PROYECTO FIN DE GRADO

TÍTULO: Recopilación y procesamiento de los datos de un inversor solar extraídos de la nube de Huawei mediante API REST

AUTOR: Sergio Roca Montesa

TITULACIÓN: Ingeniería Telemática

TUTOR: Javier Malagón Hernández

DEPARTAMENTO: Ingeniería Telemática y Electrónica (DTE)

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE: Jerónimo López-Salazar Codes

TUTOR: Javier Malagón Hernández

SECRETARIO: Óscar Ortiz Ortiz

Fecha de lectura:

Calificación:

El Secretario,

"Es sencillo hacer
que las cosas parezcan complicadas,
pero es difícil hacer que sean sencillas"

Friedrich Nietzsche

Agradecimientos

A mi madre, que siempre ha confiado pese a todo.

A mi familia, por darme las herramientas para poder conseguirlo y acompañarme durante el largo camino.

A Marta, mi otra mitad, por motivarme y ayudarme a ser mejor en todo lo que hago.

A Francés y Diego, por las infinitas batallas que hemos librado juntos para poder llegar hasta aquí.

Resumen

En la era digital, el almacenamiento en la nube ha transformado la forma en que gestionamos y accedemos a nuestros datos. Sin embargo, aunque parece que poseemos nuestros archivos y documentos, la realidad es que nuestra relación con ellos es más compleja. Al guardar información en la nube, ésta reside en servidores de grandes corporaciones, y aunque se nos garantiza la posibilidad de consultarla y gestionarla, los términos de uso a menudo limitan nuestro control. En esencia, no somos los verdaderos propietarios de nuestros datos: la infraestructura y los mecanismos de seguridad están bajo el dominio de los proveedores del servicio, quienes también pueden tener acceso a la información, imponer restricciones, o incluso decidir cuándo y cómo se puede usar. Esto plantea inquietudes sobre la privacidad y el control, ya que, aunque nuestros datos siempre parecen estar al alcance, la verdadera soberanía sobre ellos no nos pertenece.

El propósito principal de este proyecto es garantizar que, aunque inicialmente el usuario dependa del fabricante para obtener los datos a través de una API, una vez que estos datos se descargan, el usuario se convierta en su propietario. Esto significa que, tras la obtención, el usuario tiene control total sobre la información, pudiendo almacenarla, gestionarla y utilizarla de manera independiente, sin ninguna dependencia adicional del fabricante, en este caso Huawei. Así, se asegura la autonomía y la verdadera posesión de los datos, brindando al usuario la libertad de decidir cómo proteger y utilizar su información, sin restricciones externas.

Para conseguirlo, se ha realizado una arquitectura que pide, almacena y explota los datos del inversor solar de Huawei, todo ello utilizando una Raspberry Pi que actúa como servidor, almacenando todo el software necesario para la gestión de los datos, y se encuentra en la red local de la casa del usuario. Las aplicaciones del sistema desarrollado se integran en openHAB, software de gestión del hogar digital, para visualizar los datos del inversor en la misma aplicación que el resto de los datos del hogar. Dicha integración se ha realizado utilizando el centro de control del Hogar Digital situado en la Escuela.

Palabras clave

API REST, base de datos, Grafana, hogar digital, Huawei, InfluxDB, máquina virtual, NorthBound Open API, openHAB, Python, Raspberry Pi, Visual Studio Code.

Abstract

In the digital era, cloud storage has transformed how we manage and access our data. However, while it seems that we own our files and documents, the reality is more complex. When storing information in the cloud, it resides on servers owned by large corporations, and although we are assured the ability to access and manage it, the terms of use often limit our control. In essence, we are not the true owners of our data: the infrastructure and security mechanisms remain under the service providers' control. These providers can access the information, impose restrictions, or even dictate when and how it can be used. This raises concerns about privacy and control since, although our data appears to be always within reach, true sovereignty over it does not belong to us.

The main purpose of this project is to ensure that, although users initially depend on the manufacturer to obtain data through an API, once the data is downloaded, users become its rightful owners. This means that after retrieval, users gain full control over their information, enabling them to store, manage, and use it independently without further reliance on the manufacturer, in this case, Huawei. This approach ensures autonomy and true ownership of the data, providing users the freedom to decide how to protect and utilize their information without external restrictions.

To achieve this, an architecture has been developed to request, store, and exploit data from the Huawei solar inverter, all within a virtual machine simulating a Raspberry Pi. This Raspberry Pi acts as a server, hosting all the necessary software for data management and residing within the user's local network. The system’s applications are integrated into openHAB, a digital home management platform, allowing users to visualize inverter data alongside other household data within the same application. This integration has been implemented using the Digital Home control center located at the College.

Key words

API REST, database, Grafana, Huawei, InfluxDB, NorthBound Open API, openHAB, Python, Raspberry Pi, Smart home, virtual machine, Visual Studio Code.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Lista de acrónimos

API – Application Programming Interface

DHCP – Dynamic Host Configuration Protocol

DNS – Domain Name Service

GPG – GNU Privacy Guard

GRUB – Grand Unified Bootloader

HDMI – High-Definition Multimedia Interface

HTTP – Hypertext Transfer Protocol

IoT – Internet of Things

JRE – Java Runtime Environment

JSON – JavaScript Object Notation

MAC – Media Access Control

MV – Máquina virtual

ODS – Objetivos de Desarrollo Sostenible

openHAB – Open Home Automation Bus

RAM – Random Access Memory

REST – Representational State Transfer

SSH – Secure Shell

SMB – Server Message Block

URL – Uniform Resource Location

USB – Universal Serial Bus

VNC – Virtual Network Computing

VSC – Visual Studio Code

WiFi – Wireless Fidelity

XFCE – Xforms Common Environments

1. Introducción

1.1 Marco y motivación del proyecto

En un mundo cada vez más conectado y dependiente de la tecnología, el concepto de hogar digital ha ganado una relevancia considerable. Los avances en dispositivos inteligentes y sistemas de automatización permiten a los usuarios gestionar sus hogares de manera eficiente, desde controlar la energía hasta optimizar el uso de recursos. Sin embargo, esta creciente dependencia también plantea un desafío significativo: la falta de control y propiedad real sobre los datos personales generados por estos dispositivos, que a menudo son gestionados por terceros.

Este proyecto está particularizado a un hogar que produce energía solar fotovoltaica y cuenta con un inversor de Huawei que envía los datos de producción y consumo a la nube de Huawei, que el usuario puede consultar mediante una aplicación web de Huawei.

La motivación detrás de este proyecto surge de la necesidad de devolver a los usuarios el control sobre sus datos en un entorno digital que prioriza la autonomía y la privacidad. Aunque las plataformas de fabricantes, como Huawei, proporcionan acceso a los datos a través de aplicaciones web, el usuario sigue siendo dependiente de esas aplicaciones para consultar y gestionar su información. Este proyecto se propone cambiar esa dinámica, garantizando que, una vez obtenidos los datos, el usuario se convierta en el propietario absoluto, pudiendo almacenarlos y gestionarlos sin restricciones externas.

Para lograrlo, se ha diseñado una arquitectura innovadora que permite obtener y gestionar los datos del inversor solar de Huawei de manera segura y eficiente. Utilizando una Raspberry Pi, el sistema actúa como un servidor privado en la red local del usuario, garantizando que toda la información se mantenga dentro de un entorno controlado. Dentro de esta máquina virtual se encuentra el software para hacer las peticiones a la nube de Huawei, la base de datos InfluxDB en la que se almacenan los datos obtenidos, la aplicación Grafana que explota estos datos y genera los gráficos y, finalmente, la aplicación openHAB que gestiona todos los dispositivos del hogar digital, que es donde finalmente el usuario puede consultar los datos, tanto históricos como en tiempo real. [1]

Esta solución no solo proporciona independencia y flexibilidad, sino que también refuerza la seguridad y privacidad de los datos en el hogar digital, empoderando al usuario y dándole soberanía total sobre su información.

1.2 Objetivos

El objetivo principal de este proyecto es acceder a la nube de Huawei, fabricante del inversor, y descargar los datos que este envía para poder almacenarlos y explotarlos localmente desde el sistema de control de un hogar digital.

Para conseguir este objetivo principal se proponen los siguientes objetivos específicos:

- Acceder a los datos y descargarlos mediante una aplicación que utilice la API de Huawei.
- Almacenar y explotar los datos de manera local, utilizando una base de datos, un sistema gestor de base de datos y una aplicación para mostrar de forma gráfica esos datos.
- Visualizar los datos de forma clara y cómoda creando una interfaz diferente a la ofrecida por Huawei.
- Integrar este sistema en el centro de control del hogar digital del usuario.

1.3 Estructura del resto de la memoria

En este capítulo de la memoria se ofrece una vista general de la misma enumerando todos los apartados de los que se compone este proyecto y dando una breve explicación sobre cada uno de ellos:

1 **Introducción**.
Se explica la situación tecnológica en la que se enmarca este proyecto, la motivación para su realización y el problema que busca solucionar. Adicionalmente, se explica cuál es el objetivo principal del proyecto y cuáles son los objetivos específicos que se han seguido para cumplir con el principal.

2 **Marco tecnológico.**
Se explica el estudio realizado de algunas de las tecnologías que existen y se relacionan con este proyecto, justificando la elección de cada una de ellas.

3 **Especificaciones y restricciones de diseño.**
Se explican los factores que es necesario tener en cuenta para la implementación de la solución.

4 **Implementación de la solución.**
Se detallan cada uno de los pasos seguidos para la construcción de la solución propuesta.

5 **Resultados.**
En este apartado se explican las pruebas realizadas para la comprobación de un correcto funcionamiento, así como los resultados obtenidos y sus implicaciones.

6 **Presupuesto.**
Se detalla cada uno de los aspectos monetarios que están relacionados con el desarrollo de la solución.

7 Impacto del proyecto.

Se indican las implicaciones sociales, de salud y seguridad, ambientales, económicas y tecnológicas relacionadas con el proyecto, así como la aportación a los Objetivos de Desarrollo Sostenible.

8 Conclusiones.

Se explican las diferentes fases por las que ha pasado el proyecto, se analizan los objetivos previamente descritos y cómo se han cumplido y por último se proponen algunos trabajos futuros para la ampliación del proyecto.

9 Referencias.

Se expone la lista completa de fuentes citadas en esta memoria.

2. Marco tecnológico

En este apartado se muestra el contexto tecnológico en el que se encuentra este proyecto. Se muestran varias tecnologías y se explica cuáles de ellas se han utilizado en la solución.

2.1 Huawei

Huawei es una empresa de tecnología china fundada en el año 1987. Actualmente es una de las compañías punteras en el desarrollo de dispositivos móviles, así como de equipos utilizados para mantener el hogar conectado. [2]

En el marco de este proyecto, Huawei es la empresa fabricante del inversor fotovoltaico. Por ese motivo, todos los datos generados por el inversor (energía generada, consumida, comprada, vendida, etc.) se encuentran en sus servidores. El usuario puede consultar los datos sobre su instalación en la página web proporcionada por Huawei, llamada FusionSolar. [3]

En la siguiente ilustración se puede observar un gráfico de ejemplo ofrecido por FusionSolar:

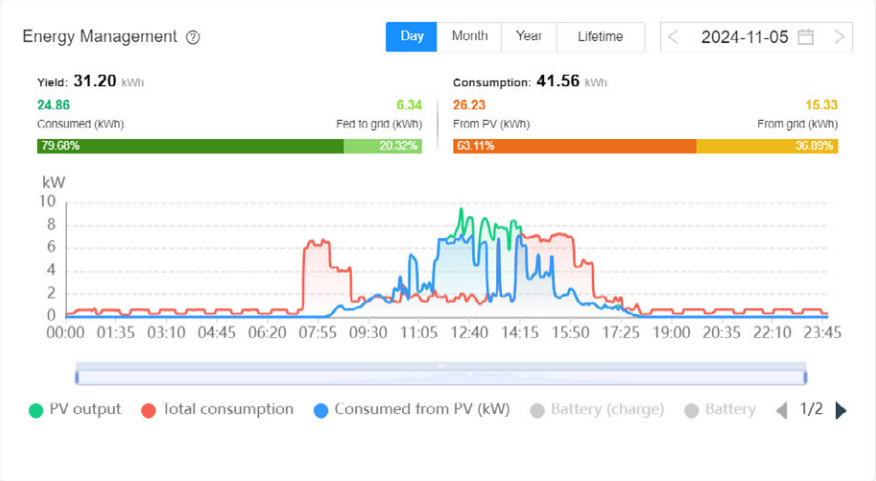

Ilustración 1 Gráfico ejemplo de FusionSolar

Esto supone un problema, ya que el usuario no es poseedor de los datos generados por su planta fotovoltaica, sino que simplemente está autorizado a consultarlos mediante la página web ofrecida por Huawei, siempre y cuando Huawei mantenga lo términos y condiciones de uso y no cese su actividad.

Por otro lado, Huawei ofrece una alternativa para acceder a los datos. Se trata de una API pública.

2.1.1 API

Una API (*Application Programming Interface*) es un conjunto de definiciones, reglas y protocolos que permite que diferentes aplicaciones o servicios se comuniquen y se integren de manera estructurada. Las APIs definen cómo los componentes de software deben

interactuar, especificando las solicitudes que pueden realizarse, cómo deben formarse estas solicitudes, y los formatos de las respuestas. [4]

Desde un punto de vista técnico, las APIs actúan como un intermediario entre diferentes sistemas, proporcionando interfaces que simplifican la complejidad de las interacciones subyacentes. En este proyecto, la API hace de intermediario entre los servidores de Huawei donde se encuentran los datos y el usuario que los pide.

Al abstraer la lógica interna de un servicio, las APIs permiten a los desarrolladores acceder a funcionalidades o datos específicos sin necesidad de comprender o manipular el código interno del software con el que están interactuando. De esta manera a través de peticiones HTTPS sencillas se pueden pedir todos los datos almacenados del inversor.

La tecnología utilizada por Huawei para ofrecer la API es *Northbound Open API*.

2.1.2 Northbound Open API

Se trata de una interfaz pública basada en el estándar REST y diseñada para integrar sistemas de terceros. [5]

REST (*Representational State Transfer*) es un estilo arquitectónico para el diseño de sistemas distribuidos, particularmente los servicios web [6].

Para que una API se considere REST debe cumplir las siguientes especificaciones:

- Arquitectura cliente-servidor que se comunica a través de peticiones HTTP/HTTPS[1].
- No tiene estado, es decir, no se almacena información del cliente. Cada petición debe incluir toda la información necesaria para ser procesada.
- Guardar información en caché para agilizar la comunicación.
- Interfaz uniforme. Todas las peticiones para el mismo recurso deben tener el mismo aspecto.
- Sistema organizado en capas. Esto hace que para el cliente sea invisible el punto final de la comunicación del lado del servidor.
- Código bajo demanda. Capacidad para que el servidor envíe al cliente códigos ejecutables cuando éste lo solicite.

Es necesario crear una cuenta en Huawei identificándose como el propietario del inversor para acceder a estos servicios.

De este modo, Huawei proporciona acceso a los datos del inversor a través de solicitudes HTTP/HTTPS, gestionadas mediante Northbound Open API. Es necesario que la primera solicitud incluya las credenciales del usuario para permitir la autenticación correspondiente.

[1] HTTP es un protocolo del nivel de aplicación del modelo TCP/IP [40] diseñado para transmitir información entre dos dispositivos conectados a internet, cliente y servidor [41].

Una vez autenticado, se recibe un *token* que será necesario incluir en las siguientes peticiones para identificarse.

En la siguiente imagen se muestra la arquitectura de la API:

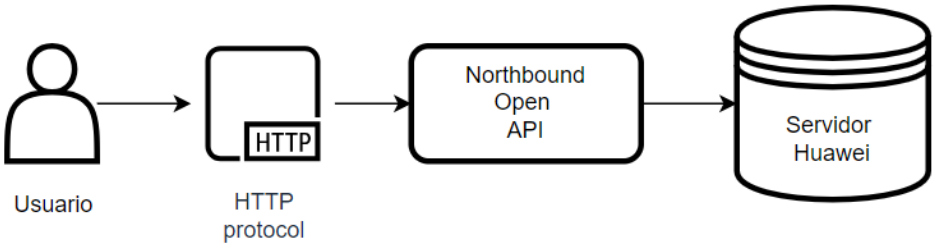

Ilustración 2 Arquitectura API

Las respuestas a estas solicitudes se envían en formato JSON, un estándar ligero de intercambio de datos. Este formato es ampliamente utilizado debido a su simplicidad para ser leído y escrito por humanos, así como por su facilidad para ser procesado y generado por los sistemas informáticos [7].

En la siguiente ilustración se muestra un documento de ejemplo JSON devuelto por la API:


```
{
    "success":true,
    "data":[
    {
        "dataItemMap":{
        "use_power":288760,
        "radiation_intensity":0.6968,
        "reduction_total_co2":18.275,
        "reduction_total_coal":7.332,
        "inverter_power":null,
        "theory_power":17559.36,
        "ongrid_power":18330,
        "power_profit":34320,
        "installed_capacity":25200,
        "perpower_ratio":0.727,
        "reduction_total_tree":999,
        "performance_ratio":89
    },
    "stationCode":"5D02E8B40AD342159AC8D8A2BCD4FAB5",
    "collectTime":1501516800000
    }
    ],
    "failCode":0,
    "params":{
        "stationCodes":"BA4372D08E014822AB065017416F254C,5D02E8B40AD342159AC8D8A2BCD4FAB5",
        "collectTime":1501862400000,
        "currentTime":1503046597854
    },
    "message":null
}
```


Ilustración 3 Código ejemplo JSON

2.2 Raspberry Pi

Una Raspberry Pi es un ordenador de placa única desarrollado por la fundación Raspberry Pi en Reino Unido [8]. Su objetivo es promover la enseñanza de ciencias de la computación y la educación tecnológica en escuelas, pero debido a su versatilidad y bajo costo, se ha convertido en una herramienta popular para una amplia gama de proyectos educativos y profesionales.

En la siguiente ilustración se muestra el tamaño real de una Raspberry Pi:

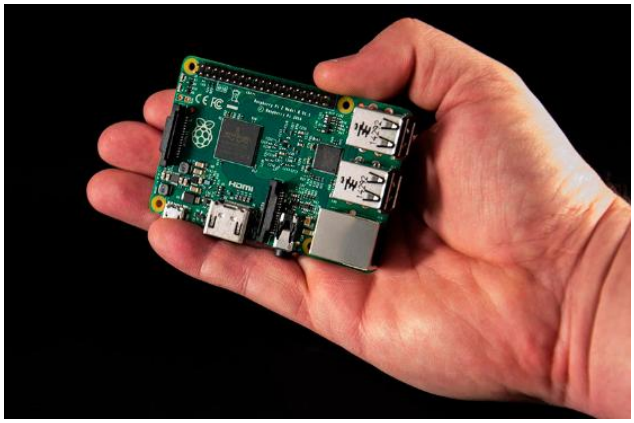

Ilustración 4 Raspberry Pi [9]

A pesar de su reducido tamaño, la Raspberry Pi ofrece todos los elementos esenciales de un ordenador como son procesador, memoria RAM, puertos USB, conectividad de red mediante Ethernet o WiFi y salida de video HDMI.

Como sistema operativo generalmente utiliza una distribución de Linux basada en Debian llamada Raspberry Pi OS (anteriormente llamado Raspbian), pero también puede ejecutar otras distribuciones de Linux como Ubuntu o incluso versiones simplificadas de Windows. [10]

Por todo lo anterior, la Raspberry Pi es una de las mejores elecciones para ejecutar el centro de control del hogar digital, ya que tiene más potencia que otras alternativas como Arduino. Por ese motivo se ha elegido utilizar una Raspberry Pi en este proyecto, pero, por simplicidad, para no depender de una Raspberry Pi real, se ha usado una máquina virtual que ejecuta el sistema operativo Raspberry Pi OS. Esta máquina virtual recrea el entorno de la Raspberry Pi, permitiendo realizar pruebas y desarrollos como si se estuviera utilizando el hardware físico, pero con la flexibilidad y el control adicionales que proporciona un entorno virtualizado, que en este proyecto ha sido VMWare Workstation.

2.2.1 VMWare Workstation

Es un software de virtualización de escritorio que permite a los usuarios ejecutar múltiples sistemas operativos en un mismo ordenador físico (máquina anfritriona). [11]

En la siguiente ilustración se muestra el menú de VMWare Workstation donde se observan diferentes máquinas virtuales listas para ser ejecutadas en la misma máquina anfitriona:

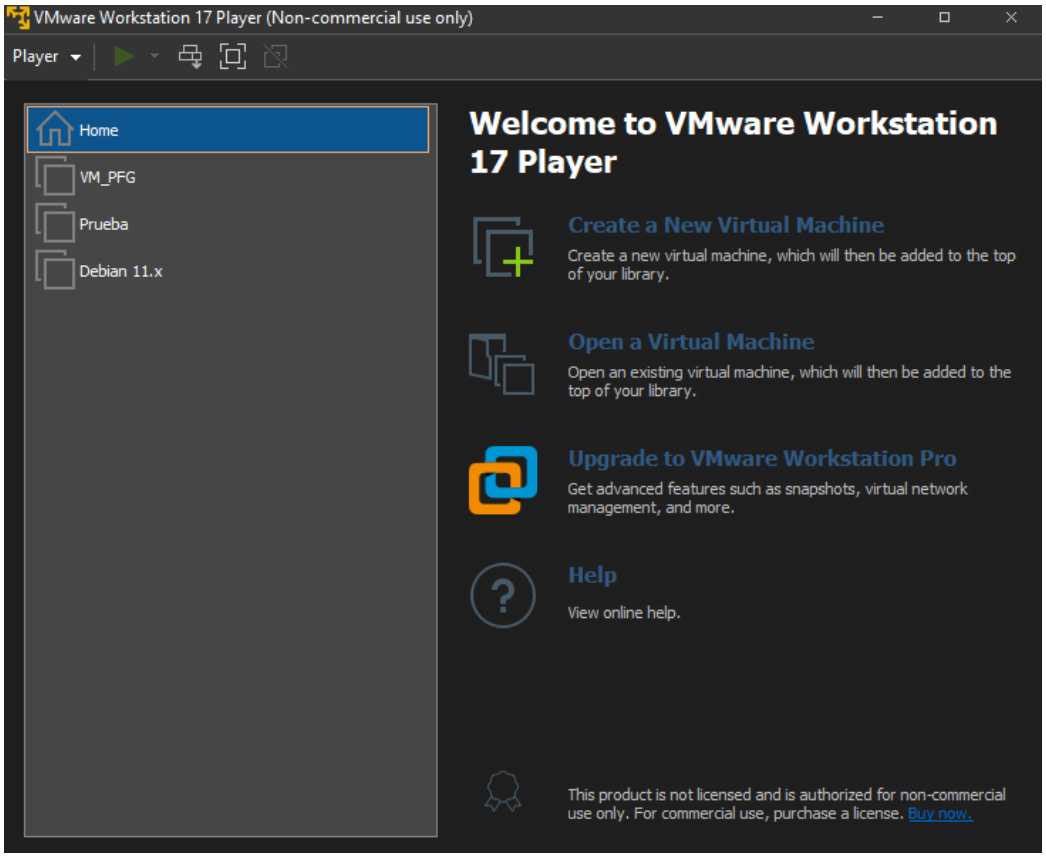

Ilustración 5 Menú principal VMWare Workstation

Algunas de las características de VMWare son:

- Soporte para múltiples sistemas operativos, como por ejemplo Windows y Linux entre otros.
- Compatibilidad de arrastre y soltado. Facilita la compartición de archivos entre la máquina anfitriona y las máquinas virtuales.
- Compatibilidad de dispositivos USB. Permite conectar dispositivos directamente a la máquina virtual.
- Modo pantalla completa. Proporciona la experiencia de un sistema operativo completo sin la interfaz gráfica de la máquina anfitriona. Esto resulta especialmente útil cuando la máquina virtual dispone de una interfaz gráfica de usuario integrada.

Para proporcionar conectividad entre la máquina virtual y la anfitriona existen diferentes tecnologías, explicadas a continuación.

2.2.2 SSH

SSH (*Secure Shell*) es un protocolo de nivel de aplicación utilizado para enviar de manera segura comandos a una máquina que se encuentra en una red no segura. [12]

Para conseguir esta seguridad, SSH encripta los datos transmitidos para protegerlos de intercepciones. Además, para acceder al servicio se requiere autenticación, que puede ser mediante usuario y contraseña o con clave pública y privada.

En la solución de este proyecto se ha utilizado SSH para enviar comandos desde la máquina anfitriona a la máquina virtual, simulando un entorno real en el que el usuario con su ordenador personal no tiene acceso directo a la Raspberry Pi, sino que debe acceder desde la red. Además, la utilización de SSH supone una ventaja en el desarrollo de la solución ya que permite copiar comandos de la máquina anfitriona, pegarlos en la línea de comandos de SSH y enviarlos a la máquina virtual, minimizando errores de escritura.

2.2.3 SMB

SMB (*Server Message Block*) es un protocolo de nivel de aplicación de compartición de archivos mediante la red. [13]

Se utiliza en entornos Windows, pero también se puede utilizar en dispositivos Unix con un software que implemente este protocolo, como por ejemplo Samba. [14]

Desde el punto de vista del usuario, una vez que ha sido configurado el acceso con la dirección IP del servidor donde se encuentran los archivos, estos son accesibles a través del explorador de archivos de Windows como si de archivos locales se tratara. En la solución de este proyecto se ha utilizado Samba para compartir los archivos de la Raspberry Pi con la máquina anfitriona. Con esto se permite que todo el código de programación se codifique en la máquina anfitriona, que cuenta con mayor potencia y fluidez.

2.2.4 VNC

El entorno real del proyecto se basa en una Raspberry Pi que se encargue de controlar el hogar digital del usuario. En este contexto, es muy posible que dicha Raspberry Pi no disponga de una pantalla, ratón y teclado conectados permanentemente para ser manejada como cualquier ordenador, puesto que se necesitaría mucho espacio para todo el conjunto.

Para solucionar este problema se ha utilizado VNC (*Virtual Network Computing*), una solución que haciendo uso del protocolo RFB (*Remote Frame Buffer*) transmite mediante la red la interfaz gráfica de usuario de un dispositivo a otro, permitiendo interactuar con ella de manera remota. [15]

Para su utilización se requiere una doble instalación. Se debe instalar VNC Server en la máquina que se quiere compartir (en este caso la Raspberry Pi), mientras que en la máquina que va a visualizar, se debe instalar VNC Viewer. La instalación completa se aborda en el apartado 4, Descripción de la solución propuesta.

