# Universidad del Valle

**Facultad de Ingeniería**
**Escuela de Ingeniería de Sistemas y Computación**
**Bases de Datos I - Poryecto: PL/SQL**

## NORMAS PARA LA ENTREGA DEL PROYECTO

1. Se van a realizar 3 entregas incrementales.
2. Se va a realizar en equipo de 5 personas.
3. Se deberá entregar una aplicación funcional.

---

🏛️ **Título de la Presentación: Diseño de Sistemas Bancarios Modernos**

**Documento de Requerimientos: Base de Datos y PL/SQL de un Banco**

**Objetivo:** Este documento detalla los requerimientos para el desarrollo de un sistema de gestión bancaria. Nos enfocaremos en el diseño de la **base de datos relacional (Oracle)** y la **lógica de negocio** implementada con **PL/SQL** para asegurar la integridad, seguridad y consistencia del sistema.

---

1️⃣ **Diapositiva: Modelo Entidad-Relación (MER)**

**Requerimientos Funcionales - Estructura de la Base de Datos**

En esta sección definimos las entidades y sus relaciones, que son la base para el diseño de nuestro sistema.

- **Cliente:**
  - **PK:** cliente_id
  - **Atributos:** nombre_completo, identificacion, direccion
- **Cuenta:**
  - **PK:** numero_cuenta
  - **FK:** cliente_id
  - **Atributos:** tipo_cuenta, saldo, estado ('activa', 'inactiva', 'bloqueada')
- **Transacción:**
  - **PK:** transaccion_id
  - **FK:** cuenta_id

    - **Atributos:** tipo_transaccion, monto, fecha_transaccion
- **Usuario:**
    - **PK:** usuario_id
    - **FK:** rol_id
    - **Atributos:** nombre_usuario, contrasena (encriptada)
- **Rol:**
    - **PK:** rol_id
    - **Atributos:** nombre_rol
- **Auditoria_Transacciones:**
    - **PK:** auditoria_id
    - **FKs:** transaccion_id, usuario_id
    - **Atributos:** fecha_operacion

---

2️⃣ **Diapositiva: Lógica de Negocio con PL/SQL**

**Requerimientos Funcionales - Implementación en la Base de Datos**

Aquí describimos la lógica que se ejecutará directamente en la base de datos para garantizar la seguridad y la integridad de los datos.

- **Paquetes (Packages):**
    - gestion_clientes_pkg: Contiene procedimientos como crear_cliente.
    - gestion_cuentas_pkg: Permite a administradores usar cambiar_estado_cuenta.
    - gestion_transacciones_pkg: Incluye procedimientos clave como realizar_deposito, realizar_retiro y la operación atómica realizar_transferencia. También tiene una función generar_historial.
    - autenticacion_pkg: Contiene la función validar_credenciales para un login seguro.
- **Triggers:**
    - trg_valida_transaccion_retiro: **BEFORE INSERT** en Transaccion. Se asegura de que la cuenta esté activa y tenga saldo suficiente antes de un retiro.
    - trg_auditoria_transacciones: **AFTER INSERT OR UPDATE** en Transaccion. Registra automáticamente cada operación en la tabla de auditoría.

- **Secuencias:**
    - Se usarán secuencias para autogenerar los IDs de las tablas **Cliente**, **Cuenta**, **Transaccion**, **Usuario** y **Auditoria_Transacciones**.

---

### 3️⃣ Diapositiva: Interfaz de Usuario y Comportamiento Reactivo

**Interfaz Visual y Dinámica (Orientada a Eventos)**

Esta parte del sistema está diseñada para ser moderna y reactiva, con actualizaciones en tiempo real y una experiencia de usuario fluida.

- **Pantalla de Inicio de Sesión:**
    - Un formulario simple que emite un evento de usuario.login.solicitado.
    - Muestra el **Dashboard** si el login es exitoso o un mensaje de error si no lo es.
- **Dashboard Principal (Reactivo):**
    - Muestra métricas clave como "Total de Cuentas Activas" que se actualizan en **tiempo real** con eventos del *backend*.
- **Gestión de Clientes y Cuentas (Dinámica):**
    - Una tabla que se actualiza instantáneamente cuando se recibe un evento del backend, como cuenta.estado.cambiado.
- **Pantalla de Transacciones (Asíncrona):**
    - Al realizar una transferencia, muestra un indicador de carga mientras espera la confirmación.
    - Actualiza el saldo y muestra una notificación de éxito al recibir el evento transaccion.realizada.

---

### 4️⃣ Diapositiva: Conclusión y Preguntas

**¡Ahora, a programar!**

Este documento de requerimientos es la base para un sistema bancario robusto y seguro. ¿Qué preguntas tienen sobre el diseño de la base de datos, la lógica en PL/SQL o el comportamiento de la interfaz?

- **Preguntas:**
    - ¿Cómo creen que la lógica de PL/SQL beneficia la **integridad** del sistema?
    - ¿Por qué es importante el uso de **triggers** y **secuencias**?
    - ¿Qué ventajas tiene una **interfaz reactiva** en un sistema como este?