PROYECTO INTEGRADOR I –TST 2024

Sprint Backlog - Semana 22: Implementación de Almacenamiento Local Temporal en ESP32

Descripción:

Definir las funcionalidades necesarias para que el middleware gestione la comunicación entre los dispositivos Edge y la API.

Definir los puntos clave de transformación de datos en el middleware.

Los puntos clave de la **transformación de datos en el middleware** son fundamentales para garantizar que la información fluya correctamente entre diferentes sistemas en un entorno de software distribuido. Los puntos más importantes:

1. **Normalización de Datos**: El middleware se encarga de estandarizar los datos que provienen de múltiples fuentes y con distintos formatos. La normalización asegura que toda la información se maneje de manera coherente y uniforme en las aplicaciones que la reciben.
2. **Conversión de Formatos**: En muchos casos, los sistemas involucrados manejan diferentes formatos de datos (XML, JSON, CSV, etc.). El middleware convierte estos formatos a uno común o al que sea requerido por el sistema de destino, permitiendo la interoperabilidad entre aplicaciones.
3. **Mapeo de Campos**: Durante la transformación, los campos de los datos de entrada se asignan a los campos de salida correspondientes, ajustando las estructuras de datos entre sistemas heterogéneos para que los datos sean comprendidos por la aplicación de destino.
4. **Filtrado de Datos**: El middleware puede filtrar información relevante o necesaria, descartando datos irrelevantes o redundantes antes de pasarlos al siguiente sistema, optimizando el rendimiento y reduciendo la carga de procesamiento.
5. **Agregación de Datos**: En algunos casos, el middleware combina datos de diferentes fuentes para crear una vista unificada o generar nuevos conjuntos de datos, necesarios para el análisis o procesamiento adicional.
6. **Validación de Datos**: Antes de transmitir los datos, el middleware verifica que cumplan con ciertos criterios o reglas de negocio predefinidas, asegurando que la información sea correcta y útil para el sistema receptor.
7. **Enriquecimiento de Datos**: El middleware puede agregar información adicional proveniente de otras fuentes o sistemas para completar los datos antes de entregarlos al sistema destino.
8. **Cifrado y Seguridad**: Durante la transformación, el middleware puede aplicar mecanismos de cifrado o anonimización para proteger la integridad y confidencialidad de los datos mientras son transmitidos entre sistemas.

Estos puntos permiten que el middleware sea un componente crucial para la integración y comunicación efectiva entre diferentes aplicaciones y servicios en entornos distribuidos.

Ejemplos prácticos para ilustrar cómo funcionan los puntos clave de la **transformación de datos en el middleware**:

1. Normalización de Datos

Ejemplo: Diferentes sistemas usan unidades distintas (uno mide peso en kilogramos y otro en libras). El middleware convierte todos los pesos a kilogramos antes de enviarlos al sistema de destino para que los datos sean coherentes y comparables.

- **Entrada: 150 lbs**
- **Transformación: 150 lbs → 68.04 kg**
- **Salida: 68.04 kg**

2. Conversión de Formatos

Ejemplo: Un sistema genera datos en formato XML, pero el sistema de destino solo acepta JSON. El middleware convierte los datos de XML a JSON antes de pasarlos.

- **Entrada:** `<user><name>John</name></user>` **(XML)**
- **Transformación: Conversión de XML a JSON**
- **Salida:** `{"user": {"name": "John"}}` **(JSON)**

3. Mapeo de Campos

Ejemplo: Un sistema utiliza el campo "user_name" y otro sistema usa "username". El middleware realiza el mapeo de campos para que la información fluya correctamente.

- **Entrada:** `{"user_name": "Alice"}`
- **Transformación: Mapeo de "user_name" a "username"**
- **Salida:** `{"username": "Alice"}`

4. Filtrado de Datos

Ejemplo: Un sistema genera varios datos, pero solo son relevantes el nombre y la fecha de nacimiento. El middleware filtra los campos innecesarios antes de enviar los datos.

- **Entrada:** `{"name": "Bob", "age": 30, "email": "bob@example.com", "dob": "1994-04-01"}`
- **Transformación: Eliminar "age" y "email"**
- **Salida:** `{"name": "Bob", "dob": "1994-04-01"}`

5. Agregación de Datos

Ejemplo: El middleware recibe datos financieros de dos fuentes diferentes (ventas y gastos) y los agrega para calcular la ganancia neta antes de enviarlos al sistema de análisis.

- **Entrada:** `{"sales": 10000, "expenses": 7000}`
- **Transformación: Agregación de datos para calcular la ganancia neta**
- **Salida:** `{"net_profit": 3000}`

6. Validación de Datos

Ejemplo: Un formulario de entrada permite a los usuarios ingresar fechas. El middleware valida que la fecha tenga el formato correcto antes de pasarla a la base de datos.

- **Entrada:** `{"date_of_birth": "1995-13-01"}`
- **Transformación: Detectar formato de fecha inválido**
- **Salida: Error, fecha no válida.**

7. Enriquecimiento de Datos

Ejemplo: El middleware recibe una solicitud de un pedido con solo un "product_id". Utiliza una base de datos interna para agregar la descripción del producto y su precio antes de enviar el pedido a la aplicación de ventas.

