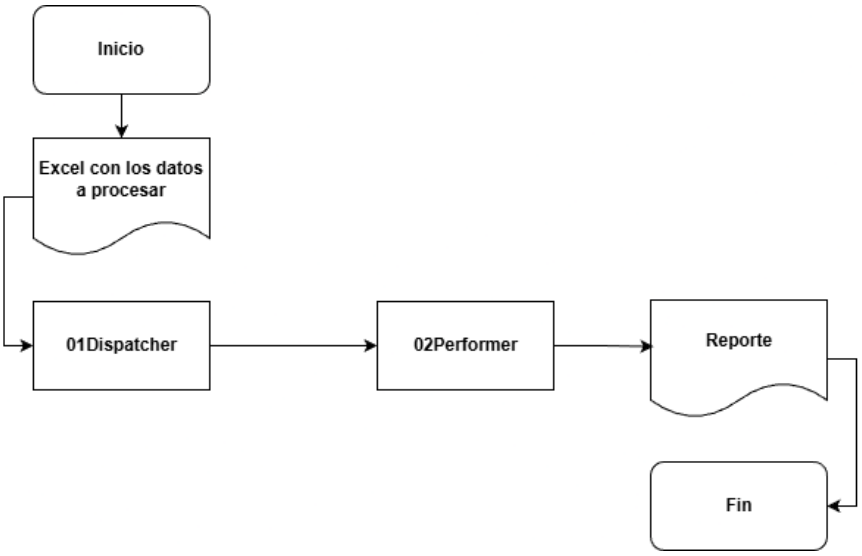
Decisiones de Arquitectura (ADR)

En esta sección se detallan las decisiones técnicas tomadas durante el desarrollo para asegurar que el bot sea escalable, resiliente y fácil de mantener.

El proyecto se divide en dos secciones 01Dispatcher y 02Performer:



01Dispatcher: Lee el archivo de Excel en la ruta especificada y agrega las tareas en cola en el Orchestrator.

02Performer: Toma las tareas que están en cola, realiza el preprocesamiento de los datos aplicando las reglas indicadas en el instructivo para posteriormente decidir si ese dato debe ser radicado en la web. Al finalizar todas las colas, crea el archivo de reporte en la ruta especificada identificando los datos radicados o no radicados mediante un id de referencia generado.

Con esta lógica se permite configurar o asignar los robots necesarios a la cola en el Orchestrator con las tareas previamente cargadas. Posterior a ello cada robot toma una tarea y comienza a aplicar el Performer.

Ventaja: Aislamiento de errores

Si el Dispatcher falla no pasa nada, la cola simplemente no se llena. Si el Dispatcher tiene éxito, pero Performer falla en la ejecución #50, las otras 49 ya están como completadas en el Orchestrator y las restantes pueden ser reintentadas automáticamente por el REFramework.

Característica	Dispatcher (Lineal)	Performer (REFramework)
Objetivo	Carga de datos inicial	Ejecución de lógica de negocio
Complejidad	Baja (Secuencia simple)	Alta (Máquina de estados)
Manejo de Cola	Add Queue Item (Escribe)	Get Transaction Item (Lee)
Fallo Crítico	El archivo no se carga	Se pierde una sola transacción (reintentable)

1. Patrón de Diseño: Robotic Enterprise Framework (REFramework) para 02_Performer

Se seleccionó el REFramework (basado en una Máquina de Estados) para el Performer, mientras que para el Dispatcher se utilizó una estructura lineal robusta. Esto se debe a:

- **Escalabilidad:** El uso de colas (FinCorp_Invoices) permite que múltiples robots trabajen simultáneamente sin duplicar esfuerzos.
- **Persistencia:** Si el robot falla físicamente, el estado de la transacción queda registrado en Orchestrator, permitiendo retomar desde el punto exacto de falla.
- **Separación de responsabilidades:** La lógica de obtención de datos (GetTransactionData), procesamiento (Process) y cierre (EndProcess) están aisladas, facilitando actualizaciones futuras (como cambiar el portal web sin tocar la lógica de Python). Además la operación GetTransactionData permite obtener, bloquear y cambiar de estado la tarea de la cola. Con esto al tener conectado otro robot no selecciona esa tarea si no que va por la siguiente con el estado “Nuevo”.

2. Resiliencia de Selectores en Entornos Dinámicos

Para garantizar que el bot no se rompa ante cambios menores en la interfaz del portal web, se implementaron las siguientes estrategias:

- **Selectores Difusos (Fuzzy Selectors):** En lugar de buscar una coincidencia exacta, el bot utiliza algoritmos de similitud para identificar elementos incluso si cambian parcialmente sus atributos.
- **Anclajes (Anchors):** Se vinculan los campos de entrada (como "Email" o "Company Name") a sus etiquetas visuales permanentes. Esto asegura que, aunque el programador de la web mueva el campo de lugar, el robot lo encuentre por su relación con el texto fijo.

3. Estrategia de Manejo de Errores (Híbrida)

Se implementó un esquema de capas para maximizar la disponibilidad del robot:

- **Manejo Local (Específico):** Se utilizan bloques Try Catch dentro de cada módulo (PRC01, PRC02) para capturar errores técnicos conocidos. Por ejemplo, si el procesamiento en Python falla, el error se mapea como un “ERROR” en el reporte de auditoría antes de que el robot intente seguir adelante.
- **Manejo Global (Sistémico):** El REFramework actúa como la red de seguridad final. Cualquier error no previsto activa el estado System Exception, el cual:
    - Toma una captura de pantalla del error.
    - Cierra de forma segura las aplicaciones (Chrome/Excel).
    - Reinicia el entorno para procesar la siguiente transacción, evitando que un error puntual detenga toda la operación.

4. Gestión Dinámica de Configuración

A diferencia de los desarrollos rígidos, este bot utiliza un archivo Config.xlsx para Dispatcher y otro para Performer, centralizando rutas, nombres de colas y carpetas de Orchestrator.