# REPORTE DE COSTO STORE ESPECIFICO 2

Fuente:  /var/www/html/qareportes/app/Http/Controllers/Cloud/CentroCostosController.php

# 1) Nombre del reporte

- Nombre: Reporte de Costo Store Específico 2
- Código / Alias: update_specific_store
- Responsable:  Equipo de Reportes Cloud
- Propósito (1 línea):

# 2) Alcance temporal

- Campo(s) de fecha utilizados:
  - first_day, last_day (generados por obtainRangeDaysByYear)
  - from_date, to_date (pasados a reportes ERPNext)

- Tipo de rango (diario / mensual / personalizado):
  - Mensual dinámico, basado en el mes actual (date("m"))
  - Se generan rangos de inicio y fin de mes (Y-m-d)
  - Puede limitarse al mes actual o retroceder 2 meses (según condición en $first_month).

- Inclusión de horas (sí/no).: Sí [00:00:00 – 23:59:59]
- Zona horaria: La del servidor

# 3) Origen de datos (tablas/APIs)

3.1 Conexión / Base: CAPACITACION

- Tabla / Recurso: report_centro_costo_store_especifico_2
- Alias (si aplica): $table
- Campos utilizados (clave → descripción):
    - id → Identificador primario autoincremental
    - agencia → Nombre de la agencia o sucursal
    - producto_servicio → Tipo de elemento (‘PRODUCTO’ o ‘SERVICIO’)
    - nombre_articulo → Nombre del artículo o servicio
    - stock_inicial → Stock inicial del periodo
    - unidades_recibidas → Cantidad ingresada en el periodo
    - unidades_vendidas → Cantidad vendida
    - merma → Cantidad perdida o desechada
    - inventario_final → Existencia final calculada
    - costo_de_compra → Costo unitario de compra
    - precio_venta → Precio unitario de venta
    - valor_inventario_final → Valor total del inventario final
    - valor_venta → Valor total de ventas
    - valor_compra → Valor total de compras
    - retabilidad → Diferencia entre valor de venta y compra
    - ano → Año del reporte
    - periodo → Mes del reporte
    - created_date → Fecha de creación del registro
    - status → Estado lógico del registro (activo = 1)

- Condiciones (WHERE) aplicadas:
    - Dinámicas según rango de fechas (first_date, second_date)
    - Filtros heredados de subconsultas (warehouse, item_group, price_list)

- Tipo de unión con otras fuentes (JOIN y llave): No aplica directamente (la consolidación se realiza por combinación de arrays en PHP)
- Observaciones (ej. particiones, índices):
    - La tabla se recrea dinámicamente por mes/año (CREATE TABLE report_centro_costo_store_especifico_2_YYYY_MM)
    - Se indexa el campo status
    - Inserciones masivas por bloques de 900 registros

3.2 Conexión / Base: CAPACITACION

- Tabla / Recurso: tabWarehouse as alm
- Alias (si aplica): $data_warehouse
- Campos utilizados (clave → descripción):
    - alm.name → Nombre del almacén

- Condiciones (WHERE) aplicadas: WHERE LOCATE('Tienda', alm.name) > 0 (solo almacenes que contienen la palabra “Tienda”)

- Tipo de unión con otras fuentes (JOIN y llave): Se cruza con warehouse de report_stock_balance mediante coincidencia exacta de nombre

- Observaciones (ej. particiones, índices):
    - Retorna únicamente almacenes activos de tipo “Tienda”
    - Estructura esperada: {status, message, data[]}

3.3 Conexión / Base: CAPACITACION

- Tabla / Recurso: tabDelivery Note
- Alias (si aplica): $data_desecho
- Campos utilizados (clave → descripción):
    - dni.warehouse → Almacén origen
    - dn.creation → Fecha de creación
    - dni.item_code → Código del artículo
    - dni.qty → Cantidad desechada

- Condiciones (WHERE) aplicadas:
    - dn.sele = 'Desecho'
    - dn.status = 'To Bill'
    - dn.creation BETWEEN first_date AND second_date

- Tipo de unión con otras fuentes (JOIN y llave): LEFT JOIN tabDelivery Note Item AS dni ON (dn.name = dni.parent)

- Observaciones (ej. particiones, índices):
    - Permite calcular el valor de merma por producto
    - API: method/send-query-database