2.3 Lenguajes y entorno de programación

El lenguaje de programación es la herramienta que permite codificar la funcionalidad básica de la solución. Específicamente se utiliza para hacer peticiones a la API de Huawei, manejar y transformar los datos que la API devuelva y finalmente almacenarlos en una base de datos. Existen multitud de lenguajes populares candidatos para este proyecto, pero a continuación se explican los dos más apropiados.

2.3.1 Java

Java es un lenguaje de programación de alto nivel orientado a objetos y de propósito general que puede ser ejecutado en cualquier máquina que tenga instalado JRE (*Java Runtime Environment).* Fue desarrollado por Sun Microsystems en 1995 y es utilizado para multitud de aplicaciones web, móviles, sistemas distribuidos o servidores entre otros. [16]

Sus características principales son:

- Orientación a objetos. Permite organizar la programación en unidades reutilizables y modulares.
- Gestión automática de memoria. Cuenta con un recolector de basura que se encarga de gestionar y liberar memoria automáticamente, liberando de esta responsabilidad a los programadores.
- Ejecución multihilo. Soporta la ejecución de múltiples hilos dentro de un mismo programa, lo que aumenta su velocidad y eficiencia.
- Bibliotecas. La biblioteca estándar proporciona un gran número de funcionalidades para todo tipo de aplicaciones.

En la siguiente ilustración se muestra un ejemplo de código desarrollado en Java:

```java
import java.util.Scanner;
class AreaOfCircle
{
   public static void main(String args[])
    {

      Scanner s= new Scanner(System.in);

         System.out.println("Enter the radius:");
         double r= s.nextDouble();
         double  area=(22*r*r)/7 ;
         System.out.println("Area of Circle is: " + area);
   }
}
```

Ilustración 6 Ejemplo de código en Java

2.3.2 Python

Python es un lenguaje de programación de alto nivel de propósito general. Es orientado a objetos, aunque también permite otros paradigmas como programación funcional y

programación estructurada. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. [17]

Sus características principales son:

- Sintaxis simple y clara. Destaca por la sencillez de su código, ya que no es necesario la utilización del carácter ";" al final de cada línea de código y para definir bloques no utiliza llaves "{}", sino que se organiza con sangrías o *indentaciones* (del inglés, *indentation*).
- Lenguaje interpretado. El código se ejecuta línea por línea sin necesidad de ser compilado previamente.
- Portabilidad. Es multiplataforma, lo que significa que se puede ejecutar en diferentes sistemas operativos como Windows, macOS o Linux.
- Gran biblioteca estándar y multitud de bibliotecas externas. Python es un lenguaje muy potente debido a la gran cantidad de bibliotecas disponibles que simplifican el código y permiten a los desarrolladores hacer aplicaciones muy complejas delegando muchas pequeñas funcionalidades a las distintas bibliotecas importadas.

En la siguiente ilustración se muestra un ejemplo de código desarrollado en Python:

```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print '    %s [label="%s' % (nodename, label)
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
   else:
      print '"];'
      children = []
      for n, child in enumerate(ast[1:]):
         children.append(dotwrite(child))
      print '    %s -> {' % nodename,
      for name in children:
         print '%s' % name,
```

Ilustración 7 Ejemplo de código en Python

2.3.3 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, lanzado por primera vez en 2015. [18]

Es ampliamente utilizado por desarrolladores debido a los siguientes aspectos:

- Es ligero y rápido, utilizando menos recursos que otros editores.
- Es multilenguaje, permitiendo el desarrollo de aplicaciones en los lenguajes más utilizados, incluyendo Java, JavaScript, Python, C, C#, C++ y muchos más.
- Es completamente personalizable gracias a las extensiones, muchas de ellas desarrolladas por la comunidad.
- Ofrece herramientas de completado automático de código y depuración.

- Tiene un terminal integrado que permite la ejecución de comandos de sistema, permitiendo la instalación de los paquetes necesarios para la aplicación desde el mismo editor de código.
- Está integrado con sistemas de control de versiones como Git.

2.3.4 Conclusión

Dado que la solución de este proyecto no requiere de orientación a objetos y que ambos lenguajes expuestos tienen una gestión de peticiones HTTP y conexión con bases de datos similares, se ha decidido utilizar Python por su biblioteca externa Pandas, extremadamente útil para el manejo de tablas de datos en estructuras llamadas *DataFrame*.

El entorno de desarrollo elegido es Visual Studio Code ya que tiene una extensión de Python oficial desarrollada por Microsoft, una herramienta de depuración integrada, es muy ligero y fácil de instalar y personalizar.

2.4 Bases de datos

El siguiente elemento de la solución es la base datos, utilizada para almacenar todos los datos históricos del inversor y seguir volcando en ella los futuros. Se han contemplado tres tecnologías de almacenamiento de datos candidatas para este proyecto

2.4.1 MySQL

MySQL es un sistema de gestión de bases de datos de código abierto. [19]

MySQL es la versión original, pero desde que Oracle adquirió MySQL, se desarrolló un nuevo gestor de base de datos llamado MariaDB como alternativa para garantizar la continuidad de un software libre y abierto.

Características principales:

- Utiliza modelos de bases de datos relacionales, lo que significa que la información se organiza en tablas compuestas de filas y columnas, donde cada columna hace referencia a un atributo específico y cada fila es un registro individual.
- Utiliza el lenguaje de consulta estructurado SQL.
- Tiene compatibilidad multiplataforma, por lo tanto, se puede ejecutar en diferentes sistemas operativos y en entornos en la nube.
- Puede ser gestionado a través de la línea de comandos o a través de una interfaz gráfica de usuario

2.4.2 SQLite

SQLite es un sistema de gestión de bases de datos ligero, rápido y autocontenido. [20]

Sus características principales son:

- No necesita un sistema gestor de base de datos para funcionar, pues todo el sistema de gestión de base de datos está incluido en un solo archivo. Ideal para una Raspberry Pi que tiene potencia y almacenamiento limitados.
- No requiere configuración ni administración adicional, lo que hace que sea muy sencillo de utilizar.
- Utiliza el lenguaje de consulta estructurado SQL.
- Es multiplataforma, tiene soporte para Windows, macOS, Linux, Android e iOS.
- Fácil integración con programas desarrollados en Python, Java y muchos lenguajes más.

Su principal desventaja es que no gestiona las series temporales de manera eficiente, por tanto, si se quiere utilizar para almacenar información de dispositivos IoT es necesario una aplicación *backend* que transforme los datos a la hora de servirlos a la aplicación encargada de explotar esos datos para generar gráficos. Esta aplicación *backend* podría ser Flask.

2.4.2.1 Flask

Flask es un *microframework* de Python que permite crear aplicaciones web de manera rápida y sencilla. [21]

La manera de incluir Flask en la solución sería crear un servidor web con Python que escuchara en un puerto determinado y, ante ciertas peticiones, obtuviera la información de la base de datos SQLite, transformara los datos para dotarlos de un eje temporal y los sirviera a la aplicación de explotación de datos. Se puede considerar Flask como una API entre la base de datos y la aplicación de explotación de los gráficos.

En la siguiente ilustración se muestra cómo se situaría el servidor Flask en la solución:

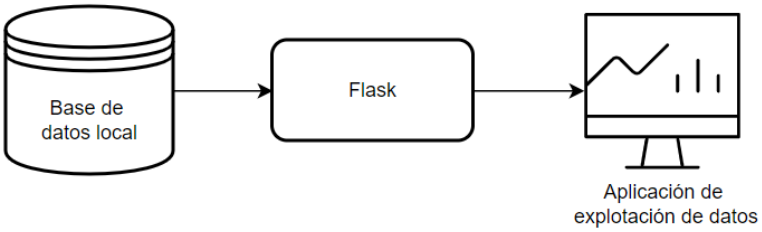

Ilustración 8 Arquitectura servidor Flask

Aunque viable, es una solución algo compleja y requiere de una aplicación Python ejecutándose constantemente en la Raspberry Pi, consumiendo recursos de esta.

2.4.3 InfluxDB

InfluxDB es una base de datos de código abierto diseñada específicamente para manejar datos de series temporales. Es especialmente adecuada para aplicaciones que requieren la recopilación, almacenamiento y análisis de grandes volúmenes de datos que cambian con el tiempo, como métricas de rendimiento, datos de sensores, *logs* y eventos. [22]

Sus características principales son:

- Optimizada para series temporales. Está diseñada para almacenar datos que se registran a intervalos de tiempo específicos.
- Alto rendimiento. Es capaz de realizar de manera muy ágil operaciones de escritura y consulta de datos.
- Utiliza un lenguaje de consulta parecido a SQL, denominado InfluxQL. Aunque para algunas aplicaciones esto puede suponer una desventaja muy condicionante, en este proyecto no lo es ya que la aplicación de explotación de datos importa automáticamente los datos sin necesidad de realizar consultas manualmente.
- Integración con herramientas de monitoreo y visualización como Grafana.

2.4.4 Conclusión

Debido a la naturaleza temporal de los datos que se manejan en este proyecto se ha elegido utilizar la base de datos InfluxDB, diseñado específicamente para ello. De esta manera se manejan los datos teniendo en cuenta el eje temporal en el que se desarrollan, cosa que facilita mucho su visualización mediante Grafana, que es la herramienta seleccionada para la explotación de los datos como se explica en la siguiente sección.

2.5 Aplicación de explotación de los datos

Una vez seleccionada la base de datos en la que se almacenan todos los datos del inversor solar, en este apartado se explica qué herramienta se ha utilizado para explotar los datos guardados y obtener de ellos los gráficos de energía generada, consumida, comprada, vendida, ingresos, etc.

2.5.1 Power BI

Power BI es una herramienta de análisis de negocios y visualización de datos desarrollada por Microsoft. Permite a los usuarios transformar datos sin procesar en información significativa a través de visualizaciones interactivas y paneles de control o *dashboards*. Power BI es utilizado por empresas y organizaciones para tomar decisiones basadas en datos de manera efectiva. [23]

Sus características principales son:

- Conectividad a múltiples fuentes de datos, incluyendo bases de datos SQL y servicios en la nube como Azure o Google Analytics.

- Visualizaciones interactivas. Ofrece la posibilidad de personalizar e interactuar con los gráficos, tablas y mapas.
- Se pueden crear informes de los gráficos visualizados.
- Actualización automática de datos para que los informes muestren siempre información actualizada.

Al ser una herramienta de Microsoft requiere de un sistema Windows para funcionar. Se puede integrar en la solución propuesta instalando la aplicación de escritorio de Power BI en el ordenador Windows del usuario, y acceder a los gráficos desde el navegador utilizando la versión web de Power BI. Esta aplicación obliga al usuario a tener un ordenador con Windows donde instalar y configurar un *software*, por lo que ha sido descartada de la solución final.

2.5.2 Grafana

Grafana es una plataforma de código abierto para la visualización y análisis de datos tanto históricos como en tiempo real. Es ampliamente utilizado en aplicaciones de monitoreo como pueden ser aplicaciones IoT. [24]

Sus características principales son:

- Visualizaciones personalizables. Se pueden crear paneles con gráficos interactivos y visualizaciones atractivas.
- Soporte para múltiples fuentes de datos, entre ellas InfluxDB, que es la base de datos seleccionada.
- Permite configurar alertas basadas en condiciones específicas y enviarlas por correo electrónico.
- Interactividad. Ofrece funciones para explorar y analizar datos mediante la creación de consultas avanzadas.
- Gestión de usuarios y permisos. Se pueden configurar los paneles creados para que solo sean accesibles a usuarios con determinados permisos. Para ello existen las *Service Accounts*, que son grupos en los que se incluyen a los usuarios y a los que se les proporciona permisos. Esos permisos pueden ser sólo de visualización, edición o administración.

2.5.3 Conclusión

Puesto que Power BI requiere de un sistema Windows se ha descartado para la solución de este proyecto. Por otro lado, Grafana puede correr en la Raspberry Pi perfectamente. Además, cuenta con integración directa para datos almacenados en InfluxDB, por lo que no se requiere de ningún software adicional que sirva o transforme los datos entre ambas tecnologías. En conclusión, se ha decidido utilizar Grafana como tecnología para generar los gráficos, obteniendo los datos de InfluxDB.

2.6 Aplicación de gestión del hogar digital

Finalmente, una vez se tienen los gráficos deseados, se van a mostrar desde a una aplicación de gestión del hogar digital. El propósito de esto es que el usuario pueda visualizar los datos de su instalación fotovoltaica en la misma aplicación que visualiza y maneja el resto de los elementos presentes en su hogar digital, tales como enchufes, interruptores, persianas, etc. Se contemplan dos tecnologías distintas, ambas expuestas a continuación, además de la decisión de cuál se ha utilizado.

2.6.1 Home Assistant

Home Assistant es una plataforma de gestión del hogar digital de código abierto que permite integrar, controlar y automatizar dispositivos de diferentes fabricantes en un único sistema. Está diseñada para ofrecer una solución sencilla y potente. [25]

Sus características principales son:

- Compatibilidad con multitud de dispositivos y tecnologías. Esto permite a los usuarios unificar sus sistemas independientemente del fabricante de cada uno de los dispositivos. Además, es compatible con tecnologías como Z-Wave, Zigbee, KNX y MQTT entre otras…
- Automatizaciones potentes. Permite la creación de automatizaciones basadas en condiciones específicas, como la hora del día, la ubicación del usuario o el estado de otros dispositivos.
- Interfaz personalizable. Ofrece un tablero en el que se pueden añadir tarjetas para controlar los dispositivos y visualizar información relevante como el clima, el consumo de energía o la actividad de los sensores.
- Privacidad. Está pensado para ejecutarse localmente, lo que permite que los datos se mantengan seguros al no salir de la propia red privada.
- Soporte para scripts. Se pueden programar eventos que se ejecuten en momentos específicos o cuando se cumplan determinadas condiciones.
- Puede ser ejecutado en diferentes sistemas operativos, incluido Raspberry Pi OS. Para este caso, se recomienda utilizar Home Assistant OS, un sistema operativo completo, con todo lo necesario para funcionar como plataforma de automatización del hogar digital, que se instala en una Raspberry Pi.
- Cuenta con una comunidad muy activa, inclusiva y formada por integrantes de todo el mundo que comparten conocimientos, experiencias y contribuyen al crecimiento del sistema.

2.6.2 openHAB

Al igual que Home Assistan, openHAB (*open Home Automation Bus)* es una plataforma de gestión del hogar digital de código abierto que permite integrar, controlar y automatizar dispositivos de diferentes fabricantes en un único sistema. Está diseñada para ofrecer una solución sencilla y potente. [26]

Sus características principales son:

- Compatibilidad con multitud de dispositivos y tecnologías. Esto permite a los usuarios unificar sus sistemas independientemente del fabricante de cada uno de los dispositivos. Además, es compatible con tecnologías como Z-Wave, Zigbee, KNX y MQTT entre otras.
- Sistema de reglas potentes. Es uno de los puntos fuertes de openHAB, pues permite a los usuarios crear reglas personalizadas para automatizar acciones basadas en condiciones específicas, como encender las luces cuando se detecta movimiento o ajustar la temperatura cuando no hay nadie en casa. Estas reglas se pueden codificar en diferentes lenguajes de scripting, lo que brinda gran flexibilidad.
- Interfaz de usuario personalizable. openHAB ofrece diferentes aplicaciones de visualización, como son Basic UI, HABPanel y Main UI entre otras, lo que de nuevo ofrece mucha flexibilidad para que el usuario diseñe y personalice paneles de control personalizados según sus necesidades.
- Es multiplataforma. Se puede ejecutar en multitud de sistemas operativos, incluyendo Linux, Windows, macOS o Raspberry Pi OS entre otros.
- Comunidad. Cuenta con una comunidad global que contribuye con nuevas características, mejoras y corrección de errores.
- *Bindings*. Son módulos que permiten extender la funcionalidad de openHAB integrando dispositivos y servicios.
- Privacidad. Permite mantener seguros los datos del hogar digital al mantenerlos en la red privada del usuario

2.6.3 Conclusión

Como se observa, ambas tecnologías comparten la mayoría de las especificaciones, por lo que cualquiera de las dos es apta para la solución de este proyecto. Se ha elegido openHAB por los siguientes motivos:

- Su potente sistema de reglas para automatizar procesos.
- Ofrece una instalación manual más sencilla que Home Assistant, ya que con esta tecnología se recomienda instalar el sistema operativo completo Home Assistan OS.
- Aunque para un usuario poco experimentado puede ser un punto negativo, la personalización por código ofrecida por openHAB permite una mayor profundidad que la de Home Assistant.
- Integración con el centro de control del Hogar Digital de la Escuela.

2.7 Resumen

Este apartado recoge, a modo de resumen, toda la información expuesta en la sección 2, Marco tecnológico.

Las tecnologías que se han seleccionado para la solución son las siguientes:

- Northbound Open API, que ofrece la API de Huawei.
- Raspberry Pi OS, virtualizado mediante VMWare Workstation, y ofreciendo conexión con la máquina anfitriona mediante SSH, SMB y VNC.
- Python como lenguaje de programación para realizar las peticiones a la API y almacenar los datos en la base de datos.
- InfluxDB como sistema de gestión de base de datos.
- Grafana para generar los gráficos de los datos procedentes de InfluxDB.
- openHAB como programa de centro de control del hogar digital donde se van a mostrar los gráficos con los datos del inversor, además del resto de elementos del hogar.

3. Especificaciones y restricciones de diseño

En este apartado se detallan las especificaciones técnicas y las restricciones que han influido en el desarrollo del proyecto. Adicionalmente se proporciona una visión general de la propuesta.

3.1 Especificaciones de diseño

A continuación se listan las especificaciones más relevantes:

- Extracción de los datos enviados por el inversor de la nube de Huawei.
- Utilización de un ordenador de coste, consumo y tamaño reducidos.
- Almacenamiento de los datos en una base de datos local .
- Explotación de los datos almacenados.
- Construcción de una arquitectura accesible para usuarios poco familiarizados con la tecnología.
- Integración de los gráficos generados en la aplicación de gestión del hogar digital, mejorando así la interfaz de Huawei.

3.2 Restricciones de diseño

3.2.1 Tecnologías de código abierto

Desarrollo de la solución utilizando tecnologías no propietarias y de código abierto, tanto software como hardware.

3.2.2 API de Huawei

Diseño de la aplicación usando la API proporcionada por Huawei, que tiene las siguientes limitaciones debido a su estricto control de flujo:

- No se pueden realizar más de cinco peticiones cada diez minutos.
- Dependiendo del dato pedido, existe un límite de peticiones diarias. Dichas restricciones se pueden observar en la siguiente tabla:

Login Interface	The timeout value is 30 minutes, which will be reset if any service invokes this OpenAPI in 30 minutes. If the XSRF-TOKEN has not expired, the user does not need to log in again.
Logout Interface	Users will be logged out upon timeout.
Power Plant List Interface	Mandatory. The recommended maximum number of API calls per day is 24. This interface is used to obtain the list of authorized plants. If the authorized plants have not changed, there is no need to invoke this interface.
Interface for Real-time Plant Data	Optional. The recommended maximum number of API calls per day is 288. The interface data is updated every 5 minutes.
Interface for Hourly Plant Data	Optional. The recommended maximum number of API calls per day is 24. The interface data is updated every hour.
Interface for Daily Plant Data	Optional. The recommended maximum number of API calls per day is 1. The interface data is updated every day.
Interface for Yearly Plant Data	Optional. The recommended maximum number of API calls per day is 1. The interface data is updated every day.
Device List Interface	This OpenAPI is mandatory if device-level data needs to be queried. The recommended maximum number of API calls per day is 1. When the device list has not changed, the data returned by this interface remains the same.
Real-time Device Data Interface	Optional. The recommended maximum number of API calls per day is 288. The interface data is updated every 5 minutes.
5-minute Device Data Interface	Optional. The recommended maximum number of API calls per day is 288. The interface data is updated every 5 minutes.
Daily Device Data Interface	Optional. The recommended maximum number of API calls per day is 24. The interface data is updated every hour.
Monthly Device Data Interface	Optional. The recommended maximum number of API calls per day is 1. The interface data is updated every day.
Yearly Device Data Interface	Optional. The recommended maximum number of API calls per day is 1. The interface data is updated every day.
Device Alarm Interface	Optional. The recommended maximum number of API calls per day is 288. The interface data is updated every 5 minutes.

Tabla 1 Restricciones de la API de Huawei

3.2.3 Raspberry Pi

Uso de Raspberry Pi como ordenador en que se alojan todas las aplicaciones. Cabe tener en cuenta dos limitaciones derivadas:

- Limitación de potencia de procesado y capacidad de almacenamiento debido las especificaciones de Raspberry.
- Limitación en utilización de software compatible, ya que herramientas como las de Power Platform de Microsoft no se pueden ejecutar debido al sistema operativo. Estas tecnologías requieren de Windows.

3.3 Arquitectura propuesta

En la siguiente ilustración se representa la arquitectura propuesta para la solución:

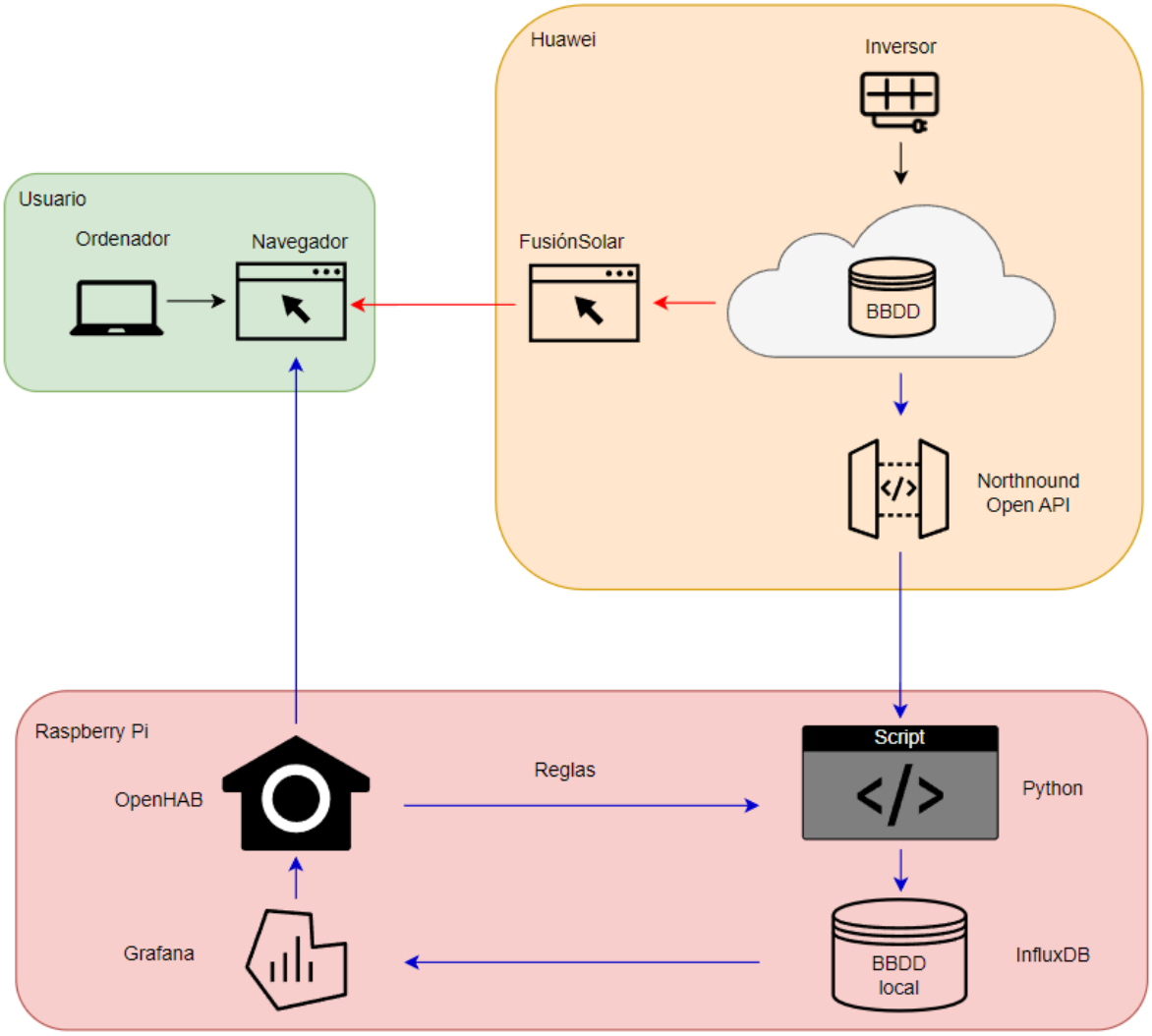

Ilustración 9 Arquitectura de la solución

Las flechas rojas indican el camino de los datos si el usuario visualiza los gráficos desde la web de FusionSolar ofrecida por Huawei. Las flechas azules indican el camino que siguen los datos en la arquitectura de este proyecto. Las flechas negras son pasos comunes, tanto si se visualiza desde la web FusionSolar de Huawei como si se utiliza la arquitectura de la solución propuesta.

Elementos destacables:

- Base de datos local. Una de las motivaciones de este proyecto es hacer que el usuario sea propietario de sus propios datos. Para ello se usa una base de datos alojada dentro de la Raspberry Pi en la casa del usuario, para que así, una vez pedidos los datos mediante el script Python que usa la API, se deje de depender de la disponibilidad de los datos ofrecida por Huawei.

- Integración con openHAB. Una ventaja de utilizar esta arquitectura es la integración de los gráficos del inversor solar en la aplicación de gestión del hogar digital, openHAB, como si de un dispositivo IoT más se tratara. De esta manera, el usuario puede consultar todos los datos de su casa en la misma plataforma, evitando tener que visualizar los datos del inversor en una web distinta.

4. Descripción de la solución propuesta

4.1 Introducción

En esta sección de la memoria se comenta detalladamente cada uno de los pasos que se han seguido para lograr desplegar y conectar toda la arquitectura que conforma la solución de este proyecto.

4.2 Despliegue de máquina virtual Raspberry Pi

Como se ha comentado anteriormente, se ha utilizado una máquina virtual de Raspberry Pi, lo que permite realizar pruebas y desarrollos como si se estuviera utilizando el hardware físico, pero con la flexibilidad y el control adicionales que proporciona un entorno virtualizado.

