AEMET – Agencia estatal de meteorología

Ingesta de datos

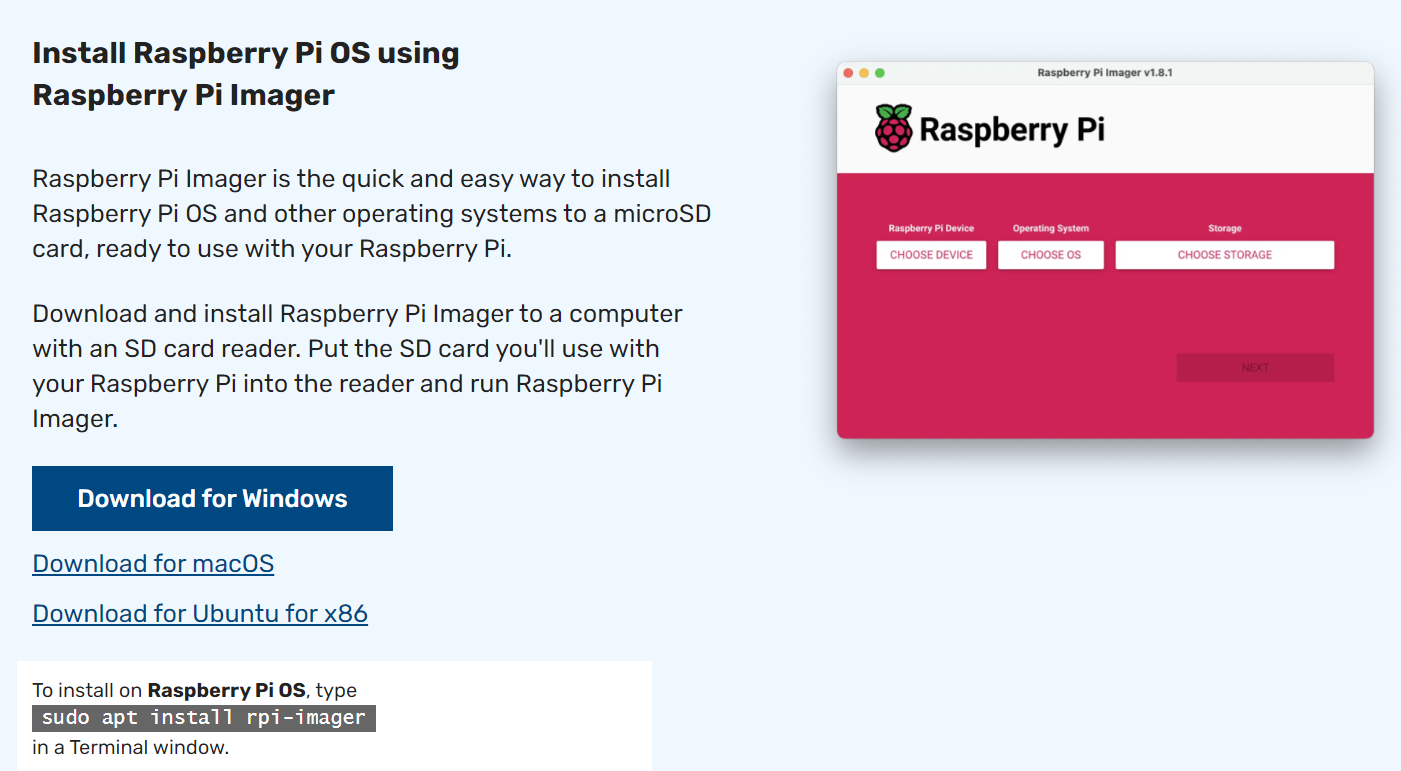
****

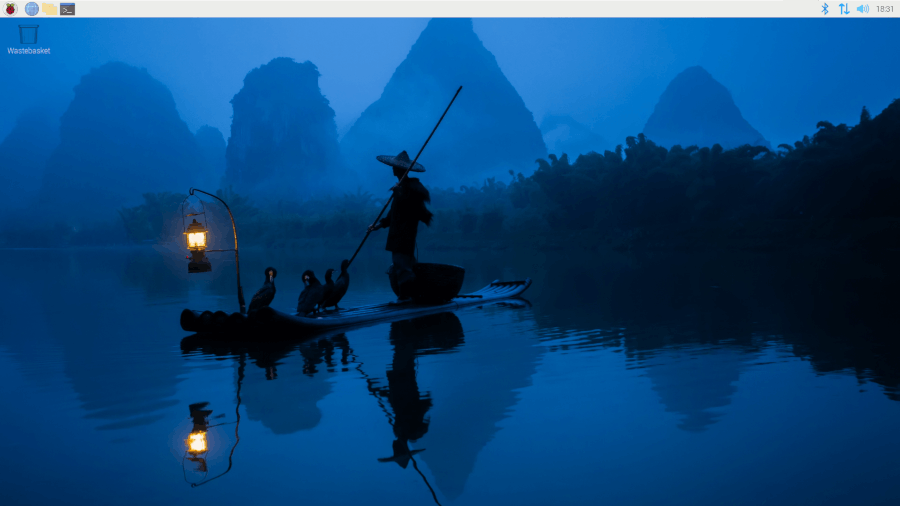
**Ficha técnica de la Raspberry Pi 5**

|  |  |
| --- | --- |
|  | Raspberry Pi 5 |
| CPU | Broadcom BCM2712  Quad-core Arm Cortex-A76 a 2,4 GHz |
| GPU | VideoCore VII  Soporta OpenGL ES 3.1 y Vulkan 1.2 |
| RAM | 8 GB LPDDR4X |
| Conectividad | Wi-Fi 5  Bluetooth 5.0 / BLE |
| Puertos | 2 x micro HDMI (hasta 2 x 4K 60Hz simultáneas)  2 x USB 3.0  2 x USB 2.0  1 x Gigabit Ethernet con PoE opcional  2 x MIPI de 4 pistas  1 x PCIe 2.0 x1  1 x GPIO 40 pines |
| Almacenamiento | Ranura microSD  Opción para unidades SSD M.2 (vía HAT opcional) |
| Otros | Botón de encendido y apagado  RTC (Real Time Clock)  Accesorios opcionales diversos |

**Instalación de SO de la Rasberry Pi**

https://www.raspberrypi.com/software/





Video de youtube de comparación de la Rasberry pi 4 y 5 e instalación del SO:

<https://www.youtube.com/watch?v=arvCnhpfE-c>

**Ingesta de datos de AEMET:**

**Para este proyecto vamos a extraer datos de AEMET a través de una pequeña aplicación en Python que se conectará al API de AEMET. Los pasos son los siguientes:**

**Paso 1: Obtención de la API\_KEY**

Para conectarse a la API necesitamos que AEMET nos proporcione una API\_KEY que recibiremos en el mail que le proporcionemos.

[**https://opendata.aemet.es/centrodedescargas/altaUsuario**](https://opendata.aemet.es/centrodedescargas/altaUsuario)

****

**Documentación de la API, datos que podemos obtener:**

[**https://opendata.aemet.es/dist/index.html?#/predicciones-especificas/Predicci%C3%B3n%20de%20monta%C3%B1a.%20Tiempo%20pasado**](https://opendata.aemet.es/dist/index.html?#/predicciones-especificas/Predicci%C3%B3n%20de%20monta%C3%B1a.%20Tiempo%20pasado)

**Paso 2: Crear un programa en Python para hacer una ingesta de datos**

Código básico para la obtención de datos que podéis usar (o no)



