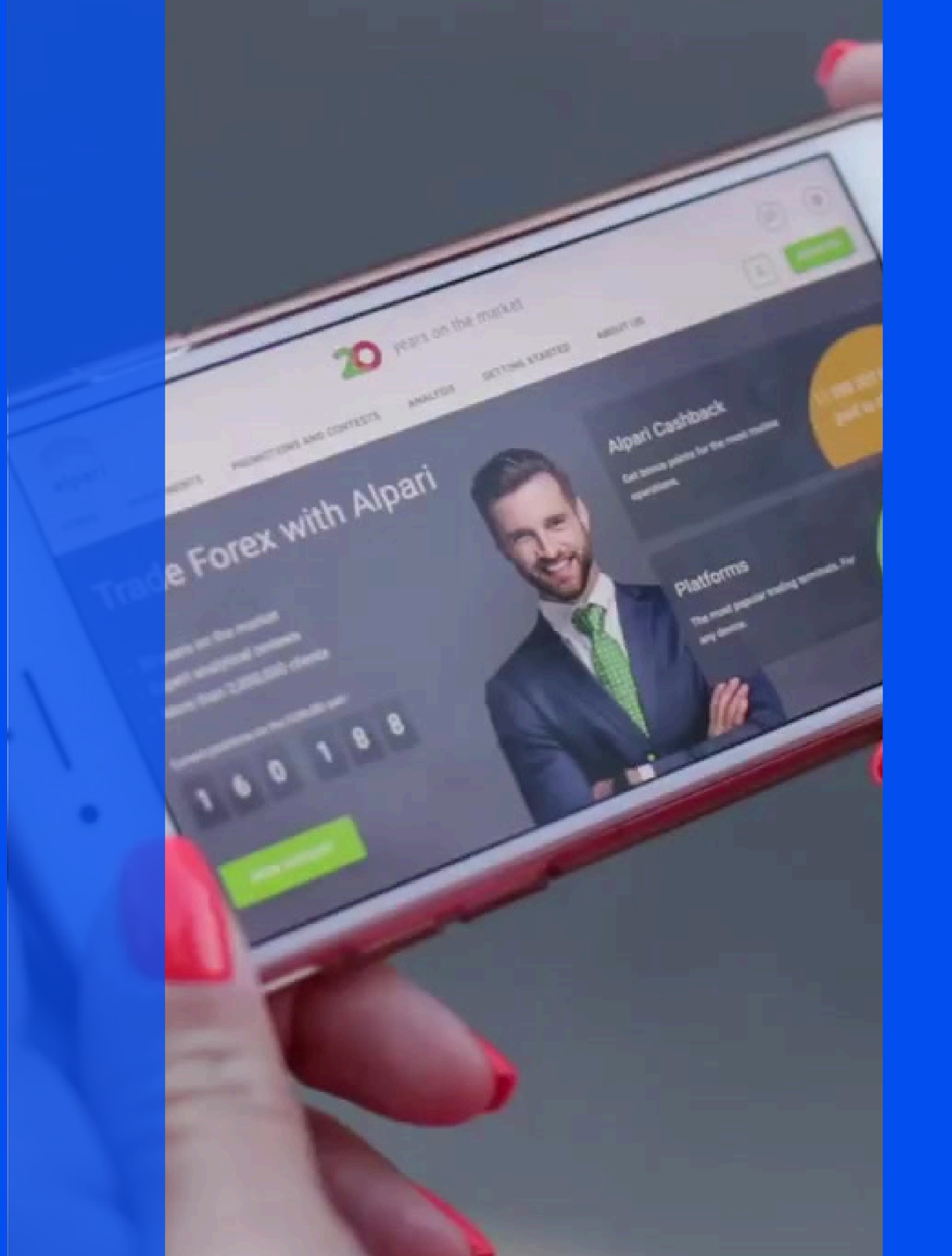
LABORATORIO 7

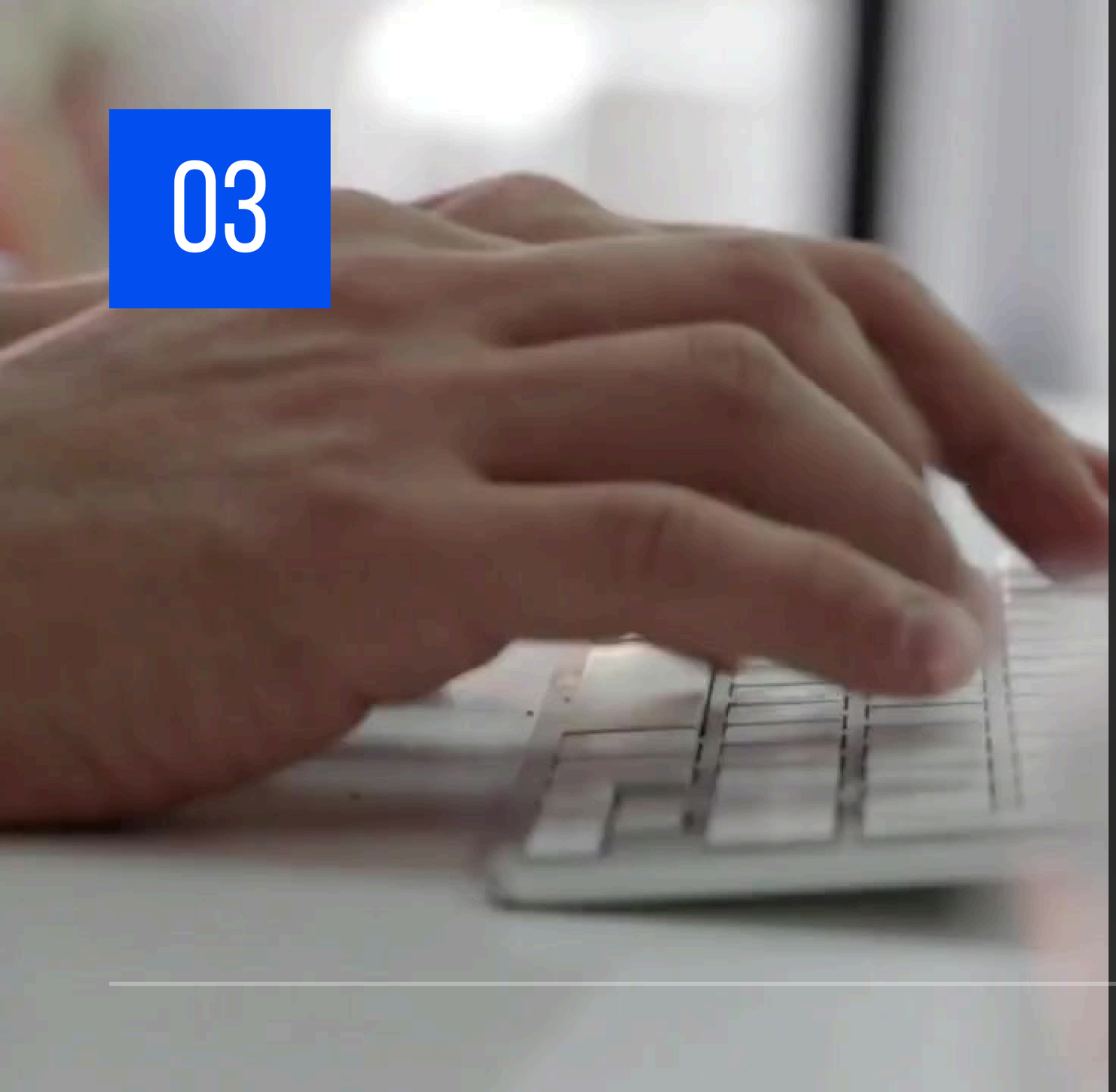
REDES
CESAR LÓPEZ # 22535
JOAQUÍN CAMPOS # 22155
GITHUB:
HTTPS://GITHUB.COM/CZAR272/LAB7_REDES.GIT

01

OBJETIVOS DEL LABORATORIO

- SIMULAR INFORMACIÓN METEOROLÓGICA USANDO DISTRIBUCIONES PROBABILÍSTICAS.
- CREAR UN PRODUCER EN PYTHON PARA ENVIAR DATOS A UN TOPIC PROPIO EN KAFKA.
- CREAR UN CONSUMER QUE INTERPRETA LOS DATOS Y LOS GRAFICA EN “TIEMPO REAL”.
- IMPLEMENTAR UN SISTEMA DE CODIFICACIÓN COMPACTA DE 3 BYTES POR LECTURA.
- ANALIZAR VENTAJAS/DESVENTAJAS DE USAR KAFKA EN ENTORNOS IOT.





