UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Universidad del Perú, Decana de América

Facultad de Ingeniería de Sistemas e Informática
E.P. de Ingeniería de Software

INFORME TÉCNICO

Sistema de gestión de donaciones HopeCare

ASIGNATURA: Base de datos II

DOCENTE: Jorge Luis Chávez Soto

ESTUDIANTE: Leslie Diaz Chambi

LIMA – PERÚ

2025

1. PRESENTACIÓN TÉCNICA

1.1 Stack tecnológico

Base de datos:

- Oracle Database 19c Enterprise Edition (Free Version)
- Pluggable Database: FREEPDB1
- Esquema: HOPECARE
- Lenguaje: PL/SQL + SQL

Backend:

- Java 17
- Spring Boot 3.2
- JDBC Template (sin ORM por requisito del curso)
- Maven como gestor de dependencias

Frontend:

- HTML5 + CSS3
- JavaScript ES6
- Thymeleaf (template engine)
- Font Awesome (iconografía)

Infraestructura:

- Servidor de aplicaciones: Tomcat embebido (Spring Boot)
- Puerto: 8080
- Context path: /hopecare

1.2 Características técnicas principales

Escalabilidad:

- Diseño normalizado (3FN)
- Índices optimizados (32 índices)
- Paginación en consultas pesadas
- Soporte para 20 usuarios concurrentes

Performance:

- Tiempo de respuesta: < 2 segundos
- Vistas materializadas para reportes
- Uso de packages compilados (PL/SQL)

Seguridad:

- Autenticación basada en hash MD5
- Control de acceso por roles
- Protección contra SQL Injection
- Auditoría completa de operaciones

2. OBJETIVOS TÉCNICOS

2.1 Objetivo general técnico

Diseñar e implementar una base de datos relacional de nivel empresarial que soporte todas las operaciones de la ONG HopeCare, garantizando:

- Integridad de datos
- Rendimiento óptimo
- Seguridad multi-nivel
- Trazabilidad completa

2.2 Objetivos específicos técnicos

1. Diseño de base de datos

- Normalización a 3FN de todas las tablas
- Modelo conceptual con 14 entidades
- Definición de 69 constraints (PK, FK, UK, CK)

2. Programación en PL/SQL

- 6 packages con 36 métodos públicos
- 8 triggers automáticos
- 8 vistas de negocio
- Funciones de seguridad (hash, validación)

3. Implementación de lógica de negocio

- Validación automática de stock

- Conversión de múltiples monedas
- Actualización atómica de inventario
- Control de integridad referencial

4. Seguridad y auditoría

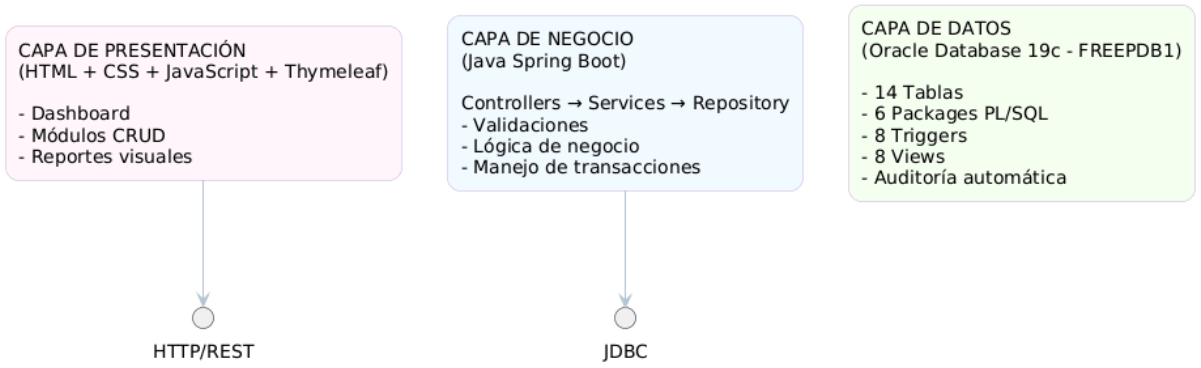
- 2 roles de usuario (ADMIN, ASSISTANT)
- Triggers de auditoría inmutables
- Protección de datos sensibles