El primer paso consiste en simular la Raspberry Pi física que se encontraría en el Hogar Digital de la Escuela. Se ha realizado un *backup* de la configuración de openHAB de la escuela y después se ha recuperado dicha información en la máquina virtual, para obtener así la interfaz que se tiene en la Escuela y poder añadir en ella la información del inversor solar.

Se ha utilizado una máquina virtual con sistema operativo Raspberry Pi OS, basado en Debian, una distribución de Linux. A continuación se muestran las indicaciones para desplegar una máquina virtual con ese sistema operativo desde cero.

4.2.1 Descarga Raspberry Pi OS

Es necesario descargar la máquina virtual del sitio oficial de Raspberry. [10]

En la página de descarga se encuentran varias opciones. Una de ellas es descargar el sistema operativo para instalarlo en una Raspberry Pi física. Esta opción de descarga se denomina “Raspberry Pi Imager”. Esto no es lo que se busca, puesto que se quiere una imagen del sistema operativo completo (extensión .iso) para después ejecutarlo en la aplicación de virtualización. Esta opción aparece más abajo en la página y se denomina “Raspberry Pi Desktop for PC and Mac”.

Al pulsar el botón “Download Raspberry Pi Desktop” se carga otra página en la que se pueden observar las características del sistema operativo.

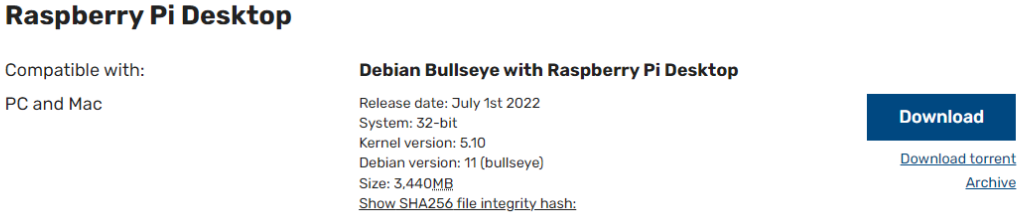

Ilustración 10 Información Raspberry Pi OS

Esta información es relevante para la configuración posterior de la máquina virtual. Como se puede observar se trata de la versión 11 de Debian de 32 bits.

4.2.2 Crear una máquina virtual e instalar Raspberry Pi OS en ella

Para la creación de la máquina virtual se deben seguir los siguientes pasos:

1. En la pantalla principal de VMWare se selecciona “Nueva máquina virtual” y a continuación la opción “Installer disk image file (iso)”, proporcionando la ubicación del archivo con extensión .iso donde está el sistema operativo descargado en el apartado anterior.
2. Se elige el sistema operativo. Aquí se debe incluir la información que se visualizó en la pantalla de descarga de Raspberry: sistema operativo Linux y versión Debian 11.x.
3. Se le da nombre a la máquina virtual y se selecciona la ubicación en la que se quiere guardar.
4. Se elige la cantidad de almacenamiento que requiere la máquina virtual. Se dejan las dos opciones que vienen por defecto: 20GB de almacenamiento y dividir el disco virtual en diferentes archivos.
5. En la siguiente pantalla se puede personalizar el hardware. Se eligen 8GB de RAM, ya que es la configuración de memoria que tiene la Raspberry Pi 5 de la escuela. En el apartado de adaptador de red se selecciona la opción “puenteado” (en inglés *bridged*), para que la máquina virtual esté conectada directamente al adaptador y tenga un IP propia en vez de tener la de la máquina anfitriona. Se cierra la pestaña actual y se pulsa “Finish”.

La máquina virtual debería estar creada en este momento. Cuando arranque lo hará del DVD que se configuró para encontrar el iso con el sistema operativo.

Se procede con la instalación del sistema operativo. Los pasos son los siguientes:

1. Lanzar la máquina virtual. Lo primero que aparece es un menú con diferentes opciones, se debe elegir “Graphic install”:

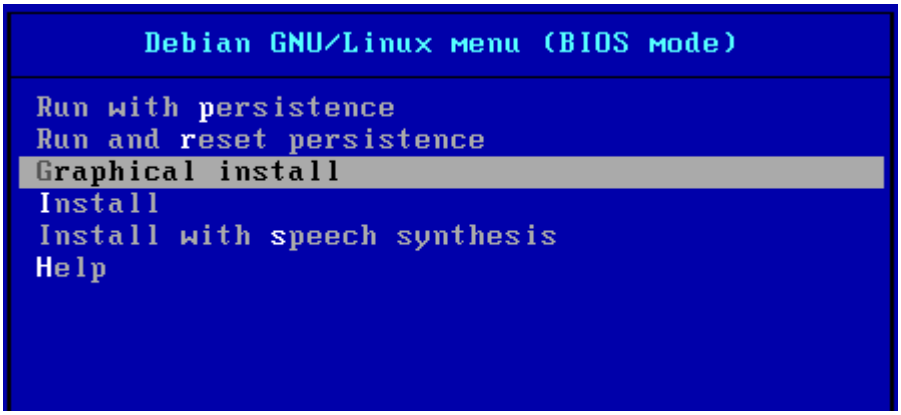

Ilustración 11 BIOS de Raspberry Pi OS

2. Se selecciona el idioma español.
3. Se selecciona utilizar el disco entero para las particiones. En la siguiente pantalla debe aparecer el disco virtual creado para esta máquina.
4. Se eligen todos los archivos en una misma partición. En la siguiente pantalla se pulsa "Continuar" para terminar de configurar las particiones y escribir los cambios en el disco.
5. Después de copiar los archivos e instalar el sistema operativo, pide instalar el GRUB. Se pide un disco duro, debe aparecer el disco virtual de esta máquina. Se elige ese disco y se pulsa "Continuar".
6. Si se pide la ubicación para instalar el GRUB, se le debe dar la partición principal. En este caso es "/dev/sda"

Después pone que la instalación ha terminado, se pulsa continuar y tras unos segundos, la máquina se reinicia y arranca ya con el sistema instalado. Se observa en la primera página que tiene una IP de la red local de la casa:

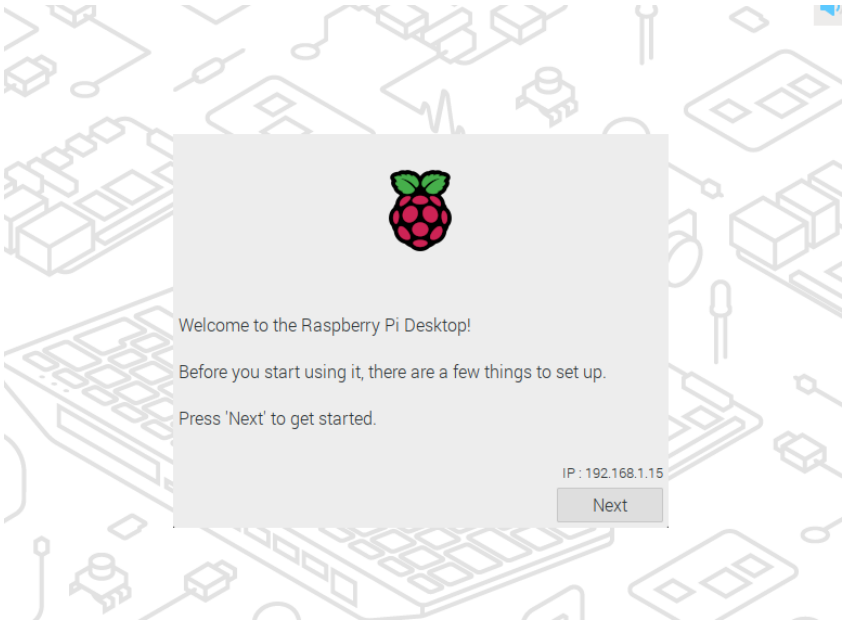

Ilustración 12 Pantalla de inicio de Raspberry Pi

Se seleccionan las configuraciones iniciales, incluyendo ubicación, idioma, nombre de usuario y contraseña. Después se ofrece la opción de actualizar los paquetes. Se haya elegido actualizar o no, es necesario reiniciar la máquina para comenzar a utilizarla.

En este proyecto se ha decido utilizar el nombre de usuario "pi", que se puede observar a lo largo de la memoria.

4.3 Conectividad Raspberry Pi

Cabe recordar que, en un entorno real, esta máquina virtual sería una Raspberry Pi física, por lo que hay dotarla de acceso remoto. El motivo de esto es que posiblemente la Raspberry Pi

se encuentre en un lugar poco accesible donde el usuario no pueda conectar una pantalla, un ratón y un teclado para manejar la Raspberry.

Para conectar la Raspberry con el ordenador del usuario se han utilizado las tecnologías expuestas en los siguientes apartados.

4.3.1 VNC

Esta es una aplicación que sigue el modelo cliente-servidor y que se utiliza para enviar la interfaz gráfica de una máquina de forma remota. En este caso se va a utilizar para enviar la interfaz gráfica de la Raspberry Pi al ordenador del usuario, para poder manejar la Raspberry Pi de forma remota.

4.3.1.1 VNC Server

Esta es la aplicación servidor que se ejecuta en la Raspberry Pi para compartir su interfaz de usuario. Los pasos para su instalación se detallan a continuación. [27]

1. Actualizar la lista de paquetes:

```
sudo apt update
```

2. Instalar la aplicación, en este caso "tightvncserver", pero existen multitud de posibilidades:

```
sudo apt install xfce4 xfce4-goodies tightvncserver
```

3. Se lanza el servicio:

```
vncserver
```

   En este momento se pide configurar una contraseña. Esta contraseña es importante ya que se pedirá posteriormente cuando desde la máquina Windows se quiera acceder a la Raspberry Pi.

4. Se edita el fichero de arranque de VNC:

```
sudo nano ~/.vnc/xtartup
```

5. Se deben añadir las siguientes líneas al fichero:

```
#!/bin/bash
xrdb $HOME/.Xresources
startxfce4 &
```

Ilustración 13 Configuración VNC Server

Con estas líneas se consigue que cuando se inicie una sesión de VNC, se carguen las configuraciones personalizadas y se inicie un entorno de escritorio XFCE para que se pueda interactuar con la Raspberry Pi de forma gráfica.

6. Se otorga permiso de ejecución al archivo "xstartup":

```
sudo chmod +x ~/.vnc/xstartup
```

7. Y por último se lanza el servicio:

```
vncserver :1
```

El término ":1" significa que se ejecuta el servicio para la pantalla 1. Esto se debe a que se pueden ejecutar diferentes instancias de VNC Server.

Es importante tener en cuenta que, aunque la Raspberry Pi va a tener un uso ininterrumpido, si por cualquier motivo se reiniciara, se debe volver a lanzar el servicio de VNC Server con el último comando citado.

4.3.1.2 VNC Viewer

Por otro lado, hay que instalar la aplicación cliente VNC Viewer en la máquina Windows.

De las muchas aplicaciones cliente posibles, se ha elegido descargar de la página oficial de RealVNC el archivo ejecutable. [15]

Se ejecuta el archivo, y se procede con la instalación del programa. En este caso, no hay que variar ningún parámetro de la instalación, se deja todo por defecto y se finaliza el proceso.

Una vez dentro de la aplicación se debe observar una pantalla como esta:

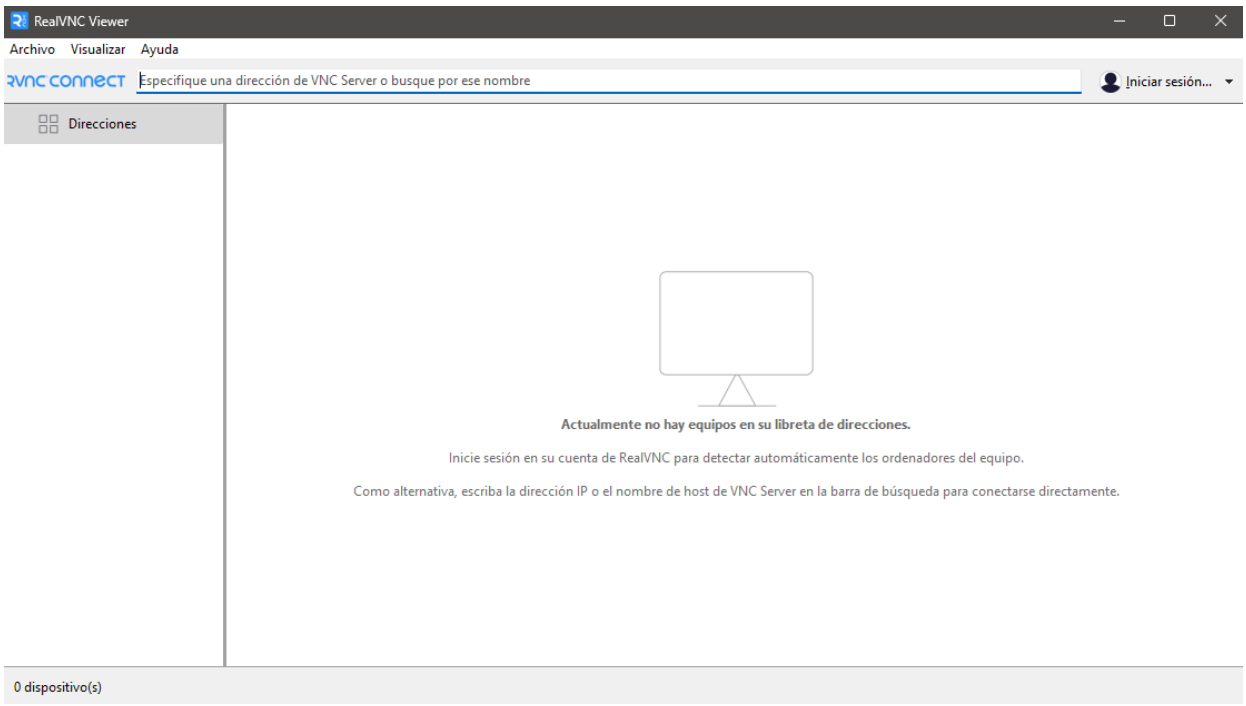

Ilustración 14 Página principal VNC Viewer

En la barra superior se debe poner la IP y puerto para conectarse al VNC Server de la Raspberry Pi. En este proyecto, sería 192.168.1.150:5901. VNC por defecto utiliza los puertos 590X, el valor de la X depende de la instancia que se esté ejecutando. Es decir, como en el apartado anterior se ha ejecutado el comando:

```
vncserver :1
```

El puerto es el 5901.

Una vez introducidos IP y puerto, se pide la contraseña que se estableció en VNC Server, y si es correcta, se debe observar la interfaz de usuario de la Raspberry Pi:

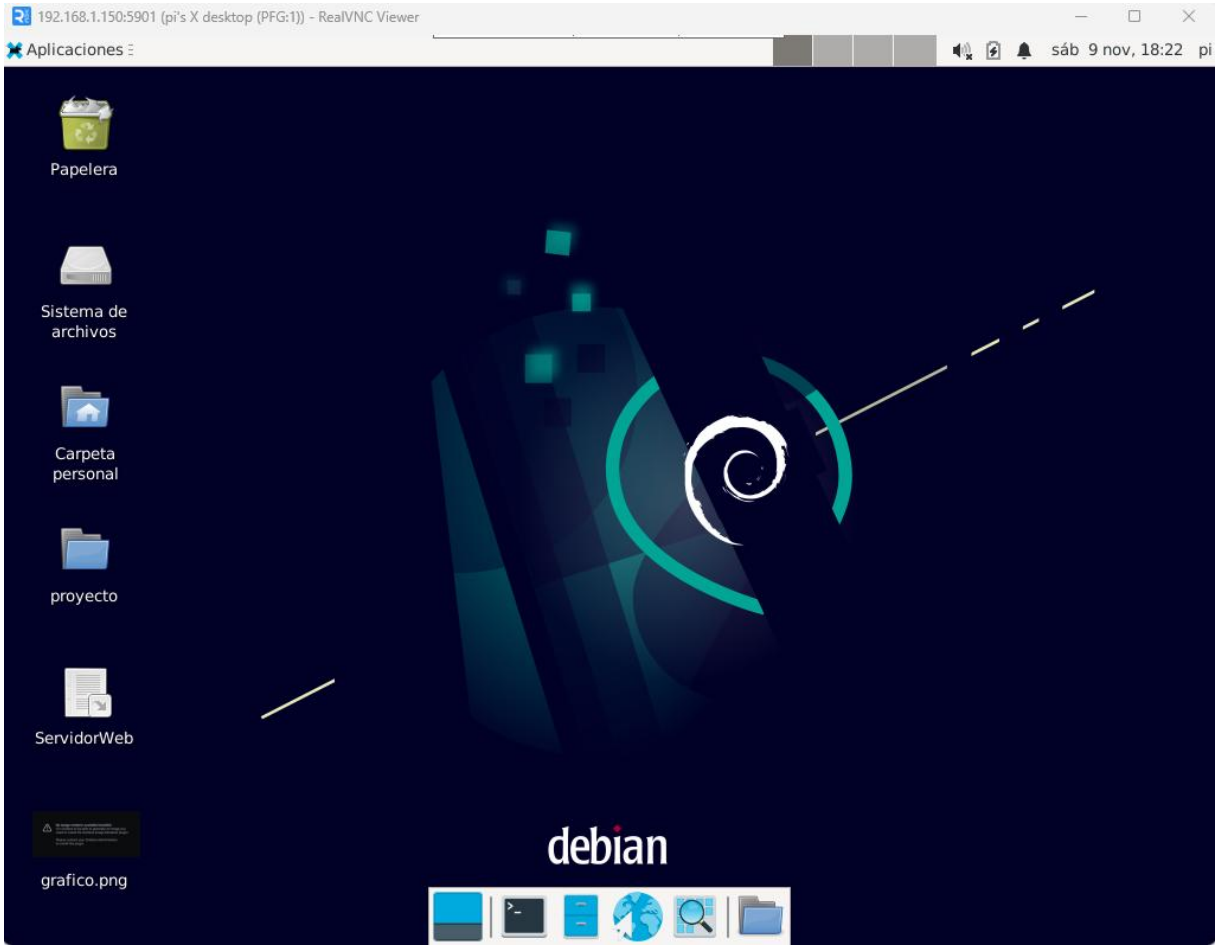

Ilustración 15 Interfaz gráfica generada por VNC

Aunque es distinta de la interfaz gráfica de la Raspberry Pi, se puede comprobar que se está dentro de la Raspberry Pi verificando el sistema de carpetas y visualizando los archivos que se hayan creado.

4.3.2 SSH

En la Raspberry Pi no es necesario instalar SSH pues ya viene incluido con el sistema operativo. Solo es necesario habilitarlo. Para ello se navega hasta el menú de configuración, situado en Preferencias/Configuración:

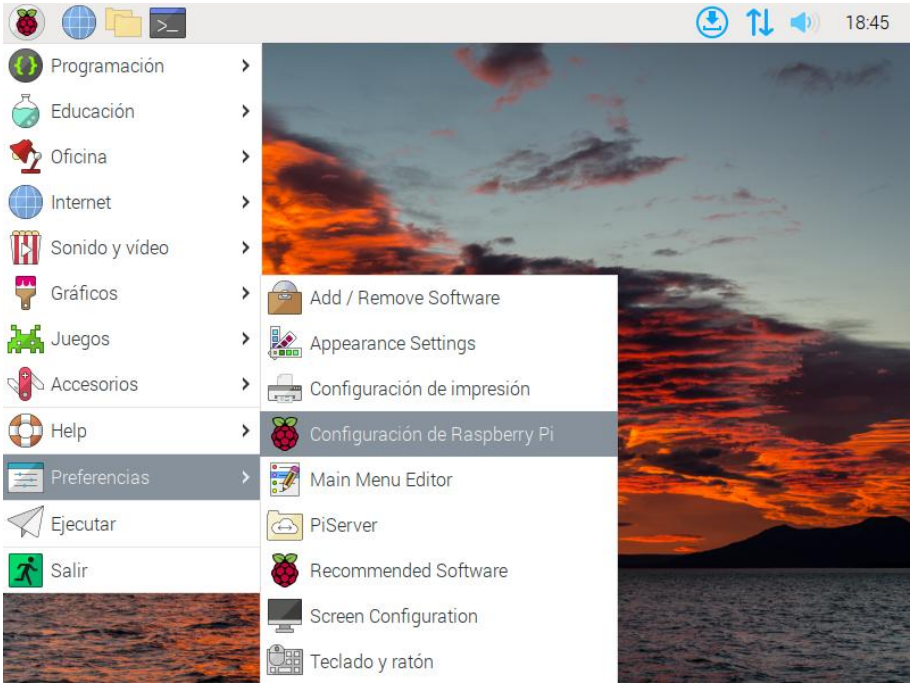

Ilustración 16 Configuración de la Raspberry Pi

En la pantalla de interfaces, se activa SSH.

También se pude hacer mediante la línea de comandos:

```
sudo raspi-config
```

Este comando abrirá una herramienta de configuración del sistema, en el que se observa una pestaña de interfaces, donde se encuentra la opción de SSH.

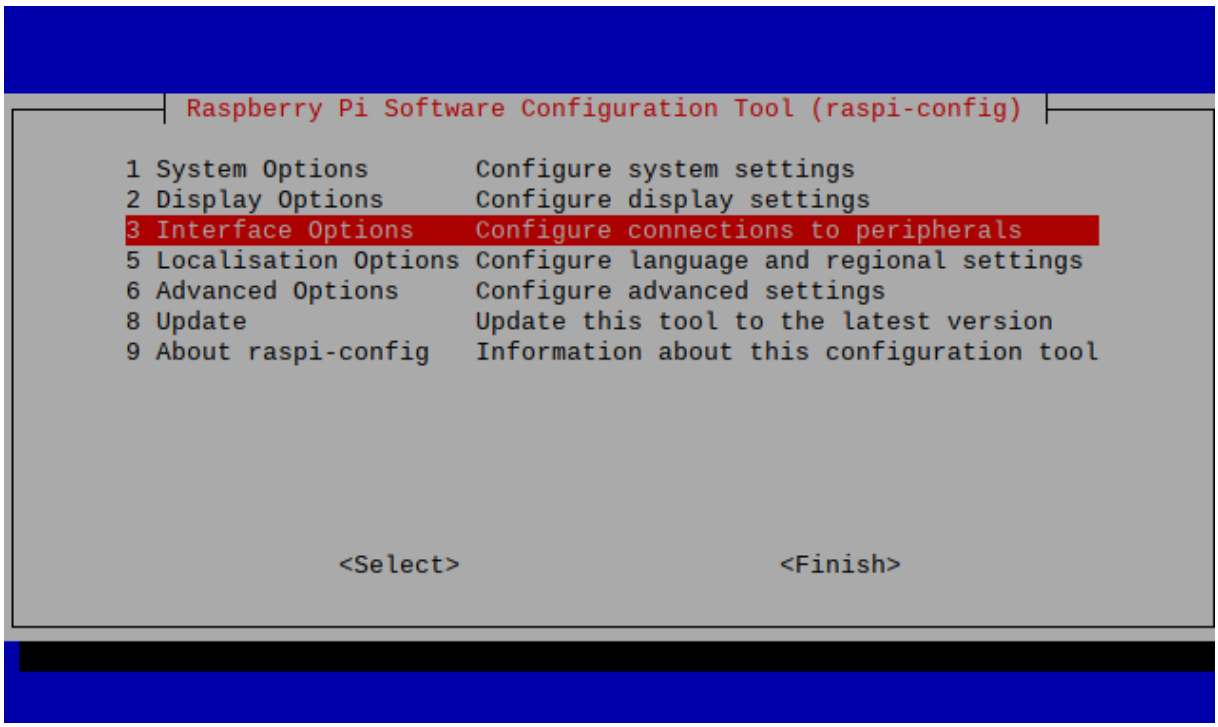

Ilustración 17 Activación SSH con comandos

Para comprobar el funcionamiento se necesita conocer la IP de la Rasberry Pi, con el siguiente comando:

```
ifconfig
```

Con este comando se debe obtener una salida parecida a la siguiente imagen:

```
pi@PFG:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.15  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::c3a7:48dc:795a:9f50  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:f0:92:7f  txqueuelen 1000  (Ethernet)
        RX packets 3432  bytes 2989210 (2.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2965  bytes 382051 (373.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 27  bytes 3720 (3.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 27  bytes 3720 (3.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 18 Configuración de red Raspberry Pi

Se observa que la dirección IP dentro de la red local es la 192.168.1.15 y la dirección de Loopback es la 127.0.0.1.

Ahora en la máquina anfitriona, Windows, se abre una interfaz de comandos y se escribe el siguiente comando:

```
ssh pi@192.168.1.15
```

El término “ssh” hace referencia al protocolo, “pi” es el nombre de usuario y “192.168.1.15” es la IP de la Raspberry Pi obtenida en el paso anterior.

La contraseña es la que se introdujo la primera vez que se arrancó la máquina, después de configurar el idioma y la zona horaria.

Una vez introducida la contraseña correctamente, el *prompt* de la máquina anfitriona debe cambiar para mostrar el de la Raspberry Pi:

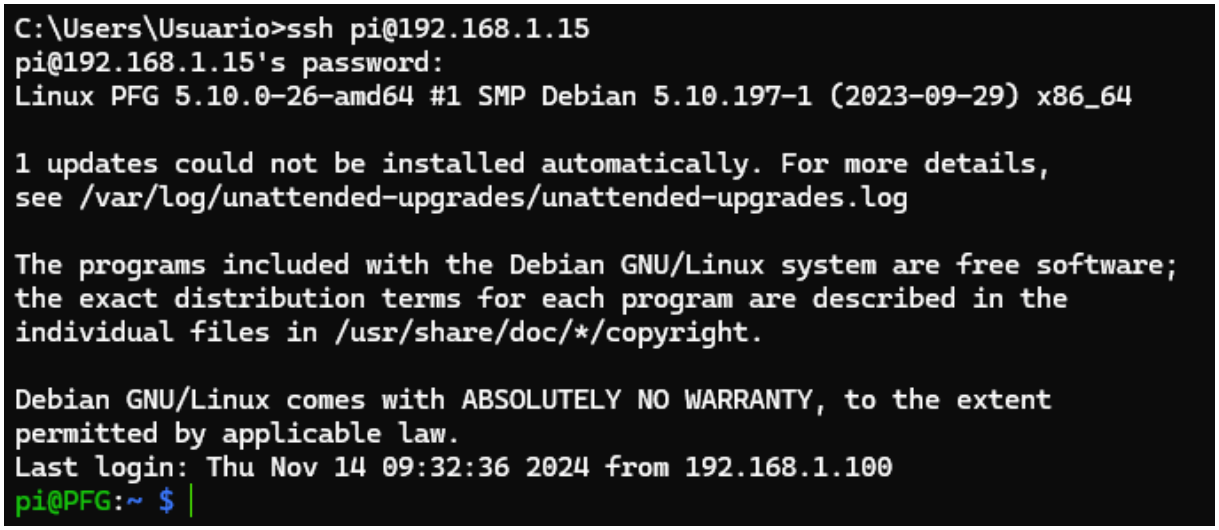

Ilustración 19 Conexión exitosa por SSH

Como se observa en la imagen se ha introducido el comando para la conexión por SSH, se ha escrito la contraseña correcta y el *prompt* se ha cambiado indicando que la conexión se ha establecido y el sistema está esperando para recibir comandos.

Como se ha comentado antes, “pi” es el nombre de usuario. Además, ahora aparece otro nombre después de “@”, en este caso “PFG”. Este nombre es configurable por el usuario en Preferencias/Configuración de Raspberry Pi/Sistema/Hostname.

En este momento se puede comprobar que los comandos se ejecutan sobre las Raspberry Pi, por ejemplo, ejecutando de nuevo el comando de configuración de red:

```
ifconfig
```

La salida debe ser la misma que la que se obtuvo cuando se ejecutó directamente sobre la Raspberry Pi.

Debido a que la conexión se debe hacer introduciendo manualmente la IP de la Raspberry Pi, es una buena práctica configurar la máquina para que siempre tenga la misma IP. De no ser así, cada vez que se quisiera conectar habría que averiguar la IP que el DHCP del *router* de la casa le haya otorgado. Y esto en el entorno real no sería posible al tener la Raspberry Pi física fuera del alcance del usuario.

Por ello, se configura la máquina para que siempre obtenga la misma dirección IP al arrancar. Se puede hacer de dos maneras distintas:

- Configurar el *router* para que reserve una dirección para la Raspberry Pi. Esto se realiza cambiando la configuración del *router*, y asignando la dirección IP que se quiera para la Raspberry Pi con la MAC de esta.
- Configurar la Raspberry Pi con una dirección estática. Para que esto no suponga un problema, es una buena práctica asignar una IP a la Raspberry Pi que se encuentre fuera del rango de direcciones asignadas por DHCP. Esta información se puede consultar y modificar en la configuración del *router*.

Debido a que el funcionamiento esperado de la Raspberry Pi es ininterrumpido, no se va a apagar o reiniciar, y que la configuración del *router* es diferente para cada fabricante, en este proyecto se ha utilizado la segunda opción.

Para cambiar la configuración de la Raspberry Pi se debe introducir el siguiente comando para editar el archivo de configuración de red:

```
sudo nano /etc/dhcpcd.conf
```

Se deben añadir las siguientes líneas al archivo:

```
interface eth0
static ip_address=192.168.1.150/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Ilustración 20 Configuración IP estática

Estas líneas informan a la Raspberry de que en la interfaz “eth 0” se tiene una IP estática, con valor 192.168.1.150 y con máscara 24. La dirección 192.168.1.1 es la del *router*, por lo que es donde se deben enviar los datos para que salgan a internet. En otras palabras, la puerta de enlace predeterminada. Finalmente se declara el servidor de nombre de dominio (DNS), con valor 192.160.1.1 que coincide con la puerta de enlace del router. Esto se hace así para utilizar el DNS que tenga configurado el *router.* Para conocer qué información se debe poner en el servidor DNS, se puede consultar en el PC mediante el siguiente comando:

```
ipconfig /all
```

Este comando muestra la información completa de la conexión de red del PC. En el adaptador que se esté usando se debe buscar la información "DNS Servers". Se guarda el archivo y se reinicia la Raspberry Pi. Los cambios deben haber sido aplicados. Se pueden comprobar los cambios reiniciando la Raspberry Pi y verificando la nueva dirección IP.