03

INFRAESTRUCTURA PROPORCIONADA

SERVIDOR KAFKA CON:

HOST: IOT.REDESUVG.CLOUD
IP: 147.182.219.133
PUERTO: 9092

TECNOLOGÍAS UTILIZADAS

- PYTHON

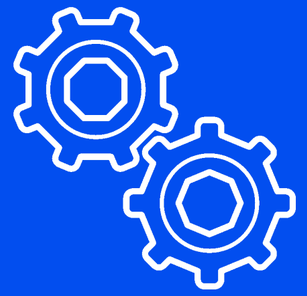
04

MATPLOTLIB

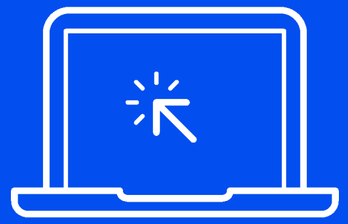
KAFKA-PYTHON

NUMPY

ARQUITECTURA GENERAL DE LA SOLUCIÓN



SENSOR SIMULADO: GENERA TEMPERATURA, HUMEDAD Y VIENTO.



PRODUCER: ENVÍA DATOS AL TOPIC DE KAFKA.

CONSUMER: RECIBE DATOS Y LOS GRAFICA EN “TIEMPO REAL”.

SIMULACIÓN DE SENSORES

- TEMPERATURA: DISTRIBUCIÓN NORMAL TRUNCADA A [0, 110] CON DECIMALES.
- HUMEDAD: DISTRIBUCIÓN NORMAL TRUNCADA A [0, 100], ENTERO.
- DIRECCIÓN DEL VIENTO: UNIFORMEMENTE ENTRE 8 VALORES: N, NO, O, SO, S, SE, E, NE.

```python
# Temperatura: distribucion normal en [0,110]
mean_temp = 25.0
std_temp = 10.0
temp = np.random.normal(loc=mean_temp, scale=std_temp)
temp = max(0.0, min(110.0, temp))  # recortar a [0,110]
temp = round(temp, 2)

# Humedad relativa: normal + enteros [0,100]
mean_hum = 60.0
std_hum = 15.0
hum = np.random.normal(loc=mean_hum, scale=std_hum)
hum = int(round(max(0, min(100, hum))))

# Direccion del viento: uniforme sobre las 8 opciones
wind = random.choice(WIND_DIRECTIONS)

return {"temperatura": temp, "humedad": hum, "direccion_viento": wind}
```

