**PROYECTO PL/SQL taller base de datos**

**“PlayVault”**

**Integrantes:**

Cristóbal Cisternas Tilleria

Sebastián Bravo Silva

**Docente:**

Wilson Fabian González Retamales

# 

**1. Contexto de Negocio**

**PlayVault** es una empresa dedicada a la venta y renta de videojuegos retro en formato físico. Opera exclusivamente mediante tiendas físicas ubicadas en distintas ciudades, sin presencia digital. Su catálogo abarca múltiples plataformas como PS5, Xbox, PC y Nintendo Switch.

**Desafíos Operativos**

* Gestión eficiente del inventario
* Control de edad en productos con clasificación restringida
* Coordinación de repartidores para envíos y retiros en tienda
* Aplicación automática de multas por retrasos

El sistema desarrollado utiliza una base de datos relacional y objetos PL/SQL para automatizar procesos y centralizar la información clave.

**2. Datos Procesados e Información Relevante**

**Entidades Principales**

* VENTA, RENTA, RESEÑA, DETALLE\_VENTA
* Entidades complementarias: PRODUCTO, CLIENTE, SUCURSAL

**KPIs Generados**

* Clientes con mayor número de compras
* Ingresos por sucursal
* Productos sin reseña
* Producto más vendido
* Rentas con retraso
* Promedio de calificación por producto
* Total de ventas por mes

Cada KPI se almacena en una tabla específica y se actualiza automáticamente mediante triggers que ejecutan procedimientos del paquete KPI\_ANALISIS.

**3. Procedimientos y Funciones**

**Definición y características**

Los procedimientos y funciones en PL/SQL son bloques de código que encapsulan lógica de negocio. Los procedimientos ejecutan tareas sin devolver valores directamente, mientras que las funciones devuelven un valor y pueden usarse en sentencias SQL.

Ventajas y desventajas

* Procedimientos: permiten modificar datos y procesar grandes volúmenes, pero no se pueden usar directamente en SELECT.
* Funciones: devuelven valores precisos y se pueden usar en consultas, pero no deben modificar datos.

**Propósito en el proyecto**

Los procedimientos se usaron para calcular KPIs y actualizar tablas. Las funciones permiten consultar métricas específicas como ventas por cliente o promedio de calificaciones.

**Procedimientos implementados**

* Calcular\_KPI\_Clientes\_Top: cuenta ventas por cliente.
* Calcular\_KPI\_Ingresos\_Sucursal: suma ingresos por sucursal desde ventas y rentas.
* Calcular\_KPI\_Productos\_Sin\_Reseña: detecta productos sin reseñas.
* Calcular\_KPI\_Producto\_Top: suma unidades vendidas por producto.
* Calcular\_KPI\_Rentas\_Retraso: calcula días de retraso en devoluciones.
* Calcular\_KPI\_Reseñas\_Promedio: calcula promedio de calificaciones por producto.
* Calcular\_KPI\_Ventas\_Mes: agrupa ventas por mes.

**Funciones implementadas**

* Total\_Ventas\_Global(): total de todas las ventas.
* Total\_Ventas\_Cliente(p\_id\_cliente): total individual de compras.
* Promedio\_Calificacion\_Producto(p\_id\_producto): promedio de calificaciones de un producto.
* Total\_Rentas\_Retrasadas(): cantidad de rentas con retraso.

**4. Paquetes PL/SQL. Paquetes PL/SQL. Paquetes PL/SQL**

El paquete **KPI\_ANALISIS** agrupa todos los procedimientos relacionados con KPIs. Esto mejora la modularidad, reutilización del código y facilita el mantenimiento.

**5. Triggers Implementados**

Los triggers se ejecutan automáticamente ante eventos como **INSERT**, **DELETE** o **UPDATE**. Cada KPI tiene su propio trigger que se activa cuando cambia la tabla base correspondiente, permitiendo que los indicadores se actualicen en tiempo real.

**Triggers implementados**

* trg\_kpi\_clientes\_top → sobre VENTA.
* trg\_kpi\_ingresos\_sucursal\_venta → sobre VENTA.
* trg\_kpi\_ingresos\_sucursal\_renta → sobre RENTA.
* trg\_kpi\_productos\_sin\_reseña → sobre RESEÑA.
* trg\_kpi\_producto\_top → sobre DETALLE\_VENTA.
* trg\_kpi\_rentas\_retraso → sobre RENTA.
* trg\_kpi\_reseñas\_promedio → sobre RESEÑA.
* trg\_kpi\_ventas\_mes → sobre VENTA.

