Validación y Testing - Día 3

Change Status

completed · 40 min

Learning Objectives

1. Entender importancia del testing en pipelines de datos
2. Aprender tipos básicos de tests (unitarios, integración)
3. Comprender validación de calidad de datos
4. Conocer estrategias de testing automatizado

Activities and Learning

Task 1: Testing Unitario Básico (10 minutos)

¿Por qué hacer testing en pipelines de datos?

Los pipelines procesan datos críticos. El testing asegura que:

Los datos se procesan correctamente
Los errores se detectan temprano
Los cambios no rompen funcionalidad existente

Tipos de testing:

Unitario: Prueba componentes individuales
Integración: Prueba cómo componentes trabajan juntos
Validación: Prueba calidad de datos

Ejemplo simple de test unitario:

```
def test_suma_datos():
    # Arrange
    datos = [1, 2, 3, 4, 5]

    # Act
    resultado = sum(datos)

    # Assert
    assert resultado == 15, f"Expected 15, got {resultado}"
    print("✅ Test passed!")
```

Task 2: Validación de Calidad de Datos (10 minutos)

¿Qué validar en los datos?

Completitud: ¿Faltan datos?
Exactitud: ¿Son correctos los datos?
Consistencia: ¿Son coherentes los datos?
Actualidad: ¿Están actualizados?

Ejemplo de validación básica:

```
def validar_datos_ventas(datos):
    validaciones = {
        'completitud': len(datos) > 0,
        'exactitud': all(row['precio'] > 0 for row in datos),
        'consistencia': all('fecha' in row for row in datos)
    }
    return validaciones
```

Task 3: Testing de Integración (10 minutos)

¿Cómo probar que todo funciona junto?

El testing de integración verifica que los componentes del pipeline trabajen correctamente como un sistema completo.

Estrategia básica:

Probar cada componente individualmente

Probar integración entre componentes
Probar el pipeline completo con datos reales

Ejemplo conceptual:

```python
def test_pipeline_completo():
    # Simular datos de entrada
    datos_entrada = [{'id': 1, 'valor': 100}]

    # Ejecutar pipeline paso a paso
    datos_limpios = limpiar_datos(datos_entrada)
    datos_transformados = transformar_datos(datos_limpios)
    resultado = guardar_datos(datos_transformados)

    # Verificar resultado final
    assert resultado['exito'] == True
```