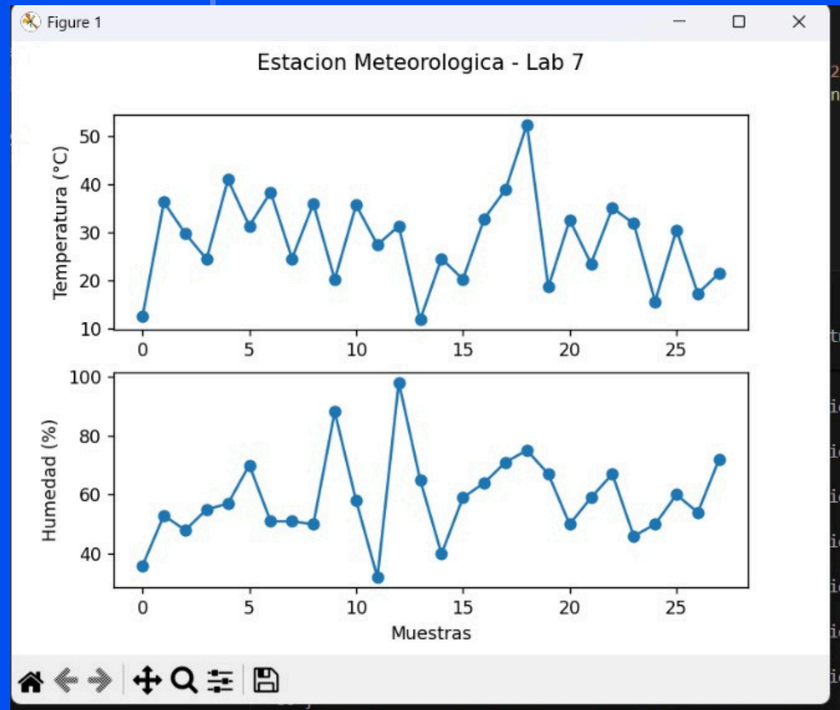
06

PRODUCER (JSON)

```python
TOPIC = "22535"


def main():
    producer = KafkaProducer(
        bootstrap_servers=["iot.redesuvg.cloud:9092"],
        value_serializer=lambda v: json.dumps(v).encode("utf-8"),
    )

    print(f"Enviando datos al topic '{TOPIC}'...")

    try:
        while True:
            lectura = generar_lectura()
            print("Enviando:", lectura)

            producer.send(TOPIC, value=lectura)
            producer.flush()

            delay = random.randint(5, 10)
            time.sleep(delay)
    except KeyboardInterrupt:
        print("\nDeteniendo producer...")
    finally:
        producer.close()
```

```
(venv) PS C:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Redes\Labs\Lab 7\Lab7_Redes> python produc
Enviando datos al topic '22535'...
Enviando: {'temperatura': 24.43, 'humedad': 70, 'direccion_viento': 'S'}
Enviando: {'temperatura': 23.93, 'humedad': 75, 'direccion_viento': 'NE'}
Enviando: {'temperatura': 17.61, 'humedad': 56, 'direccion_viento': 'O'}
Enviando: {'temperatura': 28.71, 'humedad': 87, 'direccion_viento': 'SO'}
Enviando: {'temperatura': 23.44, 'humedad': 47, 'direccion_viento': 'S'}
Enviando: {'temperatura': 23.84, 'humedad': 59, 'direccion_viento': 'SE'}
Enviando: {'temperatura': 23.94, 'humedad': 61, 'direccion_viento': 'NO'}
Enviando: {'temperatura': 42.44, 'humedad': 53, 'direccion_viento': 'NO'}
Enviando: {'temperatura': 45.29, 'humedad': 50, 'direccion_viento': 'N'}
Enviando: {'temperatura': 7.43, 'humedad': 74, 'direccion_viento': 'O'}
Enviando: {'temperatura': 17.57, 'humedad': 83, 'direccion_viento': 'S'}
Enviando: {'temperatura': 10.62, 'humedad': 12, 'direccion_viento': 'S'}
Enviando: {'temperatura': 15.83, 'humedad': 96, 'direccion_viento': 'E'}
Enviando: {'temperatura': 23.65, 'humedad': 59, 'direccion_viento': 'SO'}
Enviando: {'temperatura': 24.03, 'humedad': 40, 'direccion_viento': 'O'}
Enviando: {'temperatura': 19.81, 'humedad': 39, 'direccion_viento': 'SE'}
Enviando: {'temperatura': 45.1, 'humedad': 53, 'direccion_viento': 'NO'}
Enviando: {'temperatura': 32.95, 'humedad': 46, 'direccion_viento': 'N'}
```

CONSUMER (JSON)




```
Enviando: {'temperatura': 21.43, 'humedad': 72, 'direccion_viento': '
SE'}
Enviando: {'temperatura': 27.17, 'humedad': 64, 'direccion_viento': '
N'}
```

```
NO'}
Recibido: {'temperatura': 18.54, 'humedad': 67, 'direccion_viento': '
O'}
Recibido: {'temperatura': 32.57, 'humedad': 50, 'direccion_viento': '
N'}
Recibido: {'temperatura': 23.42, 'humedad': 59, 'direccion_viento': '
NO'}
Recibido: {'temperatura': 35.04, 'humedad': 67, 'direccion_viento': '
O'}
Recibido: {'temperatura': 31.79, 'humedad': 46, 'direccion_viento': '
NO'}
Recibido: {'temperatura': 15.55, 'humedad': 50, 'direccion_viento': '
S'}
Recibido: {'temperatura': 30.47, 'humedad': 60, 'direccion_viento': '
S'}
Recibido: {'temperatura': 17.18, 'humedad': 54, 'direccion_viento': '
SO'}
Recibido: {'temperatura': 21.43, 'humedad': 72, 'direccion_viento': '
SE'}
Recibido: {'temperatura': 27.17, 'humedad': 64, 'direccion_viento': '
N'}
```

```python
# Guardamos historia de lecturas
temps = []
hums = []
winds = []

# Configuracion inicial de la grafica
plt.ion()
fig, (ax1, ax2) = plt.subplots(2, 1)
fig.suptitle("Estacion Meteorologica - Lab 7")

for msg in consumer:
    data = msg.value
    print("Recibido:", data)

    temps.append(data["temperatura"])
    hums.append(data["humedad"])
    winds.append(data["direccion_viento"])

    if len(temps) > 50:
        temps.pop(0)
        hums.pop(0)
        winds.pop(0)

    # Actualizar grafica
    ax1.clear()
    ax2.clear()

    ax1.plot(temps, marker="o")
    ax1.set_ylabel("Temperatura (°C)")

    ax2.plot(hums, marker="o")
    ax2.set_ylabel("Humedad (%)")
    ax2.set_xlabel("Muestras")

    plt.pause(0.01)
```