4.3.3 Samba

Para poder utilizar el protocolo SMB en un dispositivo Linux, es necesario instalar algún software que lo implemente. En este proyecto se ha utilizado Samba.

Las instrucciones de instalación se detallan a continuación: [28]

1. Se actualiza la lista de paquetes disponibles, revisando los repositorios configurados y descargando la última lista de versiones y paquetes posibles de cada uno:

```
sudo apt update
```

2. Se ejecuta el siguiente comando para instalar Samba y los archivos binarios necesarios para configurar y ejecutar Samba:

```
sudo apt-get install samba samba-common-bin
```

3. Se modifica el archivo de configuración, por ejemplo, con el editor nano ejecutando:

```
sudo nano /etc/samba/smb.conf
```

4. Se incluyen en el archivo las siguientes líneas:

```
[pi]
  comment=pi user
  path=/home/pi
  browseable=Yes
  writeable=Yes
  only guest=no
  public=no
  create mask=0777
  directory mask=0777
```

Ilustración 21 Directorio "pi" compartido por Samba

El directorio "pi" es en el que se van a alojar todos los archivos fuente desarrollados en el proyecto.

5. Para comprobar la correcta instalación y configuración se ejecuta el siguiente comando:

```
testparm
```

Se debe observar algo parecido a siguiente ilustración:

```
pi@PFG:~ $ testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed
Server role: ROLE_STANDALONE
```

Ilustración 22 Samba configurado correctamente

Para conectarse, se debe escribir "\\192.168.1.150" en la barra de direcciones del explorador de archivos de la máquina Windows.

Para acceder se pide usuario y contraseña, pero no se ha configurado ninguna, por lo que no se puede acceder. Para configurar un usuario y una contraseña en Samba, se ejecuta el siguiente comando en la Raspberry Pi:

```
sudo smbpasswd -a pi
```

En este caso se ha decidido que el usuario sea "pi". Justo después del comando, el sistema pide introducir una nueva contraseña, y después repetirla. Una vez finalizado este proceso, se tendrá acceso a la Raspberry desde la máquina Windows con las credenciales configuradas. De esta forma, los archivos de la Raspberry Pi que se hayan compartido con Samba serán accesibles desde la red.

4.4 InfluxDB

InfluXB es la tecnología que se ha elegido como gestor de base datos debido al tratamiento temporal que realiza con los datos y su integración con Grafana.

Los pasos de instalación se detallan a continuación: [29]

1. Se verifica la última versión de los paquetes:

```
sudo apt update
sudo apt upgrade -y
```

2. Se instalan dependencias necesarias:

```
sudo apt install -y apt-transport-https ca-certificates curl
```

3. Se agrega el repositorio oficial de InfluxDB:

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
```

```
echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee
/etc/apt/sources.list.d/influxdata.list
```

4. Se instala InfluxDB:

```
sudo apt update
sudo apt install influxdb
sudo apt install influxdb-client
```

5. Se habilita el servicio para que arranque automáticamente y, después, se arranca:

```
sudo systemctl enable influxdb
sudo systemctl start influxdb
```

6. En este momento, el servicio debe estar instalado y arrancado. Para comprobarlo se ejecuta el siguiente comando para ver su estado:

```
sudo systemctl status influxdb
```

En la siguiente ilustración se muestra un ejemplo de la salida de este comando:

```
pi@PFG:~ $ sudo systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time series database
     Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-11-09 19:48:04 CET; 1h 27min ago
       Docs: man:influxd(1)
   Main PID: 624 (influxd)
      Tasks: 8 (limit: 4915)
     Memory: 204.3M
        CPU: 17.109s
     CGroup: /system.slice/influxdb.service
             └─624 /usr/bin/influxd -config /etc/influxdb/influxdb.conf

nov 09 20:35:19 PFG influxd[624]: ts=2024-11-09T19:35:19.268163Z lvl=info msg="Executing query" log_id=0slZvUrl000 serv>
nov 09 20:35:19 PFG influxd[624]: [httpd] 127.0.0.1 - - [09/Nov/2024:20:35:19 +0100] "GET /query?db=Inversor&epoch=ms&q>
nov 09 20:35:19 PFG influxd[624]: [httpd] 127.0.0.1 - - [09/Nov/2024:20:35:19 +0100] "GET /query?db=Inversor&epoch=ms&q>
nov 09 20:35:19 PFG influxd[624]: ts=2024-11-09T19:35:19.269241Z lvl=info msg="Executing query" log_id=0slZvUrl000 serv>
nov 09 20:35:19 PFG influxd[624]: [httpd] 127.0.0.1 - - [09/Nov/2024:20:35:19 +0100] "GET /query?db=Inversor&epoch=ms&q>
nov 09 20:39:20 PFG influxd[624]: ts=2024-11-09T19:39:20.952125Z lvl=info msg="Cache snapshot (start)" log_id=0slZvUrl0>
nov 09 20:39:21 PFG influxd[624]: ts=2024-11-09T19:39:21.031269Z lvl=info msg="Snapshot for path written" log_id=0slZvU>
nov 09 20:39:21 PFG influxd[624]: ts=2024-11-09T19:39:21.031504Z lvl=info msg="Cache snapshot (end)" log_id=0slZvUrl000>
nov 09 20:48:08 PFG influxd[624]: ts=2024-11-09T19:48:08.954652Z lvl=info msg="Retention policy deletion check (start)" >
nov 09 20:48:08 PFG influxd[624]: ts=2024-11-09T19:48:08.954731Z lvl=info msg="Retention policy deletion check (end)" l>
lines 1-21/21 (END)
```

Ilustración 23 Estado de InfluxDB

Como se observa, el servicio está activo y corriendo (*running*).

Como última comprobación, se ejecuta el comando:

```
influx
```

En la siguiente ilustración se observa cómo se cambia el *prompt*, lo que significa que ahora se está dentro de la consola de InfluxDB:

```
pi@PFG:~ $ influx
Connected to http://localhost:8086 version 1.6.7~rc0
InfluxDB shell version: 1.6.7~rc0
>
```

Ilustración 24 Consola de comandos de InfluxDB

Una vez en la consola de InfluxDB, se puede crear ya la base de datos, en la que posteriormente se alojarán los datos que se recojan mediante la API, con el siguiente comando:

```
CREATE DATABASE Inversor
```

En este proyecto se ha decidido llamar a la base de datos “Inversor”. El nombre no es realmente relevante, se puede elegir cualquiera.

Dentro de esa base de datos “Inversor”, se van a alojar varias tablas o *measuruments.* Estas tablas no se crean manualmente, sino que son creadas por los scripts de Python. Cada vez que se va a escribir en la base de datos, si la tabla indicada existe se añaden los datos y, si no existe, se crea esa nueva tabla en ese momento.

Las tablas de este proyecto son: “horarios”, “diarios”, “mensuales” y “anuales”. Una por cada tipo de dato que se desea almacenar.

En las tablas “anuales”, “mensuales” y “diarios” se encuentran las siguientes columnas o campos:

- Ahorro CO2. Es el ahorro de dióxido de carbono debido a la utilización de la energía generada por la instalación fotovoltaica. Se mide en toneladas. En la base de datos es un dato de tipo *float.*
- Ahorro carbon. Es el ahorro de carbón debido a la utilización de la energía generada por la instalación fotovoltaica. Se mide en toneladas. En la base de datos es un dato de tipo *float.*
- Autoconsumo. Es la energía consumida proveniente del inversor. Se mide en kWh. En la base de datos es un dato de tipo *float.*
- Capacidad instalada. Es la potencia total que la instalación puede generar bajo condiciones óptimas. Se mide en kW. En la base de datos es un dato de tipo *float.*
- Consumo total. Es la suma del autoconsumo más la energía comprada. Se mide en kWh. En la base de datos es un dato de tipo *float.*
- Energía comprada. Se trata de la energía que ha sido necesario comprar cuando se requería más energía que la que el inversor estaba generando en ese momento. Se mide en kWh. En la base de datos es un dato de tipo *float.*
- Energía vendida. Se trata de la energía sobrante. Cuando el inversor ha generado más energía de la que se necesitaba en ese momento, el sobrante se vende a la red eléctrica. Se mide en kWh. En la base de datos es un dato de tipo *float.*
- Ingresos. Ingresos generados debido a la energía vendida. Se mide en €. En la base de datos es un dato de tipo *float.*
- time. No es una columna o campo como el resto, sino que es el valor temporal de las medidas. En la base de datos se representa con un número entero. Dicho número entero es el valor en nanosegundos desde la época Unix. Es decir, es el tiempo transcurrido desde el 1 de enero de 1970 mostrado en nanosegundos. [30]

En la tabla “horarios” únicamente se encuentran los campos “Autoconsumo”, “Energía generada”, “Energía vendida” e “Ingresos”, puesto que la API solo proporciona esos datos cuando se realizan peticiones con carácter horario.

4.5 Scripts Python

Python es el lenguaje elegido para codificar los scripts que piden los datos a Huawei utilizando la API, los transforman para poder hacer un mejor tratamiento y finalmente los almacenan en la base de datos InfluxDB. Al estar usando Raspberry Pi OS, Python viene directamente instalado con el sistema operativo. Es necesario tener instalada la versión 3.8 o superior de

Python para que todos los scripts funcionen. Para comprobar la versión se ejecuta el siguiente comando:

```
python3 --version
```

Los scripts se han situado en la carpeta “Proyecto” en el escritorio de la Raspberry Pi. Es importante conocer esta ubicación ya que es necesaria a la hora de crear las reglas y decir la ruta de los scripts que éstas deben lanzar.

En la siguiente ilustración se muestra el diagrama de clases de Python que se ha desarrollado en la solución:

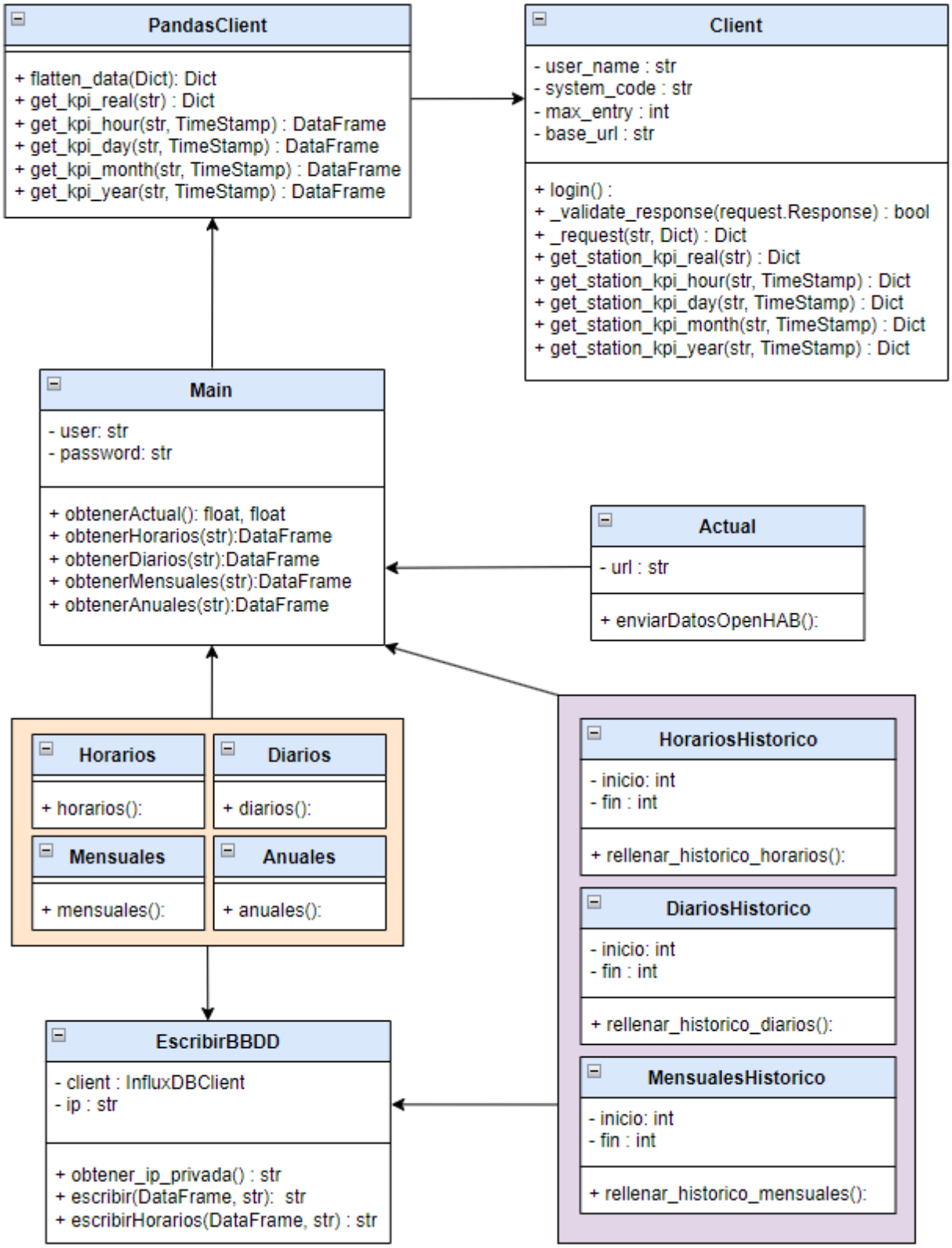

Ilustración 25 **Diagrama de clases Python**

Para el desarrollo de la lógica de programación, se ha tomado el proyecto FusionSolar de EnergieID de GitHub como base. [31]

Dicho proyecto define la clase `Client.py` del diagrama, donde se encuentran las funciones que lanzan la petición a la API, generando una petición HTTP con método POST para pedir los

datos. También sugiere una implementación de la clase `PandasClient.py`, pero en este proyecto no se ha utilizado dicha sugerencia, sino que se ha implementado una solución propia.

Solo `Client.py` y `PandasClient.py` son catalogadas como clase, debido a que son las únicas definiciones que utilizan el paradigma de programación orientada a objetos. El resto simplemente agrupan funciones y variables, pero no definen objetos.

Dentro de la caja naranja se encuentras los scripts `Horarios.py`, `Diarios.py`, `Mensuales.py` y `Anuales.py`. Se recogen dentro de la misma caja ya que se comportan de igual manera, piden los datos horarios, diarios, mensuales o anuales respectivamente dependiendo de la fecha en el momento de la ejecución. Estos scripts son lanzados desde openHAB mediante reglas, explicado en el apartado 4.7.4 Reglas. De esta manera se consigue automatizar y garantizar que los datos se encuentran siempre actualizados.

Por otro lado, los scripts de la caja morada están diseñados para rellenar la base de datos con todos los datos desde el momento de la instalación del inversor. Estos scripts reciben como argumento de programa el año desde el que se quiere comenzar a recoger y el año hasta el que se quiere hacerlo: inicio y fin respectivamente. Estos scripts deben ser lanzados desde la propia Raspberry Pi por la consola de comandos. La ejecución de estos scripts se explica en el anexo A.1 Manual de usuario. Cabe destacar la ausencia de un script `AnualesHistorico.py`. El motivo de esta ausencia se debe a que la API devuelve todos los registros anuales cuando se solicita uno de ellos, por tanto, con el script `Anuales` de la caja naranja ya se satisface la necesidad de obtener los registros históricos de datos anuales.

Todos los scripts comentados anteriormente utilizan el script `EscribirBBDD.py` para almacenar los datos en la base de datos.

El script `Actual.py` se encuentra separado porque, a pesar de hacer peticiones como los demás scripts comentados, el tratamiento de los datos es distinto. Estos datos no son almacenados en la base de datos puesto que son registros en tiempo real. Estos datos son enviados directamente a openHAB para su visualización.

Todos los scripts de la caja naranja, la caja morada y el script `Actual.py`, invocan a las funciones contenidas en el script `Main.py`, que invocan a las funciones de la clase `PandasClient.py` y que a su vez invocan a las funciones de la clase `Client.py`, siendo estas las que finalmente lanzan la petición a la API.

A continuación se explica más detalladamente cada uno de los elementos del diagrama.

4.5.1 Clase Client.py

La clase `Client.py` es la encargada de enviar las peticiones a la API y recibir sus respuestas. Para ello define las siguientes variables:

- `user_name`. Es el nombre de usuario dado de alta en la cuenta de Huawei para utilizar la API, comentado en el apartado 2.1.2 Northbound Open API.
- `system_code`. Contraseña de la cuenta.
- `max_entry`. Número máximo de intentos de autenticación contra la API.
- `base_url`. Parte común de la URL a la que después se le concatenará una cadena de caracteres según la petición que se desee hacer. Esto es porque todos los *items* de openHAB comparten una parte de la de la URL, puesto que lo único que cambia es el final, donde se introduce el nombre del *item.* Por ejemplo, para acceder al *item* llamado "A", la URL sería: http://ip_openhab:8080/rest/items/A, y para un *item* "B" la URL sería: http://ip_openhab:8080/rest/items/B, por lo que se observa que la parte inicial de la URL es compartida.

Y los siguientes métodos:

- `login()`. Se invoca cuando se crea una nueva instancia de esta clase y lanza una petición contra la API en la que se envían el nombre de usuario y contraseña para autenticarse. En este procedimiento los datos viajan de manera segura ya que la petición se realiza mediante HTTPS.
- `_validate_response(request.Response):bool`. Este método se invoca cuando se recibe una respuesta de la API. Se pasa como argumento la petición recibida y el método verifica si la respuesta contiene alguno de los códigos de error conocidos. Si no encuentra ninguno, devuelve *true* haciendo saber que la petición ha sido exitosa. En caso contrario, el valor devuelto es *false*.
- `_request(str, Dict):Dict`. Este es el método que envía la petición a la API. Como parámetros tiene una cadena de caracteres que es el final de la URL, la parte a concatenar a la variable base_url, y un diccionario que son los códigos de estación y las fechas en las que se quieren conocer los datos. El valor devuelto es un diccionario generado a través del JSON recibido de la API con los datos solicitados.
- `get_station_kpi_real(str):Dict`. Este método recibe el código de estación como parámetro e invoca a `_request()` con el método de la API deseado y el código de estación para obtener los datos en tiempo real. Después, devuelve el diccionario obtenido.
- `get_station_kpi_hour(str, TimeStamp):Dict`. Este método recibe el código de estación y la fecha, transforma la fecha a un *timestamp* e invoca a `_request()` con el método de la API deseado, el código de estación y la nueva fecha. Después, devuelve el diccionario obtenido con los datos horarios.
- `get_station_kpi_day(str, TimeStamp):Dict`. Este método recibe el código de estación y la fecha, transforma la fecha a un *timestamp* e invoca a `_request()` con el método de la API deseado, el código de estación y la nueva fecha. Después, devuelve el diccionario obtenido con los datos diarios.
- `get_station_kpi_month(str, TimeStamp):Dict`. Este método recibe el código de estación y la fecha, transforma la fecha a un *timestamp* e invoca a

`_request()` con el método de la API deseado, el código de estación y la nueva fecha. Después, devuelve el diccionario obtenido con los datos mensuales.

- `get_station_kpi_year(str, TimeStamp):Dict`. Este método recibe el código de estación y la fecha, transforma la fecha a un *timestamp* e invoca a `_request()` con el método de la API deseado, el código de estación y la nueva fecha. Después, devuelve el diccionario obtenido con los datos anuales.

4.5.2 Clase PandasClient.py

La clase `PandasClient.py` hereda de la clase `Client.py` y define los siguientes métodos:

- `flatten_data(Dict):Dict`. Este método se ha diseñado para modificar el formato de los datos, concretamente para hacer que la columna de tiempos sea mostrada como una columna más y facilitar el procesado de datos. Recibe un diccionario con los datos y devuelve un nuevo diccionario con la columna temporal modificada.
- `get_kpi_real(str):Dict`. Recibe el código de estación e invoca a `get_station_kpi_real()`. Devuelve el diccionario obtenido. Funciona como una función intermedia que no aporta funcionalidad, pero se ha decidido codificar así para unificar el criterio con los otros tipos de dato (anual, mensual, diario y horario).
- `get_kpi_hour(str, TimeStamp):DataFrame`. Recibe el código de estación y la fecha. Invoca a `get_station_kpi_hour()` y con el diccionario que le devuelve, crea un *DataFrame* de Pandas. Se modifica la columna de la fecha del *DataFrame* para hacerla legible y devuelve dicho *Dataframe*.
- `get_kpi_day(str, TimeStamp):DataFrame`. Recibe el código de estación y la fecha. Invoca a `get_station_kpi_day()` y con el diccionario que le devuelve, crea un *DataFrame* de Pandas. Se modifica la columna de la fecha del *DataFrame* para hacerla legible. En este punto se corrige el desfase temporal de un día incluido por la API. Se devuelve dicho *Dataframe*.
- `get_kpi_month(str, TimeStamp):DataFrame`. Recibe el código de estación y la fecha. Invoca a `get_station_kpi_month()` y con el diccionario que le devuelve, crea un *DataFrame* de Pandas. Se modifica la columna de la fecha del *DataFrame* para hacerla legible. En este punto se corrige el desfase temporal de un mes incluido por la API. Se devuelve dicho *Dataframe*.
- `get_kpi_year(str, TimeStamp):DataFrame`. Recibe el código de estación y la fecha. Invoca a `get_station_kpi_year()` y con el diccionario que le devuelve, crea un *DataFrame* de Pandas. Se modifica la columna de la fecha del *DataFrame* para hacerla legible. En este punto se corrige el desfase temporal de un año incluido por la API. Se devuelve dicho *Dataframe*.

4.5.3 Script Main.py

En este script se definen las siguientes variables:

- `user`. Nombre de usuario de la cuenta de la API.
- `password`. Contraseña de la cuenta de la API.

En este punto el usuario debe introducir sus credenciales, que serán pasadas a la clase `Client` al crear la instancia.

Se definen además las siguientes funciones:

- `obtenerActual():float, float`. Esta función invoca a `get_kpi_real()` pasando como argumento el código de estación. Almacena en un diccionario el valor devuelto y lo descompone en variables, para finalmente devolver solo las de interés, en este caso la energía generada y los ingresos del día.
- `obtenerHorarios(str):DataFrame`. Esta función recibe la fecha por parámetro y la convierte en un *timestamp* de la zona horaria deseada. Invoca a la función `get_kpi_hour()` con el código de estación y la fecha con su nuevo formato para finalmente devolver el *DataFrame* obtenido.
- `obtenerDiarios(str):DataFrame`. Esta función recibe la fecha por parámetro y la convierte en un *timestamp* de la zona horaria deseada. Invoca a la función `get_kpi_day()` con el código de estación y la fecha con su nuevo formato para finalmente devolver el *DataFrame* obtenido.
- `obtenerMensuales(str):DataFrame`. Esta función recibe la fecha por parámetro y la convierte en un *timestamp* de la zona horaria deseada. Invoca a la función `get_kpi_month()` con el código de estación y la fecha con su nuevo formato para finalmente devolver el *DataFrame* obtenido.
- `obtenerAnuales(str):DataFrame`. Esta función recibe la fecha por parámetro y la convierte en un *timestamp* de la zona horaria deseada. Invoca a la función `get_kpi_year()` con el código de estación y la fecha con su nuevo formato para finalmente devolver el *DataFrame* obtenido.

4.5.4 Script Actual.py

Este script se utiliza para conocer los datos del inversor en tiempo real. Es lanzado cada cinco minutos mediante la regla `actual.rules` de openHAB. Se define la siguiente variable:

- `url`. Cadena de caracteres recibida como argumento de programa. Esta es la URL base a la que se deben enviar las actualizaciones de los *items* de openHAB.

Y la siguiente función:

- `enviarDatosOpenHAB()`. Esta función invoca a `obtenerActual()`, almacena las dos variables devueltas y genera dos peticiones que envía a openHAB, una con cada variable. Muestra un mensaje si el envío ha sido exitoso o si se ha producido un error, y en ese caso, informa del código de error.

4.5.5 Script Horarios.py

Este script es lanzado todos los días a las 23:45 mediante la regla `horarios.rules` de openHAB.

Este script se utiliza para recoger los 24 datos horarios del día, uno por cada hora.

Se define la función `horarios()`, que invoca a `obtenerHorarios()` con la fecha actual en formato *aaaammdd* y como cadena de caracteres. Con el *DataFrame* obtenido invoca a la función `escribirHorarios()`, a la que además le pasa la tabla de la base de datos en la que se tienen que escribir los datos del *DataFrame*. Muestra un mensaje informando de que se han guardado los datos para esa fecha.

4.5.6 Script Diarios.py

Este script es lanzado el último día de cada mes a las 23:45 mediante la regla `diarios.rules` de openHAB.

Este script se utiliza para recoger los datos diarios del mes, uno por cada día del mes en curso.

Se define la función `diarios()`, que invoca a `obtenerDiarios()` con la fecha actual en formato *aaaammdd* y como cadena de caracteres. Con el *DataFrame* obtenido invoca a la función `escribir()`, a la que además le pasa la tabla de la base de datos en la que se tienen que escribir los datos del *DataFrame*. Muestra un mensaje informando de que se han guardado los datos para esa fecha.

4.5.7 Script Mensuales.py

Este script es lanzado cada 31 de diciembre a las 23:45 mediante la regla `mensuales.rules` de openHAB.

Este script se utiliza para para recoger los 12 datos mensuales del año, uno por cada mes.

Se define la función `mensuales()`, que invoca a `obtenerMensuales()` con la fecha actual en formato *aaaammdd* y como cadena de caracteres. Con el *DataFrame* obtenido invoca a la función `escribir()`, a la que además le pasa la tabla de la base de datos en la que se tienen que escribir los datos del *DataFrame*. Muestra un mensaje informando de que se han guardado los datos para esa fecha.

4.5.8 Script Anuales.py

Este script es lanzado cada 31 de diciembre a las 23:45 mediante la regla `anuales.rules` de openHAB.

Este script se utiliza para recoger los datos anuales, uno por cada año desde que se instaló el inversor.

Se define la función `anuales()`, que invoca a `obtenerAnuales()` con la fecha actual en formato *aaammdd* y como cadena de caracteres. Con el *DataFrame* obtenido invoca a la función `escribir()`, a la que además le pasa la tabla de la base de datos en la que se tienen que escribir los datos del *DataFrame*. Muestra un mensaje informando de que se han guardado los datos para esa fecha.

4.5.9 Script HorariosHistorico.py

Este script es lanzado manualmente por el usuario y cumple la función de recopilar todos los datos horarios desde que se instaló el inversor. Recibe dos argumentos de programa, el año desde el que se quiere empezar a recopilar y el año hasta el que se quiere hacerlo. Se define una función, `rellenar_historico_horarios()`, que recorre cada día, cada mes y cada año dentro del rango establecido invocando a la función `obtenerHorarios()` para todos esos días. Después invoca a la función `escribirHorarios()` pasando el *DataFrame* y la tabla en la que se tienen que guardar los datos.

4.5.10 Script DiariosHistorico.py

Este script es lanzado manualmente por el usuario y cumple la función de recopilar todos los datos diarios desde que se instaló el inversor. Recibe dos argumentos de programa, el año desde el que se quiere empezar a recopilar y el año hasta el que se quiere hacerlo. Se define una función, `rellenar_historico_diarios()`, que recorre cada mes y cada año dentro del rango establecido invocando a la función `obtenerDiarios()` para todos esos meses. Después invoca a la función `escribir()` pasando el *DataFrame* y la tabla en la que se tienen que guardar los datos.

4.5.11 Script MensualesHistorico.py

Este script es lanzado manualmente por el usuario y cumple la función de recopilar todos los datos mensuales desde que se instaló el inversor. Recibe dos argumentos de programa, el año desde el que se quiere empezar a recopilar y el año hasta el que se quiere hacerlo. Se define una función, `rellenar_historico_mensuales()`, que recorre cada año dentro del rango establecido invocando a la función `obtenerMensuales()` para todos esos años. Después invoca a la función `escribir()` pasando el *DataFrame* y la tabla en la que se tienen que guardar los datos.

4.5.12 Script EscribirBBDD.py

En este script se proporciona la persistencia de los datos obtenidos de la API. Para ello se definen las siguientes variables:

- `client`. Esta variable aloja una instancia de un objeto de la clase `InfluxDBClient.py`. Este objeto se utiliza para conectar y escribir en la base de datos.

- `ip`. En esta variable se almacena la IP privada de la máquina en la que se está ejecutando el script, la Raspberry Pi, para poder indicar dónde se encuentra la base de datos cuando se conecta con el objeto `client`.

Y las siguientes funciones:

- `obtener_ip_privada()`. Esta función obtiene la IP privada de la máquina en la que se ejecuta el script, la Raspberry Pi, para almacenarla en la variable `ip`.
- `escribir(DataFrame, str):str`. Esta función es invocada desde los scripts `Diarios.py`, `Mensuales.py`, `Anuales.py`, `DiariosHistorico.py` y `MensualesHistorico.py`. Como parámetro recibe el *DataFrame* que debe guardar en la base de datos y la tabla en la que debe hacerlo. Devuelve un mensaje que informa de que la escritura en la base de datos ha sido exitosa.
- `escribirHorarios(DataFrame, str):str`. Esta función es invocada desde los scripts `Horarios.py` y `HorariosHistorico.py`. Como parámetro recibe el *DataFrame* que debe guardar en la base de datos y la tabla en la que debe hacerlo. Devuelve un mensaje que informa de que la escritura en la base de datos ha sido exitosa.

Esta doble función de escribir en la base de datos se debe a que los datos horarios devueltos por la API tienen menos campos que los diarios, mensuales y anuales. En estas funciones se recorren los *DataFrame* para extraer todos sus datos y guardarlos en la base de datos por lo que, como los *DataFrame* de los datos horarios tienen contenidos distintos, se necesita una función distinta.

A continuación se explica un ejemplo de funcionamiento para obtener los datos diarios:

En primer lugar, openHAB lanzará el script `Horarios` cada día por la noche para recoger los datos horarios generados durante el día. Este script, invoca a la función `obtenerHorarios()` del script `Main.py` y le pasa como argumento la fecha actual. Esta función hace una primera transformación de la fecha que recibe, e invoca al método `get_kpi_moth()` de la clase `PandasClient.py`. Como argumentos le pasa la fecha, y el código de estación, obtenido al instanciar la clase `PandasClient.py`. El método `get_kpi_month()` invoca al método `get_station_kpi_month()` de la clase `Client.py`, pasándole la fecha y el código de estación. El método `get_station_kpi_month()` lanza la petición a la API y devuelve un diccionario con los datos. El método `get_kpi_month()` recibe ese diccionario, lo modifica con el método `flatten_data(Dict):Dict` y lo transforma en un *DataFrame* de Pandas para simplificar su manejo. Este *DataFrame* viaja de vuelta hasta el script `Horarios`, donde se invoca a la función `escribirHorarios()` del script `EscribirBBDD.py`. En este script se realiza la conexión con la base de datos InfluxDB mediante `InfluxDBClient.py`, importado desde el paquete "influx". Esta función recibe el *DataFrame*, descarta las columnas vacías o repetidas enviadas por la API e introduce los datos en la tabla de la base de datos indicada como "measurement".

Si se solicitan los datos horarios, este devolverá 24 registros, cada uno correspondiente con una de la fecha que se le haya indicado. Si se le piden datos diarios, devuelve un registro por cada día del mes que se haya indicado. Si se piden datos mensuales, devuelve 12 registros, uno por cada mes del año que se haya indicado. Finalmente, si se piden datos anuales, la API devuelve un registro por cada año que la instalación lleve en funcionamiento.

4.6 Grafana

Grafana es la aplicación seleccionada para generar los gráficos con los datos obtenidos.

4.6.1 Instalación

Para instalarlo, se han seguido los pasos de la página oficial de Grafana. [32]

1. Primero se instalan los paquetes requeridos:

```
sudo apt-get install -y apt-transport-https software-properties-common wget
```

2. Se agrega la clave GPG de Grafana:

```
sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo
tee /etc/apt/keyrings/grafana.gpg > /dev/null
```

3. Se añade el repositorio de versiones estables:

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com
stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

4. Se actualiza la lista de paquetes disponibles:

```
sudo apt-get update
```

5. Se instala la versión gratuita de Grafana:

```
sudo apt-get install grafana
```

Para verificar la instalación se ejecutan las siguientes comprobaciones:

1. Se ejecuta el comando vara visualizar el estado de actividad del servicio:

```
sudo systemctl status grafana-server
```

Se debe visualizar algo como lo que se ve en la siguiente ilustración:

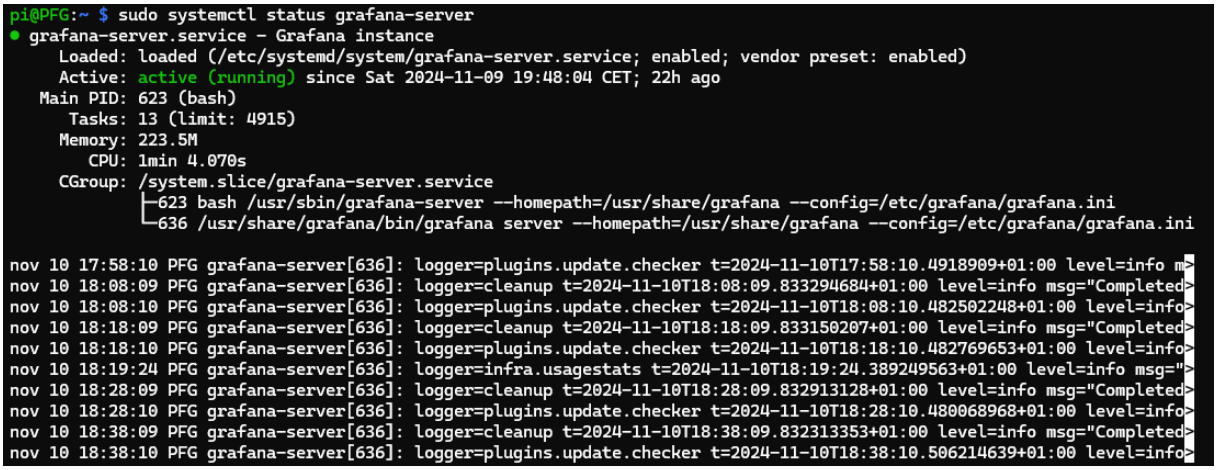

Ilustración 26 Estado del servicio de Grafana

En la ilustración se observa que el servicio de Grafana se encuentra activo y corriendo (*running*).

2. Por otro lado, desde cualquier dispositivo que se encuentre en la misma red local de la casa en la que se encuentre la Raspberry Pi, se puede entrar en un navegador, y en la barra de búsqueda poner: "*ip_raspberry*:3000". Esto debería dirigir a la página principal de Grafana en la que se pide usuario y contraseña:

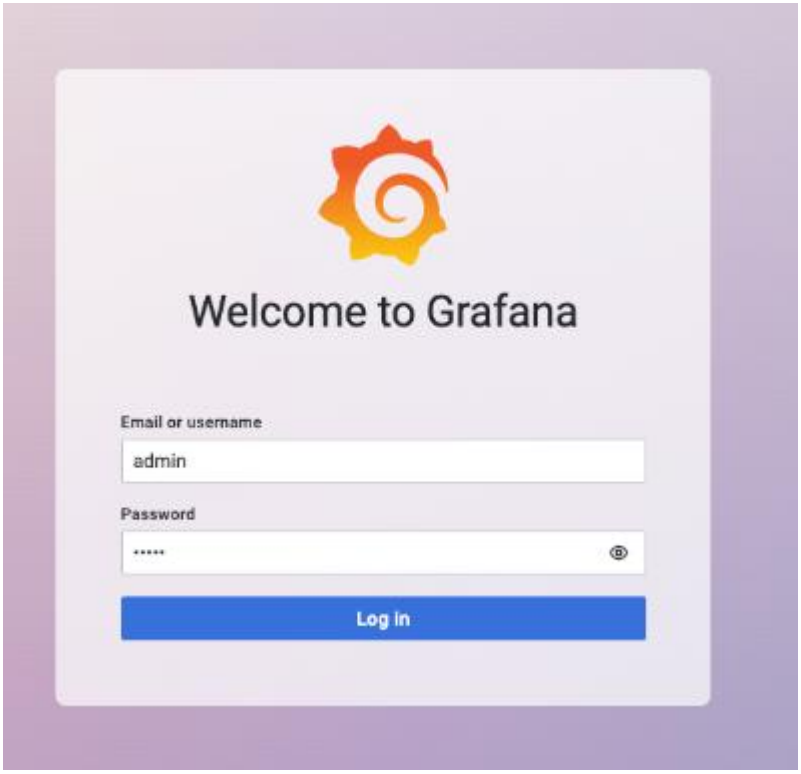

Ilustración 27 Inicio de sesión Grafana

La primera vez que se inicia sesión el nombre de usuarios es “admin” y la contraseña es “admin”. Cuando se introducen estos datos, Grafana pide cambiar la contraseña por motivos de seguridad.

4.6.2 Configuración de los gráficos

Una vez instalado el servicio, se debe con figurar la conexión con InfluxDB para hacer posible el acceso a los datos. Para ello, se siguen los siguientes pasos:

1. En el menú lateral de Grafana se busca la opción “Connections/Data sources”.
2. Se añade una nueva fuente de datos y se selecciona InfluxDB entre las opciones.
3. Se abre una página de configuración. A continuación, se detallan los campos que se deben rellenar:
   - Name: A gusto del usuario. En este proyecto se ha dejado el valor por defecto “influxdb”.
   - En el campo HTTP, se debe rellenar el apartado URL con el siguiente contenido: http://*ip_servidor_influx*:8086. Como InfluxDB se instaló y se ejecuta en la misma Raspberry Pi, se pude poner la dirección de Loopback, 127.0.0.1.
   - En la sección InfluxDB Details se debe rellenar el nombre de la base de datos, dentro del campo Database. En el apartado anterior ya se creó la base de datos y se nombró “Inversor”, por lo tanto, este campo se debe rellenar con esa información. Como método HTTP se debe configurar GET, puesto que solo se van a visualizar datos, no se ve a escribir nada en la base de datos.

Si todo ha ido bien, se tiene la conexión establecida con la base de datos. El siguiente paso es visualizar los gráficos. Para ello, en el menú lateral se pulsa sobre “Dashboards”. Se crea uno nuevo y se elige la opción de nueva visualización. En este momento pregunta de qué fuente de datos se van a obtener estos, se debe elegir InfluxDB. Se abre una pestaña de edición para elegir los parámetros del gráfico. Por defecto se utiliza la base de datos “Inversor” que se indicó durante la conexión. Para configurar un gráfico es necesario elegir la tabla de la base de datos, “measurement”, así como el campo, “field”, que es la columna que se quiere visualizar. Por último, queda seleccionar un rango de fechas acorde con el eje temporal de los datos que se quieren visualizar y el gráfico será visible.

En la siguiente imagen se muestra un ejemplo de configuración de un gráfico que muestra los datos anuales de energía generada:

Ilustración 28 Configuración de gráfico anual en Grafana

En este proyecto se han realizado cuatro paneles, o *dashboard*, uno para cada tipo de dato. Los paneles de los datos diarios, mensuales y anuales contienen los mismos gráficos, uno de energía generada y consumida, uno de energía comprada y vendida, uno de ingresos y uno de ahorro de carbón y CO2. Para los datos horarios se tienen solo tres gráficos ya que la API devuelve menos registros para este tipo de datos. Lo gráficos generados para los datos horarios son: uno de energía generada y autoconsumo, uno de energía vendida y uno de ingresos.

4.6.3 Compartir los gráficos generados

Se pueden compartir los gráficos generados de dos formas. La primera de ellas es compartir el panel completo, que compartirá todos los gráficos que se encuentren en él, visualizando en todos el mismo rango temporal. Para ello, se debe pulsar el botón “Share”, situado en la parte superior en la vista general del panel:

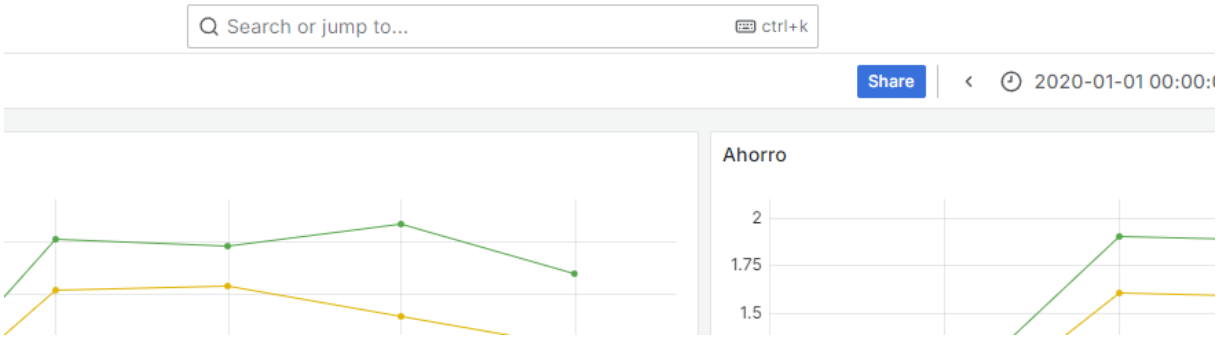

Ilustración 29 Botón de compartir en Grafana

Al pulsarlo, se despliega el siguiente mensaje emergente:

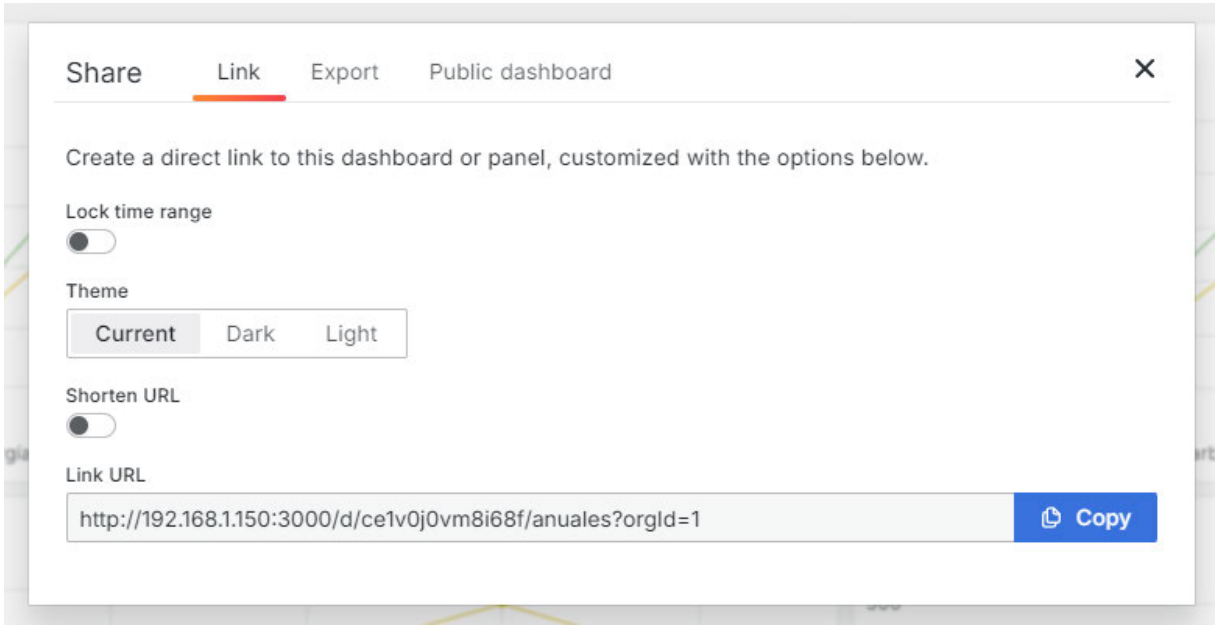

Ilustración 30 Configuración al compartir gráficos en Grafana

En este proyecto se ha utilizado la opción “Link”, lo que genera una URL que después se introducirá en openHAB. Se ha desmarcado la opción “Lock time range”, pues se quiere poder cambiar el rango temporal visualizado desde openHAB. También se puede elegir si compartir el gráfico en tema claro u oscuro, en este caso, se ha utilizado el tema claro.

Para compartir únicamente un gráfico, se deben seleccionar los tres puntos que aparecen en la parte superior derecha de cada uno de los gráficos:

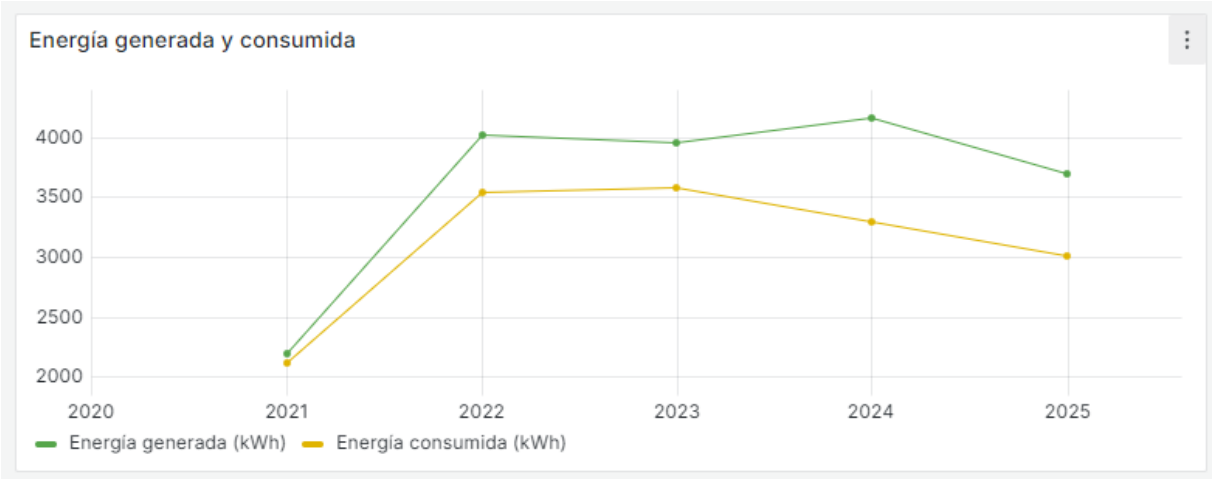

Ilustración 31 Gráfico de Grafana

En ese momento se despliega un menú en el que se observa la opción “Share”:

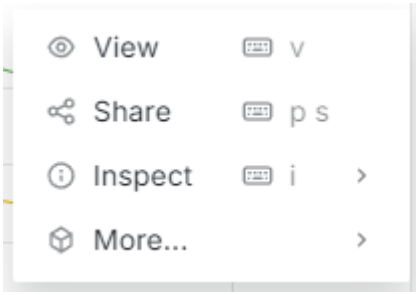

Ilustración 32 Opciones de un gráfico de Grafana

Esto lleva al mismo menú mostrado en Ilustración 30 Configuración al compartir gráficos en Grafana, donde se puede copiar la URL para compartir únicamente un gráfico.

Todos estos gráficos se muestran en el apartado Resultados.

4.7 openHAB

OpenHAB es la aplicación que se ha elegido para la gestión del hogar digital. En ella se van a integrar los gráficos del inversor solar generados por Grafana.

4.7.1 Instalación

Los pasos de la instalación se pueden encontrar en la página oficial de openHAB. [33]

El método de instalación utilizado es el de instalación de paquetes de repositorio o *package repository installation*, que es el recomendado por openHAB.

1. El primer paso es instalar Java. En este caso, Raspberry Pi OS ya trae Java incorporado por lo que este paso no es necesario.

   La versión de Java compatible con openHAB 4.2.1 es la 17. Para consultar la versión de Java instalada se ejecuta el siguiente comando:

```
java --version
```

2. Se añade la clave GPG de openHAB:

```
curl -fsSL "https://openhab.jfrog.io/artifactory/api/gpg/key/public" | gpg --dearmor > openhab.gpg
sudo mkdir /usr/share/keyrings
sudo mv openhab.gpg /usr/share/keyrings
sudo chmod u=rw,g=r,o=r /usr/share/keyrings/openhab.gpg
```

3. Se añade el repositorio de la versión estable:

```
echo 'deb [signed-by=/usr/share/keyrings/openhab.gpg]
https://openhab.jfrog.io/artifactory/openhab-linuxpkg stable main' |
sudo tee /etc/apt/sources.list.d/openhab.list
```

4. Se refresca la lista de paquetes disponibles:

```
sudo apt-get update
```

5. Se instala la última versión de openHAB, que en el momento de la realización de este proyecto es la versión 4.2.1:

```
sudo apt-get install openhab
```

6. Se inicia el servicio:

```
sudo systemctl start openhab.service
```

7. Se habilita el arranque automático al iniciar la Raspberry Pi:

```
sudo systemctl daemon-reload
sudo systemctl enable openhab.service
```

8. En este momento, openHAB ya está instalado y debería estar funcionando en la Raspberry Pi. Para comprobarlo se ejecuta el comando:

```
sudo systemctl status openhab.service
```

En la siguiente ilustración se ve la salida del comando:

```
pi@PFG:~ $ sudo systemctl status openhab.service
● openhab.service - openHAB - empowering the smart home
     Loaded: loaded (/lib/systemd/system/openhab.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-11-09 19:48:08 CET; 24h ago
       Docs: https://www.openhab.org/docs/
             https://community.openhab.org
   Main PID: 1061 (java)
      Tasks: 135 (limit: 4915)
     Memory: 506.4M
        CPU: 2min 38.092s
     CGroup: /system.slice/openhab.service
             └─1061 /usr/bin/java -XX:-UsePerfData -Dopenhab.home=/usr/share/openhab -Dopenhab.conf=/etc/openhab -Dopen>

nov 09 19:48:08 PFG systemd[1]: Started openHAB - empowering the smart home.
lines 1-13/13 (END)
```

Ilustración 33 Estado del servicio de openHAB

Se debe observar que está activo, sin errores y corriendo (*running*).

Igual que se realizó con Grafana, si desde un navegador de la red local se busca *ip_raspberry*:8080, el navegador debería mostrar la página de registro de openHAB:

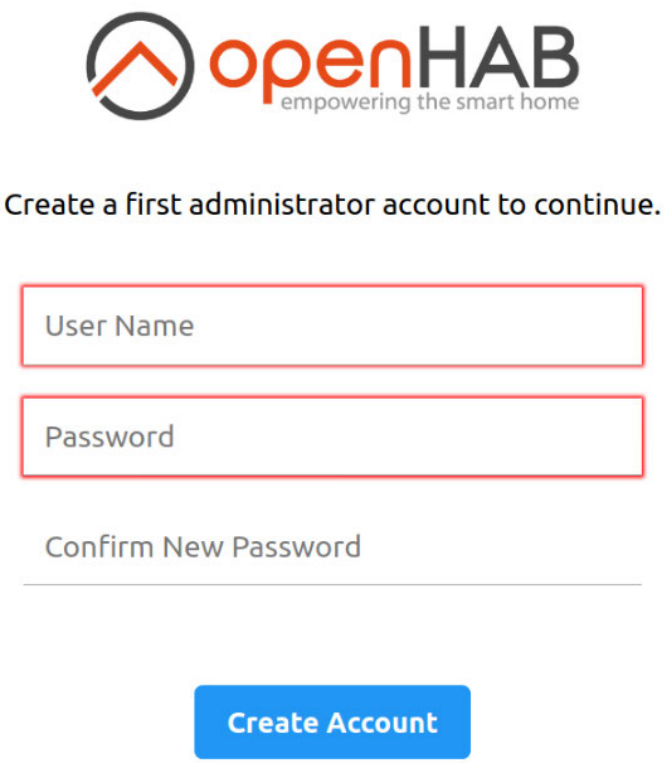

Ilustración 34 Página de registro de openHAB

Se introduce nombre de usuario y contraseña deseados. En este momento openHAB ya está listo para ser configurado.

4.7.2 Directorios compartidos por Samba

Para poder modificar los archivos de openHAB desde el PC, es necesario compartir los directorios por Samba. Para ello se incluyen las siguientes líneas en el fichero “smb.conf”, como se comentó en el apartado 4.3.3 Samba:

```
[openHAB-userdata]
  comment=openHAB userdata
  path=/var/lib/openhab
  browseable=Yes
  writeable=Yes
  only guest=no
  public=no
  create mask=0777
  directory mask=0777

[openHAB-conf]
  comment=openHAB site configuration
  path=/etc/openhab
  browseable=Yes
  writeable=Yes
  only guest=no
  public=no
  create mask=0777
  directory mask=0777

[openHAB-logs]
  comment=openHAB logs
  path=/var/log/openhab
  browseable=Yes
  writeable=Yes
  only guest=no
  public=no
  create mask=0777
  directory mask=0777
```

Si el usuario quiere incluir fotos o iconos en su configuración de openHAB, es necesario alojar esos archivos en los directorios compartidos por Samba. Como se comentó anteriormente, puesto que el usuario que se va a utilizar para acceder a los archivos con Samba es el usuario "pi", con el siguiente comando se le otorgan los permisos de lectura y escritura necesarios:

```
sudo chown -hR pi:pi /etc/openhab /var/lib/openhab /var/log/openhab
```

Después es necesario reiniciar el servicio de Samba:

```
sudo service smbd restart
```

4.7.3 Configuración de la interfaz gráfica

En este apartado se detalla la configuración necesaria para la visualización de todos los datos en openHAB. Se parte de una copia de seguridad del contenido de la interfaz Main UI de la página de openHAB de la Escuela. Toda la información del inversor ha sido integrada en ella, creando para ello una página aparte dentro de la interfaz Main UI de openHAB.

4.7.3.1 Datos en tiempo real

Los datos en tiempo real van a ser enviados a openHAB desde un script de Python, concretamente desde el script `Actual.py` mediante peticiones HTTP con método POST. Este script será lanzado por una de las reglas de openHAB, explicadas más adelante. La URL a la que se debe enviar estas peticiones es distinta para cada uno de los elementos.

Es necesario crear un elemento o *item* de openHAB para cada uno de los datos que se quieren manejar. En este proyecto se ha creado un *item* para la energía generada del día y otro para los ingresos del día.

Al crear estos *items*, openHAB permite consultarlos o modificarlos mediante una URL con el siguiente formato: http://*ip_openHAB*:8080/rest/items/*nombre_item*. En este proyecto, *ip_openHAB* es 192.168.1.150, pero como está en la misma máquina, se puede utilizar "localhost", y *nombre_item* es el nombre que se le haya asignado a cada uno de los elementos creados.

Por tanto, cada vez que se envíe una petición HTTP POST a alguna de estas URLs, el valor del *item* será actualizado.

En la siguiente imagen se muestran los *items* creados:

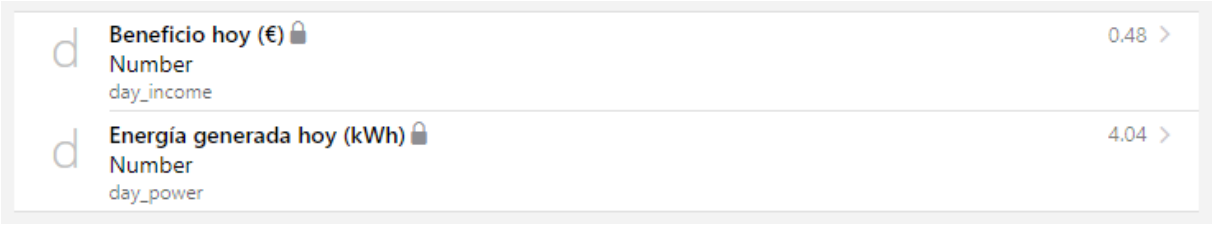

Ilustración 35 *Items* openHAB

El primer *item*, "Beneficio hoy", tiene como nombre "day_income", por lo que este es el término que se debe escribir al final de la URL para actualizar su valor. De la misma forma debe hacerse con el segundo, "Energía generada hoy", que tiene como nombre "day_power".

4.7.3.2 Creación de páginas de openHAB

La estructura seguida para la organización de todos los gráficos generados es la siguiente:

Una página llamada "Inversor". En esta página se muestran cinco pestañas distintas:

- General: Se incluyen los datos en tiempo real antes comentados además de un gráfico de potencia generada y consumida por cada tipo de dato. Es decir, un gráfico para datos horarios, un gráfico para datos diarios, uno para datos mensuales y otro para datos anuales.
- Anuales: Se incluyen cuatro gráficos en los que se extrae la información del inversor anualmente. Estos cuatro gráficos son: energía generada y consumida, ahorro de CO2 y carbón, energía comprada y vendida a la red eléctrica y, finalmente, ingresos.
- Mensuales: Se incluyen cuatro gráficos en los que se extrae la información del inversor mensualmente. Estos cuatro gráficos son: energía generada y consumida, ahorro de CO2 y carbón, energía comprada y vendida a la red eléctrica y, finalmente, ingresos.
- Diarios: Se incluyen cuatro gráficos en los que se extrae la información del inversor diariamente. Estos cuatro gráficos son: energía generada y consumida, ahorro de CO2 y carbón, energía comprada y vendida a la red eléctrica y, finalmente, ingresos.
- Horarios: Se incluyen tres gráficos en los que se extrae la información del inversor por horas. Estos tres gráficos son: energía generada y auto consumida, energía vendida a la red eléctrica y, finalmente, ingresos. En este caso los gráficos son distintos a los anteriores debido a que la API proporciona menos información para datos horarios.

Estas pestañas, se deben configurar como páginas aparte y quitar la visibilidad desde el menú lateral. Posteriormente, se deben añadir como pestañas a la página “Inversor”, que sí que es visible desde el menú lateral:

Ilustración 36 Acceso directo a la página Inversor en openHAB

La creación de cada una de las páginas que van a contener los gráficos es muy similar, cambiando únicamente valores como nombres y URLs de los gráficos. Por tanto, se explica con detalle cómo se crea una de ellas, repitiendo el proceso para las demás.

Para la creación de la página “General” se debe ir a Configuración/Pages y seleccionar una nueva página. El tipo de página deseado es el que aparece como “layout”, puesto que se quiere configurar una disposición de objetos.

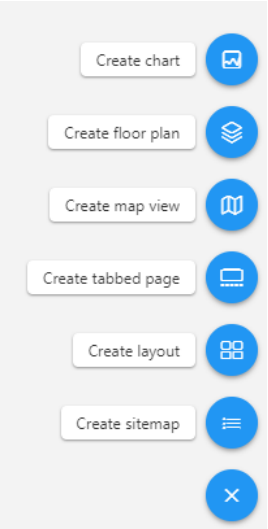

Ilustración 37 Tipos de página disponibles en openHAB

Se añade un bloque, tres filas y dos columnas en cada fila, quedando la disposición vacía así:

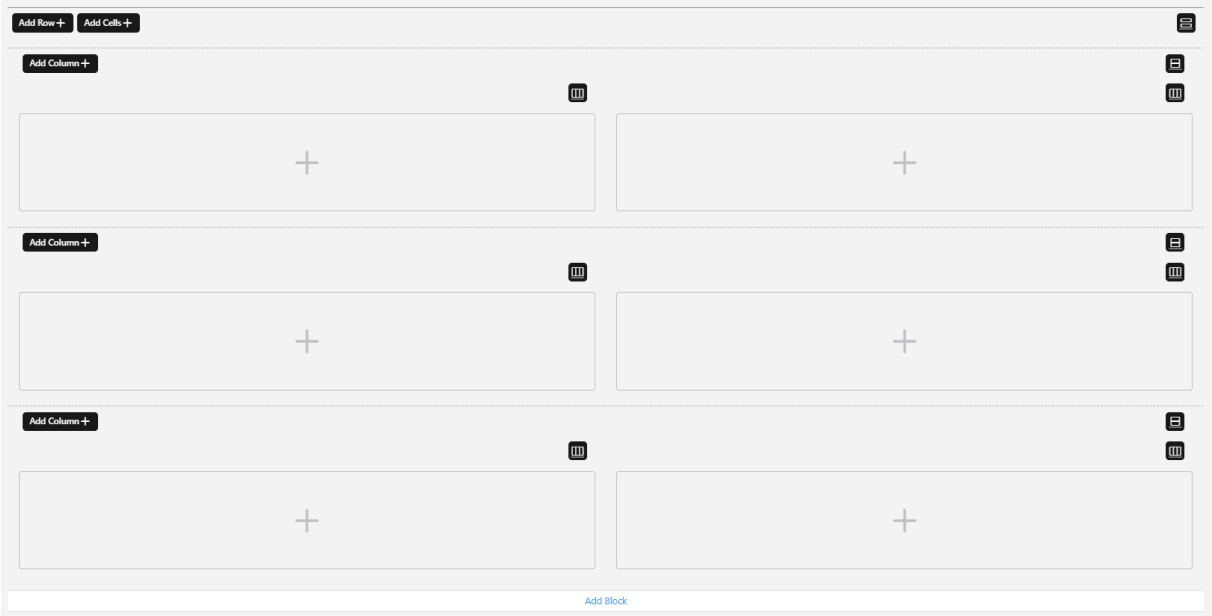

Ilustración 38 Disposición vacía de elementos de la página "General" de openHAB

En este momento se dispone de seis espacios para rellenar con *widgets*, los cuales pueden ser:

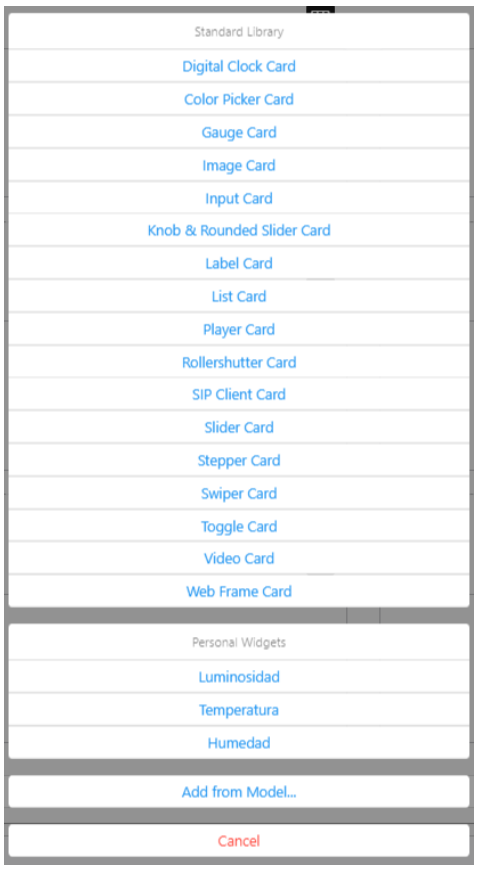

Ilustración 39 Widgets disponibles en openHAB

Para los dos *widgets* de la primera fila se selecciona “Label Card”, puesto que van a mostrar información de texto.

Se abre el panel de configuración y en él se debe rellenar el título que se le quiera dar y en el campo *item* se debe elegir el *item* del dato en tiempo real, creado en el apartado anterior.

Para los cuatro *widgets* restantes, se elige la opción “Web Frame Card”, ya que en la configuración permite introducir una URL, que debe ser la URL obtenida en Grafana para compartir el gráfico. En esta página, como se van a mostrar los gráficos de energía generada y consumida en términos horarios, diarios, mensuales y anuales no se quiere compartir el eje temporal, por lo que se debe compartir el gráfico único, no el panel completo para cada uno de ellos.

Para la creación de la página Inversor, como esta página va a alojar a todas las demás, se debe elegir la opción de página con pestañas, o en inglés *tabbed page*. Una vez dentro, se deben seleccionar todas las pestañas que se van a alojar dentro de esta página:

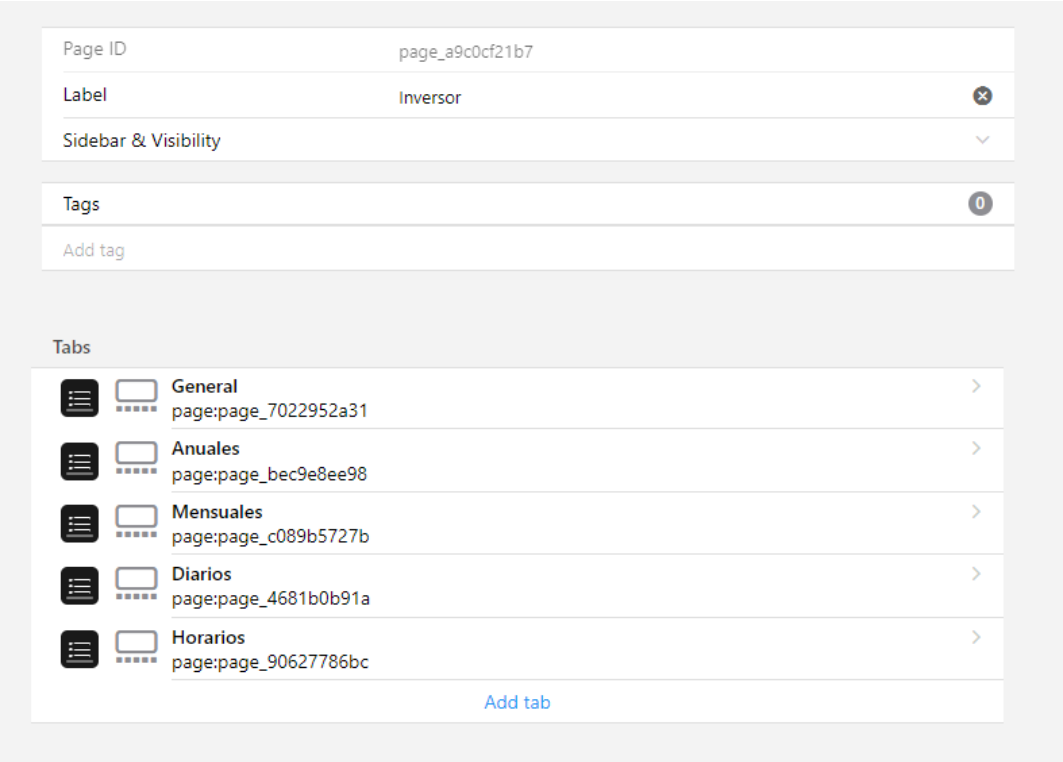

Ilustración 40 Configuración de la página "Inversor" de openHAB

4.7.4 Reglas

Las reglas son una herramienta de openHAB utilizada para la automatización de procesos. En este proyecto la utilidad que se les ha dado ha sido la de lanzar los scripts de Python para pedir los datos a Huawei mediante la API y guardarlos en la base de datos.

Para programar que estas reglas se ejecuten cada cierto tiempo se utiliza el formato *cron*, que es un estándar para programar la ejecución de tareas automatizadas en sistemas Linux. En este formato, se define una secuencia de tiempos con cinco o seis campos separados por espacios. Estos campos son: segundos, minutos, hora, día del mes, mes y día de la semana, en ese orden y en caso de ser seis campos. Si se construye con cinco, se excluyen los segundos. [34]

Si se cumple que la fecha actual coincide con los campos de minutos, hora y mes del año (en el caso de que se contemplen los segundos también se incluyen en este grupo) y además una de las otras dos condiciones de día, día del mes o día de la semana, entonces se lanza la tarea programada.

Como ejemplo la sentencia 10 14 * * *, se ejecuta a las 14:10 de cada día, y la sentencia 10 14 * * 1 se ejecuta a las 14:10 el primer día de la semana, es decir, el lunes.

De esta forma, se garantiza que la base de datos se siga rellenando una vez se ha concluido la fase de desarrollo. En este proyecto se han creado cinco reglas, todas ellas expuestas a continuación.

Los ficheros de las reglas se encuentran en el apartado A.3 Ficheros de las reglas de openHAB. En ellos se observa que se debe indicar la ruta en la que se encuentran los scripts de Python.

4.7.4.1 Datos en tiempo real

La regla `actual.rules` lanza cada cinco minutos el script `Actual.py` de Python que pide los datos en tiempo real y los envía a openHAB vía HTTP. Para ello obtiene la dirección IP privada de la máquina en la que se está ejecutando, con ella construye una URL base con formato http://*ip_obtenida*:8080/rest/items/ y lanza el script pasándole como argumento la URL generada. El script recoge esa URL, pide los datos en tiempo real a la API, genera las dos URLs para los *items* partiendo de la URL base que ha sido recibida y envía por HTTP los datos a openHAB.

4.7.4.2 Datos horarios

La regla `horarios.rules` lanza cada día a las 23:45 el script `Horarios.py` de Python, que pide los datos horarios y los guarda en la base de datos.

4.7.4.3 Datos diarios

La regla `diarios.rules` lanza el último día de cada mes a las 23:45 el script `Diarios.py` de Python, que pide los datos diarios y los guarda en la base de datos.

4.7.4.4 Datos mensuales

La regla `mensuales.rules` lanza el día 31 de diciembre a las 23:45 el script `Mensuales.py` de Python, que pide los datos mensuales y los guarda en la base de datos.

4.7.4.5 Datos anuales

La regla `anuales.rules` lanza el día 31 de diciembre a las 23:45 el script `Anuales.py` de Python, que pide los datos anuales y los guarda en la base de datos.

5. Resultados

En este apartado se muestran los resultados del proyecto una vez finalizada su etapa de desarrollo.

Este proyecto tiene dos objetivos principales. El primero de ellos es hacer que el usuario sea propietario de los datos generados por su inversor, y por tanto solo depender de Huawei a la hora de pedir los datos a la API, como alternativa a usar la aplicación Fusion Solar de Huawei. El segundo objetivo es incorporar los datos del inversor a OpenHAB, que es el centro de control de todos los dispositivos inteligentes del hogar, para de esta forma unificar en una sola aplicación la visualización y gestión de todos los dispositivos, incluido el inversor solar.

A continuación se muestran ilustraciones sobre cuál es el resultado de la integración de los datos con openHAB.

Primero, se muestra la página inicial de la interfaz Main UI de openHAB, donde se pueden observar diversos enlaces y accesos directos del Hogar Digital de la Escuela. En relación con el inversor, se observan cinco *widgets*, dentro del bloque nombrado “Inversor”, que actúan como enlace a las diferentes pestañas la página “Inversor” comentadas en el apartado anterior. Además, se observa en el menú de la izquierda otro acceso a la página “Inversor”:

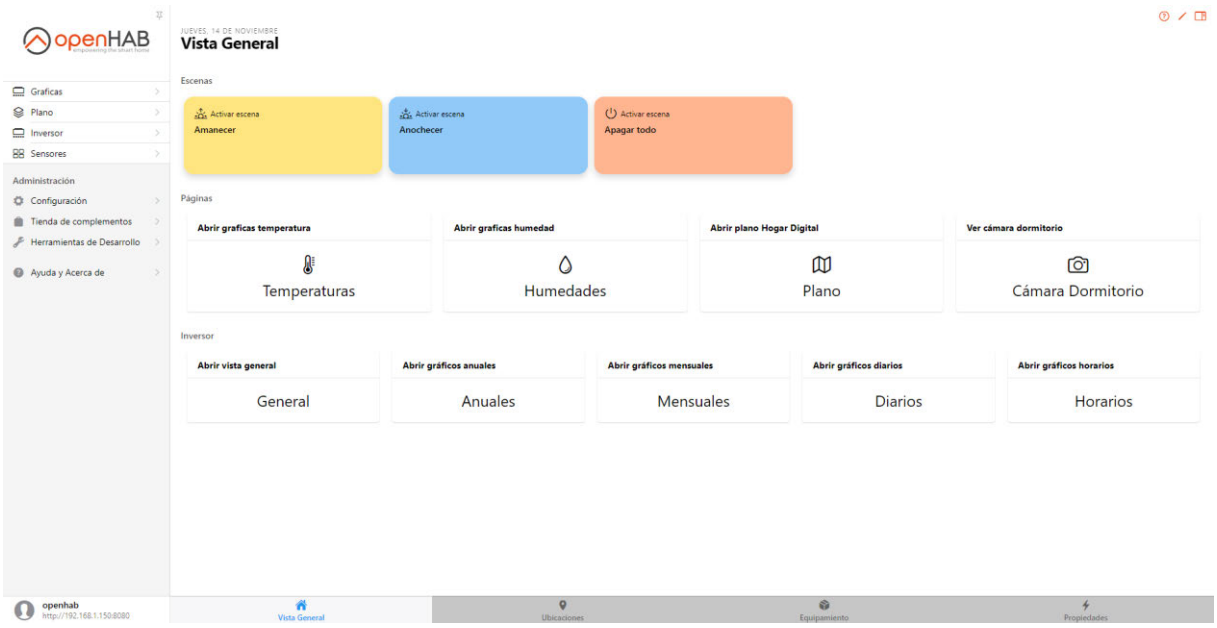

Ilustración 41 Página principal de openHAB

Tanto si se pulsa el *widget* “General” como si se pulsa la página del inversor del menú izquierdo, openHAB redirige a la página dedicada en exclusiva para los datos del inversor, y más concretamente a su pestaña de nombre “General”. Esta pestaña ofrece una visión de la energía generada y consumida, dispuesta en cuatro gráficos en los que se pueden ver los datos horarios, diarios, mensuales y anuales. Además, en la parte superior se observan dos datos numéricos. Estos datos corresponden con la energía total generada a lo largo del día y los

ingresos que ha producido la instalación. El periodo de actualización de ambos datos es de cinco minutos, como se comenta en el apartado 4.7.4 Reglas.

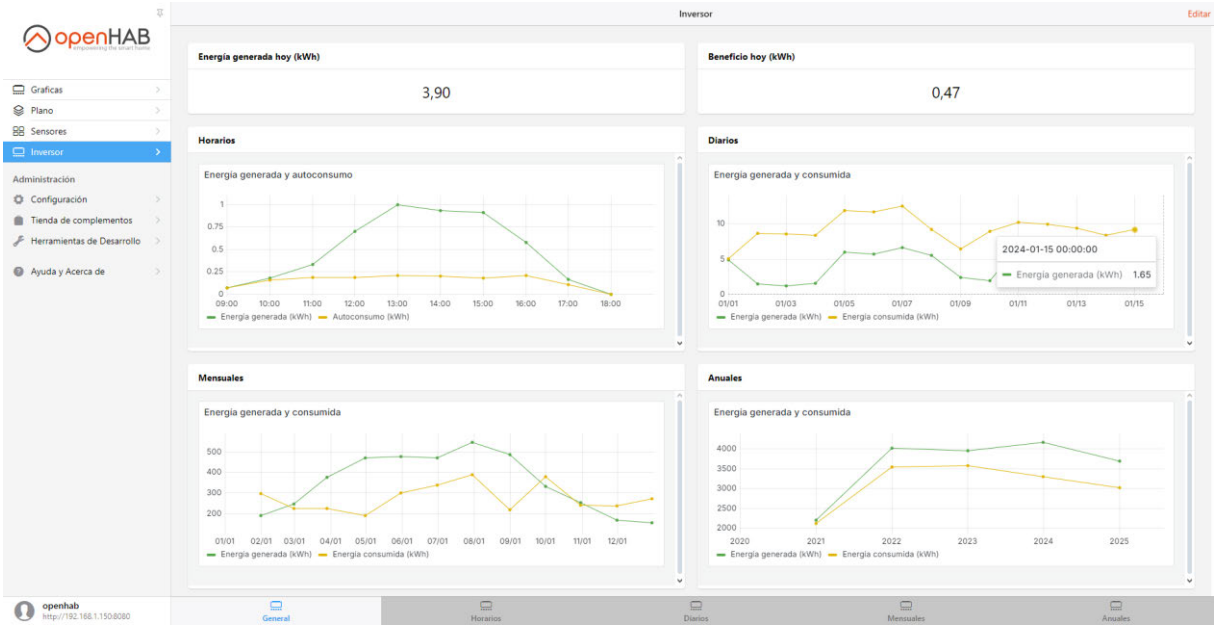

Ilustración 42 Pestaña "General" de la página "Inversor" de openHAB

Se observa que en el gráfico de diarios hay una pequeña ventana emergente. Se trata de la pestaña que surge al pasar el cursor por encima del gráfico, informando de la fecha y valor exactos.

En la siguiente imagen se observa uno de los gráficos más de cerca:

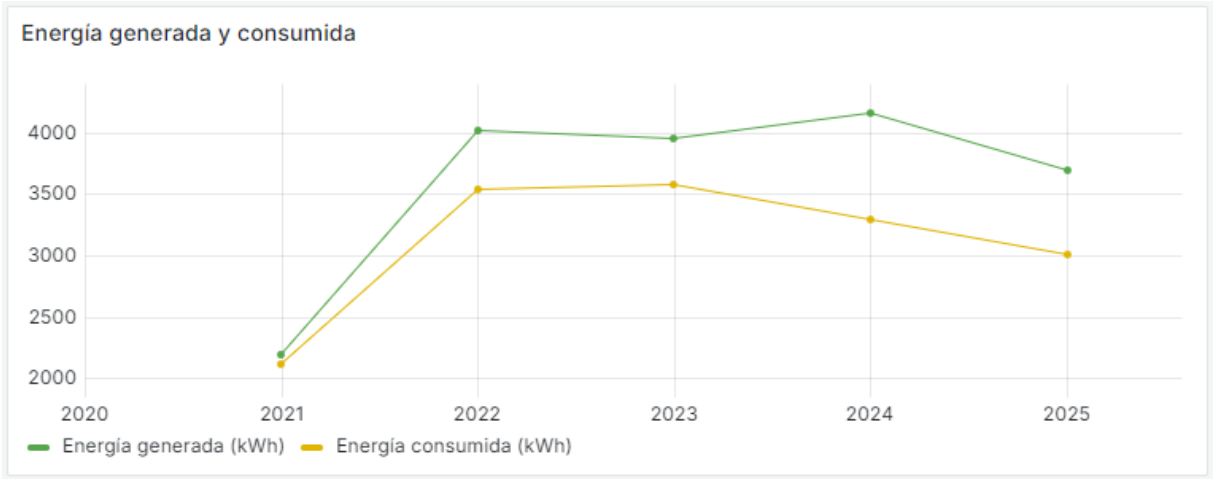

Ilustración 43 Gráfico anual de energía generada y consumida

Además de la pestaña "General", hay cuatro pestañas más. La primera de ellas se refiere solo a los datos de carácter anual, mostrando cuatro gráficas en las que se puede observar la energía generada y consumida, el ahorro de carbón y CO2, la energía comprada y vendida y los ingresos generados, todo ello desglosando la información anualmente.

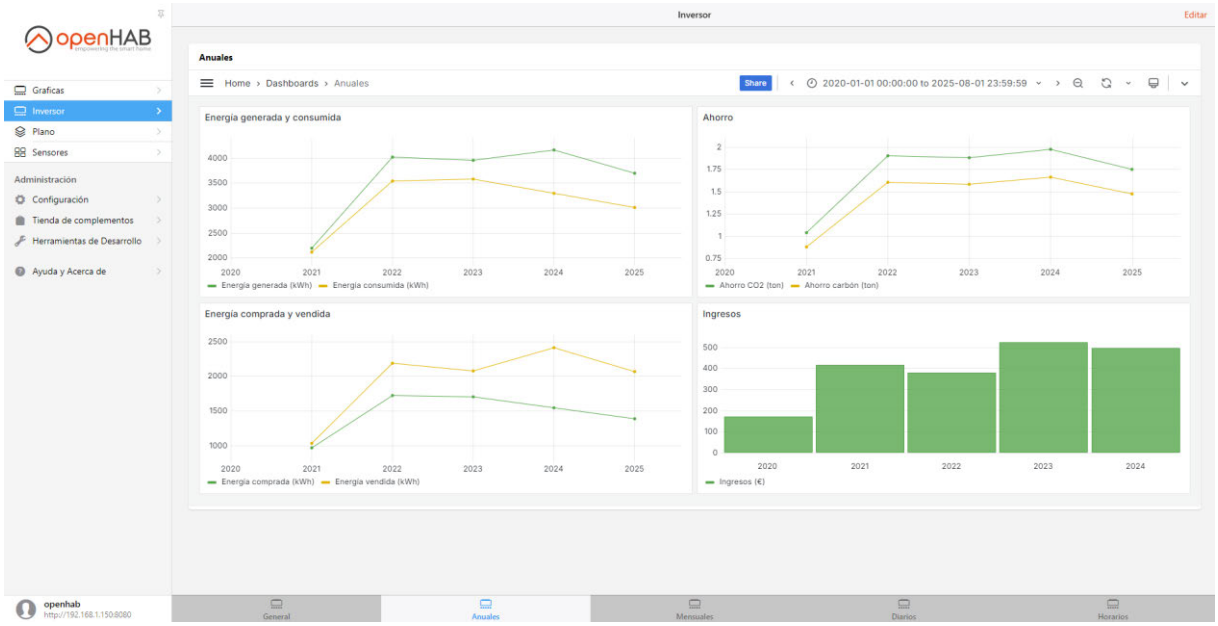

Ilustración 44 Pestaña "Anuales" de la página "Inversor" de openHAB

La siguiente pestaña es la de los datos mensuales, que ofrece los mismos cuatro gráficos que la pestaña de anuales pero esta vez separando la información mensualmente.

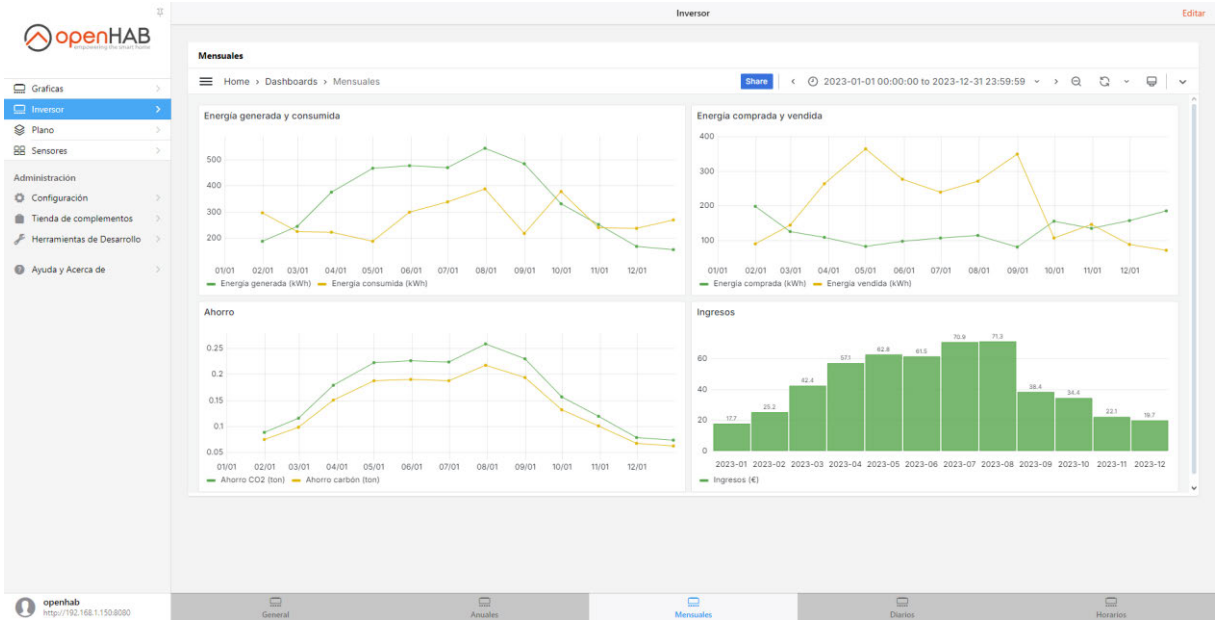

Ilustración 45 Pestaña "Mensuales" de la página "Inversor" de openHAB

La siguiente pestaña se trata de los datos diarios que, de nuevo, ofrece la misma información que las anteriores, pero separando esta vez la información diariamente.

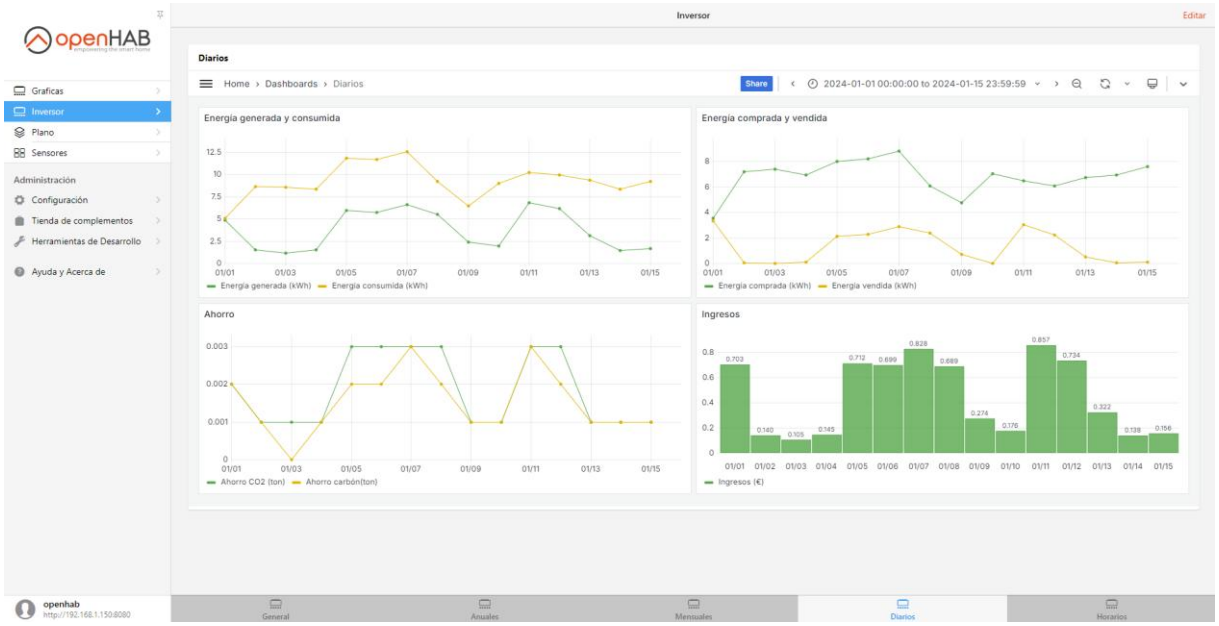

Ilustración 46 Pestaña "Diarios" de la página "Inversor" de openHAB

Por último, se encuentra la pestaña de “Horarios”. Esta pestaña, a diferencia de las tres anteriores muestra unos gráficos algo distintos. Esto se debe a que la API devuelve menos información cuando se realiza una petición de datos horarios que cuando se realiza una petición de datos diarios, mensuales o anuales. En este caso se muestran tres gráficos. El primero de ellos ofrece la información de la energía generada y el autoconsumo, siendo el autoconsumo la energía consumida que viene directamente del inversor, sin incluir la comprada de la red como en los casos anteriores. El segundo gráfico muestra únicamente la energía vendida, puesto que la API no ofrece la información horaria sobre la energía comprada. Finalmente, el tercer gráfico muestra los ingresos generados, en este caso, igual que las demás pestañas.

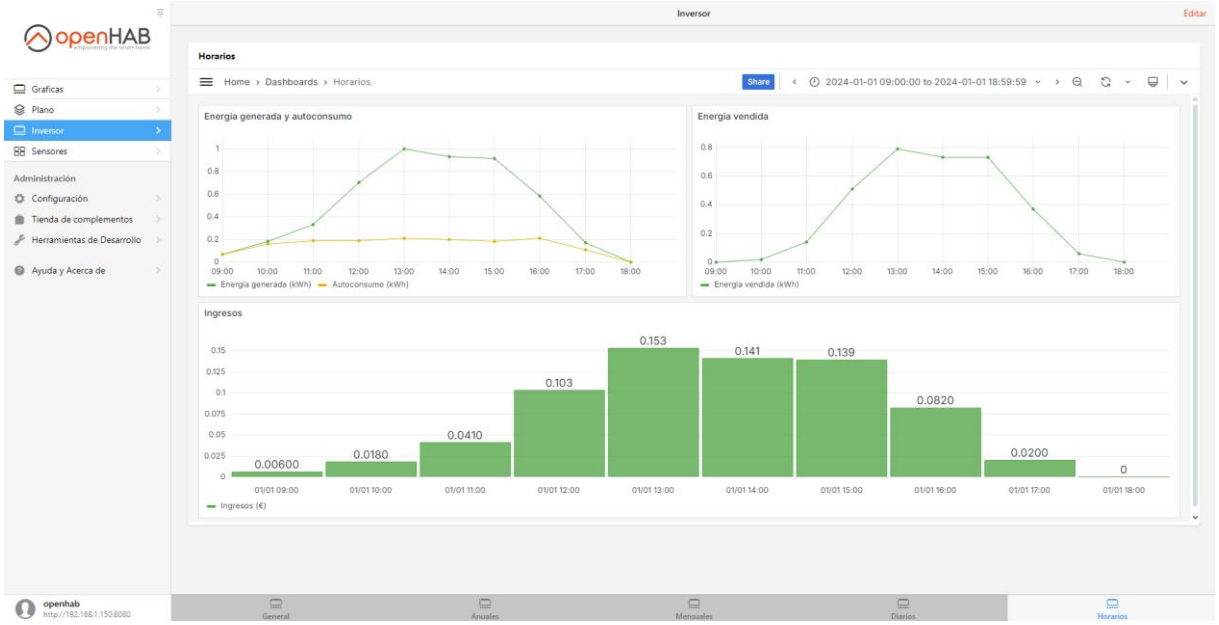

Ilustración 47 Pestaña "Horarios" de la página "Inversor" de openHAB

En la parte superior de los gráficos, al lado del botón “Share”, se encuentra el selector de fecha:

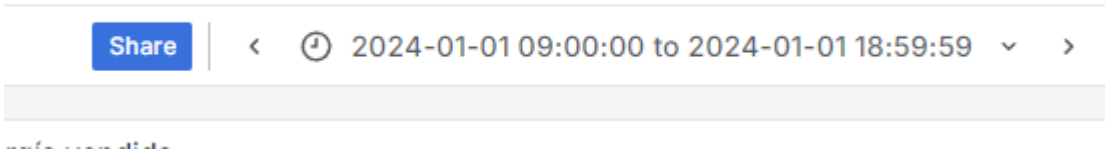

Ilustración 48 Selector de fecha de Grafana

En él se puede seleccionar el rango de fechas que se desean visualizar en el gráfico, incluyendo las horas:

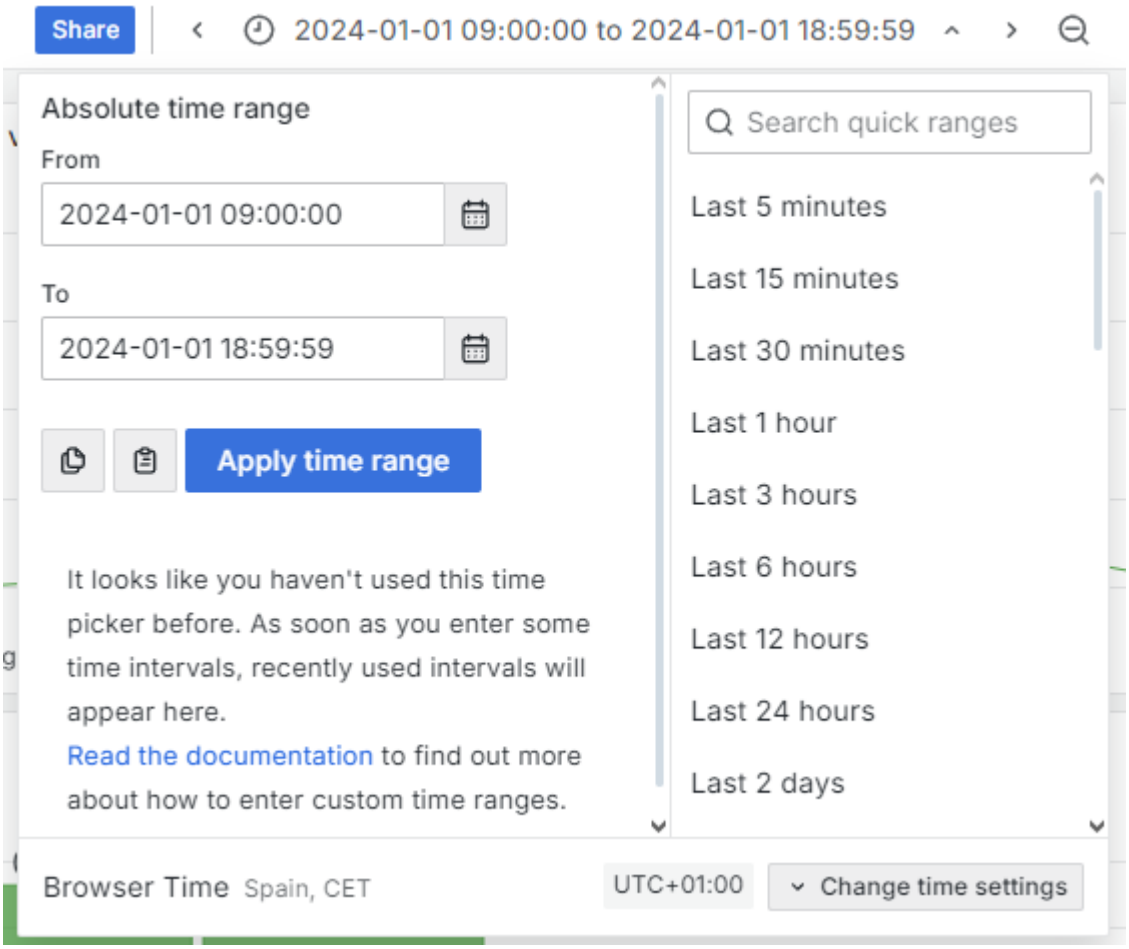

Ilustración 49 Configuración de fecha de Grafana

De esta forma se puede modificar el rango temporal que se muestra en cada gráfico de Grafana desde la página de openHAB.

Además de esta esta manera de visualizar los gráficos, que corresponde con la interfaz *Main UI* de openHAB, en los anexos se proporciona la explicación para configurar una visualización alternativa, haciendo uso de la interfaz *Basic UI* de openHAB que usa *sitemaps*.

6. Presupuesto

En esta sección se presenta el coste de cada uno de los aspectos necesarios para el desarrollo del proyecto.

6.1 Coste debido a dispositivos físicos

6.1.1 Ordenador personal del usuario

La Raspberry Pi va a ser la encargada de alojar y manejar todo el software del proyecto, pero como se comentó previamente, se ha utilizado un ordenador conectado con la Raspberry Pi de forma remota.

Existe una amplia gama de ordenadores, cada uno de ellos con precios muy distintos. El ordenador se utiliza para el desarrollo del proyecto, por lo que se descartan los ordenadores de la gama de acceso ya que sus especificaciones son demasiado limitadas. El ordenador que se ha elegido es un HP 15-fd0103ns, con un valor de 899€ en el momento de la realización de este proyecto [35]. Este ordenador tiene la capacidad suficiente para desarrollar el proyecto, además de incluir el sistema operativo Windows dentro del precio.

La vida útil de un ordenador portátil depende de la gama de este, yendo desde los 2 a los 6 años [36]. Como el ordenador utilizado es de gama media, se toma como vida útil 4 años. Este proyecto se ha desarrollado a lo largo de 6 meses, por lo que el precio relativo del ordenador con respecto a este proyecto es de 112,38€, lo que corresponde con 0,125 veces el precio total del ordenador.

6.1.2 Instalación de placas solares

En este proyecto se utiliza el Hogar Digital de la Escuela que cuenta con una instalación de placas solares y una Raspberry Pi, por lo que este presupuesto no ha sido utilizado, pero se incluye en esta memoria a modo informativo.

Se ha utilizado la web de OctopusEnergy para realizar un cálculo aproximado del coste de una instalación fotovoltaica. Se ha supuesto una casa con 100m² de superficie y un gasto mensual de electricidad de entre 75€ y 100€. [37]

Para una casa con dichas características la herramienta propone una instalación de siete paneles solares con un presupuesto de instalación de 5.500€

6.1.3 Raspberry Pi

Este proyecto se ha desarrollado con una máquina virtual con Raspberry Pi OS, puesto que el escenario real supone que el usuario tiene una Raspberry Pi física. El último modelo disponible

es la Raspberry Pi 5. Esta se puede encontrar en Kubii, que es un distribuidor oficial, por un precio de 89,40€ para el modelo de 8GB de memoria RAM. [38]

6.2 Coste debido a elementos software

Todo el software que se ha explicado en esta memoria es de código abierto, por tanto, el coste debido a software es de 0€.

6.3 Coste de recursos humanos

El salario de un ingeniero *junior* en España es de 15,68€/h. [39]

En este proyecto se han dedicado 330 horas, por tanto, el precio de recursos humanos total es de 5.174,40€.

6.4 Coste total

6.4.1 Coste del prototipo

El coste del prototipo es el que tiene en cuenta toda la fase de desarrollo.

La siguiente tabla muestra el coste del prototipo. No se tiene en cuenta la instalación fotovoltaica y la Raspberry Pi puesto que para el desarrollo del proyecto se han utilizado los recursos disponibles en el Hogar Digital de la Escuela.

Concepto	Cantidad	Coste (€)	Coste total (€)
Ordenador personal	0,125	899	112,375
Hora de trabajo de un ingeniero *junior*	330	15,68	5174,4
		Total	**5286,775**

Tabla 2 Coste del prototipo

6.4.2 Coste de producción industrial

Este es el coste que supondría vender el producto, y no se tiene en cuenta toda la fase de desarrollo, sino que se valora el tiempo invertido en instalación, montaje y puesta a punto de la solución de este proyecto.

Si el usuario ya cuenta con una Raspberry Pi y una instalación fotovoltaica el coste industrial es el mostrado en la siguiente tabla:

Concepto	Cantidad	Coste (€)	Coste total (€)
Hora de trabajo de un ingeniero junior	10	15,68	156,8
Beneficio (%)	10	156,8	15,68
		Total	**172,48**

1Tabla 3 Coste industrial reducido

Si el usuario necesita realizar la instalación fotovoltaica y la Raspberry Pi, el coste es el siguiente:

Concepto	Cantidad	Coste (€)	Coste total (€)
Instalación placas solares	1	5500	5500
Raspberry Pi	1	89,4	89,4
Hora de trabajo de un ingeniero junior	10	15,68	156,8
Beneficio (%)	10	156,8	15,68
		Total	**5761,88**

Tabla 4 Coste industrial completo

En ambos costes se observa un beneficio. Este beneficio es un 10% que aplica únicamente a las horas de trabajo, correspondiente con la rentabilidad de la venta del proyecto.

7. Impacto del proyecto

En este capítulo se manifiestan las implicaciones sociales, de salud y seguridad, ambientales, económicas, tecnológicas o industriales que están relacionadas con el proyecto, así como la aportación a los ODS (Objetivos de Desarrollo Sostenible), visibles en la siguiente ilustración:

Ilustración 50 Objetivos de Desarrollo Sostenible (ODS)

7.1 Implicaciones sociales

Este proyecto facilita a las personas el monitoreo y la gestión de la energía solar en sus hogares, lo que puede promover un mayor compromiso con el uso sostenible de la energía y una conciencia ambiental más profunda.

En relación con los ODS:

- ODS 7 – Energía asequible y no contaminante. Contribuye al acceso a energía limpia y sostenible, promoviendo el uso eficiente de recursos renovables.
- ODS 11 – Ciudades y comunidades sostenibles. Mejora la gestión energética a nivel doméstico, fomentando la sostenibilidad en las comunidades.

7.2 Implicaciones de salud y seguridad

La gestión eficiente de la energía solar puede contribuir a la reducción de la contaminación ambiental, lo que a largo plazo impacta positivamente en la salud de las personas.

En relación con los ODS:

- ODS 3 – Salud y bienestar. Ayuda a crear un entorno más saludable al reducir la dependencia de fuentes de energía contaminantes.
- ODS 13 – Acción por el clima. Promueve la mitigación del cambio climático al facilitar el uso de energía renovable.

7.3 Implicaciones ambientales

El proyecto impulsa el uso de energía solar, lo que contribuye a la reducción de emisiones de gases de efecto invernadero y apoya la transición hacia fuentes de energía más limpias.

En relación con los ODS:

- ODS 12 – Producción y consumo responsable. Facilita una mejor gestión y optimización del consumo de energía en los hogares al conocer en todo momento la producción y el consumo de esta.

7.4 Implicaciones económicas

Permite a los usuarios optimizar su consumo de energía, reduciendo costos y aumentando la eficiencia económica del hogar. Además, puede contribuir al desarrollo del sector de la tecnología energética.

En relación con los ODS:

- ODS 8 – Trabajo decente y crecimiento económico. Potencial para generar nuevas oportunidades de empleo y desarrollo económico en el sector de la energía renovable.

7.5 Implicaciones tecnológicas e industriales

Impulsa la integración de tecnologías avanzadas como uso de APIs de terceros, bases de datos y plataformas de visualización, lo que contribuye al avance de soluciones digitales en el sector de la energía renovable.

En relación con los ODS:

- ODS 9 – Industria, innovación e infraestructura. Fomenta la modernización de la infraestructura tecnológica y la innovación en el manejo de datos energéticos.
- ODS 7 – Energía asequible y no contaminante. Mejora la eficiencia tecnológica del uso de energía solar, haciendo que la energía renovable sea más accesible y práctica.

8. Conclusiones

En este apartado se comentan las conclusiones del proyecto, los objetivos que se han cumplido, las diferentes fases por las que ha pasado y, por último, los trabajos futuros.

8.1 Objetivos del proyecto

Los objetivos que se definieron en el anteproyecto son los siguientes:

- Acceder a los datos que genera el inversor y descargarlos mediante una aplicación que utilice la API de Huawei.
- Almacenar y explotar los datos de manera local, utilizando una base de datos, un servidor y una aplicación.
- Visualizar los datos de forma clara y cómoda, creando una interfaz diferente a la ofrecida por Huawei.

El cumplimiento de estos objetivos se explica en el siguiente apartado, en el que se detallan las fases por las que ha pasado el proyecto.

8.2 Fases del proyecto

El proyecto ha seguido un desarrollo lineal con las siguientes fases:

8.2.1 Obtener de los datos utilizando la API.

Esta fase cumple con el primer objetivo *"Acceder a los datos y descargarlos mediante una aplicación que utilice la API de Huawei".* Para ello se han utilizado una serie de scripts programados en Python en los que se ha conseguido el acceso a los datos del inversor que tiene Huawei mediante una API utilizando el protocolo HTTPS.

Una dificultad encontrada en este punto ha sido la modificación y filtrado de los datos, para conseguir guardar sólo lo datos estrictamente necesarios y en el formato correcto para que sean legibles y accesibles por el resto de las aplicaciones involucradas en este proyecto.

8.2.2 Almacenar los datos en una base de datos para asegurar su persistencia.

Esta fase se enmarca en el segundo objetivo *"Almacenar y explotar los datos de manera local, utilizando una base de datos, un servidor y una aplicación"*. Python ha sido de nuevo el motor utilizado para guardar los datos, realizando una conexión a una base de datos InfluxDB, diseñada para almacenar, de forma eficiente, datos con series temporales.

8.2.3 Explotar los datos para obtener gráficos y poder analizarlos.

Esta fase termina de cumplir el segundo objetivo *"Almacenar y explotar los datos de manera local, utilizando una base de datos, un servidor y una aplicación".* Para conseguirlo se ha utilizado una Raspberry Pi en la que están alojadas todas las aplicaciones, trabajando como un

servidor. Una de esas tecnologías es Grafana, que es la aplicación utilizada para generar los gráficos con los datos almacenados.

8.2.4 Exportar los resultados al centro de control del hogar digital.

Esta fase cumple con el tercer objetivo *"Visualizar los datos de forma clara y cómoda creando una interfaz diferente a la ofrecida por Huawei".* Para ello se han exportado los gráficos de Grafana a la aplicación de openHAB, que es la aplicación que hace de centro de control, consiguiendo así integrar los gráficos con el resto de los elementos inteligentes que tiene el hogar digital.

Esto conlleva dificultades debido a la autenticación requerida por Grafana para acceder a los gráficos. Finalmente se decidió permitir la autenticación anónima para que cualquier usuario en posesión de la URL pueda acceder al gráfico. Esto no supone ningún riesgo de seguridad dado que la Raspberry Pi opera en la red local de la casa del usuario.

8.3 Trabajos futuros

En este apartado se proponen diferentes posibilidades para dar continuidad y ampliar el proyecto:

- Proporcionar una capa extra de seguridad. Como se ha comentado anteriormente, se ha configurado Grafana con autenticación anónima. Se propone configurar la autenticación de Grafana mediante *tokens*, de esta manera solo el usuario que esté en posesión del token con los premisos necesarios podrá visualizar el gráfico.
- Mejorar el algoritmo de recogida de datos históricos. Actualmente el script que recoge los datos históricos recibe dos argumentos, el primero es el año en el que se realizó la instalación y por tanto el año en el que se empezó a registrar datos (*año inicio*), y el segundo es el año límite hasta el que se quieren recoger los datos (*año fin*). Con el funcionamiento actual, se piden los datos desde el primer día del *año inicio* hasta el último día del *año fin*. Este algoritmo resulta poco eficiente ya que dependiendo de cuándo se realizó la instalación, se van a realizar muchas peticiones sin respuesta. Por ejemplo, si la instalación se realizó en noviembre de 2020, el script pide los datos desde el 1 de enero de 2020, lo que supone un gran número de peticiones sin respuesta y más tiempo del necesario para rellenar la base de datos.
- Si en un futuro la API proporciona el dato de energía consumida en tiempo real, igual que se recogen la energía generada y los ingresos del día, se podría incorporar este dato, y programar reglas que hagan uso de él. Por ejemplo, si la energía generada es mayor que la consumida, hacer que se encienda algún elemento del hogar, como la depuradora de la piscina en verano o la calefacción eléctrica en invierno.

9. Referencias

[1] Avast, «¿Qué es un hogar inteligente?,» [En línea]. Available: https://www.avast.com/es-es/c-what-is-a-smart-home. [Último acceso: 4 Noviembre 2024].

[2] Huawei, «Información corporativa,» [En línea]. Available: https://www.huawei.com/es/corporate-information. [Último acceso: 4 Noviembre 2024].

[3] Huawei, «FusionSolar Login,» [En línea]. Available: https://eu5.fusionsolar.huawei.com/unisso/login.action. [Último acceso: 4 Noviembre 2024].

[4] IBM, «¿Qué es una API (interfaz de programación de aplicaciones)?,» [En línea]. Available: https://www.ibm.com/es-es/topics/api. [Último acceso: 4 Noviembre 2024].

[5] Huawei, «SmartPVMS 24.2.0 Northbound API Reference,» [En línea]. Available: https://support.huawei.com/enterprise/en/doc/EDOC1100358266/3b606afb/introduction-to-northbound-open-apis. [Último acceso: 4 Noviembre 2024].

[6] RedHat, «REST,» [En línea]. Available: https://www.redhat.com/es/topics/api/what-is-a-rest-api#rest. [Último acceso: 4 Noviembre 2024].

[7] JSON.ORG, «Introducing JSON,» [En línea]. Available: https://www.json.org/json-en.html. [Último acceso: 4 Noviembre 2024].

[8] R. Pi, «About Raspberry Pi,» [En línea]. Available: https://www.raspberrypi.com/. [Último acceso: 4 Noviembre 2024].

[9] C. L. Vegas, «Raspberry Pi, un ordenador potente,barato, multifuncional y de tamaño bolsillo,» [En línea]. Available: https://www.callemayor.es/raspberry-pi-ordenador-potente-barato-multifuncional-tamano-bolsillo/. [Último acceso: 14 Noviembre 2024].

[10] R. Pi, «Raspberry Pi OS,» [En línea]. Available: https://www.raspberrypi.com/software/. [Último acceso: 4 Noviembre 2024].

[11] VMWare, «About VM Ware Workstation,» [En línea]. Available: https://www.vmware.com/info/workstation-pro/faq. [Último acceso: 4 Noviembre 2024].

[12] CloudFlare, «What is SSH? | Secure Shell (SSH) protocol,» [En línea]. Available: https://www.cloudflare.com/learning/access-management/what-is-ssh/. [Último acceso: 5 Noviembre 2024].

[13] Microsoft, «Server Message Block Overview,» [En línea]. Available: https://learn.microsoft.com/es-es/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831795(v=ws.11). [Último acceso: 5 Noviembre 2024].

[14] Samba, «About Samba,» [En línea]. Available: https://www.samba.org/. [Último acceso: 5 Noviembre 2024].

[15] RealVNC, «Descargue VNC® Connect en el equipo que desee controlar,» [En línea]. Available: https://www.realvnc.com/es/connect/download/vnc/. [Último acceso: 5 Noviembre 2024].

[16] Java, «What is Java technology and why do I need it?,» [En línea]. Available: https://www.java.com/en/download/help/whatis_java.html. [Último acceso: 5 Noviembre 2024].

[17] Python, «About Python,» [En línea]. Available: https://www.python.org/. [Último acceso: 5 Noviembre 2024].

[18] Microsoft, «About Visual Studio Code,» [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 5 Noviembre 2024].

[19] MySQL, «About MySQL,» [En línea]. Available: https://www.mysql.com/. [Último acceso: 5 Noviembre 2024].

[20] SQLite, «What Is SQLite?,» [En línea]. Available: https://www.sqlite.org/index.html. [Último acceso: 5 Noviembre 2024].

[21] PyPi, «Flask,» [En línea]. Available: https://pypi.org/project/Flask/. [Último acceso: 5 Noviembre 2024].

[22] InfluxData, «About InfluxData,» [En línea]. Available: https://www.influxdata.com/. [Último acceso: 5 Noviembre 2024].

[23] Microsoft, «Power BI,» [En línea]. Available: https://www.microsoft.com/es-es/power-platform/products/power-bi?msockid=30c1535418356cb4207d47a119e56d18. [Último acceso: 5 Noviembre 2024].

[24] Grafana, «About Grafana,» [En línea]. Available: https://grafana.com/docs/grafana/latest/introduction/. [Último acceso: 6 Noviembre 2024].

[25] HomeAssistant, «About Home Assistant,» [En línea]. Available: https://www.home-assistant.io/. [Último acceso: 6 Noviembre 2024].

[26] openHAB, «About openHAB,» [En línea]. Available: https://www.openhab.org/docs/. [Último acceso: 6 Noviembre 2024].

[27] clouding.io, «Cómo instalar y configurar VNC Server en Linux,» [En línea]. Available: https://help.clouding.io/hc/es/articles/360010658340-Cómo-instalar-y-configurar-VNC-Server-en-Linux. [Último acceso: 9 Noviembre 2024].

[28] FlotaDigital, «Instalar y configurar Samba en Ubuntu 20.04,» [En línea]. Available: https://flotadigital.com/tutoriales2/instalar-y-configurar-samba-en-ubuntu-20-04/. [Último acceso: 9 Noviembre 2024].

[29] openHAB, «InfluxDB+Grafana persistence and graphing,» [En línea]. Available: https://community.openhab.org/t/influxdb-grafana-persistence-and-graphing/13761/86. [Último acceso: 9 Noviembre 2024].

[30] desarrolloweb.com, «Qué es la Unix epoch (época Unix),» [En línea]. Available: https://desarrolloweb.com/faq/unix-epoch-epoca. [Último acceso: 28 Noviembre 2024].

[31] EnergieID, «Proyecto FusionSolar,» [En línea]. Available: https://github.com/EnergieID/FusionSolar. [Último acceso: 10 Noviembre 2024].

[32] Grafana, «Install Grafana on Debian or Ubuntu,» [En línea]. Available: https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/. [Último acceso: 10 Noviembre 2024].

[33] OpenHAB, «openHAB on Linux,» [En línea]. Available: https://www.openhab.org/docs/installation/linux.html. [Último acceso: 10 Noviembre 2024].

[34] IBM, «Formato cron de UNIX,» [En línea]. Available: https://www.ibm.com/docs/es/db2/11.5?topic=task-unix-cron-format. [Último acceso: 11 Noviembre 2024].

[35] E. C. Inglés, «HP 15-fd0103ns,» [En línea]. Available: https://www.elcorteingles.es/electronica/A51887863-portatil-hp-15-fd0103ns-i7-16gb-1tb-ssd-156-

w11/?stype=search_redirect&parentCategoryId=999.7624603013&color=Plata. [Último acceso: 28 Noviembre 2024].

[36] MiniTool, «¿Cuánto tiempo dura un ordenador portátil?,» [En línea]. Available: https://www.minitool.com/es/respaldar-datos/cuanto-dura-un-ordenador-portatil.html. [Último acceso: 28 Noviembre 2024].

[37] OctopusEnergy, «Presupuesto intalación fotovoltaica,» [En línea]. Available: https://octopusenergy.es/solar/simulador/presupuesto?houseType=house&consumption=100&roof_size=135.8&location=-3.5605856067617196%2C40.39802986052783. [Último acceso: 6 Noviembre 2024].

[38] Kubii, «Comprar Raspberry Pi 5,» [En línea]. Available: https://www.kubii.com/es/microordenadores/4106-1832-raspberry-pi-5-3272496315938.html#/ram-8_gb?src=raspberrypi. [Último acceso: 6 Noviembre 2024].

[39] Jooble, «Ingeniero junior salarios,» [En línea]. Available: https://es.jooble.org/salary/ingeniero-junior#hourly. [Último acceso: 28 Noviembre 2024].

[40] Avast, «¿Qué es TCP/IP y cómo funciona?,» [En línea]. Available: https://www.avast.com/es-es/c-what-is-tcp-ip. [Último acceso: 4 Noviembre 2024].

[41] CloudFlare, «¿Qué es HTTP?,» [En línea]. Available: https://www.cloudflare.com/es-es/learning/ddos/glossary/hypertext-transfer-protocol-http/. [Último acceso: 4 Noviembre 2024].

[42] HP, «Portátil HP 15 -fd0113ns,» [En línea]. Available: https://www.hp.com/es-es/shop/product.aspx?id=A42C2EA&opt=ABE&sel=NTB. [Último acceso: 6 Noviembre 2024].

[43] Jobted, «Sueldo del Ingeniero en España,» [En línea]. Available: https://www.jobted.es/salario/ingeniero. [Último acceso: 6 Noviembre 2024].

[44] Google, «DNS público de Google para tus dispositivos,» [En línea]. Available: https://developers.google.com/speed/public-dns/docs/using. [Último acceso: 9 Noviembre 2024].

Anexos

A.1 Manual de usuario

En este anexo se explica cómo se debe hacer para almacenar en la base de datos los registros históricos que generó el inversor.

Los scripts que realizan esta función son `HorariosHistorico.py`, `DiariosHistorico.py` y `MensualesHistorico.py`. Dado que estos scripts se van a ejecutar una única vez, no se ha creado ningún automatismo que los lance.

Los tres scripts esperan los mismos argumentos: el año de inicio y el año de fin. Con el año de inicio se hace referencia al año en que se instaló el inversor, por lo que es el momento en que deberían registrar los primeros datos. Con el año de fin se hace referencia al año hasta el que se quieren recoger los datos, generalmente el año en curso. Si por cualquier motivo se quisiera recoger un periodo de años concreto, no es necesario que el inicio y el fin sean las fechas de instalación y el año en curso, sino que pueden ser las del periodo deseado.

Puesto que los scripts se van a lanzar por comandos en la Raspberry Pi, se recomienda situarse en el directorio en el que se hayan ubicado los archivos Python. Una vez en ese directorio se lanza el comando, en caso de querer recoger los datos mensuales desde el año 2022 al 2024:

```
python3 MensualesHistorico.py --inicio 2022 --fin 2024
```

En la siguiente ilustración se observa una ejecución del programa:

```
pi@PFG:~ $ cd Desktop/proyecto/
pi@PFG:~/Desktop/proyecto $ python3 MensualesHistorico.py --inicio 2022 --fin 2024
Datos mensuales.  Datos guardados exitosamente en InfluxDB de la fecha 20220101
Datos mensuales.  Datos guardados exitosamente en InfluxDB de la fecha 20230101
Datos mensuales.  Datos guardados exitosamente en InfluxDB de la fecha 20240101
pi@PFG:~/Desktop/proyecto $
```

Ilustración 51 Ejecución script MensualesHistorico.py

Para rellenar la base de datos con toda la información que ha generado el inversor y que está en la nube de Huawei, se debe realizar el mismo procedimiento con los scripts `DiariosHistorico.py` y `HorariosHistorico.py`.

A.2 Ficheros del proyecto

En el siguiente enlace se encuentran los archivos necesarios para el desarrollo de este proyecto: https://github.com/sergioroca00/PFG.git .

Dichos archivos son:

- Python. Todos los scripts mencionados en 4.5 Scripts Python.
- openHAB. Se encuentran los archivos con extensión ".sitemap", ".rules" y ".items".

A.3 Ficheros de las reglas de openHAB

Regla para lanzar el script `Actual.py`:

```
rule "Datos en tiempo real"
when
    Time cron "0 */5 * * * ?" // Cada 5 minutos
then
    // Ejecutar el script Python
    val String ip = executeCommandLine(Duration.ofSeconds(5), "hostname", "-
I").trim.split(" ").get(0)
    val String url = "http://" + ip + ":8080/rest/items/"
    val String resultado = executeCommandLine(Duration.ofSeconds(60), "/usr/bin/python3",
"/home/pi/Desktop/proyecto/Actual.py", "--url", url)
    logInfo("Datos en tiempo real", "Resultado del script: " + resultado)
end
```

Regla para lanzar `Horarios.py`:

```
rule "Horarios"
when
    Time cron "0 45 23 * * ?" // A las 23:45 cada día
then
    // Ejecutar el script Python
    val String resultado = executeCommandLine(Duration.ofSeconds(60), "/usr/bin/python3",
"/home/pi/Desktop/proyecto/Horarios.py")
    logInfo("Datos horarios", "Resultado del script: " + resultado)
end
```

Regla para lanzar el script `Diarios.py`:

```
rule "Diarios"

when

    Time cron "0 45 23 L * ?" // A las 23:45 el último día de cada mes

then

    // Ejecutar el script Python

    val String resultado = executeCommandLine(Duration.ofSeconds(60), "/usr/bin/python3",
"/home/pi/Desktop/proyecto/Diarios.py")

    logInfo("Datos diarios", "Resultado del script: " + resultado)

end
```

Regla para lanzar el script Mensuales.py:

```
rule "Mensuales"
when
    Time cron "0 45 23 31 12 ?" // Cada 31 de diciembre
then
    // Ejecutar el script Python
    val String resultado = executeCommandLine(Duration.ofSeconds(60), "/usr/bin/python3",
"/home/pi/Desktop/proyecto/Mensuales.py")
    logInfo("Datos mensuales", "Resultado del script: " + resultado)
end
```

Regla para lanzar el script Anuales.py:

```
rule "Anuales"

when

    Time cron "0 45 23 31 12 ?" // Cada 31 de diciembre a las 23:45

then

    // Ejecutar el script Python

    val String resultado = executeCommandLine(Duration.ofSeconds(60), "/usr/bin/python3",
"/home/pi/Desktop/proyecto/Anuales.py")

    logInfo("Datos anuales", "Resultado del script: " + resultado)

end
```

Estos archivos se deben alojar en la siguiente ruta: “/etc/openhab/rules/”.

A.4 Ficheros de los items de openHAB

En este anexo se encuentran los ficheros de los dos *items* creados en openHAB. El primero se corresponde con la energía generada en ese día, `day_power.items`:

```
Number day_power "Energía generada hoy (kWh) [%.2f]"
```

El Segundo se correspone con los ingresos de ese día, `day_income.items`:

```
Number day_income "Beneficio hoy (€) [%.2f]"
```

El término "[%.2f]" se utiliza para mostrar el valor únicamente con dos cifras decimales.

Estos archivos se deben alojar en la siguiente ruta: "/etc/openhab/items/".

A.5 Configuración de sitemaps en openHAB

Para visualizar la información mediante los *sitemaps* de openHAB se deben seguir los siguientes pasos.

A.5.1. Instalación de Basic UI en openHAB

Basic UI es otra interfaz de usuario de openHAB diferente a Main UI que es la que se ha usado en este proyecto. Esta alternativa busca la simplicidad de uso y se basa en la información contenida en los ficheros *sitemaps*.

Para poder utilizarla, es necesario instalarla desde la página principal de openHAB. Para ello, en el menú lateral se pulsa sobre la opción "Tienda de complementos" y en la barra de búsqueda se introduce "basic". Se debe visualizar lo siguiente:

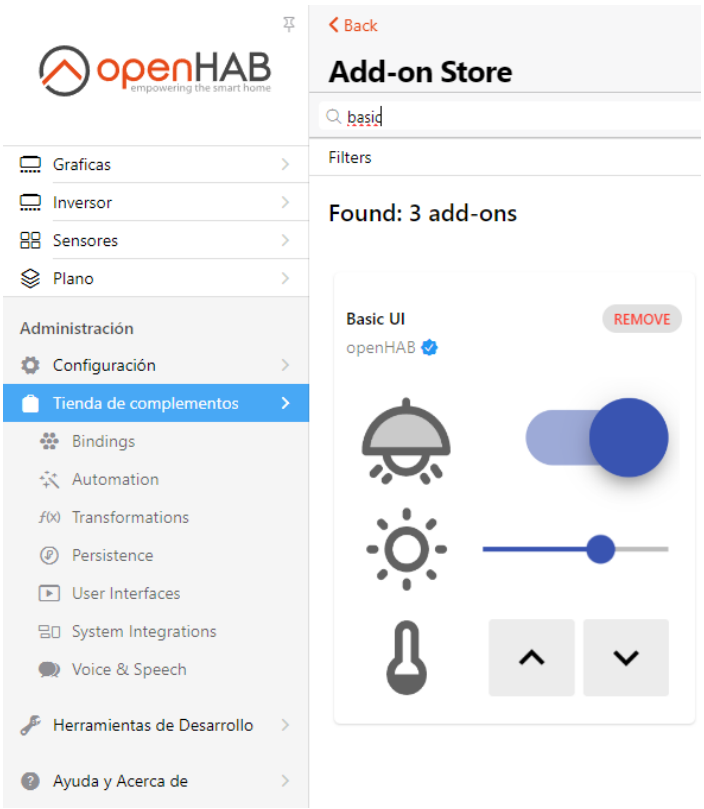

Ilustración 52 Instalación de Basic UI de openHAB

En este caso se ve el mensaje "Remove" puesto que ya está instalada. De no ser así, en ese mensaje pondría "Install".

Una vez instalada, la forma de acceder a ella es introducir la siguiente URL en un el navegador de cualquier dispositivo que esté en la misma red local donde está la raspberry: "http://*ip_raspberry*:8080/basicui/app" y se deben observar los *sitemaps* creados:

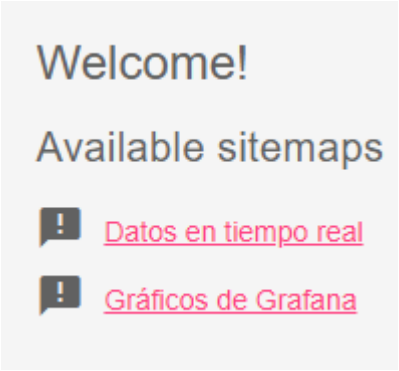

Ilustración 53 *Sitemaps* en openHAB

Donde entrando en cada uno de esos enlaces se puede ver la información que muestra el sitemap:

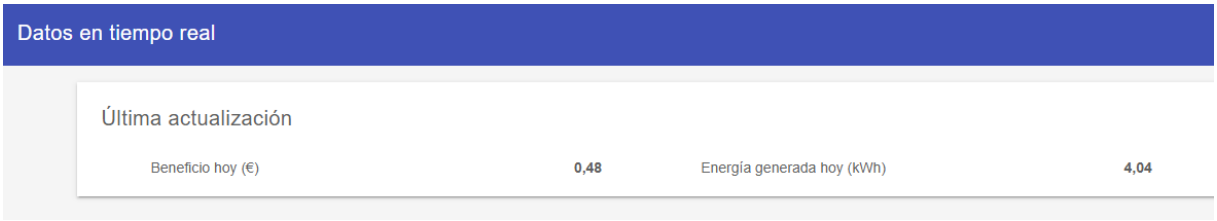

Ilustración 54 *Sitemap* actual.sitemaps

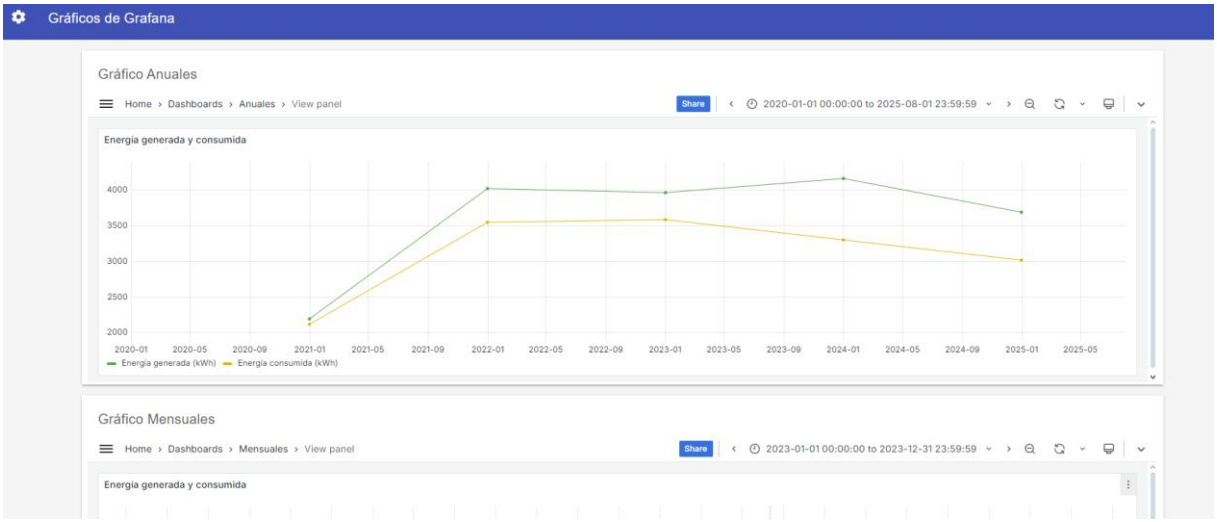

Ilustración 55 *Sitemap* grafico.sitemaps

A.5.2. Creación del archivo sitemap

Se debe crear un archivo sitemap en la Raspberry Pi, en la ubicación "/etc/openhab/sitemaps/" con el nombre que se desee.

El contenido del *sitemap* depende de lo que se quiera visualizar. En este caso, se ha decidido mostrar los mismos cuatro gráficos que se muestran en la interfaz Main UI de openHAB dentro de la pestaña "General" de la página "Inversor". La URL de cada uno de los gráficos es la obtenida en Grafana como se explicó en el apartado 4.6.3 Compartir los gráficos generados. Por tanto, el contenido del archivo es el siguiente:

```
sitemap graficos label="Gráficos de Grafana"
{
    Frame label="Gráfico Anuales" {
        Webview
url="http://127.0.0.1:3000/d/ce1v0j0vm8i68f/anuales?orgId=1&viewPanel=1&theme=light"
height=12
    }
    Frame label="Gráfico Mensuales" {
        Webview
url="http://127.0.0.1:3000/d/de1v15lojaozkd/mensuales?orgId=1&viewPanel=1&theme=light"
height=12
    }

    Frame label="Gráfico Diarios" {
        Webview
url="http://127.0.0.1:3000/d/fe1v1ddo72by8f/diarios?orgId=1&viewPanel=1&theme=light"
height=12
    }
    Frame label="Gráfico Horarios" {
        Webview
url="http://127.0.0.1:3000/d/ce1v1inyg3thcd/horarios?orgId=1&viewPanel=1&theme=light"
height=12
    }
}
```

También se ha creado otro *sitemap* para visualizar los valores en tiempo real, la energía generada y los ingresos. Este archivo toma el nombre de "actual.sitemap" y su contenido es el siguiente:

```
sitemap actual label="Datos en tiempo real"
{
    Frame label="Última actualización" {
        Text item=day_income
        Text item=day_power
    }
}
```