5. Integración con aplicación

- API REST limpia
- Uso de JDBC y +sSimpleJdbcCall
- Manejo de excepciones robusto

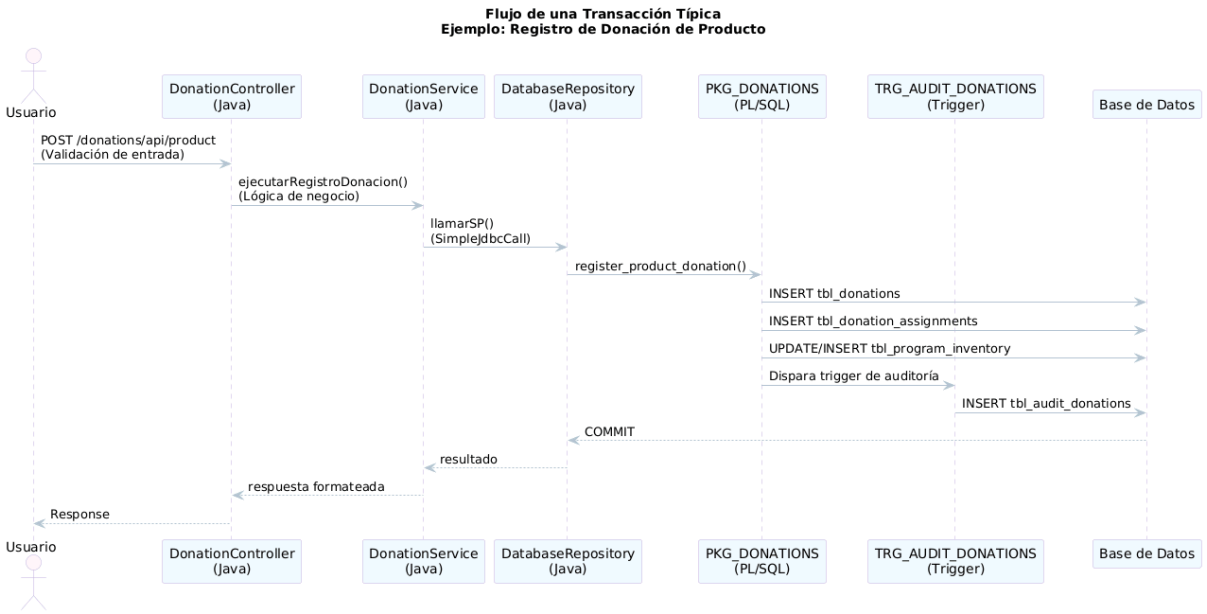
3. ARQUITECTURA DEL SISTEMA

3.1 Arquitectura de 3 capas

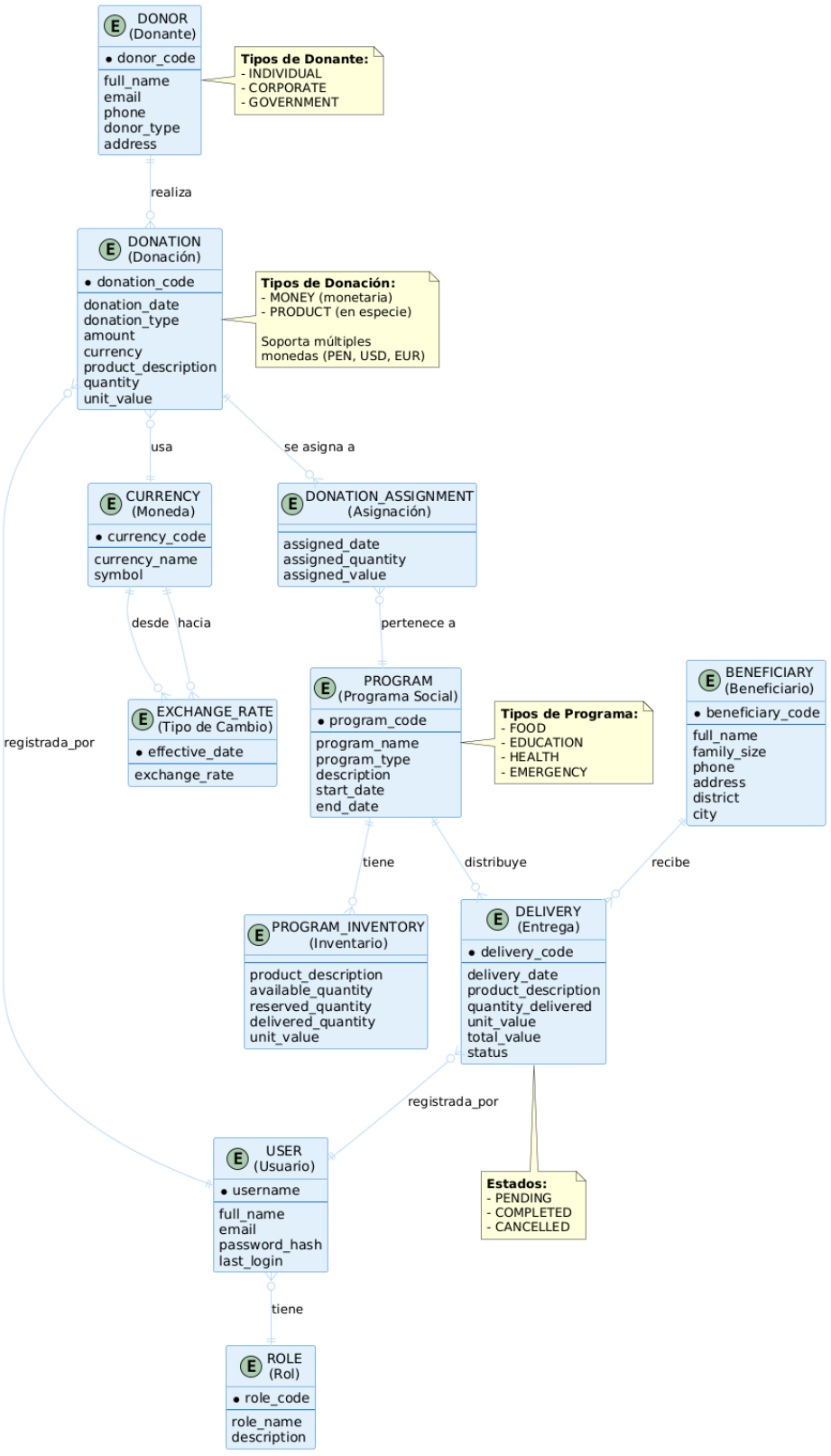


3.2 Flujo de una transacción típica

Ejemplo: Registro de donación de producto



4. MODELO DE DATOS CONCEPTUAL



4.1 Entidades principales

Catálogos (3):

1. **Currencies** - Monedas soportadas (PEN, USD, EUR)
2. **Donation Types** - Tipos de donación (Money, Product)
3. **Roles** - Roles de usuario (Admin, Assistant)

Entidades Core (4): 4. **Donors** - Donantes (individuales, corporativos, gubernamentales) 5. **Beneficiaries** - Familias/personas beneficiarias 6. **Programs** - Programas sociales (Food, Education, Health, Emergency) 7. **Users** - Usuarios del sistema

Transaccionales (5): 8. **Donations** - Registro de donaciones 9. **Donation Assignments** - Asignación de donaciones a programas 10. **Program Inventory** - Inventario disponible por programa 11. **Deliveries** - Entregas a beneficiarios 12. **Exchange Rates** - Tasas de cambio históricas

Auditoría (2): 13. **Audit Donations** - Trazabilidad de cambios en donaciones 14. **Audit Deliveries** - Trazabilidad de cambios en entregas

