86	-0,84	-0,08%				
13	2023-06-05	40:22:D8:72:98:1C	C449C	1012,50	1011,39	-1,11	-0,11%				
14	2023-06-04	40:22:D8:72:98:1C	C449C	1011,70	1011,51	-0,19	-0,02%				
15	2023-06-03	40:22:D8:72:98:1C	C449C	1011,20	1009,58	-1,62	-0,16%				
16	2023-06-02	40:22:D8:72:98:1C	C449C	1009,70	1009,45	-0,25	-0,02%				
17	2023-06-01	40:22:D8:72:98:1C	C449C	1009,50	1009,58	0,08	0,01%				
18		Medias:		1012,38	1011,41	-0,96	-0,10%				
19											
20	18/06/2023		C449C	1013,10	1012,22	-0,88	-0,09%	1013,18	0,08	0,01%	
21	19/06/2023		C449C	1012,60	1011,97	-0,63	-0,06%	1012,93	0,33	0,03%	0,02%
22	20/06/2023		C449C	1012,10	1011,00	-1,10	-0,11%	1011,96	-0,14	-0,01%	0,01%
23	21/06/2023		C449C	1014,20	1012,89	-1,31	-0,13%	1013,85	-0,35	-0,03%	0,00%
24	22/06/2023		C449C	1016,10	1014,84	-1,26	-0,12%	1015,80	-0,30	-0,03%	-0,01%
25	23/06/2023		C449C	1015,90	1014,88	-1,02	-0,10%	1015,84	-0,06	-0,01%	-0,01%
26	24/06/2023		C449C	1014,40	1013,51	-0,89	-0,09%	1014,47	0,07	0,01%	0,00%
27	25/06/2023			1011,90	1010,86	-1,04	-0,10%	1011,82	-0,08	-0,01%	-0,01%
28	26/06/2023			1009,50	1008,27	-1,23	-0,12%	1009,23	-0,27	-0,03%	-0,01%
29	27/06/2023			1010,10	1008,93	-1,17	-0,12%	1009,89	-0,21	-0,02%	-0,01%
30	28/06/2023			1013,60	1012,36	-1,24	-0,12%	1013,32	-0,28	-0,03%	-0,01%
31	29/06/2023			1015,70	1014,57	-1,13	-0,11%	1015,53	-0,17	-0,02%	-0,01%
32	30/06/2023			1015,40	1014,61	-0,79	-0,08%	1015,57	0,17	0,02%	-0,01%
33		Factor Correctivo	-0,96			%Error:	-0,10%		Nuevo % Error:	-0,01%	
34											

Figura 7.36: Hoja de cálculo con las conclusiones para la Presión máxima

Capítulo 8

Conclusiones

Finalmente hemos llegado a un punto donde, además de haber construido un sensor que recoge datos, también se ha diseñado y construido un ecosistema global. Existe una plataforma (Zeus) donde se pueden consultar y realizar los análisis que se pretendían sobre los datos recopilados, tanto desde la AEMET como desde nuestros sensores.

Se han cumplido los propósitos marcados en el inicio de nuestro proyecto, partiendo de una primera idea muy básica se han ido implementando las diversas funcionalidades recogidas en nuestra propuesta.

Este proyecto se ha dividido en diversos tipos de tareas atendiendo a su naturaleza:

- Montaje del sensor a nivel de hardware (electrónica).
- Programación con lenguaje C sobre placas ESP32.
- Programación de un Servicio VB.net: el proceso importaAemet.
- Programación de un Servicio Web SOAP usando VB.net para el API que será invocado por los sensores.
- Trabajos de Administración y configuración de un Servidor, en este caso un Windows Server 2022.
- Trabajos de implantación de un motor de Base de Datos, en nuestro caso, Microsoft SQL Server.

Por tanto se ha tratado de un proyecto multidisciplinar, donde de una manera orquestada se ha trabajado con diferentes tecnologías: electrónica, desarrollos a nivel de hardware, a nivel de servicios o procesos y por último a nivel de Sistemas operativos.

El aprendizaje adquirido no sólo ha sido amplio sino constante desde el inicio del proyecto. Desde las primeras tareas se ha seguido el siguiente esquema: Análisis previo - Desarrollo - Fase de pruebas. Antes de pasar a la siguiente fase se volvía a testear en paralelo con la anterior, de esta manera se iba comprobando, paso a paso, la totalidad del proyecto.

Superada cada fase únicamente restaba avanzar hacía la siguiente, eso sí, con paso firme. Existe un viejo refrán que dice:

"*Inteligente es aquel que sabe donde quiere ir y más inteligente aún el que sabe donde ya no tiene que volver*"

Es evidente que siempre se presentan inconvenientes y problemas no previstos inicialmente a lo largo de cualquier proceso y éste no podía ser una excepción.

Durante el desarrollo se han detectado errores que, en ocasiones, no eran debidos al código que se acababa de añadir. Por ejemplo el estar trabajando sin éxito con un determinado sensor y después de unos días ver que la ficha técnica del mismo era errónea; o comprobar que no había registros en la base de datos porque no se estaban grabando debido a un desbordamiento de valores. A buen seguro que ante una situación futura similar ya sabremos que revisar y como actuar.

Algún que otro sensor quemado por alimentarlo con mas voltaje del permitido, un display que no mostraba nada y luego descubrir que el potenciómetro que regula el brillo estaba defectuoso, fueron otros hechos un tanto curiosos y, por supuesto, que supusieron la consiguiente perdida de tiempo.

Con todo y con ello, la experiencia adquirida al igual que la satisfacción ha sido enorme.

Queremos destacar el desarrollo de la Plataforma Web Zeus. La programación de este tipo de plataformas era sobre la que teníamos menos experiencia y, como era de suponer, sobre la que nos surgieron más dudas a la hora de afrontarla.

Tal y como se detalla en la parte final del Capítulo de Pruebas, se concluye que las diferencias obtenidas tras los análisis de los datos, son realmente bajas, habiéndose reducido aún más al aplicar los factores de corrección calculados según la metodología indicada.

Por último me gustaría terminar con una cita, de Rudyard Kipling, que siempre conviene tener presente. Considero que son cuestiones básicas muy bien condensadas en una única frase:

"*Seis honrados servidores me enseñaron cuanto sé;*
sus nombres son cómo, cuándo, dónde, qué, quién y por qué."

8.1. Trabajo futuro

Como con cualquier otra tarea cotidiana en nuestro día a día, nos puede ocurrir que, al concluirla, nos quedemos con un extraño sabor de boca. No del todo malo, pero si un tanto peculiar.

Es habitual, quizás muy habitual - por fortuna para nosotros - que a lo largo del camino recorrido nos encontremos con opciones que, a priori, nunca se nos hubieran ocurrido hasta que surgen una vez llegamos a un determinado punto.

Ese punto de incertidumbre, de continuar hacia adelante o de borrar lo escrito y rehacer el planteamiento seguido hasta ese momento, es similar al de tirar una moneda al aire y obrar dependiendo, si la misma cae cara arriba o cara abajo.

Según decía el autor canario Don Benito Pérez Galdós:

"*La experiencia es una llama que no alumbra sino quemando*"

Así pues, y trsd de muchas quemaduras - y no me refiero con el soldador - y siguiendo una una idea previa, se ha alcanzado un resultado. Resultado que puede ser mejor o peor, pero a fin de cuentas es, sin duda, un resultado. ¿Mejorable? Sin duda.

Como propuestas de mejoras y de cara a una versión 2.0 de Zeus, se proponen:

- Cálculo dinámico de los factores de corrección: se podría incluir el cálculo de los factores de corrección de forma dinámica, de manera que los datos obtenidos nos permitan afinar la sensibilidad del sensor.
- Alertas: Funcionalidad que permitiría definir alertas que se dispararían cuando el valor de una determinada variable se encuentre fuera de un umbral preestablecido. Dichas alertar podrían ser a través de correo electrónico o por mensajes SMS.
- AutoWifi: En este primer prototipo del sensor, la configuración de la red Wifi se realiza en el propio código, siendo necesario una nueva compilación de este a la hora de poner en marcha un nuevo sensor. Existen librerías en el mercado, como por ejemplo WifiManager, que permite, configurar la red Wifi de un dispositivo ESP32 desde un móvil sin tener que recompilar el código.
- Sensores adicionales: Tanto el sensor como el Display del sensor Zeus utilizan el protocolo de comunicación serie I^2C, por tanto, podemos seguir añadiendo sensores en serie y así poder tomar otro tipo de medidas (p.ej. humedad del suelo, nivel de CO_2, un segundo sensor de temperatura, etc.)
- Retro iluminación manual del Display: En determinadas condiciones lumínicas la visualización del Display del sensor Zeus no es cómoda, se requeriría un pulsador manual que, a petición del usuario, retroalimentará el Display el tiempo necesario para una correcta lectura.