3.4 Conexión / Base: CAPACITACION

- Tabla / Recurso: tabDelivery Note Item

- Alias (si aplica): $data_desecho
- Campos utilizados (clave → descripción):
  - dni.warehouse → Almacén origen
  - dn.creation → Fecha de creación
  - dni.item_code → Código del artículo
  - dni.qty → Cantidad desechada

- Condiciones (WHERE) aplicadas:
  - dn.sele = 'Desecho'
  - dn.status = 'To Bill'
  - dn.creation BETWEEN first_date AND second_date

- Tipo de unión con otras fuentes (JOIN y llave): LEFT JOIN tabDelivery Note Item AS dni ON (dn.name = dni.parent)

- Observaciones (ej. particiones, índices):
  - Permite calcular el valor de merma por producto
  - API: method/send-query-database

3.5 Conexión / Base: CAPACITACION

- Tabla / Recurso: method/erpnext.stock.report.stock_balance.stock_balance.report_stock_balance
- Alias (si aplica): $get_report_stock_balance
- Campos utilizados (clave → descripción):
  - warehouse → Nombre o código del almacén
  - item_name → Nombre del artículo o producto
  - item_group → → Grupo o categoría del ítem
  - opening_qty → Cantidad inicial en inventario
  - in_qty → Cantidad ingresada al almacén
  - out_qty → Cantidad salida del almacén
  - val_rate → Valor unitario o tasa de valoración del producto

- Condiciones (WHERE) aplicadas:
  - company = 'Shalom Empresarial'
  - item_group = 'TIENDA'

- from_date, to_date dinámicos

- Tipo de unión con otras fuentes (JOIN y llave):
  - Combina con tabWarehouse (por nombre de almacén)
  - Cruza con get_desechos (por warehouse e item_code)

- Observaciones (ej. particiones, índices):
  - Endpoint protegido con token de autenticación
  - Retorna estructura JSON con message (array de registros)

3.6 Conexión / Base: CAPACITACION

- Tabla / Recurso: method/erpnext.stock.report.stock_ledger.stock_ledger.report_stock
- Alias (si aplica): $get_report_stock_ledger
- Campos utilizados (clave → descripción):
  - item_name → Nombre del artículo o producto
  - actual_qty → Cantidad actual disponible en inventario
  - valuation_rate → Valor unitario o tasa de valoración del ítem
  - qty_after_transaction → Cantidad total después de la transacción
  - in_qty → Cantidad ingresada en la transacción
  - out_qty → Cantidad retirada o despachada en la transacción

- Condiciones (WHERE) aplicadas:
  - from_date, to_date, company = 'Shalom Empresarial'
  - warehouse = 'ALMACEN GENERAL-TIENDA - SE'

- Tipo de unión con otras fuentes (JOIN y llave):
  - Relación por item_name con los productos de get_report_stock_balance

- Observaciones (ej. particiones, índices):
  - Permite calcular stock inicial y final detallado
  - Respuesta JSON con niveles [message][1] para los datos válidos

3.7 Conexión / Base: CAPACITACION

- Tabla / Recurso: tabItem Price
- Alias (si aplica): $pricelist_respo

- Campos utilizados (clave → descripción):
  - price_list_rate → Precio de venta estándar

- Condiciones (WHERE) aplicadas:
  - price_list = 'Venta estándar'
  - selling = 1
  - docstatus = 0
  - item_name = <producto>

- Tipo de unión con otras fuentes (JOIN y llave):
  - Se asocia por item_name con registros del balance de stock

- Observaciones (ej. particiones, índices):
  - Se consulta individualmente por cada producto
  - Orden descendente por creation (último precio vigente)

# 4) Filtros globales del reporte

- Inclusiones (must-have):
  - Solo artículos con grupo 'TIENDA'
  - Solo almacenes cuyo nombre contenga 'Tienda'
  - Solo movimientos de la empresa 'Shalom Empresarial'

- Exclusiones (reglas de descarte):
  - Desechos no clasificados fuera de rango
  - Precios con docstatus ≠ 0

- Reglas por estado / habilitado: Campos status = 1 (activos en tabla destino)
- Parámetros externos (si el usuario puede filtrarlo): No se expone interfaz de usuario directa (parámetros internos calculados).