4.2 Relaciones principales

DONORS (1) ————————< (N) DONATIONS

DONATIONS (1) ————< (N) DONATION_ASSIGNMENTS

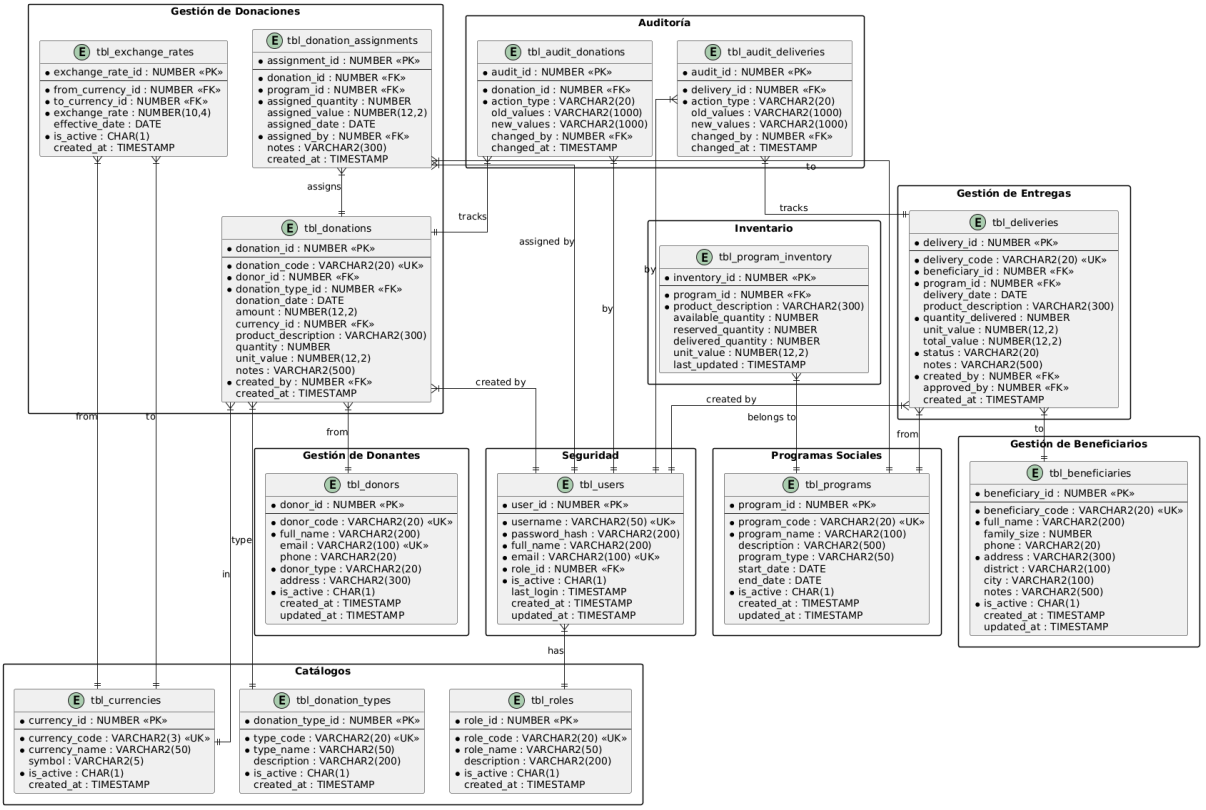
PROGRAMS (1) ————————< (N) DONATION_ASSIGNMENTS

PROGRAMS (1) ————————< (N) PROGRAM_INVENTORY

PROGRAMS (1) ————————< (N) DELIVERIES

BENEFICIARIES (1) < (N) DELIVERIES

5. MODELO DE DATOS LÓGICO



5.1 Normalización

Todas las tablas están en 3FN :

1FN: No hay grupos repetidos, todos los atributos son atómicos
2FN: No hay dependencias parciales de la clave primaria
3FN: No hay dependencias transitivas

Ejemplo de Normalización:

MAL (No normalizado):

DONATION(id, donor_name, donor_email, amount, program_name, program_type)

BIEN (3FN):

DONOR(id, name, email)

PROGRAM(id, name, type)

DONATION(id, donor_id FK, program_id FK, amount)

5.2 Áreas de información

1. Gestión de catálogos

- tbl_currencies (3 registros)

- tbl_donation_types (2 registros)
- tbl_roles (2 registros)

