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

**Este documento detalla los requerimientos para el desarrollo de un sistema de gestión bancaria, enfocándose en el diseño de la base de datos con Oracle y la lógica de negocio programada con PL/SQL para garantizar la integridad y consistencia del sistema.**

**1. Requerimientos Funcionales para el Modelo Entidad-Relación (MER)**

**Estos requerimientos definen la estructura inicial de la base de datos.**

**Entidades y Atributos**

* **Entidad: Cliente**
  + **cliente\_id (PK)**
  + **nombre\_completo**
  + **identificacion**
  + **direccion**
* **Entidad: Cuenta**
  + **numero\_cuenta (PK)**
  + **cliente\_id (FK)**
  + **tipo\_cuenta**
  + **saldo**
  + **estado (Ej. 'activa', 'inactiva', 'bloqueada')**
* **Entidad: Transaccion**
  + **transaccion\_id (PK)**
  + **cuenta\_id (FK)**
  + **tipo\_transaccion**
  + **monto**
  + **fecha\_transaccion**
* **Entidad: Usuario**
  + **usuario\_id (PK)**
  + **nombre\_usuario**
  + **contrasena**
  + **rol\_id (FK)**
* **Entidad: Rol**
  + **rol\_id (PK)**
  + **nombre\_rol**
* **Entidad: Auditoria\_Transacciones**
  + **auditoria\_id (PK)**
  + **transaccion\_id (FK)**
  + **usuario\_id (FK)**
  + **fecha\_operacion**

**2. Requerimientos Funcionales de PL/SQL (Ajustados)**

**Esta sección describe la lógica de negocio que se implementará directamente en la base de datos para garantizar la integridad y la seguridad.**

**2.1. Paquetes, Procedimientos y Funciones**

* **Paquete gestion\_clientes\_pkg:**
  + **Procedimiento crear\_cliente: Registra un nuevo cliente y utiliza una secuencia (seq\_clientes) para generar su ID.**
* **Paquete gestion\_cuentas\_pkg:**
  + **Procedimiento cambiar\_estado\_cuenta: Permite a un usuario administrativo modificar el estado de una cuenta.**
* **Paquete gestion\_transacciones\_pkg:**
  + **Procedimiento realizar\_deposito: Incrementa el saldo de una cuenta y valida que el estado sea activa.**
  + **Procedimiento realizar\_retiro: Decrementa el saldo y valida que el estado sea activa y que el saldo sea suficiente.**
  + **Procedimiento realizar\_transferencia: Realiza una operación atómica que llama a los procedimientos de retiro y depósito. Valida que el estado de ambas cuentas sea activa.**
  + **Función generar\_historial: Utiliza un cursor para obtener y reportar las transacciones en un rango de fechas.**
* **Paquete autenticacion\_pkg:**
  + **Función validar\_credenciales: Recibe un p\_nombre\_usuario y una p\_contrasena. Consulta la tabla Usuario y retorna el usuario\_id si las credenciales son correctas. Para mayor seguridad, la contraseña se debe validar contra un *hash* almacenado en la base de datos. En caso de fallo, debe devolver null o lanzar una excepción.**

**2.2. Triggers y Secuencias**

* **Secuencias: Se crearán secuencias para las tablas Cliente, Cuenta, Transaccion, Usuario y Auditoria\_Transacciones para asignar IDs de forma automática.**
* **Trigger trg\_valida\_transaccion\_retiro: Un trigger BEFORE INSERT en la tabla Transaccion que verifica que la cuenta esté activa y que el saldo sea suficiente antes de un retiro.**
* **Trigger trg\_auditoria\_transacciones: Un trigger AFTER INSERT OR UPDATE en la tabla Transaccion que registra cada operación en la tabla Auditoria\_Transacciones.**

**3. Interfaz de Usuario Visual y Dinámica (Orientada a Eventos)**

**Estos requerimientos describen el comportamiento reactivo de la interfaz y proporcionan *mockups* visuales para su diseño.**

**3.1. Pantalla de Inicio de Sesión**

**Esta será la primera pantalla que verá el usuario. Contendrá un formulario con campos para el nombre de usuario y la contraseña, y un botón de Ingresar. Al presionar el botón, la interfaz enviará un evento de usuario.login.solicitado y esperará la respuesta del backend. Si el login es exitoso, la aplicación emitirá un evento usuario.login.exitoso y redirigirá al usuario al Dashboard. En caso de credenciales incorrectas, se mostrará un mensaje de error.**

**3.2. Dashboard Principal (Reactivo)**

**Este mockup visualiza el panel de control del sistema bancario. Las métricas clave como "Total de Cuentas Activas" o el "Saldo Total del Banco" se actualizarán en tiempo real cuando se reciba un evento del backend, sin que el usuario tenga que refrescar la página.**

**3.3. Gestión de Clientes y Cuentas (Dinámica)**

**Esta pantalla muestra una tabla de cuentas. La interfaz se actualizará de forma instantánea al recibir un evento cuenta.estado.cambiado. Por ejemplo, si un gerente cambia el estado de una cuenta, todos los usuarios que están viendo esa pantalla verán el cambio reflejado al instante.**

**3.4. Pantalla de Transacciones (Asíncrona)**

**Este mockup ilustra la experiencia de realizar una transferencia de fondos. Al enviar el formulario, la interfaz muestra un indicador de carga mientras espera la confirmación. Una vez que el backend emite el evento transaccion.realizada, la pantalla actualiza el saldo y muestra una notificación de éxito.**