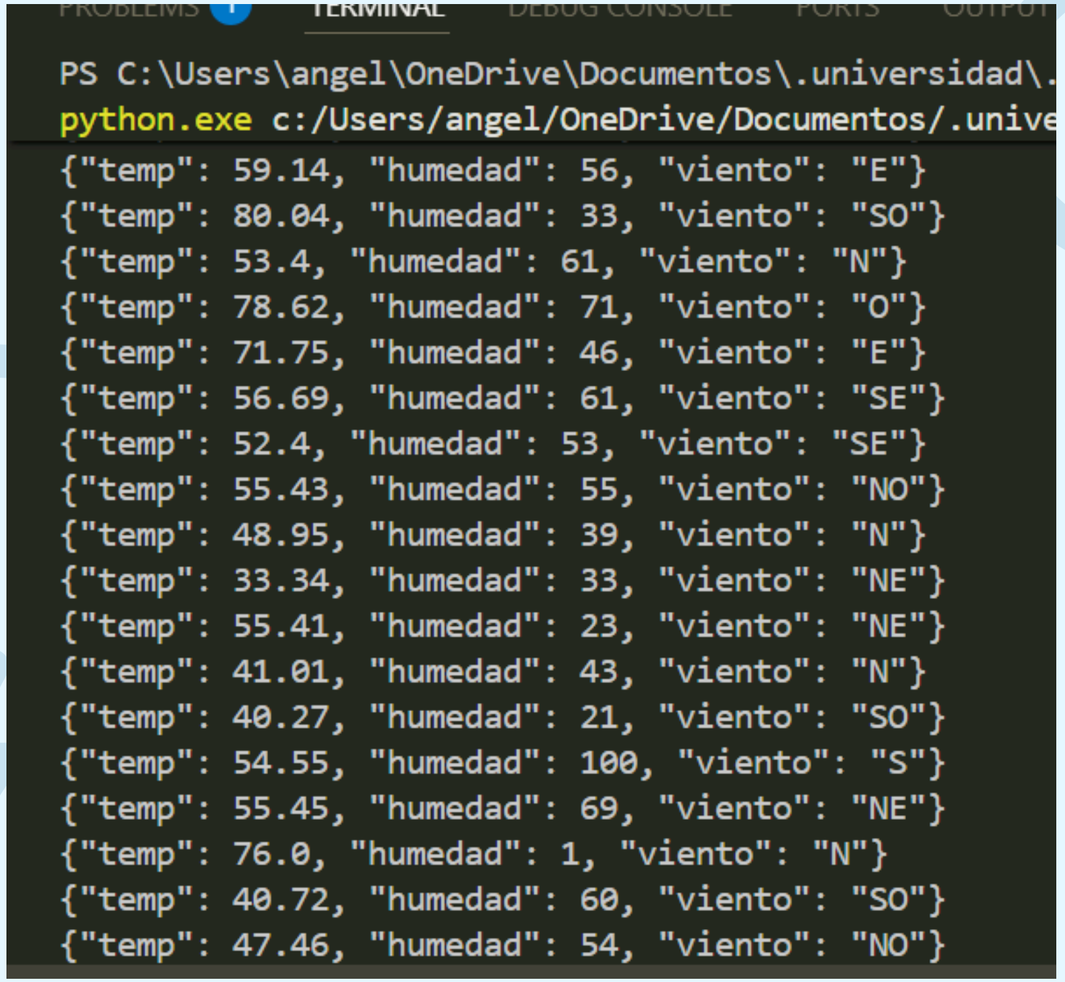
Laboratorio 7

Francis Aguilar, 22243
Angela García, 22869

Enlace del repositorio: https://github.com/angelargd8/Lab7-redes

3.1 Simulación de un sensor



En está parte, se simuló un sensor, creando un productor que generará el sensor de temperatura, de humedad relativa y de dirección de viento. teniendo en cuenta la distribución normal para la creación de los datos.

- **¿A qué capa pertenece JSON/SOAP según el Modelo OSI y por qué?**
  - Pertenecen a la capa 6, la de presentación, porque esta es la responsable de la traducción, codificación y formateo de los datos. Y la función de JSON y SOAP es convertir los datos a un formato estándar para aplicaciones.
- **¿Qué beneficios tiene utilizar un formato como JSON/SOAP?**
  - Los beneficios que tienen es la interoperabilidad porque permite que los diferentes sistemas en diferentes lenguajes y plataformas. Por otro lado, está la portabilidad porque pueden viajar en cualquier protocolo, como HTTP, Kafka, WebSockets y otros. Y por último, que tienen independencia de transporte, no depende de TCP o UDP, sino que puede ser usado en microservicios, colas de mensajes, Kafka, APIs y bases de datos

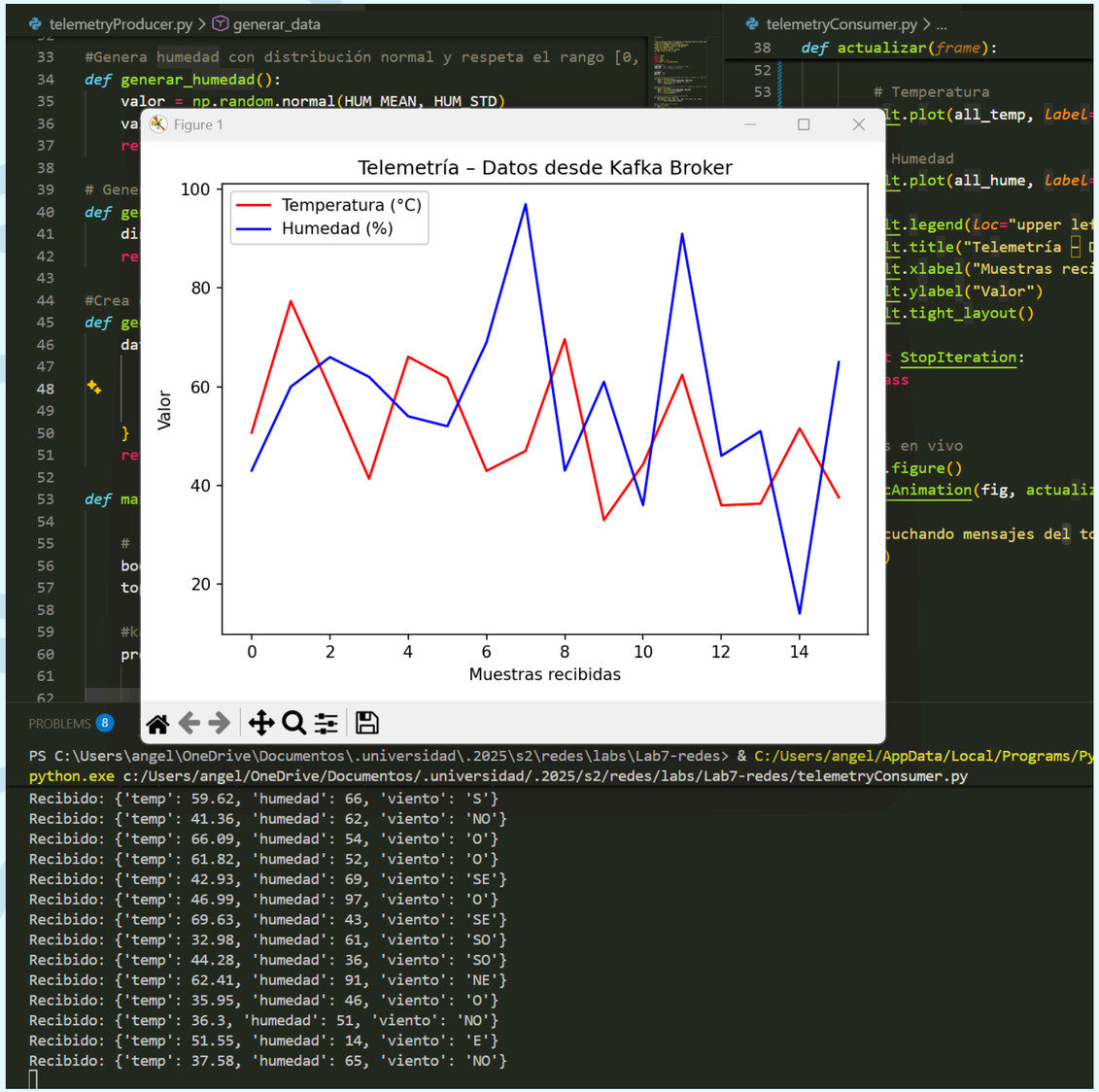
3.2 Envío de datos al Server Edge

```
PS C:\Users\angel\OneDrive\Documentos\.universidad\.2025\s2\redes\1
ams/Python/Python312/python.exe c:/Users/angel/OneDrive/Documentos/
yProducer.py
Enviando datos al servidor Kafka...

Enviando: {'temp': 42.93, 'humedad': 69, 'viento': 'SE'}
Enviando: {'temp': 46.99, 'humedad': 97, 'viento': 'O'}
Enviando: {'temp': 69.63, 'humedad': 43, 'viento': 'SE'}
Enviando: {'temp': 32.98, 'humedad': 61, 'viento': 'SO'}
Enviando: {'temp': 44.28, 'humedad': 36, 'viento': 'SO'}
Enviando: {'temp': 62.41, 'humedad': 91, 'viento': 'NE'}
```

Se modificó el productor para que enviará los datos al servidor Kafka entre cada 15 a 30 segundos. En donde este se queda corriendo y mandando datos hasta que es interrumpido. Para ello se creó un Kafka producer, que envia datos al boostrap server, teniendo un topic.

3.3 Consumir y desplegar datos Meteorológicos



Ya con la data publicada y se ha generado en topics
el consumer se suscribe al topic y esucucha los mensajes entrantes
siendo el consumidor el que tiene el rol de los elementos que estan detras del edge.

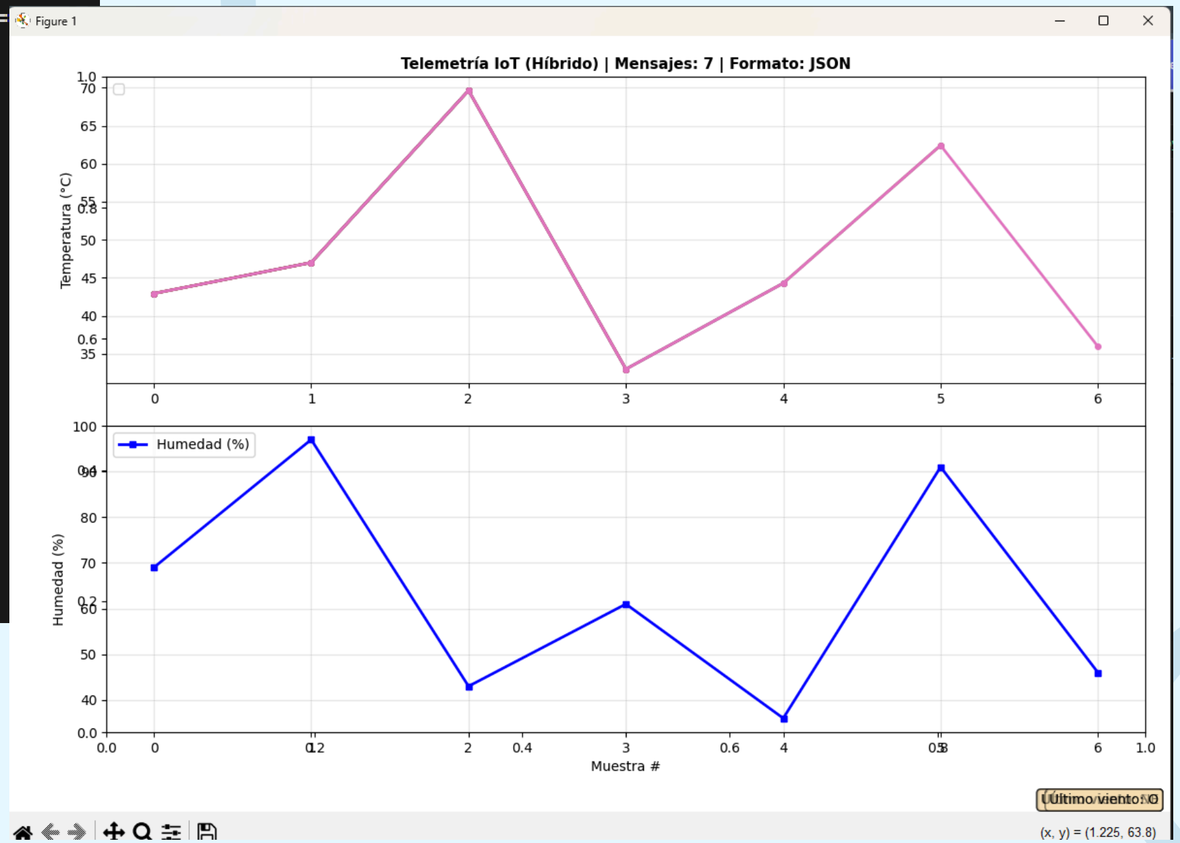
- **¿Qué ventajas y desventajas considera que tiene este acercamiento basado en Pub/Sub de Kafka?**
  - La ventaja que tiene es que al ser productores y consumidores, el productor no necesita saber quien lo consume y el consumidor no necesita saber cómo se produjo el mensaje. Por otro lado Kafka permite guardar los mensajes en el disco por un tiempo definido, entonces permite re-procesar datos, reconstruir estados y agregar nuevos consumidores sin perder historial. Una desventaja que tiene es que este no permite queries complejas, índices ni nada de eso porque no es una base de datos. Además que, este requiere de una infraestructura, por lo tanto, no es ideal para ambientes con bajo hardware, baja disponibilidad de red o dispositivos muy limitados.
- **¿Para qué aplicaciones tiene sentido usar Kafka? ¿Para cuáles no?**
  - Las aplicaciones en las que tiene sentido usar, es en aquellas que se requiere de procesamiento de datos en tiempo real, IoT, sensores, logs streaming, transacciones y telemetría. También en sistemas de monitoreo, piplines de datos, microservicios desacoplados, en la integración de múltiples productores y consumidores. Y en los que no son en aplicaciones pequeñas o simples, comunicación estrictamente síncrona, reemplazar una base de datos, sistema con latencias muy bajas, sistemas con cambios de red y IoT muy limitados.

3.4 IoT en entornos con restricciones

```
==============================
 MENSAJE #5
==============================
Datos originales:
   Temperatura: 78.84°C
   Humedad: 53%
   Viento: SE

Payload comprimido:
  Hex: 7B31AD
  Dec: [123, 49, 173]
  Bytes: b'{1\xad'
  Tamaño: 3 bytes

 Enviado correctamente
```



Se crea un encode y decode en el producer y consumer para poder comprimir los datos en las restricciones

- **¿Qué complejidades introduce el tener un payload restringido (pequeño)?**
  - Tener un payload restringido introduce varias complejidades importantes. En primer lugar, obliga a implementar funciones de encode y decode más elaboradas, ya que la información debe compactarse para caber en pocos bits. Esto también incrementa la probabilidad de cometer errores durante el empaquetado o la interpretación de los datos. Además, se pierde legibilidad y flexibilidad, porque el formato deja de ser intuitivo y cada campo debe estar cuidadosamente calculado y posicionado. Finalmente, un payload pequeño genera problemas de compatibilidad, ya que agregar nuevos campos o modificar el tamaño de los existentes puede romper el formato actual y exigir cambios en todos los dispositivos que consumen ese payload.
- **¿Cómo podemos hacer que el valor de temperatura quepa en 14 bits?**
  - Para que el valor de la temperatura quepa en 14 bits, se utiliza una estrategia basada en transformar la representación del número antes de codificarlo. Como necesitamos transmitir temperaturas entre 0 y 110 °C con dos decimales, primero se multiplica el valor original por 100 para convertirlo en un entero (por ejemplo, 25.67 °C pasa a 2567). Esto elimina los decimales sin perder precisión. El rango resultante va de 0 a 11000, y al calcular cuántos bits se requieren, obtenemos que $\log_2(11000) \approx 13.42$, por lo que 14 bits son suficientes para representar cualquier valor dentro de ese rango. Como $2^{14} = 16384$, se confirma que este tamaño de campo cubre completamente el rango necesario.

- **¿Qué sucedería si ahora la humedad también es tipo float con un decimal? ¿Qué decisiones tendríamos que tomar en ese caso?**
  - Si la humedad pasa de ser un entero a un valor flotante con un decimal, aumenta de forma significativa la cantidad de valores posibles que deben representarse y, por lo tanto, la cantidad de bits necesarios. En lugar de los 7 bits actuales para humedad entera, un valor entre 0.0 y 100.0 con un decimal genera 1001 combinaciones posibles, lo que requiere 10 bits. Esto provoca que el payload ya no quepa en los 24 bits disponibles, pasando de [14 temp][7 humedad][3 viento] = 24 bits a [14 temp][10 humedad][3 viento] = 27 bits. Para ajustarnos nuevamente al límite, hay que tomar decisiones de compromiso. Una opción viable es reducir la precisión de la temperatura, bajando de dos decimales a uno. Al multiplicar la temperatura por 10 en lugar de por 100, su rango pasa a 0–1100, lo cual cabe en 11 bits.
- **¿Qué parámetros o herramientas de Kafka podrían ayudarnos si las restricciones fueran aún más fuertes?**
  - Si las restricciones del payload fueran aún más fuertes, Kafka ofrece mecanismos como la compresión para reducir significativamente el tamaño de los mensajes. Algoritmos como gzip, snappy, lz4 y zstd permiten equilibrar compresión, velocidad y uso de CPU según las necesidades: gzip comprime más pero es lento, mientras que snappy y lz4 son rápidas y adecuadas para IoT; zstd ofrece el mejor balance general. Al aplicar compresión, especialmente en batch, es posible reducir el tamaño de los mensajes hasta en un 70%, ayudando a cumplir con límites estrictos sin perder demasiada precisión.