

Workflow Studio — Context Pack

Objetivo: mantener un resumen único, estable y completo del proyecto para no perder contexto entre sesiones. Usalo como “manual del proyecto” y como semilla al abrir hilos nuevos.

1) Stack y alcance

- **Front (editor visual):** WebForms + JS clásico (workflow.ui.js) y también prototipo React (canvas de pruebas).
 - **Backend:** ASP.NET Web Forms (.NET Framework **4.8, C#**).
 - **BD:** SQL Server Express.
 - **Motor:** MotorFlujoMinimo con *handlers* básicos + `ManejadorSql` (nodo `data.sql`).
-

2) Páginas y responsabilidades

- **WorkflowUI.aspx**
 - Editor (toolbox + canvas).
 - Exporta JSON / C# y **Guarda en SQL** usando `__doPostBack('WF_SAVE', '')` con `hfWorkflowJson`.
 - Panel “Probar motor en servidor” (para test rápido con el motor local).
 - **WF_Definiciones.aspx**
 - Lista definiciones en BD, ver JSON.
 - **WF_Instancias.aspx** (con `<%@ Page Async="true" %>`)
 - Lista instancias (filtro por definición).
 - Acciones: **Datos, Log, Reejecutar** (todas asíncronas y *awaited*).
 - **Poliza_Nueva.aspx**
 - Página de negocio que dispara un **workflow** seleccionado y muestra el Id de la instancia creada.
-

3) Tablas SQL (esquema estable)

```
-- dbo.WF_Definicion
(Id int PK IDENTITY,
Codigo nvarchar(50), Nombre nvarchar(200), Version int,
Activo bit, FechaCreacion datetime, CreadoPor nvarchar(100),
JsonDef nvarchar(max))

-- dbo.WF_Instancia
(Id bigint PK IDENTITY,
WF_DefinicionId int FK,
Estado nvarchar(20), FechaInicio datetime, FechaFin datetime null,
DatosEntrada nvarchar(max) null, DatosContexto nvarchar(max) null,
CreadoPor nvarchar(100) null)
```

```
-- dbo.WF_InstanciaLog
(WF_InstanceId bigint FK, FechaLog datetime,
 Nivel nvarchar(20), Mensaje nvarchar(max),
 NodoId nvarchar(50) null, NodoTipo nvarchar(100) null)
```

4) Motor y namespaces

- Namespace Web (UI + Motor mínimo): Intranet.WorkflowStudio.WebForms
- MotorFlujoMinimo contiene:
 - Modelos: NodeDef, EdgeDef, WorkflowDef.
 - ContextoEjecucion (incluye HttpClient con BaseAddress en intranet).
 - MotorFlujo (ejecución secuencial por aristas y etiquetas).
 - Handlers: HStart (util.start), HEnd (util.end), HLogger (util.logger), HIf (control.if), HHHttpRequest (http.request).
 - MotorDemo.EjecutarAsync(...) registra handlers por defecto y ejecuta.
- Namespace Runtime (servicios de negocio): Intranet.WorkflowStudio.Runtime
- WorkflowRuntime → CrearInstanciaYEjecutarAsync(defId, json, user) y ReejecutarInstanciaAsync(instId, user).
- ManejadorSql (data.sql) — ejecuta commandText / query con parameters, guarda sql.rows en ctx.Estado.

5) JSON del workflow (contrato)

```
{
  "StartNodeId": "nStart",
  "Nodes": {
    "nStart": { "Id": "nStart", "Type": "util.start", "Parameters": {} },
    "nSql": { "Id": "nSql", "Type": "data.sql", "Parameters": {
      "connectionStringName": "DefaultConnection",
      "commandText": "UPDATE Tabla SET Campo=@valor WHERE Id=@id",
      "parameters": { "valor": "${input.Algo}", "id": 1 }
    }},
    "nIf": { "Id": "nIf", "Type": "control.if", "Parameters": {
      "expression": "${sql.rows} == 1" }
    },
    "nOk": { "Id": "nOk", "Type": "util.logger", "Parameters": {
      "level": "Info", "message": "OK" }
    },
    "nErr": { "Id": "nErr", "Type": "util.logger", "Parameters": {
      "level": "Error", "message": "Sin filas" }
    },
    "nEnd": { "Id": "nEnd", "Type": "util.end", "Parameters": {} }
  },
  "Edges": [
    { "From": "nStart", "To": "nSql", "Condition": "always" },
    { "From": "nSql", "To": "nIf", "Condition": "always" },
    { "From": "nIf", "To": "nOk", "Condition": "true" },
    { "From": "nIf", "To": "nErr", "Condition": "false" },
    { "From": "nOk", "To": "nEnd", "Condition": "always" },
    { "From": "nErr", "To": "nEnd", "Condition": "always" }
  ]
}
```

```

        { "From": "nErr", "To": "nEnd", "Condition": "always" }
    ]
}

```

Etiquetas de salida: `always`, `true`, `false`, `error`, etc. El `control.if` usa expresiones simples (ej: `#{payload.status} == 200`, `#{sql.rows} == 1`).

6) Flujo de trabajo típico (end-to-end)

1. En `WorkflowUI.aspx` armá el grafo y **Export JSON**.
2. **Guardar en SQL** (usa `hfWorkflowJson + __doPostBack('WF_SAVE', '')`).
3. En `WF_Definiciones.aspx` verificá que aparece la definición y su JSON.
4. En `Poliza_Nueva.aspx`:
5. Completá datos de negocio → **Enviar** → crea instancia y ejecuta runtime.
6. En `WF_Instancias.aspx`:
7. Filtrá por la definición → ver **Datos** (entrada y contexto) y **Log**.
8. Podés **Reejecutar** (crea una nueva instancia con el mismo input).

7) Convenciones importantes

- **No cambiar nombres** (archivos, ids, tipos de nodo).
- Cuando pidas cambios, devolvemos **archivos completos** para evitar regresiones.
- **Bootstrap local** (sin CDN) por política de Intranet.
- Restaurar canvas tras postback: usar `hfWorkflowJson + window.__WF_RESTORE` en `Page_Load`.
- **Conexiones SQL**: usar `connectionStringName = DefaultConnection` (en `web.config`).

8) Problemas conocidos / TODO

- Posición de nodos: persistir `position` en `Parameters` (o tabla aparte) para reponer layout.
- `http.request` en intranet: preferir rutas relativas (`/Api/Ping.ashx`) o `simularStatus` para pruebas.
- Inspector: agregar validaciones por tipo de nodo.

9) Semilla para abrir hilos nuevos

Copiá/pegá este bloque al comienzo de un hilo nuevo y seguimos sin perder el contexto:

```

Proyecto: Workflow Studio (ASP.NET WebForms, .NET 4.8, C#) – Omar.
Páginas: WorkflowUI.aspx, WF_Definiciones.aspx, WF_Instancias.aspx (Async),
          Poliza_Nueva.aspx.
Tablas: WF_Definicion, WF_Instancia, WF_InstanciaLog.
Motor: MotorFlujoMinimo (HStart, HEnd, HLogger, HIf, HHttpRequest) +

```

```
ManejadorSql; WorkflowRuntime con CrearInstanciaYEjecutarAsync/  
ReejecutarInstanciaAsync.  
JSON: { StartNodeId, Nodes: { Id, Type, Parameters }, Edges: [{ From, To,  
Condition }] }.  
Convenciones: sin cambios de nombres; archivos completos en cada ajuste;  
Bootstrap local; __doPostBack('WF_SAVE','') + hfWorkflowJson.
```

10) Cómo pedir cambios sin regressões

- “Te paso WF_Instancias.aspx.cs completo, agregá X sin tocar Y”.
- “Actualizá ManejadorSql para soportar OUTPUT”.
- “Extendé HIf para >=, <= (mantener compatibilidad).”

Estado actual: Editor UI estable, guardado a SQL funcionando, runtime ejecuta (data.sql, if, logger), vistas de definiciones e instancias operativas y async. Listo para iterar en nodos de negocio y persistencia de layout.