- **Entrada:** `{"product_id": 123}`
- **Transformación: Agregar descripción y precio consultando una base de datos**
- **Salida:** `{"product_id": 123, "description": "Laptop", "price": 1000}`

8. Cifrado y Seguridad

Ejemplo: Un sistema necesita enviar información sensible (número de tarjeta de crédito) a otro sistema. El middleware cifra esta información antes de transmitirla.

- **Entrada:** `{"credit_card_number": "1234-5678-9876-5432"}`
- **Transformación: Aplicar cifrado de datos**
- **Salida:** `{"credit_card_number": "****-****-****-5432"}`

Estos ejemplos muestran cómo el middleware maneja la transformación de datos en diferentes contextos, garantizando la integridad, coherencia y seguridad de la información mientras se intercambia entre sistemas.

Usando un lenguaje como **Python** para ilustrar cómo se podría implementar estos puntos clave de la **transformación de datos en el middleware** con ejemplos de código. utilizando algunas librerías comunes como **`json`** para manipulación de datos y ejemplos básicos de transformaciones.

1. Normalización de Datos

En este ejemplo, convertimos una unidad de libras a kilogramos.

```python
def normalize_weight(weight_lbs):
    # Conversión de libras a kilogramos
    weight_kg = weight_lbs * 0.453592
    return round(weight_kg, 2)

# Ejemplo de uso
weight_in_lbs = 150
normalized_weight = normalize_weight(weight_in_lbs)
print(f"Peso normalizado: {normalized_weight} kg")
```

2. Conversión de Formatos

Convertir los datos en formato XML a JSON usando **`xmltodict y json`.**

```python
import xmltodict
import json

def convert_xml_to_json(xml_data):
    # Convertir XML a diccionario
    dict_data = xmltodict.parse(xml_data)
    # Convertir diccionario a JSON
    json_data = json.dumps(dict_data)
    return json_data

# Ejemplo de uso
xml_data = "<user><name>John</name></user>"
json_result = convert_xml_to_json(xml_data)
print(json_result)
```

3. Mapeo de Campos

Mapear un campo de "**user_name**" a "**username**".

```python
def map_fields(data):
    # Cambiar "user_name" a "username"
    if "user_name" in data:
        data["username"] = data.pop("user_name")
    return data

# Ejemplo de uso
data = {"user_name": "Alice"}
mapped_data = map_fields(data)
print(mapped_data)
```

4. Filtrado de Datos

Filtrar los campos innecesarios de los datos.

```python
def filter_data(data):
    # Filtrar solo los campos necesarios
    return {key: data[key] for key in ["name", "dob"] if key in data}

# Ejemplo de uso
data = {"name": "Bob", "age": 30, "email": "bob@example.com", "dob": "1994-04-01"}
filtered_data = filter_data(data)
print(filtered_data)
```

5. Agregación de Datos

Agregar datos de ventas y gastos para calcular la ganancia neta.

```python
def calculate_net_profit(data):
    # Calcular la ganancia neta
    net_profit = data['sales'] - data['expenses']
    return {"net_profit": net_profit}

# Ejemplo de uso
financial_data = {"sales": 10000, "expenses": 7000}
profit = calculate_net_profit(financial_data)
print(profit)
```

6. Validación de Datos

Validar que una fecha esté en un formato correcto.

```python
from datetime import datetime

def validate_date(date_string):
    try:
        # Intentar convertir la fecha a formato datetime
        datetime.strptime(date_string, '%Y-%m-%d')
        return True
    except ValueError:
        return False

# Ejemplo de uso
date = "1995-13-01"
is_valid = validate_date(date)
print(f"Fecha válida: {is_valid}")
```

7. Enriquecimiento de Datos

Enriquecer un producto añadiendo más información de una base de datos.

```python
def enrich_product(product_data, product_db):
    # Agregar datos adicionales del producto
    product_id = product_data["product_id"]
    if product_id in product_db:
        product_data.update(product_db[product_id])
    return product_data

# Ejemplo de uso
product_data = {"product_id": 123}
product_db = {
    123: {"description": "Laptop", "price": 1000},
    124: {"description": "Mouse", "price": 20}
}
enriched_data = enrich_product(product_data, product_db)
print(enriched_data)
```

8. Cifrado y Seguridad

Cifrar el número de tarjeta de crédito antes de enviarlo.

```python
def mask_credit_card(credit_card_number):
    # Enmascarar todo excepto los últimos 4 dígitos
    return "****-****-****-" + credit_card_number[-4:]

# Ejemplo de uso
credit_card_number = "1234-5678-9876-5432"
masked_number = mask_credit_card(credit_card_number)
print(masked_number)
```

Explicación

- **Normalización** convierte valores numéricos de un sistema de medida a otro.
- **Conversión de Formatos** permite adaptar los datos de un sistema (XML) a otro (JSON).
- **Mapeo de Campos** ajusta los nombres de variables o campos entre sistemas.
- **Filtrado** selecciona solo los campos de datos que son relevantes para la operación.
- **Agregación** permite sumar o combinar datos de diferentes fuentes.
- **Validación** asegura que los datos cumplan con el formato o estándar requerido.

- **Enriquecimiento** añade información adicional a los datos utilizando fuentes externas.
- **Cifrado** protege información sensible, como tarjetas de crédito, en su transmisión.

Este conjunto de ejemplos ilustra cómo se podrían manejar transformaciones de datos dentro de un middleware usando Python.