# 5) Transformaciones y Reglas de negocio

- Derivaciones de campos (cómo se calculan):
  - inventario_final = opening_qty + in_qty - out_qty - merma
  - valor_inventario_final = inventario_final * val_rate
  - valor_venta = out_qty * price_list_rate
  - valor_compra = out_qty * val_rate
  - retabilidad = valor_venta - valor_compra

- Mapeos de estado:
  - item_group == 'SERVICIOS' → producto_servicio = 'SERVICIO'
  - En caso contrario → 'PRODUCTO'

- Reglas de validación (p.ej., sólo registros validados): Retorna error si alguno de los servicios ERP no devuelve status=true.

- Reglas condicionales (si X entonces Y): Si no hay branch, se obtiene de warehouse (partido por “-” para nombre de sucursal).

# 6) Estructura de salida

- Tabla/archivo destino: report_centro_costo_store_especifico_2
- Particionado / sufijo: Se crea tabla mensual: report_centro_costo_store_especifico_2_YYYY_MM
- Clave(s) primaria(s) o únicas: id (INT AUTO_INCREMENT PRIMARY KEY)
- Columnas del reporte (nombre → tipo → descripción):
  - id → INT → Identificador único del registro (PK)
  - agencia → VARCHAR(255) → Nombre de la tienda o sucursal
  - producto_servicio → ENUM → Clasificación del ítem (producto / servicio)
  - nombre_articulo → VARCHAR(225) → Descripción o nombre completo del artículo
  - stock_inicial → FLOAT → Cantidad disponible al inicio del periodo
  - unidades_recibidas → FLOAT → Total de unidades ingresadas durante el periodo
  - unidades_vendidas → FLOAT → Total de unidades vendidas o despachadas

    - merma → VARCHAR(20) → Cantidad perdida o deteriorada
    - inventario_final → FLOAT → Stock final calculado (stock_inicial + recibidas - vendidas - merma)
    - costo_de_compra → VARCHAR(20) → Costo unitario de adquisición
    - precio_venta → FLOAT → Precio unitario de venta
    - valor_inventario_final → FLOAT → Valor monetario del inventario final
    - valor_venta → FLOAT → Ingreso total generado por ventas
    - valor_compra → FLOAT → Costo total de adquisición
    - rentabilidad → FLOAT → Diferencia entre valor_venta y valor_compra
    - ano → INT → Año correspondiente al periodo
    - periodo → VARCHAR(20) → Nombre del mes o periodo (ej. “ENERO”)
    - created_date → DATETIME → Fecha y hora de creación del registro
    - status → TINYINT → Estado lógico del registro (1=activo, 0=inactivo)

- Ordenamiento por defecto: No explícito en SQL; implícito por inserción.

# 7) Frecuencia y operación

- Frecuencia de actualización: PENDIENTE (Esperando info de Eduardo)
- Ventana que recalcula (días/meses): Último mes (por defecto), con opción de extender 2 meses atrás
- Tamaño de lote / paginado (chunking): Inserción por array_chunk de 900 registros
- Tolerancia a fallos / reintentos:
    - Manejo de try-catch con rollback de transacción
    - Registro en tabla emp_reportes_validation con status=0

- Tiempos de ejecución esperados: PENDIENTE (Esperando info de Eduardo)

# 8) Calidad y controles

- Validaciones previas/post:

    - Verifica existencia de tabla destino (exist_table_bd)
    - Crea índice status si la tabla es nueva
    - Inserta registro de validación con fechas procesadas.

- Controles de duplicados: La tabla se recrea por mes → evita duplicidad de periodo
- Métricas de completitud (qué se monitorea):
    - Conteo de registros por rango de fechas validado en emp_reportes_validation.

# 9) Dependencias externas

- APIs / servicios y endpoints:
    - APICAPACITACION . "method/send-query-database"
    - APICAPACITACION . "method/erpnext.stock.report.stock_balance.stock_balance.report_stock_balance"
    - APICAPACITACION . "method/erpnext.stock.report.stock_ledger.stock_ledger.report_stock"

- Autenticación / headers:
    - Authorization: token fc60669957e6bdc:b207d472e96c9df
    - Content-Type: application/json

- Mapeos y contratos de datos:
    - Todos los endpoints devuelven message o response en JSON
    - Campo valor indica validez (true/false).

# 10) Historial de cambios

2025-10-21 14:00 - Elian Franco Arroyo Rodas - Documentación inicial

---

Revisión #1
Creado 2025-10-24 15:35:28 -05 por Elian
Actualizado 2025-10-24 15:35:48 -05 por Elian