PRIMERA ESCUELA DE ARTE MULTIMEDIAL

Trabajo Practico Integrador

Carrera: Tecnicatura Superior en Análisis de Sistemas
Asignatura: Diseño de Sistemas
Año lectivo: 2025
Docente: Lic. Matias Velasquez

Contexto del Problema

Una empresa de marketing digital necesita mejorar su gestión de información sobre clientes potenciales. Actualmente, los datos de los clientes se recogen diariamente de varias fuentes externas, pero el proceso es manual y propenso a errores. La empresa quiere automatizar este flujo de trabajo para asegurar que los datos sean precisos y estén disponibles para su análisis en todo momento.

El objetivo es desarrollar un sistema que centralice la información de los clientes en una base de datos única y accesible, actualizando automáticamente los datos mediante un trabajo diario que recolecta información de varias fuentes y la almacena de manera persistente. Una vez confirmada la actualización de datos, el sistema generará automáticamente reportes en Excel, facilitando el análisis por parte del equipo de marketing. Además, implementará una arquitectura basada en eventos, donde la finalización del proceso de actualización desencadena la generación del reporte sin intervención manual.

El sistema debe integrarse con los diferentes sistemas de gestión de las sucursales para recolectar automáticamente la información de ventas y stock al final de cada jornada. Estos datos pueden obtenerse de múltiples fuentes:

APIs de los sistemas de facturación de cada sucursal, que proporcionan registros de ventas en formatos JSON o XML. Estas APIs pueden variar entre sucursales, por lo que el sistema deberá ser capaz de estandarizar los datos recibidos.

APIs públicas de información de productos, utilizadas para completar datos faltantes o enriquecer la información almacenada, como nombres comerciales, descripciones, categorías o fabricantes.

Archivos CSV o Excel exportados desde sistemas heredados, que deben ser procesados para extraer la información relevante.

Registros internos de inventario de cada sucursal, provenientes de bases de datos locales o sistemas de gestión de stock.

Dado que los datos pueden provenir de fuentes heterogéneas, el sistema deberá contar con un proceso de limpieza y normalización de datos, que incluya:

Conversión de formatos: Unificar los datos recibidos desde distintas fuentes en una estructura común.

Validación de registros: Verificar que las transacciones de venta contengan todos los campos obligatorios (fecha, productos, cantidades, precios, etc.) y descartar registros incompletos o con errores evidentes.

Corrección de valores inconsistentes: Ajustar formatos de fechas, convertir monedas si es necesario y corregir identificadores de productos erróneos.

Enriquecimiento de datos: Si un registro de venta solo contiene un identificador de producto, completar los campos faltantes (nombre, categoría, proveedor) consultando una API externa.

Eliminación de duplicados: Evitar la carga redundante de registros al comparar los datos con transacciones previas ya almacenadas.

Una vez procesada, la información se almacenará en una base de datos centralizada, organizada en tablas que permitan consultar tanto las ventas históricas como los niveles de stock en tiempo real.

Orientaciones para el Diseño

1. **Análisis de Entidades y Atributos**:
   - Los estudiantes deberán identificar las entidades necesarias y sus atributos fundamentales, como los que se utilizan para describir a los clientes y a los reportes generados.

2. **Modelado UML**:
   - Crear un diagrama de clases que represente el diseño del sistema y sus interrelaciones.
   - Utilizar diagramas de actividades para detallar el flujo de generación de reportes basado en eventos.

3. **Arquitectura del Sistema:**
   - Aplicar un modelo C4 para representar la arquitectura del sistema, detallando los contenedores y componentes clave.
   - Considerar una arquitectura RESTful para facilitar la interacción con otros sistemas o servicios web.

4. **Persistencia y Mapeo Objetos-Relacional**:
   - Implementar un mecanismo para la persistencia de datos, adecuado a las tecnologías preferidas por los estudiantes (como Go con GORM, .NET con Entity Framework, etc.).

5. **Jobs y Patrones de Diseño/Arquitectura**:
   - Configurar un trabajo programado para la actualización diaria de datos.

- Utilizar patrones de diseño como Observer para manejar la reacción basada en eventos tras la finalización del job.
- Implementar un patrón de arquitectura basada en eventos para la generación automática de reportes.

Entregas del Trabajo Práctico Integrador

Primer Parcial:

1. Especificación de Requerimientos del Software (SRS)
   - Identificar y documentar requerimientos del sistema.

2. Diagramas UML
   - Crear diagramas que incluyan la estructura y comportamiento del sistema.

3. Diseño de Arquitectura
   - Usar el modelo C4 para describir el sistema a nivel de contexto, contenedores y componentes.

4. Persistencia
   - Implementar un ejemplo de Mapeo Objeto-Relacional para demostrar el entendimiento de la persistencia.

Segundo Parcial:

1. Actualización de Documentación
   - Ajustar cualquier cambio necesario en la documentación.

2. Principios SOLID y Patrones de Diseño
   - Revisar el uso eficaz de los principios SOLID y aplicar un patrón de diseño adecuado al sistema.

3. Arquitectura REST y Mecanismo Basado en Eventos
   - Diseñar y documentar una API RESTful / Sistema Event Driven.

Entrega Final:

- Compilación de Documentación y Código
- Presentación completa del trabajo práctico junto con la documentación respectiva.

- Evaluación Teórica
- Conjunto de preguntas para evaluar el entendimiento de los conceptos aplicados en el proyecto.