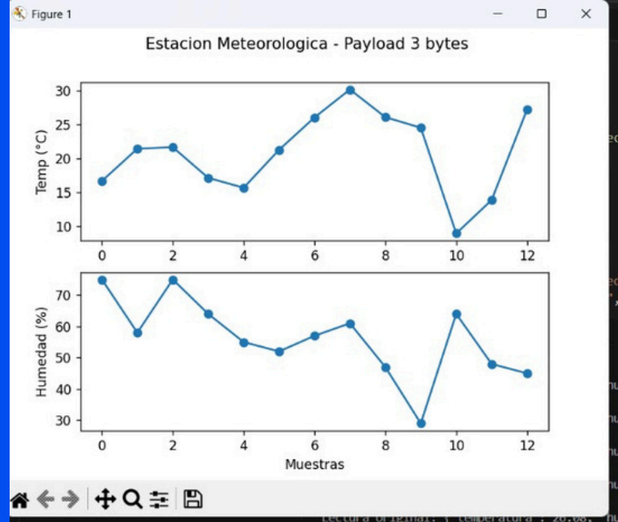
```python
def main():
    consumer = KafkaConsumer(
        TOPIC,
        bootstrap_servers=["iot.redesuvg.cloud:9092"],
        auto_offset_reset="latest",
        enable_auto_commit=True,
        group_id="grupo-lab7",
        value_deserializer=lambda m: json.loads(m.decode("utf-8")),
    )
```

PRODUCER (PAYLOAD DE 3 BYTES)

```
(venv) PS C:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Redes\Labs\Lab 7\Lab7_Redes> python producer
_bytes.py
[3 BYTES] Enviando datos al topic '22535'... Ctrl+C para salir.
Lectura original: {'temperatura': 34.84, 'humedad': 61, 'direccion_viento': 'NE'} -> bytes: b'6q\xef'
Lectura original: {'temperatura': 29.34, 'humedad': 55, 'direccion_viento': 'E'} -> bytes: b'-\xd9\xbe'
Lectura original: {'temperatura': 38.37, 'humedad': 65, 'direccion_viento': 'SE'} -> bytes: b';\xf6\r'
Lectura original: {'temperatura': 24.61, 'humedad': 75, 'direccion_viento': 'NO'} -> bytes: b'&vY'
Lectura original: {'temperatura': 25.67, 'humedad': 52, 'direccion_viento': 'E'} -> bytes: b'(\x1d\xa6'
Lectura original: {'temperatura': 21.6, 'humedad': 70, 'direccion_viento': 'O'} -> bytes: b'!\xc22'
Lectura original: {'temperatura': 23.75, 'humedad': 32, 'direccion_viento': 'SO'} -> bytes: b'%\x1d\x03'
Lectura original: {'temperatura': 16.14, 'humedad': 38, 'direccion_viento': 'NE'} -> bytes: b'\x1997'
Lectura original: {'temperatura': 5.62, 'humedad': 36, 'direccion_viento': 'N'} -> bytes: b'\x08\xc9 '
Lectura original: {'temperatura': 28.13, 'humedad': 64, 'direccion_viento': 'SE'} -> bytes: b'+\xf6\x05'
Lectura original: {'temperatura': 32.45, 'humedad': 66, 'direccion_viento': 'NE'} -> bytes: b'2\xb6\x17'
Lectura original: {'temperatura': 18.91, 'humedad': 55, 'direccion_viento': 'NE'} -> bytes: b'\x1d\x8d\xbf'
Lectura original: {'temperatura': 33.81, 'humedad': 78, 'direccion_viento': 'N'} -> bytes: b'4\xd6p'
Lectura original: {'temperatura': 29.63, 'humedad': 55, 'direccion_viento': 'NE'} -> bytes: b'.M\xbf'
Lectura original: {'temperatura': 14.98, 'humedad': 45, 'direccion_viento': 'SO'} -> bytes: b'\x17ik'
Lectura original: {'temperatura': 29.56, 'humedad': 86, 'direccion_viento': 'N'} -> bytes: b'.2\xb0'
ento': 'NE'} -> bytes: b'\x1a\xbe\x07'
Lectura original: {'temperatura': 15.65, 'humedad': 55, 'direccion_viento': 'N'} -> bytes: b'\x18u\xb8'
```

```python
TOPIC = "22535"


def main():
    producer = KafkaProducer(bootstrap_servers=["iot.redesuvg.cloud:9092"])

    print(f"[3 BYTES] Enviando datos al topic '{TOPIC}'... Ctrl+C para salir.")

    try:
        while True:
            lectura = generar_lectura()
            payload = encode_lectura(lectura)

            print("Lectura original:", lectura, "-> bytes:", payload)

            producer.send(TOPIC, value=payload)
            producer.flush()

            delay = random.randint(15, 30)
            time.sleep(delay)
    except KeyboardInterrupt:
        print("\nDeteniendo producer...")
    finally:
        producer.close()
```

