Universidad del Valle de Guatemala

Facultad de Ingeniería

Laboratorio 9

Redes

Mariana David 201055

Pablo Escobar 20936

Guatemala 12 de noviembre del 2023

Desarrollo

3.1 Respuestas de las preguntas otorgadas

Responda: ¿A qué capa pertenece JSON/SOAP según el Modelo OSI y porqué?

JSON y SOAP no son protocolos de red y, por lo tanto, no pertenecen directamente a ninguna capa del Modelo OSI. Sin embargo, se pueden utilizar en varias capas del modelo, dependiendo del contexto de implementación. En el caso de SOAP que se utiliza comúnmente en la capa de aplicación (Capa 7) del Modelo OSI, ya que define un protocolo para la comunicación entre aplicaciones a través de mensajes XML. Sin embargo, también puede ser encapsulado en protocolos de transporte como HTTP, lo que lo llevaría a la capa de transporte (Capa 4).

Por otra parte, considerando el caso del JSON, al ser un formato de intercambio de datos más ligero, generalmente se utilzaría en la capa de aplicación (Capa 7) del Modelo OSI. Además, que puede ser transportado a través de diversos protocolos de transporte, como HTTP, pero no está vinculado directamente a una capa específica.

Responda: ¿Qué beneficios tiene utilizar un formato como JSON/SOAP?

La elección entre JSON y SOAP ofrece beneficios significativos en el ámbito de la interoperabilidad, legibilidad y estructuración de datos. Ambos formatos son independientes del lenguaje y la plataforma, facilitando la comunicación eficiente entre sistemas diversos. Además, su legibilidad para los humanos simplifica el desarrollo y la depuración, mientras que la capacidad de organizar la información de manera estructurada facilita el procesamiento y manipulación de los datos.

La extensibilidad de JSON y SOAP permite la incorporación de campos adicionales o elementos según sea necesario, sin comprometer la compatibilidad con versiones anteriores. Asimismo, la compatibilidad con una amplia variedad de lenguajes de

programación y plataformas hace que ambos formatos sean ideales para integrar sistemas distribuidos.

En particular, SOAP sigue estándares de la industria y ofrece funcionalidades como seguridad y transacciones, lo que puede ser especialmente beneficioso en entornos empresariales. En resumen, la elección entre JSON y SOAP se basará en los requisitos específicos del sistema y en el contexto de implementación.

3.2 Envío de Datos al Server Edge

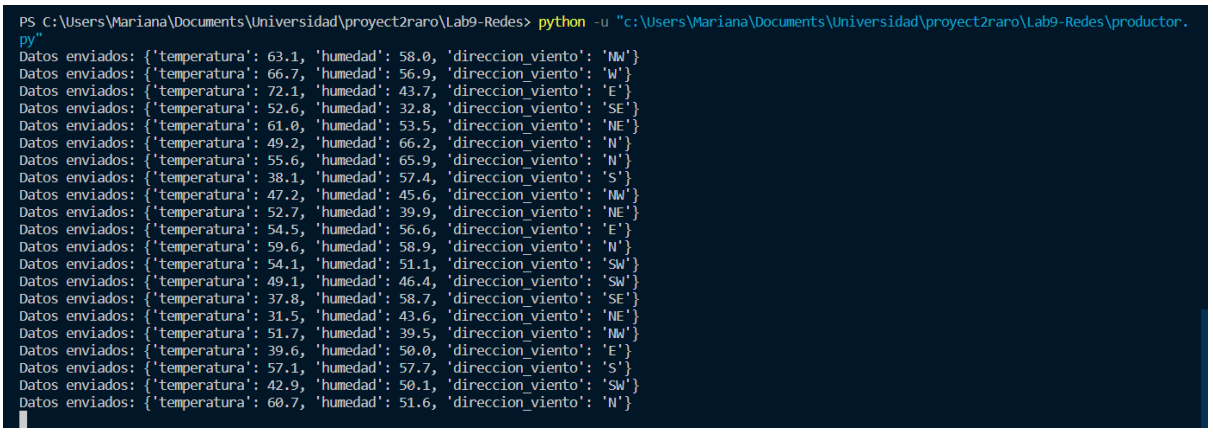


Figura 1. Implementación exitosa de envío de datos al server

3.3 Consumir y desplegar datos meteorológicos

Responda: ¿Qué ventajas y desventajas considera que tiene este acercamiento basado en Pub/Sub de Kafka?

