Casos de Aplicación de Microservicios y Propuesta de Solución

---

Caso 1: Sistema de Certificación y Facturación (Fragmentación por Bases de Datos - SQL Server y Oracle)

Contexto del caso: Una organización necesita gestionar el proceso de facturación, separando la lógica y persistencia de los encabezados de factura (FACTURACION) y los detalles de cada una (FACTURA_DETALLE) debido a que utilizan tecnologías heterogéneas: SQL Server y Oracle. Además, se necesita emitir un certificado digital de la factura (CERTIFICADO_FACTURA), consolidando información de ambas bases de datos.

Propuesta de Solución:

- **Modelo de Base de Datos:**
    - **SQL Server:**
        - **Tabla: FACTURACION (CAE, nit_clte, fecha_hora_certificado, fecha_hora_validacion, id_vendedor)**
        - **Tabla: VENDEDOR (id, nombre)**
    - **Oracle:**
        - **Tabla: FACTURA_DETALLE (id, cae, id_producto, cantidad, precio, descuento)**
        - **Tabla: PRODUCTO (id, nombre, stock)**
    - **MySQL:**
        - **Tabla: CERTIFICADO_FACTURA (id, cae, certificado, factura)**
- **Arquitectura:**
    - **Microservicio 1 (SQL Server) y (Oracle): Maneja facturas y vendedores (encabezado), maneja los detalles y productos de las facturas.**
    - **Microservicio 2 (MySQL): Genera el certificado uniendo datos del encabezado y detalles.**
- **Framework:**

    - **Flask para cada microservicio.**
    - **SQLAlchemy con "binds" para separar las conexiones.**
- **API:**
    - **Microservicio 1 expone: /microservice/v1/get/facturacion/<cae>**
    - **/microservice/v1/post/factura-detalle**
    - **Microservicio 2 consume /get/facturacion/<cae> para generar certificado con los datos.**

Caso 2: Sistema de Gestión Comercial para Cadena de Tiendas Minoristas (PostgreSQL)

Contexto del caso: Una empresa minorista desea implementar un sistema de microservicios para administrar sus operaciones comerciales incluyendo tiendas, vendedores, clientes, regiones, ciudades y segmentos. Esta arquitectura permitirá escalar el sistema de forma modular y mantener una organización clara y extensible de sus datos.

Propuesta de Solución:

- **Modelo de Base de Datos (PostgreSQL):**
    - **Tablas principales:**
        - **DIM_REGION, DIM_CIUDAD (con relación uno a muchos)**
        - **DIM_GERENTE (gestión administrativa)**
        - **DIM_TIENDA (metadatos de tiendas)**
        - **DIM_VENDEDOR (vendedores asociados a tiendas)**
        - **DIM_CLIENTE (información personal de clientes)**
        - **DIM_SEGMENTO (tipos de clientes)**
        - **DIM_CLIENTE_SEGMENTO (relación n a n entre clientes y segmentos)**
        - **DIM_VENDEDOR_TIENDA (historial de asociación vendedor-tienda)**
- **Arquitectura:**

    - **Microservicio por dominio:**
        - **Microservicio Clientes**
        - **Microservicio Tiendas**
        - **Microservicio Vendedores**
        - **Microservicio Segmentos**
        - **Microservicio Ciudades y Regiones**
        - **Microservicio Gerentes**
- **Framework:**
    - **Flask en todos los microservicios**
    - **SQLAlchemy como ORM**
    - **PostgreSQL como motor de base de datos**
- **API:**
    - **Endpoints RESTful por entidad:**
        - **/api/clientes, /api/segmentos, /api/vendedores, etc.**
        - **Relaciones cruzadas manejadas por microservicios compuestos (por ejemplo: cliente_segmento)**
- **Integración y validación cruzada:**
    - **Middleware para validaciones como claves UUID, unicidad y consistencia entre entidades**
    - **Validación de datos cruzando con otras tablas: por ejemplo, al registrar una tienda se valida que la ciudad exista, o al asignar un vendedor a tienda, que ambos existan y estén activos**

Ventajas de la Solución Propuesta:

- **Modularidad clara en la lógica de negocio**
- **Modelos formalizados hasta la 5 forma normal.**
- **Microservicios desacoplados y desplegables por separado**

- **Validez en el dominio gracias al uso de llaves UUID y constraints cruzadas**
- **Consistencia y escalabilidad para organizaciones con muchas tiendas y vendedores**
- **Flexibilidad para futuras integraciones (por ejemplo, analítica o gestión de KPIs)**