Implementación de Pipeline End-to-End - Día 2

Change Status

pending 40 min

Learning Objectives

1. Entender componentes básicos de un pipeline end-to-end
2. Aprender gestión de estado y dependencias entre tareas
3. Comprender estrategias de manejo de errores
4. Conocer importancia del monitoreo y debugging

Activities and Learning

Task 1: Componentes de un Pipeline End-to-End (10 minutos)

¿Qué es un pipeline end-to-end?

Un pipeline end-to-end conecta todas las fases del procesamiento de datos: desde capturar datos crudos hasta generar insights consumibles.

Componentes principales:

Ingesta: Captura datos desde fuentes externas
Procesamiento: Limpieza, transformación y validación
Almacenamiento: Persistencia de datos procesados
Orquestación: Coordinación de todos los componentes

Ejemplo conceptual:

```
# Pipeline simplificado
pipeline = {
    'ingesta': 'Capturar datos de APIs',
    'procesamiento': 'Limpiar y transformar',
    'almacenamiento': 'Guardar en base de datos',
    'orquestacion': 'Airflow coordina todo'
}
```

Task 2: Gestión de Estado y Dependencias (10 minutos)

¿Por qué gestionar estado en pipelines?

Los pipelines distribuidos necesitan compartir información entre tareas que se ejecutan en diferentes momentos y lugares.

Conceptos clave:

Estado: Información que persiste entre ejecuciones
Contexto: Datos compartidos entre componentes
Dependencias: Orden de ejecución requerido

Ejemplo simple de gestión de estado:

```
# Estado básico con Redis
estado_pipeline = {
    'componente_actual': 'procesamiento',
    'datos_procesados': 15000,
    'timestamp': '2024-01-15T10:30:00'
}
```

Task 3: Manejo de Errores y Monitoreo (10 minutos)

¿Por qué es importante el manejo de errores?

Los pipelines en producción fallan. Un buen manejo de errores asegura que los problemas se detecten, reporten y resuelvan automáticamente.

Estrategias básicas:

Reintentos: Re-ejecutar tareas fallidas
Circuit breakers: Detener ejecución cuando hay fallos repetidos
Logging: Registrar todo lo que sucede
Alertas: Notificar cuando hay problemas

Ejemplo de manejo de errores:

```python
# Estrategia simple de reintentos
def ejecutar_con_reintento(funcion, max_reintentos=3):
    for intento in range(max_reintentos):
        try:
            return funcion()
        except Exception as e:
            if intento == max_reintentos - 1:
                raise e
            print(f"Intento {intento + 1} falló, reintentando...")
```