En el contexto de Kafka, el enfoque de Publicar/Suscribir (Pub/Sub) presenta diversas ventajas. Primero, posibilita una escalabilidad eficiente al permitir que productores y consumidores se escalen de manera independiente. Además, ofrece flexibilidad al facilitar la conexión entre sistemas distribuidos y heterogéneos, eliminando la necesidad de que los productores y consumidores se conozcan directamente. La tolerancia a fallos se fortalece mediante la replicación de datos en varios nodos, asegurando la disponibilidad incluso en situaciones de fallos. Asimismo, la arquitectura Pub/Sub permite el desacoplamiento entre productores y consumidores, posibilitando cambios en uno sin afectar al otro.

No obstante, este enfoque también presenta desventajas a considerar. La implementación y gestión de una arquitectura Pub/Sub pueden volverse más complejas en comparación con enfoques más simples, lo que podría implicar un mayor esfuerzo en términos de desarrollo y mantenimiento. Además, puede haber un cierto overhead asociado con la necesidad de mantener la infraestructura de Kafka, especialmente en casos donde la comunicación distribuida no sea imperativa. En resumen, la elección de adoptar el enfoque Pub/Sub de Kafka dependerá de los requisitos específicos y las consideraciones de complejidad en cada contexto de implementación.

Responda: ¿Para qué aplicaciones tiene sentido usar Kafka? ¿Para cuáles no?
La utilización de Kafka resulta pertinente en diversas aplicaciones, especialmente aquellas que involucran sistemas distribuidos, como microservicios o entornos IoT, donde se requiere una comunicación eficiente entre componentes distribuidos. Además, Kafka es idóneo para aplicaciones que demandan procesamiento en tiempo real, como el análisis de telemetría o registros de eventos, y para aquellas de alta criticidad que necesitan una tolerancia a fallos robusta.

En contraste, el uso de Kafka podría no ser adecuado para aplicaciones más simples y de menor envergadura que no necesitan comunicación distribuida. En proyectos de pequeña escala, donde la complejidad de Kafka supera los beneficios, su implementación puede considerarse excesiva. Asimismo, en situaciones donde la comunicación síncrona directa entre componentes es más apropiada y no se requiere la asincronía proporcionada por Kafka, su adopción podría no ser la opción más eficiente.

Por ende, la elección de utilizar o no Kafka dependerá de los requisitos específicos de cada aplicación, y mientras ofrece soluciones eficientes para sistemas distribuidos y aplicaciones en tiempo real, podría no ser la mejor opción para proyectos más simples o de menor escala.

```
PS C:\Users\Mariana\Documents\Universidad\proyect2raro\Lab9-Redes> python -u "c:\Users\Mariana\Documents\Universidad\proyect2raro\Lab9-Redes\consumidor
.py"
Datos recibidos: {'temperatura': 48.5, 'humedad': 51, 'direccion_viento': 'NW'}
Datos recibidos: {'temperatura': 59.7, 'humedad': 45, 'direccion_viento': 'SW'}
Datos recibidos: {'temperatura': 63.400000000000006, 'humedad': 69, 'direccion_viento': 'E'}
Datos recibidos: {'temperatura': 53.2, 'humedad': 41, 'direccion_viento': 'W'}
Datos recibidos: {'temperatura': 37.599999999999994, 'humedad': 50, 'direccion_viento': 'SE'}
Datos recibidos: {'temperatura': 37.099999999999994, 'humedad': 50, 'direccion_viento': 'E'}
Datos recibidos: {'temperatura': 75.8, 'humedad': 40, 'direccion_viento': 'NE'}
Datos recibidos: {'temperatura': 42.7, 'humedad': 48, 'direccion_viento': 'NE'}
Datos recibidos: {'temperatura': 52.7, 'humedad': 50, 'direccion_viento': 'SW'}
Datos recibidos: {'temperatura': 64.7, 'humedad': 52, 'direccion_viento': 'N'}
Datos recibidos: {'temperatura': 58.3, 'humedad': 65, 'direccion_viento': 'N'}
Datos recibidos: {'temperatura': 48.7, 'humedad': 57, 'direccion_viento': 'E'}
Datos recibidos: {'temperatura': 63.400000000000006, 'humedad': 37, 'direccion_viento': 'NE'}
Datos recibidos: {'temperatura': 46.900000000000006, 'humedad': 39, 'direccion_viento': 'W'}
Datos recibidos: {'temperatura': 41.599999999999994, 'humedad': 55, 'direccion_viento': 'N'}
Datos recibidos: {'temperatura': 62.0, 'humedad': 44, 'direccion_viento': 'NW'}
Datos recibidos: {'temperatura': 63.2, 'humedad': 43, 'direccion_viento': 'SE'}
```

