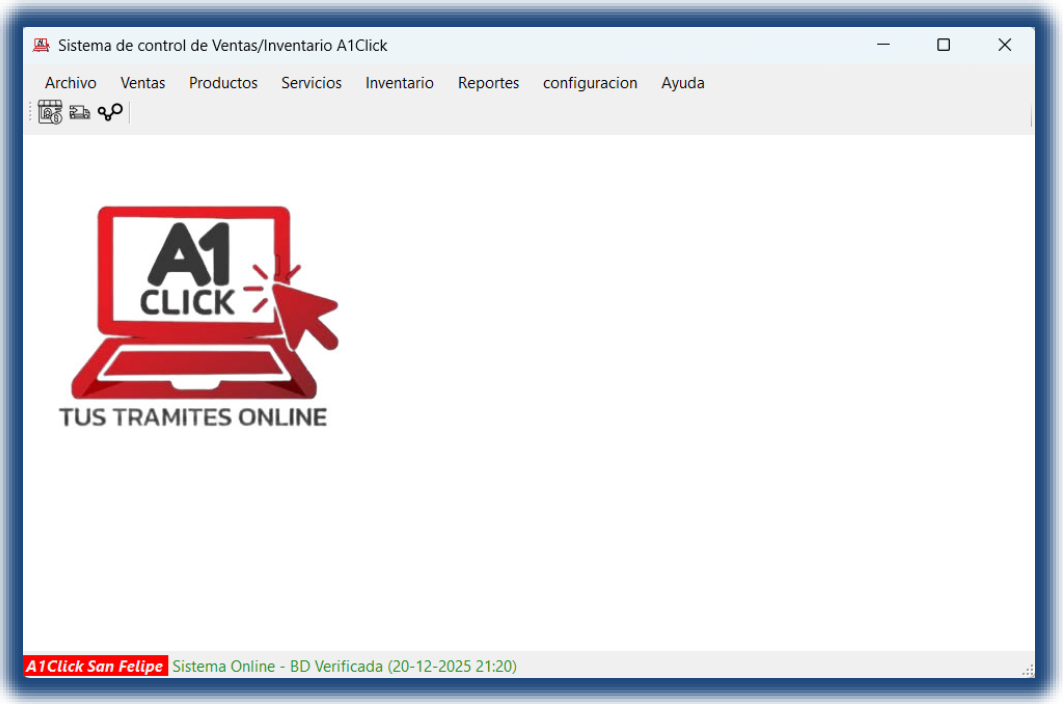


ARQUITECTURA DE PERSISTENCIA Y CONTINUIDAD OPERATIVA

Proyecto: A1Click (Backend) **Motor:** Microsoft SQL Server

Rol: Ingeniero de Software Full Stack (.NET / SQL)

Autor: Leonardo Ahumada

Repositorio: leoahumadausa-Balam/A1Click_ShowCase

Contenido

1. INTRODUCCIÓN Y FILOSOFÍA DE DISEÑO

El backend de A1Click no fue concebido como un simple repositorio de datos pasivo, sino como un Sistema de Gestión de Reglas de Negocio activo. La arquitectura se fundamenta en tres pilares de ingeniería para garantizar la viabilidad del negocio en un entorno productivo real:

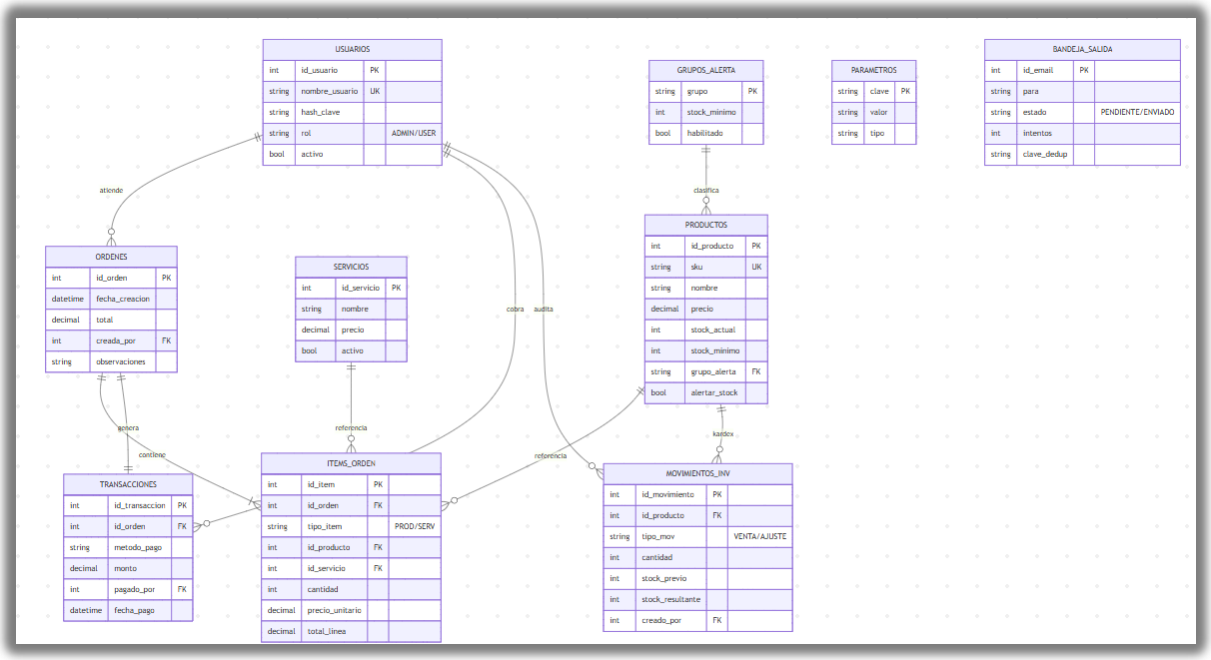
Integridad Transaccional (ACID): Garantía absoluta de consistencia entre inventario y caja.

Resiliencia Asíncrona: Desacople de dependencias externas (como servicios de correo) para asegurar latencia cero en la venta.

Auditoría Forense: Trazabilidad inmutable de cada cambio en los datos maestros.

2. MODELADO DE DATOS HÍBRIDO

Para resolver la problemática de un negocio que vende tanto productos físicos (control de stock estricto) como servicios intangibles (trámites, impresiones), se diseñó un modelo relacional normalizado capaz de gestionar polimorfismo en las líneas de venta.



Modelo Entidad-Relación (DER)

2.1 Integridad Referencial Declarativa

Se utilizaron Check Constraints complejos para validar reglas de negocio a nivel de motor, impidiendo que errores en la capa de aplicación corrompan la base de datos.

```
-- Exclusividad: o producto o servicio (no ambos, no ninguno)
CONSTRAINT CK_items_exclusivo
       CHECK (
              (tipo_item = 'PRODUCTO' AND id_producto IS NOT NULL AND
id_servicio IS NULL)
              OR (tipo_item = 'SERVICIO' AND id_servicio IS NOT NULL AND
id_producto IS NULL)
   )
```

SNIPPET SQL: Validación Exclusiva en Ítems

3. ARQUITECTURA TRANSACCIONAL Y ATOMICIDAD

La operación crítica del sistema (la venta) involucra múltiples escrituras: cabecera, detalles, rebaje de stock, registro financiero y auditoría. Para evitar estados inconsistentes (ej: "se cobró, pero no se descontó el stock"), se implementó una arquitectura basada en Table-Valued Parameters (TVP).

3.1 Inyección Masiva de Datos (TVP)

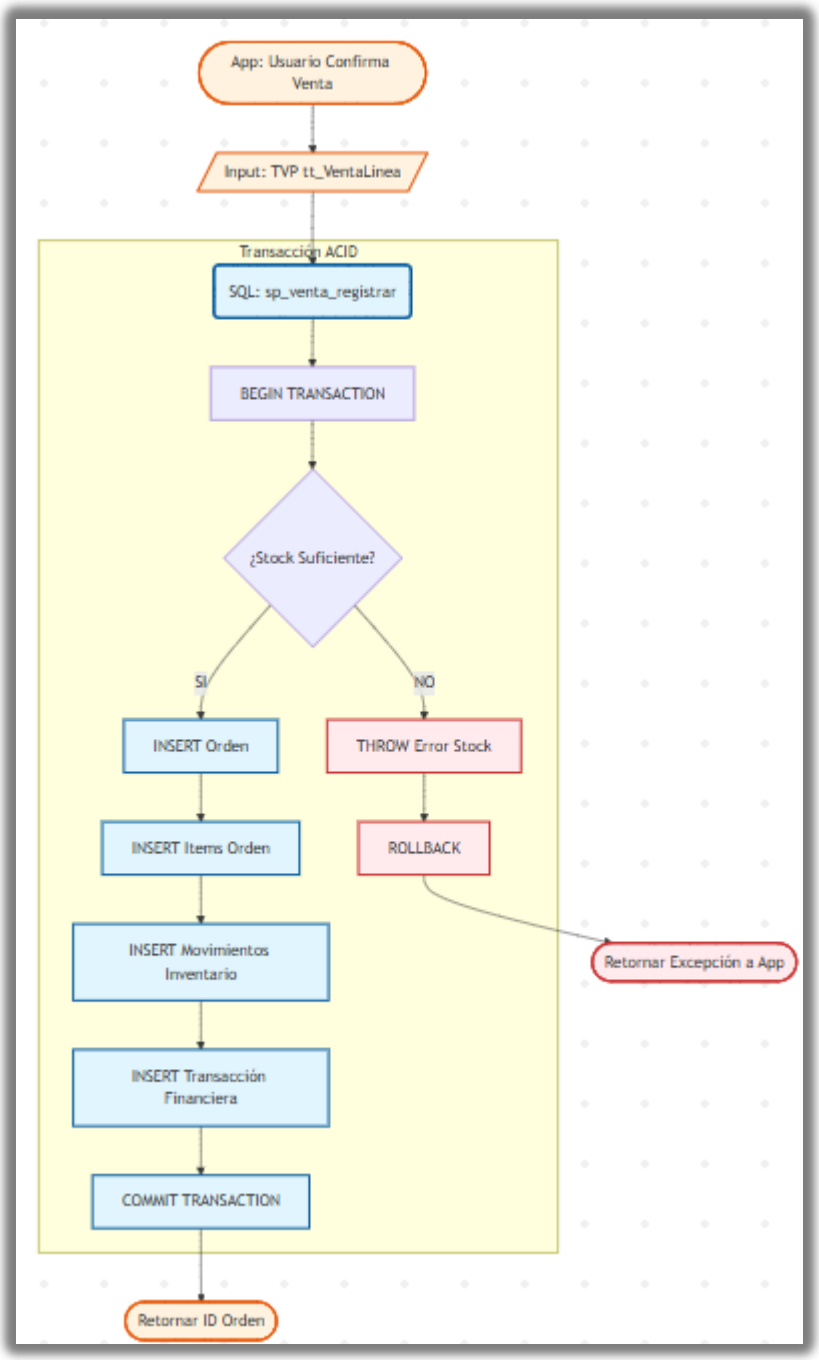
En lugar de realizar múltiples llamadas a la base de datos (chatty interface), la aplicación envía la venta completa en una sola estructura de datos en memoria.

```
CREATE TYPE dbo.tt_VentaLinea AS TABLE
(
    TipoItem        VARCHAR(10)   NOT NULL
        CHECK (TipoItem IN (''PRODUCTO'',''SERVICIO'')),

    IdProducto      INT           NULL,   -- requerido si TipoItem=PRODUCTO
    IdServicio      INT           NULL,   -- requerido si TipoItem=SERVICIO

    Cantidad        DECIMAL(12,3) NOT NULL CHECK (Cantidad > 0),
    PrecioUnitario  DECIMAL(12,2) NOT NULL CHECK (PrecioUnitario >= 0),

    -- Coherencia de referencia (uno u otro, no ambos):
    CHECK (
        (TipoItem = ''PRODUCTO'' AND IdProducto IS NOT NULL AND IdServicio IS
NULL)
        OR
        (TipoItem = ''SERVICIO'' AND IdServicio IS NOT NULL AND IdProducto IS
NULL)
    )

    -- Si quisieras PK, recuerda: las columnas deben ser NOT NULL.
    -- PRIMARY KEY NONCLUSTERED (TipoItem, IdProducto)  -- solo si IdProducto
fuera NOT NULL
);
```

SNIPPET SQL: Definición de Tipo TVP

3.2 Orquestación ACID

Un único Procedimiento Almacenado actúa como orquestador transaccional. Si cualquiera de los pasos falla (ej: stock insuficiente en la línea 50), el motor realiza un `ROLLBACK` total automático.



Flujo de Transacción de Venta

4. PATRÓN "OUTBOX" PARA MENSAJERÍA RESILIENTE

Uno de los mayores desafíos técnicos fue manejar las notificaciones (Alertas de Stock y Backups) sin bloquear la experiencia de usuario (UX) ni depender de la estabilidad de Internet en el momento de la venta.

Se implementó el patrón de arquitectura **Transactional Outbox**:

1. **Detección:** Un `Trigger` detecta la condición crítica (ej: Stock bajo).
2. **Encolado:** El mensaje se inserta en una tabla local `bandeja_salida` instantáneamente (milisegundos).
3. **Procesamiento:** Un proceso de fondo (`SQL Server Agent Job`) despacha los correos de forma asíncrona.

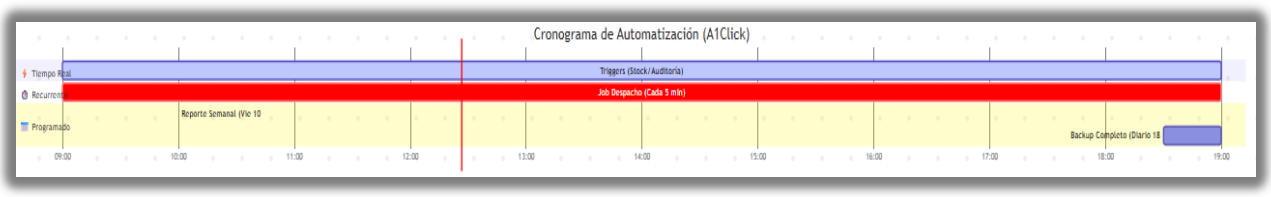
```sql
/* Lógica de Encolado con Deduplicación (Anti-Spam) */
INSERT INTO dbo.bandeja_salida
    (para, asunto, cuerpo, estado, intentos, clave_dedup)
SELECT
    @para_destinatario,
    N'[A1Click] Stock crítico: ' + e.sku,
    N'Producto: ' + e.nombre + ... (Detalle del mensaje),
    'PENDIENTE',
    0,
    -- Clave única para evitar correos duplicados del mismo problema
    N'CRITICO:' + e.sku
FROM @entra e
WHERE NOT EXISTS (
    SELECT 1
    FROM dbo.bandeja_salida b
    WHERE b.clave_dedup = N'CRITICO:' + e.sku
      AND b.estado = 'PENDIENTE'
);
```

SNIPPET SQL: Lógica de Encolado (Trigger)

Beneficio de Ingeniería: Este diseño desacopla la lógica de negocio de la infraestructura. Si el servidor de correo (Gmail/SMTP) se cae, **la venta no se detiene**. Los correos quedan en estado **PENDIENTE** y el sistema reintenta automáticamente mediante políticas de *Retry* configurables.

5. AUTOMATIZACIÓN Y DEVOPS (SQL AGENT)

El sistema fue diseñado para ser "Zero-Maintenance" en el día a día. Se programaron agentes inteligentes que gestionan la salud del sistema.



Cronograma de Automatización

5.1 Política de Retención de Datos

Para evitar el llenado del disco del servidor, se implementó un script de PowerShell embebido en SQL Agent que aplica una política de retención, eliminando respaldos antiguos automáticamente.

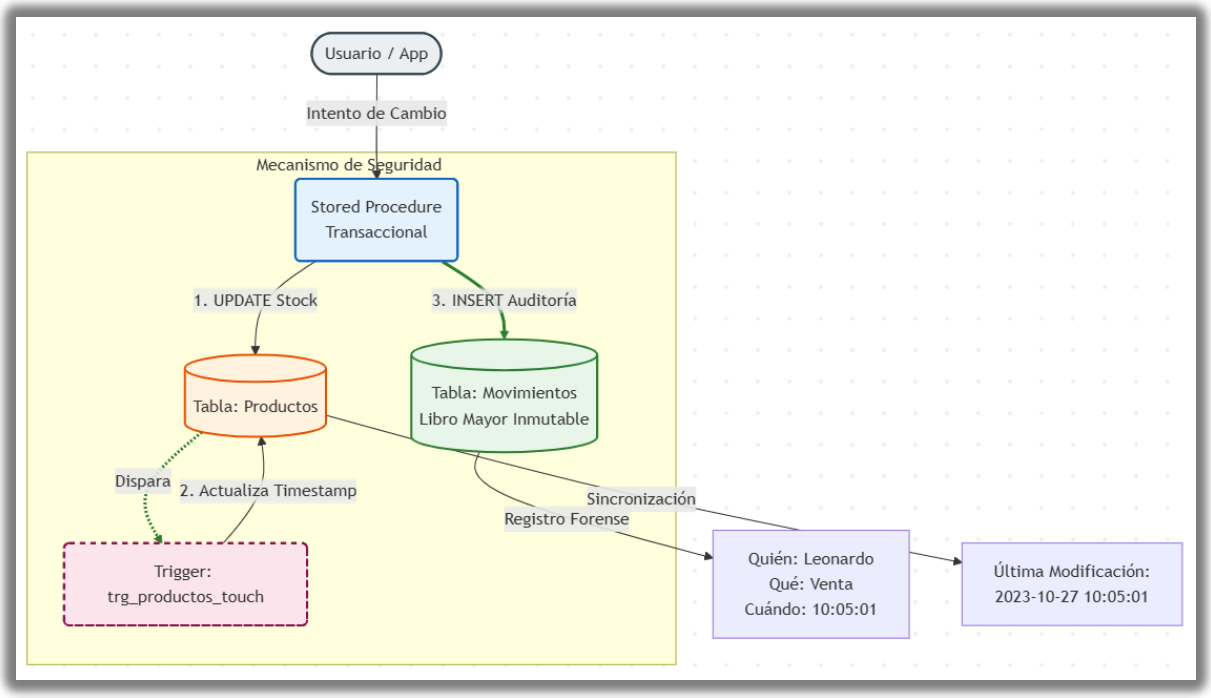
```
@command    = N'$ErrorActionPreference=''Stop''; $dir=''C:\A1Click\backups''; Start-
Sleep -Seconds 2; $files = Get-ChildItem -Path $dir -Filter ''A1Click_*.bak'' |
Sort-Object LastWriteTime -Descending; if ($files.Count -gt 1) { $files | Select-
Object -Skip 1 | Remove-Item -Force; }'
```

SNIPPET SQL: Paso de PowerShell para Retención

6. SEGURIDAD Y AUDITORÍA FORENSE

Más allá de la autenticación de usuarios, la base de datos implementa seguridad a nivel de datos mediante un "Libro Mayor Inmutable" (`movimientos_inventario`).

- **Trazabilidad:** No es posible modificar el stock directamente sin dejar rastro. Cada cambio genera una línea en el kardex indicando `Usuario`, `Motivo` y `Timestamp`.
- **Detección de Cambios:** Triggers de auditoría (`trg_productos_touch`) actualizan marcas de tiempo automáticamente, permitiendo sincronización eficiente con sistemas externos.



Flujo de trazabilidad

CONCLUSIÓN TÉCNICA

La arquitectura de datos de **A1Click** demuestra un nivel de madurez propio de software empresarial. Al priorizar la integridad referencial y la asincronía, se entregó una solución que no solo funciona correctamente bajo condiciones ideales, sino que es capaz de recuperarse de fallos y proteger el activo más valioso del cliente: sus datos de inventario y caja.

Esta solidez se sustenta en tres decisiones de diseño fundamentales:

1. **Defensa en Profundidad (Constraints):** Más allá de las validaciones en la interfaz de usuario, la base de datos actúa como la última línea de defensa. La implementación estricta de **Constraints** (CHECK, UNIQUE, FOREIGN KEY) blinda la lógica de negocio a nivel de motor. Esto garantiza matemáticamente que no existan precios negativos, transacciones huérfanas o inconsistencias entre insumos y productos, independientemente de posibles errores en la capa de aplicación.

2. **Rendimiento y Escalabilidad (Índices):** El sistema fue diseñado pensando en el crecimiento futuro. La estrategia de **Indexación** no fue aleatoria, sino quirúrgica (índices filtrados para colas de tareas, índices de cobertura para reportes). Esto asegura que la experiencia de usuario (UX) se mantenga fluida y las consultas sean de costo computacional constante O(1) o logarítmico O(log n), incluso cuando el volumen histórico de ventas crezca a millones de registros.

3. **Ecosistema de Automatización Activa:** Quizás el mayor diferenciador de ingeniería es la orquestación entre **Triggers, Procedimientos Almacenados y Jobs**. El sistema no es un repositorio pasivo; es un organismo reactivo:

    - Los **Triggers** actúan como sensores en tiempo real (detectando cambios de stock).
    - Los **Procedimientos (SPs)** encapsulan la lógica compleja de manera segura.
    - Los **Jobs del SQL Agent** procesan estas señales de manera asíncrona.

Esta sinergia permite funcionalidades avanzadas como el "Despacho de Alertas Resiliente", donde la venta nunca se bloquea por problemas de red, delegando la comunicación a procesos de fondo que se autogestionan.

En definitiva, **A1Click** no solo resuelve la problemática operativa actual de la papelería, sino que establece una base tecnológica auditable, segura y mantenible, preparada para escalar sin deuda técnica.