Bibliografía

[1] F. Charte Ojeda, *Programación con Visual Basic 2005*. Anaya Multimedia, 2005.

[2] F. Llaugé Dausá, *Iniciación a la Meteorología*. Marcombo Boixareu Editores, Barcelona, 1986.

[3] O. Torrente Artero, *Arduino. Curso práctico de formación*. Libros RC, Madrid, 2013.

[4] A. Mata Estorellas, *Turbo C. Iniciación y programación avanzada*. Editorial Paraninfo,S.A., Madrid, 1991.

[5] R. Soukup, *A fondo Microsoft SQL Server*. Microsoft Press, 1997.

[6] V. Ceylan. (2023) Serene. [Online]. Available: https://marketplace.visualstudio.com/items?itemName=VolkanCeylan.SereneSerenityApplicationTemplate

[7] S. Software. (2023) Serenity. [Online]. Available: https://serenity.is/

[8] M. Electrónica. (2023) Mk electrónica. [Online]. Available: https://mkelectronica.com/

Apéndice A

Código fuente firmware Sensor Zeus

En este anexo mostramos el código fuente desarrollado e implementado en nuestra placa principal ESP32-WROOM-D.

```
/***************************************
 * Incluir librerias
 ***************************************/
#include <Wire.h>
#include <WiFi.h>
#include <Adafruit_BMP085.h>
#include <BH1750FVI.h>
#include "SparkFunHTU21D.h"
#include <LiquidCrystal_I2C.h>

/***************************************
 * Definición parámetros WIFI
 ***************************************/
// Datos de nuestra red WiFi
// Por seguridad se ocultan con asteriscos
#define WIFISSID "************"
#define PASSWORD "**********************"

/***************************************
 * Datos del Host (servidor remoto)
 ***************************************/
// URL/IP y Puerto del Servidor remoto
// Por seguridad se oculta con *.*.*.*
const char* host = "*.*.*.*";
const int httpPort = 80;

/***************************************
 * Datos del Servidor ntp
 ***************************************/
#define NTP_SERVER     "pool.ntp.org"
#define UTC_OFFSET     0
#define UTC_OFFSET_DST 0
struct tm timeinfo;

/***************************************
 * Instancias de los Sensosres
 ***************************************/
// Creamos una instancia de Lightsensor.
Adafruit_BMP085 bmp;
```

```
40  //Creamos una instancia del objeto  myHumudity
41 BH1750FVI LightSensor(BH1750FVI::k_DevModeContLowRes);
42 HTU21D myHumidity;
43 //Asignamos dirección I2C al Display
44 LiquidCrystal_I2C lcd(0x27,20,4);

46 /****************************************
47  * Definición de Variables globales
48  ****************************************/
49 char c_temp[150];
50 #define RELAY1 16
51 // Identificador del Sensor
52 String nameSensor = "E1802";
53 String registrerSensor ="";
54 // Variable para almacenar la MAC del Sensor
55 String macSensor="";
56 // Variable para almacenar la fecha/hora actual.
57 //String fechaActual="18/02/2023 18:02";//
58 char fechaActual [20];

60 /****************************************
61  * Definición de Variables globales (Lineas)
62  ****************************************/
63 // Variable para la cadena a enviar en el servicio
64 String Linea0;
65 // Variable que contiene la linea 1 del LCD
66 String Linea1;
67 // Variable que contiene la linea 2 del LCD
68 String Linea2;
69 // Variable que contiene la linea 3 del LCD
70 String Linea3;
71 // Variable que contiene la linea 4 del LCD
72 String Linea4;

74 /****************************************
75  * Funciones principales
76  ****************************************/
77 void setup() {
78     //Ajustamos el puerto RELAY1 como de salida.
79     pinMode(RELAY1, OUTPUT);
80     //Puerto serial a 115200bps
81     Serial.begin(115200);
82     //Inicializamos el Display LCD
83     lcd.init();
84     lcd.clear();
85     lcd.backlight();
86     //Inicializamos la red wifi con los datos nuestra red wifi
87     WiFi.begin(WIFISSID, PASSWORD);
88     // Mensaje de control: conectando a la wifi...
89     Serial.println();
90     Serial.print("Esperando conexion WiFi...");
```

```
91      // Bucle de control para controlar el acceso a la red WiFi
92      while (WiFi.status() != WL_CONNECTED) {
93          Serial.print(".");
94          delay(500);
95      }
96      //Mensajes de Control sobre el acceso a la WiFi
97      Serial.println("");
98      Serial.println("WiFi Connectado");
99      Serial.println("Direccion: ");
100     Serial.println(WiFi.localIP());
101     macSensor=WiFi.macAddress();
102     Serial.println(macSensor);
103     //Control de acceso al sensor
104     if (!bmp.begin()) {
105         Serial.println("No se pudo encontrar un sensor BMP085 válido,
              verifique el cableado!");
106         muestraError("Err02: No sensor",10000);
107         while (1) {}
108     }
109     //Activamos el sensor.
110     LightSensor.begin();
111     myHumidity.begin();
112 }

114 void loop() {
115     // Desactivamos relé para apagar el LCD
116     digitalWrite(RELAY1,LOW);
117     // Obtenemos la fecha/hora desde pool.ntp.org
118     getFechaHora();
119     // Leemos los datos del sensor
120     obtenerMedidas();
121     // Activamos el relé para encender el LCD
122     digitalWrite(RELAY1,HIGH);
123     // Mostramos los datos en el LCD
124     mostrarMedidas();
125     // Envio los datos al Host remoto
126     enviarDatos(Linea0);
127     // Desactivo relé para apagar el LCD
128     digitalWrite(RELAY1,LOW);
129     // Esperamos 15 min para repetir bucle
130     delay(900000);
131     // Limpio el LCD antes de volver a imprimir
132     lcd.clear();
133 }


136 void obtenerMedidas(){
137     /**************************************
138     * Leemos el valor de cada uno de los sensores y lo asignamos a sus
          respectivas variables
139     **************************************/
```

```
140     //String* Datos[5];
141     float temperatura = bmp.readTemperature();
142     float presion = bmp.readPressure();
143     float d_presion = (presion)/100;
144     float altitud = bmp.readAltitude();
145     //uint16_t lux = LightSensor.GetLightIntensity();
146     int lux = LightSensor.GetLightIntensity();
147     float humedad = myHumidity.readHumidity();
148     // Construimos la cadena a devolver.
149     Linea0 =  (nameSensor + "#" + temperatura + "#" + humedad + "#" +
            presion + "#" + lux+ "#" + altitud + "#" + macSensor);
150     // Linea 1 del LCD
151     Linea1=String("ZEUS - Sensor: " + nameSensor);
152     // Linea 2 del LCD
153     Linea2=String(fechaActual);
154     String c_temperatura = String(temperatura,2);
155     String c_humedad = String(humedad,2);
156     // Linea 3 del LCD
157     Linea3=String("Tem"+ String((char)223) + "C  Pre.hPa" + " Hum.%");
158     //Linea3=String ("T=" + c_temperatura + (char)223 + " C" + "H=" +
            c_humedad + "%");
159     // Linea 4 del LCD
160     String c_presion = String(d_presion,2);
161     String c_lux = String(lux);
162     //Linea4=String ("P=" + c_presion + "hPa" + "L=" + c_lux + " L");
163     Linea4=String(c_temperatura + "  " + c_presion + " " + c_humedad);
164 }

166 void mostrarMedidas(){
167     //Mostrar en Display
168     lcd.setCursor(0,0);
169     lcd.print(Linea1);
170     lcd.setCursor(0,1);
171     lcd.print(Linea2);
172     lcd.setCursor(0,2);
173     lcd.print(Linea3);
174     lcd.setCursor(0,3);
175     lcd.print(Linea4);
176 }

178 void conectaWifi(bool pActiva){
179     if (pActiva==true){
180         // Conectamos la WIFI
181         //Inicializamos la red wifi con los datos nuestra red wifi
182         WiFi.begin(WIFISSID, PASSWORD);
183         Serial.println("");
184         // Mensaje de control: conectando a la wifi...
185         Serial.print("Esperando conexion WiFi...");
186         //Comprobacion del acceso a la red Wifi
187         while (WiFi.status() != WL_CONNECTED) {
188           Serial.print(".");
```

```
189           delay(500);
190           Serial.println("");
191           Serial.println("WiFi Connectado");
192           Serial.println("Direccion IP : ");
193           Serial.println(WiFi.localIP());
194           macSensor=WiFi.macAddress();
195           Serial.println("Direccion MAC: ");
196           Serial.println(macSensor);
197           muestraError("Err07: Red Wifi ON ",0);
198         }
199     }else{
200         // Desconectamos la WIFI
201         muestraError("Err08: Red Wifi OFF",10000);
202         WiFi.disconnect();
203     }
204 }

206 void enviarDatos(String pCadena){
207   // Objeto WiFiClient
208   WiFiServer server(80);
209   WiFiClient client = server.available();
210   // Conectar al Host
211   if (!client.connect(host, httpPort)) {
212     Serial.println("Error: No se puede conectar al Host.");
213     muestraError("Err03: Host OFF    ",10000);
214     loop();
215   }
216   if (client) {
217     if (client.connected()) {
218       Serial.println("Sensor conectado al Server:");
219       Serial.println("---------------------------");
220       delay(1000);
221       // Objeto Body de la llamada
222       String body01 = "POST /ZeusWebService/ZeusService.asmx HTTP/1.1\
             r\n";
223       String body02 = "Content-Type: text/xml; charset=utf-8\r\n";
224       String body03 = "Host: HHH\r\n";
225       String body04 = "Content-Length: ZZZ\r\n";
226       String body05 = "\r\n";
227       String body06 = "<soap:Envelope xmlns:soap=\"http://schemas.
             xmlsoap.org/soap/envelope/\" xmlns:xsi=\"http://www.w3.org
             /2001/XMLSchema-instance\" xmlns:xsd=\"http://www.w3.org
             /2001/XMLSchema\">\r\n";
228       String body07 = " <soap:Body>\r\n";
229       String body08 = "   <putMedidasSensores1 xmlns=\"http://www.
             gopitelecom.com\">\r\n";
230       String body09 = "     <pInfo>XXX</pInfo>\r\n";
231       String body10 = "   </putMedidasSensores1>\r\n";
232       String body11 = " </soap:Body>\r\n";
233       String body12 = "</soap:Envelope>\r\n";
```

```
235       String body00 = body01 + body02 + body03 + body04 + body05 +
             body06 + body07 + body08 + body09 + body10 + body11 + body12;
236       body00.replace("HHH", String(host));
237       body00.replace("XXX", pCadena);
238       int size = body00.length();
239       body00.replace("ZZZ", String(size));

241       Serial.println(body00);
242       Serial.println(body00.length());

244       client.println();
245       client.println(body00);
246       client.println();
247       //resultEmail = sendEmail("Test", "ESP32", "Dato enviado con
             EXITO." , "jdaryanan2@gmail.com", false);
248       delay(1000);
249       client.stop();
250       Serial.println("Sensor desconectado del Server.");
251       Serial.println("-------------------------------");
252     }
253     else {
254       Serial.println("Error: Cliente no Conectado.");
255       muestraError("Err04: Server OUT  ",10000);
256     }
257     // close the connection:
258   }
259   else {
260       Serial.println("Error:Cliente no encontrado.");
261       muestraError("Err05: Cliente OUT ",10000);
262     }

264   delay(2000);
265 }

267 void enviarCorreo(String pMensaje) {
268     // Objeto WiFiClient
269   WiFiServer server(80);
270   WiFiClient client = server.available();
271   // Conectar al Host
272   if (!client.connect(host, httpPort)) {
273     Serial.println("Error: No se puede conectar al Host.");
274     loop();
275   }
276   if (client) {
277     if (client.connected()) {
278       Serial.println("Sensor conectado al Server:");
279       Serial.println("---------------------------");
280       delay(1000);
281       // Objeto Body de la llamada
282       String body01 = "POST /ZeusWebService/ZeusService.asmx HTTP/1.1\
             r\n";
```

```
283       String body02 = "Content-Type: text/xml; charset=utf-8\r\n";
284       String body03 = "Host: HHH\r\n";
285       String body04 = "Content-Length: ZZZ\r\n";
286       String body05 = "\r\n";
287       String body06 = "<soap:Envelope xmlns:soap=\"http://schemas.
             xmlsoap.org/soap/envelope/\" xmlns:xsi=\"http://www.w3.org
             /2001/XMLSchema-instance\" xmlns:xsd=\"http://www.w3.org
             /2001/XMLSchema\">\r\n";
288       String body07 = " <soap:Body>\r\n";
289       String body08 = "   <sendEmail xmlns=\"http://www.gopitelecom.
             com\">\r\n";
290       String body09 = "     <pMensaje>XXX</pMensaje>\r\n";
291       String body10 = "   </sendEmail>\r\n";
292       String body11 = " </soap:Body>\r\n";
293       String body12 = "</soap:Envelope>\r\n";

295       String body00 = body01 + body02 + body03 + body04 + body05 +
             body06 + body07 + body08 + body09 + body10 + body11 + body12;
296       body00.replace("HHH", String(host));
297       body00.replace("XXX", pMensaje);
298       int size = body00.length();
299       body00.replace("ZZZ", String(size));

301       Serial.println(body00);
302       Serial.println(body00.length());

304       client.println();
305       client.println(body00);
306       client.println();
307       delay(1000);
308       client.stop();
309       Serial.println("Sensor desconectado del Server.");
310       Serial.println("-------------------------------");
311     }
312    else {
313       Serial.println("Error: Cliente no Conectado.");
314       muestraError("Err04: Server OUT  ",10000);
315     }
316     // close the connection:
317   }
318   else {
319       Serial.println("Error:Cliente no encontrado.");
320       muestraError("Err05: Cliente OUT ",10000);
321     }
322   delay(2000);
323 }


326 void getFechaHora() {
327   configTime(UTC_OFFSET, UTC_OFFSET_DST, NTP_SERVER);
328   if (!getLocalTime(&timeinfo)) {
```

```
329     strftime (fechaActual, 20,  "Err01: pool.ntp.org", 0);
330     muestraError("Err01: pool.ntp.org",10000);
331   }else{
332     strftime (fechaActual, 20,  "  %02d/%02m/%04Y %02H:%02M", &
          timeinfo);
333     Serial.println(String(fechaActual));
334   }
335 }

337 void muestraError(String pMensaje, int pTiempoEspera){
338     digitalWrite(RELAY1,LOW);
339     lcd.setCursor(0,1);
340     lcd.print(pMensaje);
341     delay(pTiempoEspera);
342 }
```