Figura 2. Resultados de los datos por parte del consumidor

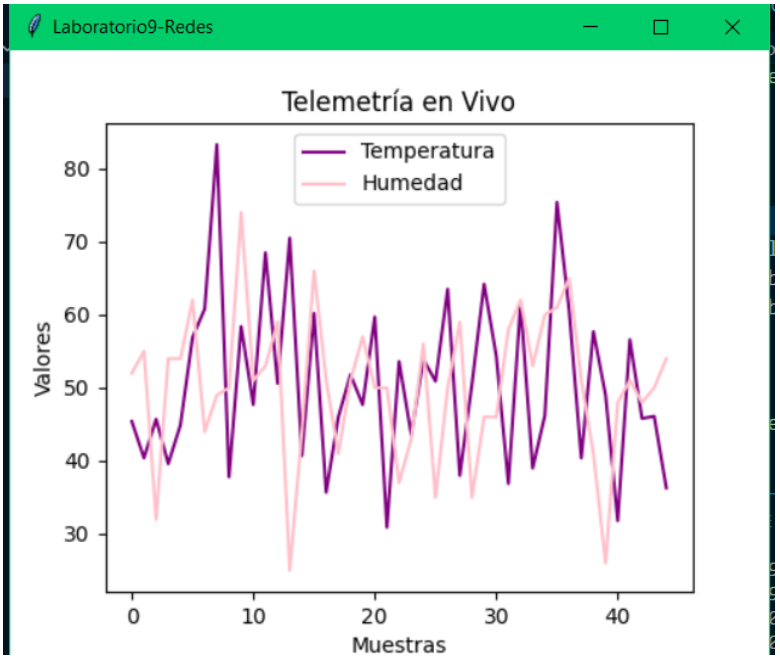


Figura 3. Diagrama de representación de resultados