**Ejemplo de trigger**

CREATE OR REPLACE TRIGGER trg\_kpi\_clientes\_top  
AFTER INSERT OR DELETE OR UPDATE ON VENTA  
FOR EACH ROW  
BEGIN  
KPI\_ANALISIS.Calcular\_KPI\_Clientes\_Top;  
END;

**6. Conclusión**

El proyecto permitió desarrollar un sistema automatizado y escalable que:

* Optimiza la gestión operativa
* Permite análisis en tiempo real
* Facilita la toma de decisiones estratégicas

**7. Anexos**

* Código completo disponible en GitHub

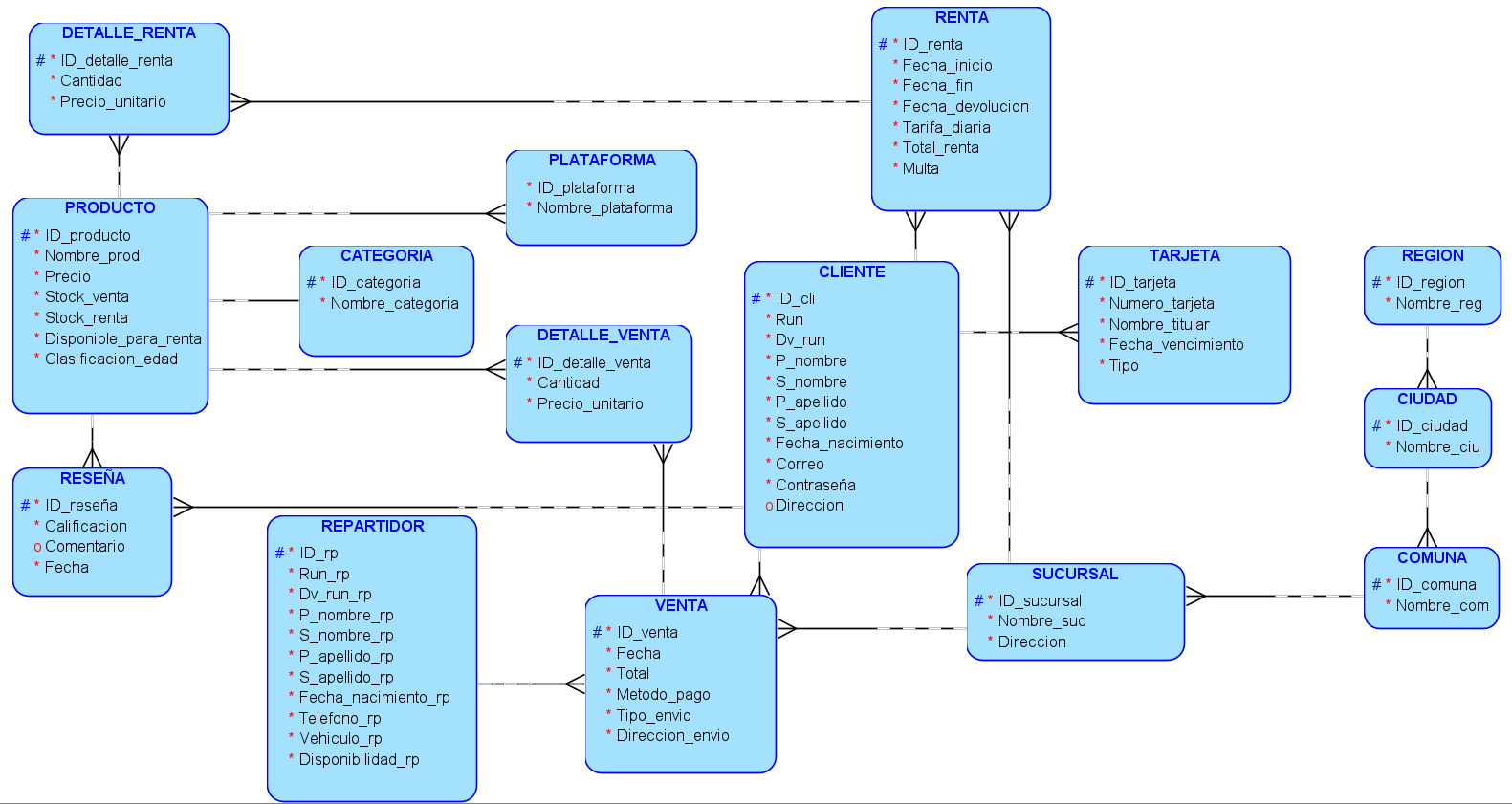
**https://github.com/crist0balact/Implementaci-n-KPIs-PlayVault-taller-base-de-datos-**

* Diagramas MER, lógico y relacional

El modelo MER incluye las siguientes entidades principales:

* Cliente, Sucursal, Producto, Repartidor, Tarjeta, Venta, Renta, Detalle\_Venta, Detalle\_Renta, Reseña, Categoría, Plataforma, Categoría, Plataforma, Región, Ciudad, Comuna.

Cada entidad está definida con sus atributos y relaciones. Por ejemplo, un cliente puede tener múltiples tarjetas, realizar ventas y rentas, y dejar reseñas. Los productos pueden pertenecer a varias categorías y plataformas. Las rentas están asociadas a tarjetas, y pueden generar multas si no se devuelven a tiempo.

*Modelo Lógico* 

*Modelo Relacional*