Apéndice B

Código fuente de "importaAEMET"

En este anexo se muestra el código fuente desarrollado con Visual Basic.Net de las funciones principales del proceso que importa los datos de la AEMET y los almacena en nuestra base de datos. Dicho proceso se denomina ImportaAemet.

Dichas funciones son:

- getEstaciones: Recupera las estaciones de la AEMET

```
1  Public Function getEstaciones() As Estaciones
2      Try
3          Dim client = New RestClient("https://opendata.aemet.es/
               opendata/api/valores/climatologicos/
               inventarioestaciones/todasestaciones")
4          client.Timeout = -1
5          Dim request = New RestRequest(Method.[GET])
6          request.AddHeader("Accept", "application/json")
7          request.AddHeader("api_key", recuperaConfiguracionZeus("
               cfg_apiKeyAemet"))
8          Dim response As IRestResponse = client.Execute(request)
9          Dim resAemet As responseAemet = JsonConvert.
               DeserializeObject(Of responseAemet)(response.Content)
10         Dim res As New Estaciones
11         If resAemet.descripcion = "exito" And resAemet.estado =
               200 Then
12             Dim webcliente As New System.Net.WebClient
13             Dim json As String = webcliente.DownloadString(
                   resAemet.datos)
14             'json = Replace(json, " ", "")
15             res.Estacion = JsonConvert.DeserializeObject(Of
                   Estacion())(json)
16             res.Resultado = "OK"
17             ' Enviar correo si éxito
18             If recuperaConfiguracionZeus("cfg_SendEmailForSuccess
                   ") = "true" Then
19                 EnviarCorreo(recuperaConfiguracionZeus("
                       cfg_EmailForSuccessReport"), "[Zeus]", "
                       Success: método getEstaciones():    " + " Res:
                        " & True.ToString, "")
20             End If
21             Return Res
22         Else
23         ' Enviar correo si éxito
```

```
24          If recuperaConfiguracionZeus("cfg_SendEmailForSuccess") =
                "True" Then
25                  EnviarCorreo(recuperaConfiguracionZeus("
                       cfg_EmailForSuccessReport"), "[Zeus]", "
                       Success: método getEstaciones():  " + " Res: "
                        & True.ToString, "")
26              End If
27          '
28          Res.Resultado = "Error: getEstaciones().Estado :  " &
               resAemet.estado & " - Descripción: " & resAemet.
               descripcion
29          Return Res
30          End If
31      Catch ex As Exception
32          If recuperaConfiguracionZeus("cfg_SendEmailForError") = "
               true" Then
33              EnviarCorreo(recuperaConfiguracionZeus("
                   cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                   todo getEstaciones():  " + Err.Description + " Res
                   : " & ex.Message, "")
34          End If
35          Return Nothing
36      End Try
37  End Function
```