Pensad sobre que vais a querer hacer un análisis, los datos que decidáis ingestar ahora los vais a utilizar durante todo el curso. ¡Así que más vale que sobre que no que falte, no escatiméis, el BIG DATA es así!!

**Paso 3: Guardar los datos en un json o varios jsons**

Los datos ingestados deben guardarse para su posterior tratamiento. Adaptar vuestro programa para crear y guardar los jsons.

Importar las librerías json y datatime:



Seleccionar el archivo json:

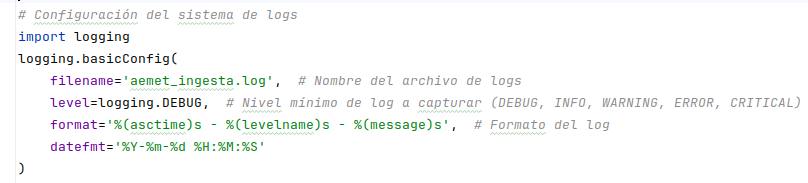


Abrir el fichero en forma de escritura:



**Paso 4: Monitoreo**

El programa debe tener manejo de errores y creación de logs.



* filename: El archivo donde se guardarán los logs (en este caso, aemet\_ingesta.log).
* level: Nivel de severidad de los mensajes que se registrarán. Con logging.DEBUG, capturas todo (desde DEBUG hasta CRITICAL).
* format: El formato de cada línea de log (hora, nivel de error, mensaje).
* datefmt: El formato de la fecha y hora.

Para escribir en un log utilizar:

* logging.info(): Para registrar el flujo normal del programa (inicio de solicitud, éxito en la obtención de datos, etc.).
* logging.error(): Para registrar cualquier error que ocurra durante el proceso (errores HTTP, problemas de conexión, etc.).
* logging.critical(): Para errores inesperados que podrían requerir atención urgente.

Bloque try – catch para el tratamiento de errores:

* requests.exceptions.HTTPError: Maneja errores HTTP como 404 o 500.
* requests.exceptions.ConnectionError: Maneja errores de conexión (no hay internet o problemas de red).
* requests.exceptions.Timeout: Maneja el caso en que la solicitud excede el tiempo de espera.
* requests.exceptions.RequestException: Captura cualquier otro error relacionado con la solicitud.
* Exception: Captura cualquier error inesperado que no esté relacionado directamente con la solicitud (por ejemplo, errores de tipo o sintaxis).