CONSUMER (PAYLOAD DE 3 BYTES)




```
producer_json.py  consumer_json.py  producer_bytes.py U  consumer_bytes.py U
ectura
edesuvg.cloud:9092"],
PORTS  GITLENS  python
(venv) PS C:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Redes\Labs\Lab 7\Lab7_Redes> python consumer_bytes.py
'humedad': 55, 'direccion_viento': 'N'}
Bytes recibidos: b'I5\xa5' -> decodificado: {'temperatura': 21.25, 'humedad': 52, 'direccion_viento': 'SE'}
Bytes recibidos: b'(\xb1\xc9' -> decodificado: {'temperatura': 26.04, 'humedad': 57, 'direccion_viento': 'NO'}
Bytes recibidos: b'/%\xec' -> decodificado: {'temperatura': 30.17, 'humedad': 61, 'direccion_viento': 'S'}
Bytes recibidos: b'(\xc1~' -> decodificado: {'temperatura': 26.08, 'humedad': 47, 'direccion_viento': 'E'}
Bytes recibidos: b'&P\xea' -> decodificado: {'temperatura': 24.52, 'humedad': 29, 'direccion_viento': 'O'}
Bytes recibidos: b'\r\xea\x05' -> decodificado: {'temperatura': 8.9, 'humedad': 64, 'direccion_viento': 'SE'}
Bytes recibidos: b'\x15\x9d\x81' -> decodificado: {'temperatura': 13.83, 'humedad': 48, 'direccion_viento': 'NO'}
Bytes recibidos: b'*\x89k' -> decodificado: {'temperatura': 27.22, 'humedad': 45, 'direccion_viento': 'SO'}
Bytes recibidos: b'\x11\xb2T' -> decodificado: {'temperatura': 11.32, 'humedad': 74, 'direccion_viento': 'S'}
OUTLINE
TIMELINE
```

```python
def main():
    consumer = KafkaConsumer(
        TOPIC,
        bootstrap_servers=["iot.redesuvg.cloud:9092"],
        auto_offset_reset="latest",
        enable_auto_commit=True,
        group_id="grupo-lab7-bytes",
        # no usamos value_deserializer, trabajamos con bytes crudos
    )

    temps = []
    hums = []
    winds = []

    plt.ion()
    fig, (ax1, ax2) = plt.subplots(2, 1)
    fig.suptitle("Estacion Meteorologica - Payload 3 bytes")
```

```python
    for msg in consumer:
        raw = msg.value  # bytes
        data = decode_lectura(raw)  # dict

        print("Bytes recibidos:", raw, "-> decodificado:", data)

        temps.append(data["temperatura"])
        hums.append(data["humedad"])
        winds.append(data["direccion_viento"])

        if len(temps) > 50:
            temps.pop(0)
            hums.pop(0)
            winds.pop(0)

        ax1.clear()
        ax2.clear()

        ax1.plot(temps, marker="o")
        ax1.set_ylabel("Temp (°C)")

        ax2.plot(hums, marker="o")
        ax2.set_ylabel("Humedad (%)")
        ax2.set_xlabel("Muestras")

        plt.pause(0.01)
```

PREGUNTAS

¿A QUÉ CAPA PERTENECE JSON/SOAP SEGÚN EL MODELO OSI Y POR QUÉ?