2. Gestión de entidades principales

- tbl_donors
- tbl_beneficiaries
- tbl_programs
- tbl_users

3. Gestión transaccional

- tbl_donations
- tbl_donation_assignments
- tbl_program_inventory
- tbl_deliveries
- tbl_exchange_rates

4. Auditoría y trazabilidad

- tbl_audit_donations
- tbl_audit_deliveries

6. MODELO DE DATOS FÍSICO

6.1 Convenciones de nomenclatura

Elemento	Convención	Ejemplo
Tablas	tbl_[nombre_plural]	tbl_donations
Secuencias	seq_[nombre_tabla]	seq_donations
Índices	idx_[tabla]_[campo]	idx_donations_donor_id
PK	pk_[tabla]	pk_donations
FK	fk_[tabla]_[referencia]	fk_donations_donor
UK	uk_[tabla]_[campo]	uk_donors_email
CK	ck_[tabla]_[campo]	ck_donations_amount
Triggers	trg_[acción]_[tabla]	trg_audit_donations
Views	vw_[descripción]	vw_donor_summary
Packages	pkg_[módulo]	pkg_donations

6.2 Especificación de tablas principales

TBL_DONORS

CREATE TABLE tbl_donors (

```
    donor_id NUMBER PRIMARY KEY,

    donor_code VARCHAR2(20) NOT NULL UNIQUE,

    full_name VARCHAR2(200) NOT NULL,

    email VARCHAR2(100) UNIQUE,

    phone VARCHAR2(20),

    donor_type VARCHAR2(20) CHECK (donor_type IN
('INDIVIDUAL','CORPORATE','GOVERNMENT')),

    address VARCHAR2(300),

    is_active CHAR(1) DEFAULT 'Y' CHECK (is_active IN ('Y','N')),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

Índices:

- idx_donors_email (UNIQUE)
- idx_donors_type
- idx_donors_active
- idx_donors_name (funcional: UPPER(full_name))

TBL_DONATIONS

```
CREATE TABLE tbl_donations (

    donation_id NUMBER PRIMARY KEY,

    donation_code VARCHAR2(20) NOT NULL UNIQUE,

    donor_id NUMBER NOT NULL,

    donation_type_id NUMBER NOT NULL,

    donation_date DATE DEFAULT SYSDATE,

    -- Money donation fields

    amount NUMBER(12,2),

    currency_id NUMBER,

    -- Product donation fields
```

```
    product_description VARCHAR2(300),

    quantity NUMBER,

    unit_value NUMBER(12,2),

    notes VARCHAR2(500),

    created_by NUMBER NOT NULL,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_donations_donor FOREIGN KEY (donor_id) REFERENCES
tbl_donors(donor_id),

    CONSTRAINT fk_donations_type FOREIGN KEY (donation_type_id) REFERENCES
tbl_donation_types(donation_type_id),

    CONSTRAINT fk_donations_currency FOREIGN KEY (currency_id) REFERENCES
tbl_currencies(currency_id),

    CONSTRAINT fk_donations_created_by FOREIGN KEY (created_by) REFERENCES
tbl_users(user_id)