3.4 Io T en Entornos con restricciones

Resultados de cambios

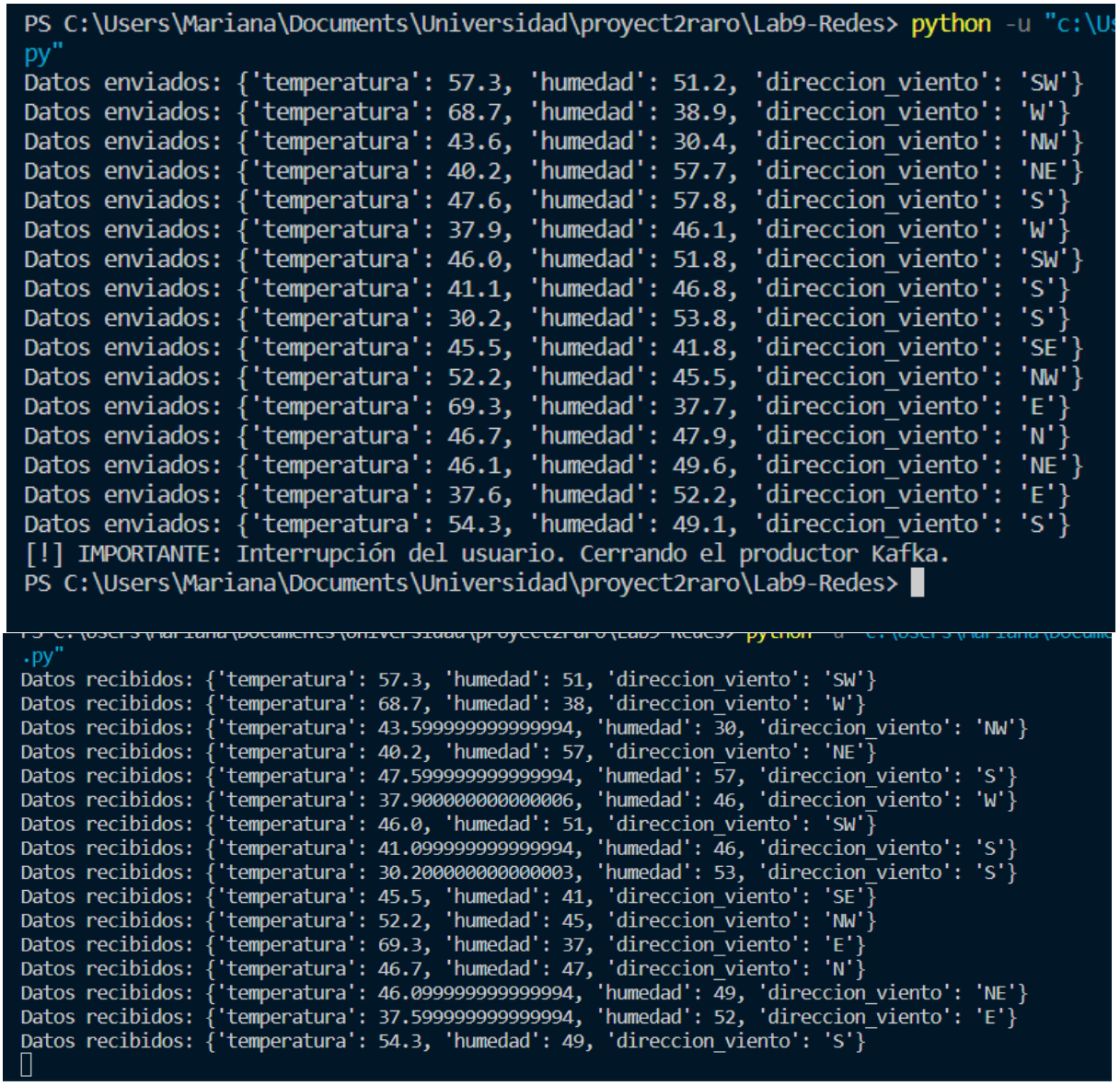
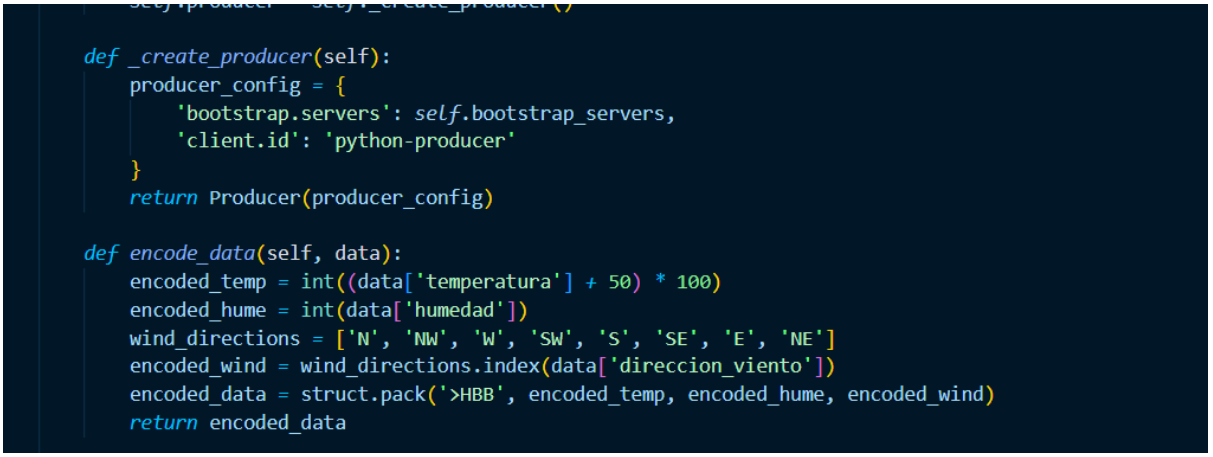