JSON y SOAP pertenecen a la capa de aplicación del modelo OSI. Esto se debe a que son formatos diseñados para que los programas se comuniquen entre sí y compartan información de una manera entendible.

¿QUÉ BENEFICIOS TIENE UTILIZAR UN FORMATO COMO JSON/SOAP?

Usar JSON o SOAP hace que el intercambio de información sea mucho más sencillo porque los datos quedan organizados y fáciles de entender. Estos formatos permiten que distintos programas, lenguajes y plataformas se comuniquen sin necesidad de acuerdos complicados.

¿QUÉ VENTAJAS Y DESVENTAJAS TIENE UN ACERCAMIENTO BASADO EN PUB/SUB DE KAFKA?

Un enfoque Pub/Sub como Kafka tiene la ventaja de que separa a quienes generan datos de quienes los consumen, lo que hace que el sistema sea más flexible y escalable. Sin embargo, una desventaja es que Kafka puede ser pesado de configurar y mantener, especialmente para proyectos pequeños.

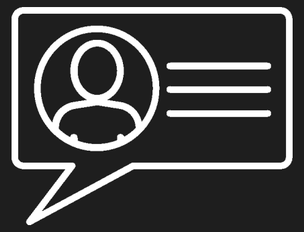
PREGUNTAS

¿PARA QUÉ APLICACIONES TIENE SENTIDO USAR KAFKA? ¿PARA CUÁLES NO?

Kafka tiene sentido en aplicaciones donde se generan muchos datos constantemente, como sensores industriales, análisis en tiempo real, registros de servicios grandes o sistemas con muchos usuarios. En cambio, no es ideal para proyectos muy pequeños

¿QUÉ COMPLEJIDADES INTRODUCE EL TENER UN PAYLOAD RESTRINGIDO (PEQUEÑO)?

Tener un payload pequeño significa que no podemos mandar los datos tal como los usamos en el programa, sino que tenemos que comprimirlos o representarlos de manera muy eficiente. Esto complica el diseño porque obliga a reducir la precisión



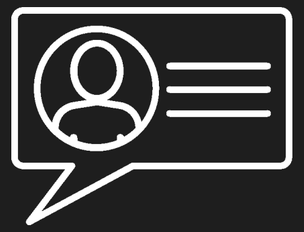
¿CÓMO PODEMOS HACER QUE EL VALOR DE TEMPERATURA QUEPA EN 14 BITS?

Para que la temperatura quepa en 14 bits, necesitamos convertirla a un número entero más pequeño. La forma más simple es multiplicarla por 100 para conservar dos decimales y luego guardar ese resultado como un entero.

PREGUNTAS

¿QUÉ SUCEDERÍA SI AHORA LA HUMEDAD TAMBIÉN ES TIPO FLOAT CON UN DECIMAL? ¿QUÉ DECISIONES TENDRÍAMOS QUE TOMAR?

Si la humedad tuviera decimales, inmediatamente ocuparía más espacio y posiblemente ya no cabría en los 3 bytes disponibles. Esto nos obligaría a decidir si queremos mantener tanta precisión o si es mejor reducirla. Podríamos, por ejemplo, redondear la humedad a un entero, o guardar solo un decimal multiplicando por 10, pero esto usaría más bits.



¿QUÉ PARÁMETROS O HERRAMIENTAS DE KAFKA PODRÍAN AYUDARNOS SI LAS RESTRICCIONES FUERAN AÚN MÁS FUERTES?

Si las restricciones fueran aún más fuertes, podríamos apoyarnos en herramientas de Kafka que ayudan a manejar mensajes pequeños o sensibles al tamaño. Por ejemplo, podríamos usar compresión para reducir el peso de los datos o ajustar configuraciones del broker para trabajar con paquetes más pequeños.