- getDatosHorariosFromAemet: Recupera los Registros Horarios desde la AEMET

```
1  Public Function getDatosHorariosFromAemet(pIdEstacion As String)
       As DatosHorariosAemet
2       Dim pCad As String = ""
3       'pCad = CDate(pFechaIni) & "," & CDate(pFechaFin) & "," &
           pIdEstacion
4       Try
5           'Dim f_Ini As String = pFechaIni.Year.ToString("0000") +
               "-" + pFechaIni.Month.ToString("00") + "-" + pFechaIni
               .Day.ToString("00") + "T00:00:00UTC"
6           'Dim f_Fin As String = pFechaFin.Year.ToString("0000") +
               "-" + pFechaFin.Month.ToString("00") + "-" + pFechaFin
               .Day.ToString("00") + "T00:00:00UTC"

8           Dim client = New RestClient("https://opendata.aemet.es/
               opendata/api/observacion/convencional/datos/estacion/"
                & pIdEstacion)
9           client.Timeout = -1
10          Dim request = New RestRequest(Method.GET)
11          request.AddHeader("Accept", "application/json")
12          request.AddHeader("api_key", recuperaConfiguracionZeus("
               cfg_apiKeyAemet"))
13          Dim response As IRestResponse = client.Execute(request)

15          Dim resAemet As responseAemet = JsonConvert.
```

```
            DeserializeObject(Of responseAemet)(response.Content)
16          Dim res As New DatosHorariosAemet
17          If resAemet.descripcion = "exito" And resAemet.estado =
               200 Then
18              Dim webcliente As New System.Net.WebClient
19              Dim json As String = webcliente.DownloadString(
                   resAemet.datos)
20              'json = Replace(json, " ", "")
21              res.Valor = JsonConvert.DeserializeObject(Of Valor())
                   (json)
22              res.Resultado = "OK"
23              If res.Valor IsNot Nothing Then
24                  res.Numero = res.Valor.Count
25              Else
26                  res.Numero = 0
27              End If
28              ' Enviar correo si éxito
29              putAemetLogAPi(pIdEstacion, "
                   getDatosHorariosFromAemet(" + pIdEstacion + ") ->
                   Recuperados: " & res.Numero.ToString & " Registros
                   .", "OK")
30              If recuperaConfiguracionZeus("cfg_SendEmailForSuccess
                   ") = "true" Then
31                  EnviarCorreo(recuperaConfiguracionZeus("
                       cfg_EmailForSuccessReport"), "[Zeus]", "
                       Success: método getDatosHorariosFromAemet(" &
                       pCad & "):  " + res.Numero.ToString + "
                       Registros. - Res: " & True.ToString, "")
32              End If
33              Return res
34          Else
35              putAemetLogAPi(pIdEstacion, "
                   getDatosHorariosFromAemet(" + pIdEstacion + ") ->
                   " & resAemet.descripcion, "NO")
36              ' Enviar correo si éxito
37              If recuperaConfiguracionZeus("cfg_SendEmailForSuccess
                   ") = "true" Then
38                  EnviarCorreo(recuperaConfiguracionZeus("
                       cfg_EmailForSuccessReport"), "[Zeus]", "
                       Success: método getDatosHorariosFromAemet(" &
                       pCad & "):  " + res.Numero.ToString + "
                       Registros. - Res: " & True.ToString, "")
39              End If
40              '
41              res.Resultado = "Error: getDatosHorariosFromAemet().
                   Estado :  " & resAemet.estado & " - Descripción: "
                    & resAemet.descripcion
42              Return res
43          End If
44      Catch ex As Exception
45          putAemetLogAPi(pIdEstacion, "getDatosHorariosFromAemet("
```

```
            + pIdEstacion + ") -> " & Err.Description, "NO")
46          If recuperaConfiguracionZeus("cfg_SendEmailForError")
               Then
47              EnviarCorreo(recuperaConfiguracionZeus("
                   cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                   todo getDatosHorariosFromAemet(" & pCad & "):  " +
                    " Res: " & Err.Description, "")
48          End If
49          Return Nothing
50      End Try
51  End Function
```

- getDatosDiariosFromAemet: Recupera los registros diarios desde la AEMET

```
1   Public Function getDatosDiariosFromAemet(pFechaIni As Date,
        pFechaFin As Date, pIdEstacion As String) As DatosDiariosAemet
2       Dim pCad As String
3       pCad = CDate(pFechaIni) & "," & CDate(pFechaFin) & "," &
           pIdEstacion
4       Dim f_Ini As String = pFechaIni.Year.ToString("0000") + "-" +
            pFechaIni.Month.ToString("00") + "-" + pFechaIni.Day.
           ToString("00") + "T00:00:00UTC"
5       Dim f_Fin As String = pFechaFin.Year.ToString("0000") + "-" +
            pFechaFin.Month.ToString("00") + "-" + pFechaFin.Day.
           ToString("00") + "T00:00:00UTC"
6       Try
7           Dim client = New RestClient("https://opendata.aemet.es/
               opendata/api/valores/climatologicos/diarios/datos/
               fechaini/" & f_Ini & "/fechafin/" & f_Fin & "/estacion
               /" & pIdEstacion)
8           client.Timeout = -1
9           Dim request = New RestRequest(Method.GET)
10          request.AddHeader("Accept", "application/json")
11          request.AddHeader("api_key", recuperaConfiguracionZeus("
               cfg_apiKeyAemet"))
12          Dim response As IRestResponse = client.Execute(request)
13          Dim resAemet As responseAemet = JsonConvert.
               DeserializeObject(Of responseAemet)(response.Content)
14          Dim res As New DatosDiariosAemet
15          If resAemet.descripcion = "exito" And resAemet.estado =
               200 Then
16              Dim webcliente As New System.Net.WebClient
17              Dim json As String = webcliente.DownloadString(
                   resAemet.datos)
18              json = Replace(json, " ", "")
19              res.DatoDiarioAemet = JsonConvert.DeserializeObject(
                   Of DatoDiarioAemet())(json)
20              res.Resultado = "OK"
21              If res.DatoDiarioAemet IsNot Nothing Then
22                  res.Numero = res.DatoDiarioAemet.Count
23              Else
```

```
24                  res.Numero = 0
25              End If
26              ' Enviar correo si éxito
27              putAemetLogAPi(pIdEstacion, "getDatosDiariosFromAemet
                    (" + pIdEstacion + " , " + f_Ini.Substring(0, 10)
                    + " , " + f_Fin.Substring(0, 10) + ") ->
                    Recuperados: " & res.Numero.ToString & " Registros
                    .", "OK")
28              If recuperaConfiguracionZeus("cfg_SendEmailForSuccess
                    ") = "true" Then
29                  EnviarCorreo(recuperaConfiguracionZeus("
                        cfg_EmailForSuccessReport"), "[Zeus]", "
                        Success: método getDatosDiariosFromAemet(" &
                        pCad & "): " + res.Numero.ToString + "
                        Registros. - Res: " & True.ToString, "")
30              End If
31              Return res
32          Else
33              ' Enviar correo si éxito
34              putAemetLogAPi(pIdEstacion, "getDatosDiariosFromAemet
                    (" + pIdEstacion + " , " + f_Ini.Substring(0, 10)
                    + " , " + f_Fin.Substring(0, 10) + ") -> " &
                    resAemet.descripcion, "NO")
35              If recuperaConfiguracionZeus("cfg_SendEmailForSuccess
                    ") = "true" Then
36                  EnviarCorreo(recuperaConfiguracionZeus("
                        cfg_EmailForSuccessReport"), "[Zeus]", "
                        Success: método getDatosDiariosFromAemet(" &
                        pCad & "): " + res.Numero.ToString + "
                        Registros. - Res: " & True.ToString, "")
37              End If
38              '
39              res.Resultado = "Error: getDatos(). Estado :  " &
                    resAemet.estado & " - Descripción: " & resAemet.
                    descripcion
40              Return res
41          End If
42      Catch ex As Exception
43          putAemetLogAPi(pIdEstacion, "getDatosDiariosFromAemet(" +
                pIdEstacion + " , " + f_Ini.Substring(0, 10) + " , "
               + f_Fin.Substring(0, 10) + ") -> " & Err.Description,
               "NO")
44          If recuperaConfiguracionZeus("cfg_SendEmailForError")
                Then
45              EnviarCorreo(recuperaConfiguracionZeus("
                    cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                    todo getDatosDiariosFromAemet(" & pCad & "): " + "
                     Res: " & Err.Description, "")
46          End If
47      Return Nothing
48      End Try
```

```
End Function
```

Además hay una serie de funciones internas y auxiliares que son:

- getCAdenaConexión: Recupera la cadena de conexión a la base de datos:

```
Public Function getCadenaConexion() As String
    Dim cadena As String = "Server=" & recuperaConfiguracionZeus(
       "cfg_SQL_Server") &
                           ";Database=" & recuperaConfiguracionZeus(
                              "cfg_SQL_DataBase") &
                           ";Trusted_Connection=True;"
    Return cadena
End Function
```

- saveDatosDiariosInZeus: Almacena la lista de datos diarios recuperados desde la AEMET en la base de datos.

```
 Public Function saveDatosDiariosInZeus(pDatos As
    DatosDiariosAemet) As Boolean
    Try
        Dim cnn As New SqlConnection(getCadenaConexion)
        Dim cmd As New SqlCommand("SELECT * FROM aemet_datos
           where 1=0", cnn)
        Dim da As New SqlDataAdapter(cmd)
        Dim builder As New SqlCommandBuilder(da)
        Dim ds As New DataSet()
        da.Fill(ds)
        Dim fila As DataRow

        For Each e In pDatos.DatoDiarioAemet
            fila = ds.Tables(0).NewRow()
            fila("indicativo") = e.indicativo
            fila("fecha") = e.fecha
            fila("tmin") = CDbl(e.tmin)
            fila("horatmin") = e.horatmin
            fila("tmax") = CDbl(e.tmax)
            fila("horatmax") = e.horatmax
            fila("tmed") = CDbl(e.tmed)
            fila("presmin") = CDbl(e.presMin)
            fila("horapresmin") = e.horaPresMin
            fila("presmax") = CDbl(e.presMax)
            fila("horapresmax") = e.horaPresMax
            fila("prec") = e.prec
            fila("sol") = CDbl(e.sol)
            fila("velmedia") = CDbl(e.velmedia)
            fila("racha") = CDbl(e.racha)
            fila("dir") = CDbl(e.dir)
```

```
29              fila("horaracha") = e.horaracha
30              fila("ult_actualizacion") = Now()

32              ds.Tables(0).Rows.Add(fila)
33              builder.GetInsertCommand()
34              da.Update(ds)
35          Next

37          da.Dispose()
38          cmd.Dispose()
39          cnn.Close()
40          'Log
41          If recuperaConfiguracionZeus("cfg_SendEmailForSuccess") =
                "true" Then
42              EnviarCorreo(recuperaConfiguracionZeus("
                   cfg_EmailForSuccessReport"), "[Zeus]", "Success: m
                   étodo saveDatosDiariosInZeus(): " + pDatos.
                   DatoDiarioAemet.Count.ToString & " Registros. " &
                   " Res: " & True.ToString, "")
43          End If
44          Return True
45      Catch ex As Exception
46          If recuperaConfiguracionZeus("cfg_SendEmailForError") = "
               true" Then
47              EnviarCorreo(recuperaConfiguracionZeus("
                   cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                   todo saveDatosDiariosInZeus(): " + Err.Description
                    + " Res: " & ex.Message, "")
48          End If
49          Return False
50      End Try
51  End Function
```

- saveDatosUltimos: Almacena en la base de datos la lista de los datos horarios recuperados desde la AEMET.

```
1  Public Function saveDatosUltimos(pValores As DatosHorariosAemet)
       As Boolean
2       Try
3           For Each e In pValores.Valor
4               Dim cnn As New SqlConnection(getCadenaConexion)
5               Dim cmd As New SqlCommand("SELECT * FROM
                    aemet_datos_ultimos where fint='" & e.fint & "'",
                    cnn)
6               Dim da As New SqlDataAdapter(cmd)
7               Dim builder As New SqlCommandBuilder(da)
8               Dim ds As New DataSet()
9               da.Fill(ds)
10              If ds.Tables(0).Rows.Count = 0 Then
11                  Dim fila As DataRow
12                  fila = ds.Tables(0).NewRow()
```

```
13                fila("indicativo") = e.idema
14                fila("fint") = e.fint
15                fila("fecha") = e.fint.ToString.Substring(0, 10)
16                fila("hora") = e.fint.ToString.Substring(11, 8)
17                fila("ta") = e.ta
18                fila("tamin") = e.tamin
19                fila("tamax") = e.tamax
20                fila("pres") = e.pres
21                fila("pres_nmar") = e.pres_nmar
22                fila("hr") = e.hr
23                fila("vmax") = e.vmax
24                fila("dmax") = e.dmax
25                fila("prec") = e.prec
26                fila("inso") = e.inso
27                fila("ult_actualizacion") = Now()
28                ds.Tables(0).Rows.Add(fila)
29                builder.GetInsertCommand()
30                da.Update(ds)
31            End If
32            da.Dispose()
33            cmd.Dispose()
34            cnn.Close()
35        Next
36        If recuperaConfiguracionZeus("cfg_SendEmailForSuccess") =
              "true" Then
37            EnviarCorreo(recuperaConfiguracionZeus("
                  cfg_EmailForSuccessReport"), "[Zeus]", "Success: m
                  étodo saveDatosUltimos(): " + " Res: " & True.
                  ToString, "")
38        End If
39        Return True
40    Catch ex As Exception
41        If recuperaConfiguracionZeus("cfg_SendEmailForError") = "
             true" Then
42          EnviarCorreo(recuperaConfiguracionZeus("
                cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                todo saveDatosUltimos(): " + Err.Description + " Res
                : " & ex.Message, "")
43        End If
44        Return False
45    End Try
46 End Function
```

- getUltimaFechaDescarga: Devuelve la fecha/hora del ultimo registro en la base de datos de una determinada estacion de la AEMET.

```
1 Public Function getUltimaFechaDescarga(pIndicativo As String) As
      Date
2     Dim res As String
3     Dim resf As Date
4     Try
```

```
5           Dim cnn As New SqlConnection(getCadenaConexion)
6           Dim cmd As New SqlCommand("SELECT top(1) * FROM
               aemet_datos where indicativo='" & pIndicativo & "'
               order by fecha desc", cnn)
7           Dim da As New SqlDataAdapter(cmd)
8           Dim builder As New SqlCommandBuilder(da)
9           Dim ds As New DataSet()
10          da.Fill(ds)
11          da.Fill(ds)
12          Dim dt As DataTable = ds.Tables(0)
13          Dim fila As DataRow
14          If dt.Rows.Count = 0 Then
15              'res = Now.Year.ToString("00") + "-" + Now.Month.
                   ToString("00") + "-" + Now.Day.ToString("00")
16              resf = CDate(Now.Year.ToString("0000") + "-01-01")
17          Else
18              For Each fila In dt.Rows
19                  res = fila("fecha")
20                  Dim f As Date = Format(CDate(res), "dd/MM/yyyy")
21                  f = DateAdd(DateInterval.Day, 1, f)
22                  resf = CDate(f)
23              Next
24          End If
25          ' Enviar correo si éxito
26          If recuperaConfiguracionZeus("cfg_SendEmailForSuccess") =
                "true" Then
27              EnviarCorreo(recuperaConfiguracionZeus("
                   cfg_EmailForSuccessReport"), "[Zeus]", "Success: m
                   étodo getUltimaFechaDescarga(): " + " Res: " &
                   True.ToString, "")
28          End If
29          Return resf
30          dt.Dispose()
31          cnn.Close()
32      Catch ex As Exception
33          If recuperaConfiguracionZeus("cfg_SendEmailForError") = "
                true" Then
34              EnviarCorreo(recuperaConfiguracionZeus("
                   cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                   todo getUltimaFechaDescarga(): " + Err.Description
                    + " Res: " & ex.Message, "")
35          End If
36          Return Nothing
37      End Try
38  End Function
```

- getEstacionesPorSincronizar: Devuelve la lista de las estaciones de la AEMET para las que se tiene que recuperar sus registros.

```
1 Public Function getEstacionesPorSincronizar() As List(Of String)
2     Dim res As New List(Of String)
```

```
3     'Try
4     Dim cnn As New SqlConnection(getCadenaConexion)
5     Dim cmd As New SqlCommand("SELECT * FROM aemet_estaciones
          where activo='SI' order by indicativo desc", cnn)
6     Dim da As New SqlDataAdapter(cmd)
7     Dim builder As New SqlCommandBuilder(da)
8     Dim ds As New DataSet()
9     da.Fill(ds)
10    Dim dt As DataTable = ds.Tables(0)
11    Dim fila As DataRow
12    If dt.Rows.Count = 0 Then
13        Return Nothing
14    Else
15        For Each fila In dt.Rows
16            '//res = fila("indicativo")
17            res.Add(Trim(fila("indicativo")))
18        Next
19    End If
20    dt.Dispose()
21    cnn.Close()
22    Return res
23 End Function
```

- putAemetLogAPi: Almacena en la base de datos el resultado de las llamadas a la API de la AEMET.

```
1 Public Function putAemetLogAPi(pIndicativo As String, pLlamada As
      String, pResultado As String) As Boolean
2     Try
3         ' Ahora insertamos
4         Dim cnn As New SqlConnection(getCadenaConexion)
5         Dim cmd As New SqlCommand("SELECT * FROM aemet_log_api
              where 1=0", cnn)
6         Dim da As New SqlDataAdapter(cmd)
7         Dim builder As New SqlCommandBuilder(da)
8         Dim ds As New DataSet()
9         da.Fill(ds)
10        Dim fila As DataRow
11        fila = ds.Tables(0).NewRow()
12        fila("Fecha") = Now
13        fila("Indicativo") = pIndicativo
14        fila("LLamada") = pLlamada
15        fila("Resultado") = pResultado

17        ds.Tables(0).Rows.Add(fila)
18        builder.GetInsertCommand()
19        da.Update(ds)
20        da.Dispose()
21        cmd.Dispose()
22        cnn.Close()
23        'Log
```

```
24          If recuperaConfiguracionZeus("cfg_SendEmailForSuccess") =
                "true" Then
25              EnviarCorreo(recuperaConfiguracionZeus("
                    cfg_EmailForSuccessReport"), "[Zeus]", "Success: m
                    étodo putAemetLogAPi(): " & " Res: " & True.
                    ToString, "")
26          End If
27          Return True
28      Catch ex As Exception
29          If recuperaConfiguracionZeus("cfg_SendEmailForError") = "
                true" Then
30              EnviarCorreo(recuperaConfiguracionZeus("
                    cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                    todo putAemetLogAPi(): " + Err.Description + " Res
                    : " & ex.Message, "")
31          End If
32          Return False
33      End Try
34  End Function
```

- deleteTablaZeus: Borra el contenido de la tabla que se pasa como parametro.

```
1  Public Function deleteTablaZeus(ByVal pTabla As String) As
       Boolean
2       Try
3           Dim Conexion As String = getCadenaConexion()
4           Dim Conn As SqlConnection = New SqlConnection(Conexion)
5           Dim SQL As String = "delete from " & pTabla
6           Conn.Open()
7           Dim Cmd As New SqlCommand(SQL, Conn)
8           Cmd.ExecuteNonQuery()
9           Conn.Close()
10          'Log
11          If recuperaConfiguracionZeus("cfg_SendEmailForSuccess") =
                "true" Then
12              EnviarCorreo(recuperaConfiguracionZeus("
                    cfg_EmailForSuccessReport"), "[Zeus]", "Success: m
                    étodo deleteTablaZeus(): " & " Res: " & True.
                    ToString, "")
13          End If
14          Return True
15      Catch ex As Exception
16          If recuperaConfiguracionZeus("cfg_SendEmailForError") = "
                true" Then
17              EnviarCorreo(recuperaConfiguracionZeus("
                    cfg_EmailForErrorReport"), "[Zeus]", "Success: mé
                    todo deleteTablaZeus(): " + Err.Description + "
                    Res: " & ex.Message, "")
18          End If
19          Return False
20      End Try
```

```
End Function
```

- EnviarCorreo: Envia un correo usando para ello el protocolo SMTPy los valores pasados como parámetros.

```
Public Function EnviarCorreo(pDestino As String, pAsunto As
    String, pMensaje As String, pFichero As String) As Boolean
    Dim pMessage As New System.Net.Mail.MailMessage()
    Dim pSMTP As New System.Net.Mail.SmtpClient
    'CONFIGURACIÓN DEL STMP
    pSMTP.Credentials = New System.Net.NetworkCredential("
       soporte@gopitelecom.es", "********")
    pSMTP.Host = "smtp.********.net"
    pSMTP.Port = 587
    pSMTP.EnableSsl = True
    ' CONFIGURACION DEL MENSAJE
    If pDestino = "" Then pDestino = recuperaConfiguracionZeus(".
       Settings.cfg_EmailForSuccessReport")
    pMessage.[To].Add(pDestino) 'Cuenta de Correo al que se le
       quiere enviar el e-mail
    pMessage.From = New System.Net.Mail.MailAddress("
       soporte@gopitelecom.es", pAsunto, System.Text.Encoding.
       UTF8) 'Quien lo envía
    pMessage.Subject = pAsunto 'Sujeto del e-mail
    pMessage.SubjectEncoding = System.Text.Encoding.UTF8 '
       Codificacion
    pMensaje = pMensaje & vbCrLf & vbCrLf & "Message Date: " &
       Now.ToString & vbCrLf & "Versión: " & VersionAplicacion()
    pMessage.Body = pMensaje  'contenido del mail
    pMessage.BodyEncoding = System.Text.Encoding.UTF8
    pMessage.Priority = System.Net.Mail.MailPriority.Normal
    pMessage.IsBodyHtml = False
    If pFichero <> "" Then
        pMessage.Attachments.Add(New Attachment(pFichero))
    End If
    'ENVIO
    Try
        pSMTP.Send(pMessage)
        'MessageBox.Show("Mensaje enviado correctamene", "Exito
           !", MessageBoxButtons.OK)
        Return True
    Catch ex As System.Net.Mail.SmtpException
        'MessageBox.Show(ex.ToString, "Error!", MessageBoxButtons
           .OK)
        Return False
    End Try
End Function
```

Apéndice C

Código fuente del servicio web "ZeusServices.asmx"

En este anexo se muestra el código fuente desarrollado con Visual Basic.Net de las funciones principales del servicio web ZeusService.asmx que será invocado desde los sensores Zeus.

Los métodos principales de este servicio son:

- VerVersión: Se devuelve la versión del servicio web publicado.

```
Public Function VerVersion() As String
    Dim pFichero As String = Server.MapPath(".") & "\bin\app_code
       .dll"
    Try
        Return ("Ver. " + FileSystem.FileDateTime(pFichero).
           ToString)
    Catch ex As Exception
        Return "Ver. Developer"
    End Try
End Function
```

- putMedidasSensores1: Método que recoge la cadena enviada por el sensor, los extrae y los almacena en la base de datos.

```
Public Function putMedidasSensores1(pInfo As String) As String
    Dim pCad As String = pInfo.ToString + " " + GetIP()
    Try
        SaveMedidasSensores1(pInfo, GetIP())
        'LOG por correo
        If ConfigurationManager.AppSettings("
           cfg_SendEmailForSuccess") Then
            EnviarCorreo("", "[ZeusService]", "Success: método
               putMedidasSensores: " + pCad + " Res: " & True.
               ToString, "")
        End If

        'getInforme()
        Return getFecha()
    Catch ex As Exception
        'LOG por correo
        If ConfigurationManager.AppSettings("
           cfg_SendEmailForError") Then
```

```
15              EnviarCorreo("", "[ZeusService]", "Error: método
                   putMedidasSensores: " + pCad + " Res: " & Err.
                   Description, "")
16          End If
17          Return ("Err: " + Err.Description)
18      End Try
19  End Function
```

- sendEmail: Función que envia un correo electrónico, usado para el envio de correos de control.

```
1  Public Function sendEmail(pMensaje As String) As Boolean
2  Try
3      ' Enviar el correo
4      EnviarCorreo("", "[ZeusService]", pMensaje, "")
5      'LOG por correo
6      If ConfigurationManager.AppSettings("cfg_SendEmailForSuccess"
           ) Then
7          EnviarCorreo("", "[ZeusService]", "Success: método
               sendEmail: " + pMensaje + " Res: " & True.ToString, ""
               )
8      End If

10     'getInforme()
11     Return True
12 Catch ex As Exception
13     'LOG por correo
14     If ConfigurationManager.AppSettings("cfg_SendEmailForError")
           Then
15         EnviarCorreo("", "[ZeusService]", "Error: método
               sendEmail: " + pMensaje + " Res: " & Err.Description,
               "")
16     End If
17     Return False
18 End Try
```

- getMedidasSensores: Recupera los registros almacenados en la base de datos.

```
1  Public Function getMedidasSensores(idSensor As String) As String
2      Try
3          Dim res As String
4          res = LoadMedidasSensores1(idSensor)
5          '
6          If ConfigurationManager.AppSettings("
               cfg_SendEmailForSuccess") Then
7              EnviarCorreo("", "[ZeusService]", "Success: método
                   getMedidasSensores: " + idSensor + " Res: " & True
                   .ToString, "")
8          End If
9          '
```

```
10          Return res
11      Catch ex As Exception
12          If ConfigurationManager.AppSettings("
                cfg_SendEmailForError") Then
13              EnviarCorreo("", "[ZeusService]", "Error: método
                    getMedidasSensores: " + idSensor + " Res: " & ex.
                    Message, "")
14          End If
15          Return Nothing
16      End Try
17  End Function
```

Además hay una serie de funciones internas y auxiliares que son:

- GetCadenaConexion: recupera la cadena de conexión a la base de datos.

```
1  Public Shared Function GetCadenaConexion() As String
2      'Devuelve la cadena de conexión a la base de datos.
3      Dim cadena As String = "Server=" & ConfigurationManager.
           AppSettings("cfg_ServerZeus") & ";" &
4                              "Database=" & ConfigurationManager.
                                   AppSettings("cfg_DataBaseZeus") &
                                   ";" &
5                              "User Id=" & ConfigurationManager.
                                   AppSettings("cfg_DataBaseUser") &
                                   ";" &
6                              "Password=" & ConfigurationManager.
                                   AppSettings("cfg_DataBasePass") &
                                   ";"
7      Return (cadena)
8  End Function
```

- SaveMedidasSensores1: Se almacena en la base de datos los valores pasados como parámetros.

```
1  Public Shared Function SaveMedidasSensores1(pInfo As String, pIp
       As String) As String
2      'pInfo = "E1802#20.10#52.59#1010.17#213#25.93#40:22:D8
           :72:98:1C"
3      'pInfo = "E1802#23.20#58.89#101488.00#196#-13.98#40:22:D8
           :72:98:1C"
4      Dim pCad As String = pInfo.ToString + " " + pIp
5      Dim pFecha As DateTime = Now()
6      Dim Datos() As String
7      Dim res As Boolean
8      Try
9          ' Split de los datos
10         Datos = pInfo.Split("#")
```

```
11          Dim cnn As New SqlConnection(GetCadenaConexion)
12          Dim cmd As New SqlCommand("SELECT * FROM
               Zeus_Sensores_Registros where 1=0", cnn)
13          Dim da As New SqlDataAdapter(cmd)
14          Dim builder As New SqlCommandBuilder(da)
15          Dim ds As New DataSet()
16          da.Fill(ds)
17          Dim fila As DataRow
18          fila = ds.Tables(0).NewRow()
19          fila("fecha") = pFecha
20          fila("idSensor") = Datos(6).ToString
21          fila("temperatura") = CDbl(Datos(1)) / 100  '
               pTemperatura
22          fila("presion") = CDbl(Datos(3)) / 100 / 100 'pPresion
23          fila("humedad") = CDbl(Datos(2)) / 100 ' pHumedad
24          fila("luminosidad") = CDbl(Datos(4)) 'pLuminosidad
25          fila("ipOrigen") = pIp
26          ds.Tables(0).Rows.Add(fila)
27          builder.GetInsertCommand()
28          da.Update(ds)
29          da.Dispose()
30          cmd.Dispose()
31          cnn.Close()
32          ' Actualiza fecha del ultimo acceso al sensor
33          updateFechaUltimoAccesoSensor(Datos(6).ToString, pFecha)
34          ' Log
35          If ConfigurationManager.AppSettings("
               cfg_SendEmailForSuccess") Then
36              EnviarCorreo("", "[ZeusService]", "Success: método
                   SaveMedidasSensores(): " + pCad + " Res: " & True.
                   ToString, "")
37          End If
38          Return getFecha()
39      Catch ex As Exception
40          If ConfigurationManager.AppSettings("
               cfg_SendEmailForError") Then
41              EnviarCorreo("", "[ZeusService]", "Success: método
                   SaveMedidasSensores(): " + pCad + " Res: " & ex.
                   Message, "")
42          End If
43          Return ("Err: " + ex.Message)
44      End Try
45  End Function
```

- updateFechaUltimoAccesoSensor: Actualiza la fecha del ultimo acceso de un determinado sensor.

```
1  Public Shared Function updateFechaUltimoAccesoSensor(pIdSensor As
       String, pFecha As DateTime) As Boolean
2      Dim pCad As String = pIdSensor + " " + pFecha.ToString
3      Dim res As Boolean = False
```

```
4       Try
5           Dim cnn As New SqlConnection(GetCadenaConexion)
6           Dim cmd As New SqlCommand("SELECT * FROM Zeus_Sensores
               where trim(idSensor)='" & Trim(pIdSensor) & "'", cnn)
7           Dim da As New SqlDataAdapter(cmd)
8           Dim builder As New SqlCommandBuilder(da)
9           Dim ds As New DataSet()
10          da.Fill(ds)
11          Dim fila As DataRow
12          If ds.Tables(0).Rows.Count <> 0 Then
13              fila = ds.Tables(0).Rows(0)
14              fila("ult_acceso") = pFecha
15              builder.GetUpdateCommand()
16              da.Update(ds)
17              da.Dispose()
18              cmd.Dispose()
19              cnn.Close()
20              res = True
21          Else
22              res = False
23          End If
24          ' Log
25          If ConfigurationManager.AppSettings("
               cfg_SendEmailForSuccess") Then
26              EnviarCorreo("", "[ZeusService]", "Success: método
                   updateFechaUltimoAccesoSensor(): " + pCad + " Res:
                    " & True.ToString, "")
27          End If
28          Return res
29      Catch ex As Exception
30          If ConfigurationManager.AppSettings("
               cfg_SendEmailForError") Then
31              EnviarCorreo("", "[ZeusService]", "Error: método
                   updateFechaUltimoAccesoSensor(): " + pCad + " Res:
                    " & ex.Message, "")
32          End If
33          Return False
34      End Try
35  End Function
```

- LoadMedidasSensores1: Recupera las medidas almacenas de un determinado sensor.

```
1  Public Shared Function LoadMedidasSensores1(idSensor As String)
       As String
2       Dim SQL As String
3       Dim pCad As String
4       SQL = "SELECT top 100 * FROM Zeus_Sensores_Registros  order
           by id_reg desc"
5       Dim conn As New System.Data.SqlClient.SqlConnection(
           GetCadenaConexion)
6       Dim ds As New System.Data.DataSet()
```

```
7       Dim adapter As New System.Data.SqlClient.SqlDataAdapter()
8       Try
9           adapter.SelectCommand = New System.Data.SqlClient.
                SqlCommand(SQL, conn)
10          adapter.Fill(ds, "reg")
11          ' Retonamos
12          Dim JSONString As String = ""
13          JSONString = JsonConvert.SerializeObject(ds)

15          'Dim file As System.IO.StreamWriter
16          'file = My.Computer.FileSystem.OpenTextFileWriter("f
                :\201812.json", False)
17          'file.WriteLine(JSONString)
18          'file.Close()

20          pCad = idSensor
21          If ConfigurationManager.AppSettings("
                cfg_SendEmailForSuccess") Then
22              EnviarCorreo("", "[ZeusService]", "Success: método
                    LoadMedidasSensores1: " + pCad + " Res: " &
                    JSONString.ToString, "")
23          End If

25          Return JSONString
26      Catch ex As Exception
27          'MsgBox(Err.Description)
28          pCad = idSensor
29      If ConfigurationManager.AppSettings("cfg_SendEmailForError")
            Then
30          EnviarCorreo("", "[ZeusService]", "Error: método
                LoadMedidasSensores1: " + pCad + " Res: " & ex.Message
                , "")
31      End If

33      Return Nothing
34      End Try


37  End Function
```

Apéndice D

Modelo de la base de datos Zeus

En este anexo se muestra la estructura detallada de la Base de Datos 'Zeus", que está compuesta de las siguientes tablas:

- "aemet-datos": Donde se almacenan los datos diarios obtenidos usando el API de la AEMET, su estructura es la siguiente:
  - id-reg bigint IDENTITY(1,1) NOT NULL,
  - indicativo nchar(5) NULL,
  - fecha nchar(10) NULL,
  - tmin float NULL,
  - horatmin nchar(6) NULL,
  - tmax float NULL,
  - horatmax nchar(6) NULL,
  - tmed float NULL,
  - presmin float NULL,
  - horapresmin nchar(6) NULL,
  - presmax float NULL,
  - horapresmax nchar(6) NULL,
  - prec nchar(10) NULL,
  - sol float NULL,
  - velmedia float NULL,
  - racha float NULL,
  - dir float NULL,
  - horaracha nchar(6) NULL,
  - ult-actualizacion datetime NULL,
- "aemet-datos-ultimos": Donde se almacenan los datos horarios obtenidos usando el API de la AEMET, su estructura es:
  - id-reg bigint IDENTITY(1,1) NOT NULL,
  - indicativo nchar(5) NULL,
  - fint nchar(19) NULL,
  - fecha nchar(10) NULL,

  - hora nchar(8) NULL,
  - ta float NULL,
  - tamin float NULL,
  - tamax float NULL,
  - pres float NULL,
  - pres-nmar float NULL,
  - hr float NULL,
  - vmax float NULL,
  - dmax float NULL,
  - prec nchar(10) NULL,
  - inso float NULL,
  - ult-actualizacion datetime NULL,

- "aemet-estaciones": Donde se almacenan las estaciones de observación registradas en la AMET. También alimentada a través del API Aemet.

  - id-reg bigint IDENTITY(1,1) NOT NULL,
  - indicativo nchar(5) NULL,
  - activo nchar(2) NULL,
  - nombre nchar(50) NULL,
  - provincia nchar(30) NULL,
  - indsinop nchar(5) NULL,
  - altitud nchar(5) NULL,
  - latitud nchar(7) NULL,
  - longitud nchar(7) NULL,
  - ult-actualizacion datetime NULL,

- "aemet-log-api": Aquí se almacena el registro de las llamadas a la API Aemet con el resultado obtenido y el éxito o fallo.

  - id-reg bigint IDENTITY(1,1) NOT NULL,
  - Fecha datetime NULL,
  - Indicativo nchar(17) NULL,
  - LLamada nchar(255) NULL,
  - Resultado nchar(2) NULL,

- "Zeus-Log": En desuso, sólo para fines de depuración.

- "Zeus-Sensores": Donde se almacenan los sensores Zeus.

  - id-reg bigint IDENTITY(1,1) NOT NULL,
  - fecha datetime NULL,

- idSensor nchar(17) NULL,
- temperatura float NULL,
- presion float NULL,
- humedad float NULL,
- luminosidad float NULL,
- ipOrigen nchar(15) NULL,

- "Zeus-Sensores-Registros": Donde se almacenan los datos que son enviados por los sensores "Zeus".

  - id-reg bigint IDENTITY(1,1) NOT NULL,
  - fecha datetime NULL,
  - idSensor nchar(17) NULL,
  - temperatura float NULL,
  - presion float NULL,
  - humedad float NULL,
  - luminosidad float NULL,
  - ipOrigen nchar(15) NULL,

Apéndice E

Modelo de la base de datos Zeus.Admin

En este anexo se muestra la estructura detallada de la Base de Datos 'Zeus", que está compuesta de las siguientes tablas:

- "Exceptions": Donde se almacenan las excepciones ocurridas en tiempo de ejecución.
  - Id bigint IDENTITY(1,1) NOT NULL,
  - GUID uniqueidentifier NOT NULL,
  - ApplicationName nvarchar(50) NOT NULL,
  - MachineName nvarchar(50) NOT NULL,
  - CreationDate datetime NOT NULL,
  - Type nvarchar(100) NOT NULL,
  - IsProtected bit NOT NULL,
  - Host nvarchar(100) NULL,
  - Url nvarchar(500) NULL,
  - HTTPMethod nvarchar(10) NULL,
  - IPAddress nvarchar(40) NULL,
  - Source nvarchar(100) NULL,
  - Message nvarchar(1000) NULL,
  - Detail nvarchar(max) NULL,
  - StatusCode int NULL,
  - SQL nvarchar(max) NULL,
  - DeletionDate datetime NULL,
  - FullJson nvarchar(max) NULL,
  - ErrorHash int NULL,
  - DuplicateCount int NOT NULL,
- "Languajes": En esta tabla se almacenan los idiomas que queremos tener disponibles.
  - Id int IDENTITY(1,1) NOT NULL,
  - LanguageId nvarchar(10) NOT NULL,
  - LanguageName nvarchar(50) NOT NULL,
- "RolesPermissions": Tabla que relaciona los permisos asignados a un determinado rol de usuarios.

  - RolePermissionId bigint IDENTITY(1,1) NOT NULL,
  - RoleId int NOT NULL,
  - PermissionKey nvarchar(100) NOT NULL,

- "Roles": Aquí se gestionan los roles de los usuarios que se quieran definir.
  - RoleId int IDENTITY(1,1) NOT NULL,
  - RoleName nvarchar(100) NOT NULL,

- "UserPermissions": Aqui se almacenan los permisos concedidos y denegados a un determinado usuario.
  - UserPermissionId bigint IDENTITY(1,1) NOT NULL,
  - UserId int NOT NULL,
  - PermissionKey nvarchar(100) NOT NULL,
  - Granted bit NOT NULL,

- "UserPreferences": Donde se almacenan y gestionan las prefenrecias de los usuarios (no se utiliza en este proyecto).
  - UserPreferenceId int IDENTITY(1,1) NOT NULL,
  - UserId bigint NOT NULL,
  - PreferenceType nvarchar(100) NOT NULL,
  - Name nvarchar(200) NOT NULL,
  - Value nvarchar(max) NULL,

- "UserRoles": Donde se almacenan los usuarios que perteneces a un determiando rol de usuarios.
  - UserRoleId bigint IDENTITY(1,1) NOT NULL,
  - UserId int NOT NULL,
  - RoleId int NOT NULL,

- "Users": Donde se gestionan los usuarios del sistema.
  - UserId int IDENTITY(1,1) NOT NULL,
  - Username nvarchar(100) NOT NULL,
  - DisplayName nvarchar(100) NOT NULL,
  - Email nvarchar(100) NULL,
  - Source nvarchar(4) NOT NULL,
  - PasswordHash nvarchar(86) NOT NULL,
  - PasswordSalt nvarchar(10) NOT NULL,
  - LastDirectoryUpdate datetime NULL,
  - UserImage nvarchar(100) NULL,
  - InsertDate datetime NOT NULL,

  - InsertUserId int NOT NULL,
  - UpdateDate datetime NULL,
  - UpdateUserId int NULL,
  - IsActive smallint NOT NULL,

- "UsersLog": Donde se almacenan los accesos al sistema por parte de los usuarios.
  - id_reg bigint IDENTITY(1,1) NOT NULL,
  - Fecha datetime NULL,
  - Username nvarchar(100) NULL,
  - Resultado nchar(100) NULL,

- "VersionInfo": Donde se almacena versiones de las migraciones de la base de datos (sólo para usos de desarrollo).
  - Version bigint NOT NULL,
  - AppliedOn datetime NULL,
  - Description nvarchar(1024) NULL