Figura 4. Proceso de encodign y decoding en los proceso de comunicación entre productor y consumidor



```
def _create_producer(self):
    producer_config = {
        'bootstrap.servers': self.bootstrap_servers,
        'client.id': 'python-producer'
    }
    return Producer(producer_config)

def encode_data(self, data):
    encoded_temp = int((data['temperatura'] + 50) * 100)
    encoded_hume = int(data['humedad'])
    wind_directions = ['N', 'NW', 'W', 'SW', 'S', 'SE', 'E', 'NE']
    encoded_wind = wind_directions.index(data['direccion_viento'])
    encoded_data = struct.pack('>HBB', encoded_temp, encoded_hume, encoded_wind)
    return encoded_data
```

Figura 5. Resultado con cambios

Responda: ¿Qué complejidades introduce el tener un payload restringido (pequeño)?

La principal complejidad de tener un payload restringido, como en el caso de entornos con limitaciones como LoRa y Sigfox, radica en la necesidad de compactar la información esencial dentro de un espacio mínimo. Al limitar el tamaño del mensaje, se requiere una cuidadosa gestión de los datos y una estrategia eficiente de codificación para garantizar que toda la información relevante pueda ser transmitida. Esto puede añadir complejidad al diseño del protocolo de comunicación y la estructura de los mensajes, ya que cada bit se convierte en un recurso valioso y debe utilizarse de manera óptima.

Responda: ¿Cómo podemos hacer que el valor de temperatura quepa en 14 bits?

Para hacer que el valor de temperatura quepa en 14 bits, se puede aplicar una técnica de escalamiento y ajuste del rango. Dado que el rango original de temperatura es de -50 a 50 grados Celsius, podemos trasladar este rango a valores positivos, ajustando así el espacio necesario. Posteriormente, se puede realizar una conversión y escalamiento para representar este nuevo rango en los 14 bits disponibles. Al utilizar técnicas inteligentes de codificación, como la representación binaria o mapeo lineal, se logra comprimir la información sin perder precisión significativa.

Responda: ¿Qué sucedería si ahora la humedad también es tipo float con un decimal? ¿Qué decisiones tendríamos que tomar en ese caso?

Si la humedad también se convierte en un valor de tipo float con un decimal, la complejidad del diseño se incrementaría notablemente. Aunque la parte entera de la humedad podría ser representada en 7 bits, el decimal adicional requeriría más espacio. Se podrían considerar estrategias, como redondear el decimal o aplicar una codificación especial, pero esto podría afectar la precisión de la medición. La toma de decisiones se centraría en equilibrar la precisión necesaria con las restricciones del payload, comprometiendo posiblemente la exactitud de los datos transmitidos.

Responda: ¿Qué parámetros o herramientas de Kafka podrían ayudarnos si las restricciones fueran aún más fuertes?

En situaciones con restricciones más fuertes, donde el tamaño del payload es crítico, se podría explorar el uso de herramientas y configuraciones específicas de Kafka para optimizar la eficiencia de la transmisión. Por ejemplo, se podrían ajustar parámetros relacionados con la compresión de datos en Kafka para reducir el tamaño de los mensajes transmitidos. Además, el particionamiento de los topics y la gestión eficiente de los offsets podrían ser estrategias para maximizar la utilización del espacio disponible. La comprensión profunda de las capacidades y configuraciones de Kafka se vuelve esencial en entornos con restricciones más estrictas.

Anexos:

LinkRepositorio: https://github.com/Marianadaso3/Lab9-Redes

Link Presentación:
https://www.canva.com/design/DAFz0xTw5DI/V_pbjZyKFA2xWQ8eOijunw/edit?utm_content=DAFz0xTw5DI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton