**Análisis del problema BPM**

**Objetivo formal del proceso**

Registrar reportes de proveedores en un portal aplicando verificaciones de fraude, cálculo de impuesto y estandarización de datos, obteniendo un balance final del proceso.

**Alcance**

**Incluye:**

- Recepción del Excel
- Validación
- Registro web
- Reporte

**Excluye:**

- Procesos posteriores a la generación del reporte.

**Roles o Stakeholders:**

Según el documento las partes interesadas son los analistas responsables de este proceso.

**Inputs y Outputs:**

**Inputs**

- Excel proveedores
- Reglas de impuestos

**Outputs:**

- Registro web
- Reporte auditoría

**Reglas de negocio:**

- Si la fecha es fin de semana se marca como ReviewRequired
- Si es Software el concepto aplica un impuesto del 0%, si es diferente entonces un 19%

**Excepciones del proceso:**

- Si falla Python en cuanto a la ejecución del proceso, el está envuelto en un try catch el cual captura el error de forma global según la arquitectura REFramework.
- Si falla Python Script internamente devuelve un “ERROR” que es capturado y mostrado en el reporte como falla técnica.
- Si el portal no carga el error es capturado por la estructura global de la arquitectura REFramework.

**Métricas:**

- Como es un sistema de colas, cada ejecución es independiente por lo cual en el Orchestrator se puede observar tiempos de ejecución del proceso completo, procesos completados, con errores técnicos y en el reporte a nivel operativo se pueden ver también procesos completados que cumplen las reglas, procesos pendientes o marcados como REVIEW o procesos fallidos por errores técnicos.

**Análisis del proceso:**

Este es el proceso de como analice el problema desde el estado actual (**AS-IS**) hacia el estado automatizado (**TO-BE**).

1. **Fase de Descubrimiento y Análisis (AS-IS)**

Identificamos un proceso manual crítico realizado por hasta 5 analistas que presentaba tres problemas principales:

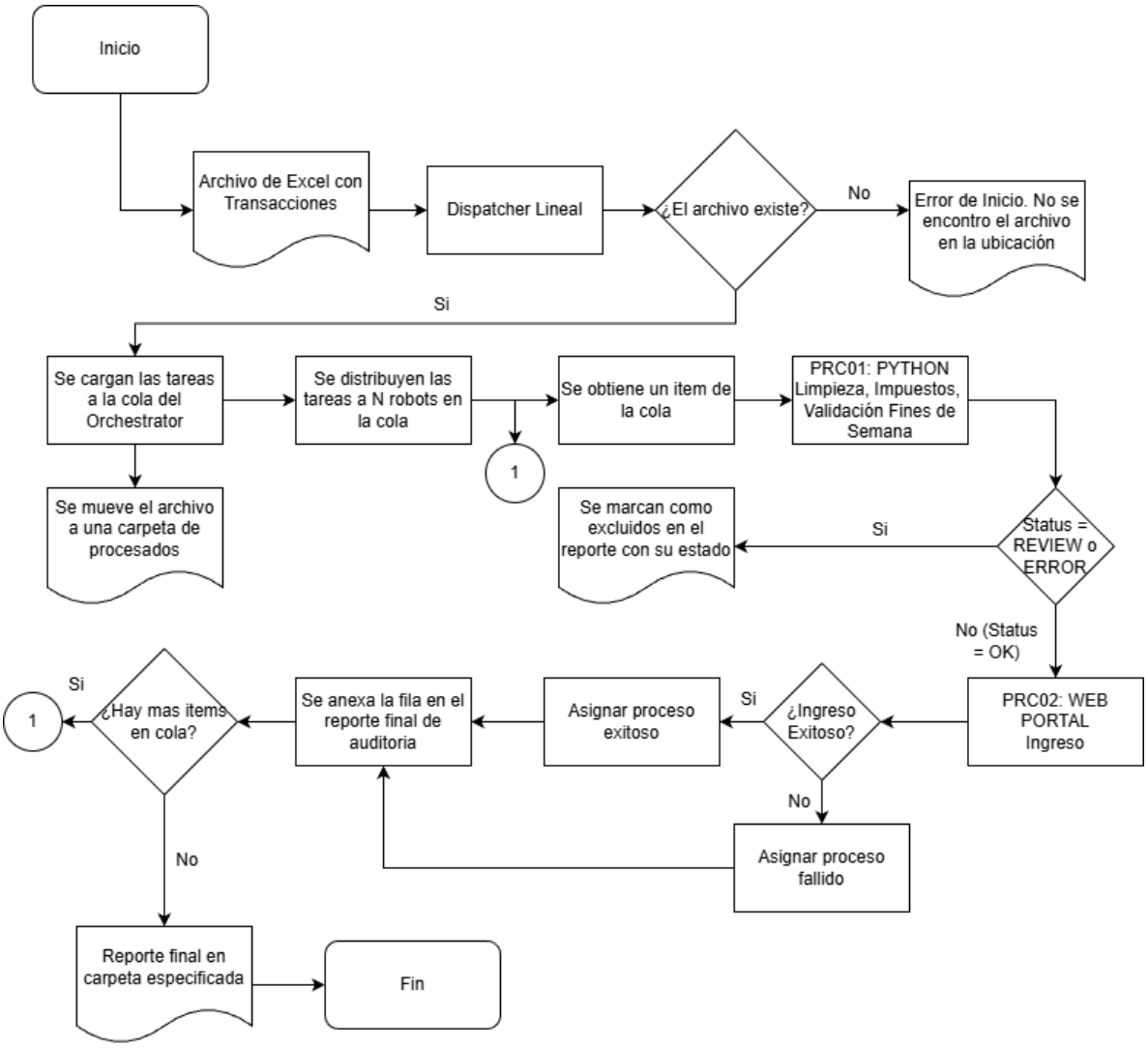
- **Cuello de Botella:** El aumento de 100 a 5,000 facturas diarias superó la capacidad humana.
- **Riesgo Operativo:** La inestabilidad del portal legado y el cambio dinámico de campos generaban errores y reprocesos.
- **Complejidad de Lógica:** Las reglas de impuestos cambiantes en "low-code" eran difíciles de mantener, creando un "código espagueti".

2. **Rediseño del Proceso (TO-BE)**

Aplicamos el principio de Desacoplamiento para que la solución sea escalable:

- **Separación de Carga y Ejecución:** Dividimos el proceso en Dispatcher (Carga) y Performer (Ejecución) para permitir el paralelismo (varios robots trabajando a la vez).
- **Caja Negra de Reglas:** Centralizamos la lógica compleja en un script de Python para asegurar la "higiene de datos" y facilitar cambios de reglas sin tocar el flujo principal.
- **Auditoría Total:** Diseñamos una salida que informa al negocio no solo qué se procesó, sino por qué se excluyó un registro (Regla de negocio vs Error técnico).

3. **Diagrama de Flujo del Proceso (Arquitectura Híbrida)**




## 4. Decisiones de Valor BPM

**Resiliencia:** Al usar Fuzzy Selectors y Anchors, el proceso no se detiene si el portal cambia visualmente, eliminando el mantenimiento constante.

**Persistencia:** Si el internet falla, el Orchestrator sabe qué factura quedó pendiente, garantizando que "no se pierda ni un solo registro". Cada robot toma 1 item en cola diferente, internamente Get Transaction Items lo asigna a ese robot.

**Gobernanza:** El reporte final de auditoría cumple con el requerimiento de transparencia total exigido por los auditores internos.