

RESUMEN EJECUTIVO

Sistema de Gestión de Inventario Abuelo Cómodo

Migración a Arquitectura Moderna

Preparado por:

Ivan Duarte

Full Stack Developer

1. Contexto y Desafío

La operación de Abuelo Cómodo ha crecido significativamente en los últimos años, gestionando un catálogo de más de 400 productos, procesando miles de órdenes mensuales desde múltiples canales de venta, y coordinando inventario entre dos ubicaciones físicas: Playa Regatas y Tienda Pilares. Este crecimiento orgánico fue soportado por una arquitectura tecnológica que, aunque funcional en sus inicios, comenzó a mostrar limitaciones estructurales que impactaban la eficiencia operativa.

El sistema anterior dependía de una cadena de herramientas interconectadas: Shopify como punto de venta, Zapier como intermediario de datos, Google Sheets como base de datos central, y Apps Script ejecutándose cada cinco minutos para procesar la lógica de negocio. Esta arquitectura, aunque ingeniosa en su momento, introducía latencias de hasta quince minutos entre la recepción de una orden y su procesamiento completo. Más crítico aún, la ausencia de normalización en los datos generaba duplicados, inconsistencias, y dependencias frágiles que solo AppSheet podía resolver mediante referencias virtuales.

El presente proyecto representa una modernización fundamental de esta infraestructura, migrando hacia una arquitectura basada en eventos con PostgreSQL como motor de base de datos, eliminando intermediarios innecesarios, y estableciendo las bases para un crecimiento escalable y sostenible.

2. Solución Implementada

La nueva arquitectura reemplaza el modelo de polling periódico por un sistema dirigido por eventos. Cuando Shopify genera una nueva orden, un webhook notifica directamente a Supabase, donde Edge Functions procesan la información en tiempo real. Los triggers de PostgreSQL ejecutan automáticamente la lógica de negocio: descomponen productos compuestos en sus componentes según las recetas definidas, generan asientos contables basados en reglas configurables, y actualizan reservas de inventario. Todo esto ocurre en menos de dos segundos, comparado con los cinco a quince minutos del sistema anterior.

El esquema de base de datos fue completamente normalizado. Las 29 tablas resultantes eliminan la redundancia de datos y establecen relaciones formales mediante llaves foráneas. Los productos con recetas complejas, algunos compuestos

por hasta 19 componentes individuales, ahora se procesan mediante triggers que consultan la tabla de recetas y generan automáticamente los registros correspondientes en `orden_item_elements`. Este nivel de automatización era imposible de lograr confiablemente en el sistema basado en hojas de cálculo.

La integración con Shopify ahora opera bidireccionalmente. Los webhooks de órdenes nuevas y fulfillment alimentan el sistema, mientras que la sincronización de inventario permite que los niveles de stock se reflejen en la tienda en línea. AppSheet mantiene su rol como interfaz de usuario para el equipo operativo, pero ahora conecta directamente con Supabase en lugar de depender de Google Sheets.

3. Resultados de la Migración

La migración de datos representó un esfuerzo considerable de limpieza, deduplicación, y transformación. El sistema legado contenía años de información acumulada con las inconsistencias típicas de una base de datos no normalizada. Tras el proceso de migración, el nuevo sistema contiene 44,638 registros distribuidos en 20 tablas activas, representando la información histórica limpia y validada de la operación.

La tabla de clientes consolida 13,099 registros únicos, eliminando duplicados que existían en el sistema anterior. El historial de órdenes preserva 4,685 transacciones con sus 9,600 líneas de detalle correspondientes. Los 7,110 registros de despacho documentan el fulfillment histórico, mientras que las 562 recetas de productos definen las relaciones de componentes para los productos compuestos que constituyen una parte significativa del catálogo.

Los 8,106 prospectos migrados representan el pipeline de ventas telefónicas, ahora integrado con el flujo de conversión a clientes mediante triggers automáticos. El inventario de 673 registros mapea los 443 productos activos contra las dos ubicaciones de almacén, con campos para cantidad disponible, reservada, y en tránsito.

4. Beneficios Operativos

La reducción en tiempo de procesamiento tiene implicaciones directas en la operación diaria. Las órdenes que antes requerían hasta quince minutos para reflejarse

completamente en el sistema ahora están disponibles en segundos. Esto permite al equipo de almacén iniciar la preparación de pedidos casi inmediatamente después de que el cliente completa su compra, mejorando los tiempos de entrega y la experiencia del cliente.

La eliminación de Zapier como intermediario reduce la dependencia de servicios externos y elimina un costo recurrente mensual. Más importante aún, elimina un punto de falla en la cadena de procesamiento. El sistema anterior fallaba silenciosamente cuando Zapier experimentaba problemas, dejando órdenes sin procesar hasta que alguien detectaba la anomalía manualmente.

La normalización de datos facilita la generación de reportes confiables. Las consultas que antes requerían fórmulas complejas en Google Sheets ahora se resuelven con queries SQL directos. La integridad referencial garantizada por PostgreSQL previene inconsistencias que antes debían detectarse y corregirse manualmente.

5. Alcance del Proyecto

El proyecto se enfocó en establecer la infraestructura fundamental y migrar el flujo crítico de procesamiento de órdenes. El alcance entregado incluye el diseño e implementación del esquema de base de datos normalizado con 29 tablas, 52 restricciones de integridad referencial, y 19 triggers que automatizan la lógica de negocio. Las dos Edge Functions implementadas manejan los webhooks de Shopify para creación de órdenes y notificación de fulfillment.

El flujo de Prospectos Telefónicos hacia conversión a órdenes fue completamente implementado, incluyendo la generación automática de identificadores de orden según el asesor asignado, el procesamiento de recetas para productos compuestos, y la generación de asientos contables según las reglas configuradas. Este flujo representa el caso de uso principal para el equipo de ventas telefónicas.

Quedan fuera del alcance actual los módulos de órdenes de compra a proveedores, traspasos entre sucursales, y la interfaz web de administración contemplada en el diseño de arquitectura. Estos componentes están diseñados pero requieren desarrollo adicional para su implementación completa. La estructura de tablas y

triggers para estos módulos ya existe en el sistema, pendiente únicamente de la migración de datos históricos y la implementación de interfaces de usuario.

6. Arquitectura Técnica

El nuevo sistema se fundamenta en Supabase como plataforma de backend, aprovechando PostgreSQL como motor de base de datos con todas sus capacidades de triggers, funciones almacenadas, y vistas materializadas. Las Edge Functions, escritas en TypeScript y ejecutándose en el runtime de Deno, procesan los webhooks entrantes con tiempos de respuesta consistentemente por debajo de los 500 milisegundos.

La lógica de negocio reside principalmente en triggers de PostgreSQL. Cuando se inserta un registro en `order_items`, tres triggers se ejecutan en secuencia: el primero procesa las recetas y genera los elementos componentes, el segundo calcula y reserva inventario, y el tercero genera los asientos contables correspondientes. Esta aproximación garantiza consistencia transaccional y elimina la posibilidad de estados intermedios inconsistentes que plagaban el sistema anterior.

AppSheet continúa siendo la interfaz principal para usuarios operativos, ahora conectando directamente a Supabase mediante su conector de PostgreSQL. Las vistas de la base de datos exponen exactamente la información que cada rol necesita, mientras que las políticas de Row Level Security de Supabase controlan el acceso a nivel de registro.

7. Consideraciones para el Futuro

La arquitectura implementada está diseñada para soportar el crecimiento del negocio. PostgreSQL escala eficientemente hasta millones de registros, y los índices implementados optimizan las consultas más frecuentes. Sin embargo, existen oportunidades de mejora que deberían considerarse en fases futuras del proyecto.

La validación de webhooks mediante HMAC debería implementarse para garantizar que solo Shopify pueda invocar las Edge Functions. Actualmente, las funciones validan la estructura del payload pero no su origen. Esta mejora representa aproximadamente dos horas de desarrollo y fortalecería significativamente la seguridad del sistema.

La sincronización bidireccional completa con Shopify, particularmente para actualizaciones de inventario desde Supabase hacia la tienda, está contemplada en el diseño pero pendiente de implementación. Esta funcionalidad permitiría que ajustes de inventario realizados en AppSheet se reflejen automáticamente en Shopify, cerrando el ciclo de sincronización.

Finalmente, la documentación técnica completa del sistema, incluyendo diagramas de arquitectura, referencia de base de datos, y guías de mantenimiento, acompañará esta entrega para facilitar la continuidad del desarrollo por parte del equipo técnico de Abuelo Cómodo.

—

Este documento forma parte de la entrega del proyecto de migración.

La documentación técnica complementaria está disponible en el repositorio del proyecto.