);
```

Índices:

- idx_donations_donor
- idx_donations_date
- idx_donations_type
- idx_donations_donor_date (compuesto)

6.3 Constraints implementados

Total: 69 constraints

Tipo	Cantidad	Nomenclatura
Primary Keys	14	pk_*
Foreign Keys	18	fk_*
Unique	12	uk_*
Check	25	ck_*

Ejemplo de check constraints:

-- Validar tipo de donante

CONSTRAINT ck_donors_type CHECK (donor_type IN ('INDIVIDUAL','CORPORATE','GOVERNMENT'))

-- Validar estado activo

CONSTRAINT ck_donors_active CHECK (is_active IN ('Y','N'))

-- Validar montos positivos

CONSTRAINT ck_donations_amount CHECK (amount > 0)

-- Validar cantidades positivas

CONSTRAINT ck_donations_quantity CHECK (quantity > 0)

7. OBJETOS DE BASE DE DATOS

7.1 Resumen de objetos

Tipo de Objeto	Cantidad	Nomenclatura
Tablas	14	tbl_*
Secuencias	14	seq_*
Índices	32	idx_*
Packages	6	pkg_*
Triggers	8	trg_*
Views	8	vw_*
Functions	3	fn_*
Procedures	3	sp_*
TOTAL	**88**	

7.2 Packages PL/SQL

PKG_DONATIONS

Procedures:

- `register_money_donation` - Registra donación monetaria
- `register_product_donation` - Registra donación de producto
- `get_donor_history` - Historial de donaciones por donante

Functions:

- `convert_to_base_currency` - Conversión automática a PEN
- `get_donor_total_contributions` - Total de contribuciones en PEN

PKG_DELIVERIES

Procedures:

- `perform_delivery` - Realiza entrega con validación de stock
- `cancel_delivery` - Cancela entrega y devuelve stock
- `get_beneficiary_history` - Historial de entregas

Functions:

- `validate_stock` - Valida disponibilidad de stock
- `get_available_stock` - Consulta stock disponible

PKG_REPORTS

Procedures:

- `total_donations_by_period` - Donaciones por período
- `donations_by_program` - Donaciones agrupadas por programa
- `top_donors` - Ranking de donantes
- `deliveries_summary` - Resumen de entregas
- `program_inventory_status` - Estado de inventario
- `donations_by_type` - Donaciones por tipo (Money vs Product)

PKG_DONORS

Procedures:

- `register_donor` - Alta de nuevo donante
- `update_donor` - Actualización de donante
- `deactivate_donor` - Baja lógica
- `search_donors` - Búsqueda por término
- `get_all_donors` - Listado completo
- `get_donor_by_id` - Consulta individual

PKG_BENEFICIARIES

Procedures:

- `register_beneficiary` - Alta de beneficiario
- `update_beneficiary` - Actualización
- `deactivate_beneficiary` - Baja lógica

- `search_beneficiaries` - Búsqueda
- `get_all_beneficiaries` - Listado completo
- `get_beneficiary_by_id` - Consulta individual
- `get_beneficiaries_by_district` - Filtro geográfico

PKG_PROGRAMS

Procedures:

- `create_program` - Creación de programa
- `update_program` - Actualización
- `deactivate_program` - Baja lógica
- `get_all_programs` - Listado completo
- `get_program_by_id` - Consulta individual
- `get_program_statistics` - Estadísticas del programa
- `search_programs` - Búsqueda

7.3 Triggers

TRG_AUDIT_DONATIONS (AFTER INSERT/UPDATE/DELETE)

- Registra automáticamente todos los cambios en tbl_audit_donations
- Captura: acción, valores anteriores, nuevos valores, usuario, timestamp

TRG_AUDIT_DELIVERIES (AFTER INSERT/UPDATE/DELETE)

- Auditoría de entregas en tbl_audit_deliveries

TRG_DONORS_UPDATE_TIMESTAMP (BEFORE UPDATE)

- Actualiza automáticamente el campo updated_at

TRG_BENEFICIARIES_UPDATE_TIMESTAMP (BEFORE UPDATE)

- Actualiza updated_at en beneficiarios

TRG_PROGRAMS_UPDATE_TIMESTAMP (BEFORE UPDATE)

- Actualiza updated_at en programas

TRG_PREVENT_DELIVERY_DELETE (BEFORE DELETE)

- Previene eliminación de entregas completadas

TRG_PROTECT_AUDIT_DONATIONS (BEFORE UPDATE/DELETE)

- Garantiza inmutabilidad de registros de auditoría

TRG_VALIDATE_DONATION_AMOUNT (BEFORE INSERT/UPDATE)

- Valida montos y cantidades positivas

7.4 Views

vw_donor_summary

- Donantes con total de donaciones, valor en PEN, última donación

vw_beneficiary_summary

- Beneficiarios con total de entregas, cantidad y valor recibido

vw_program_summary

- Programas con estadísticas completas (donaciones, entregas, beneficiarios)

vw_donation_details

- Detalle completo de donaciones con joins a 6 tablas

vw_delivery_details

- Detalle completo de entregas con información de beneficiario y programa

vw_inventory_status

- Estado actual de inventario con indicadores de stock

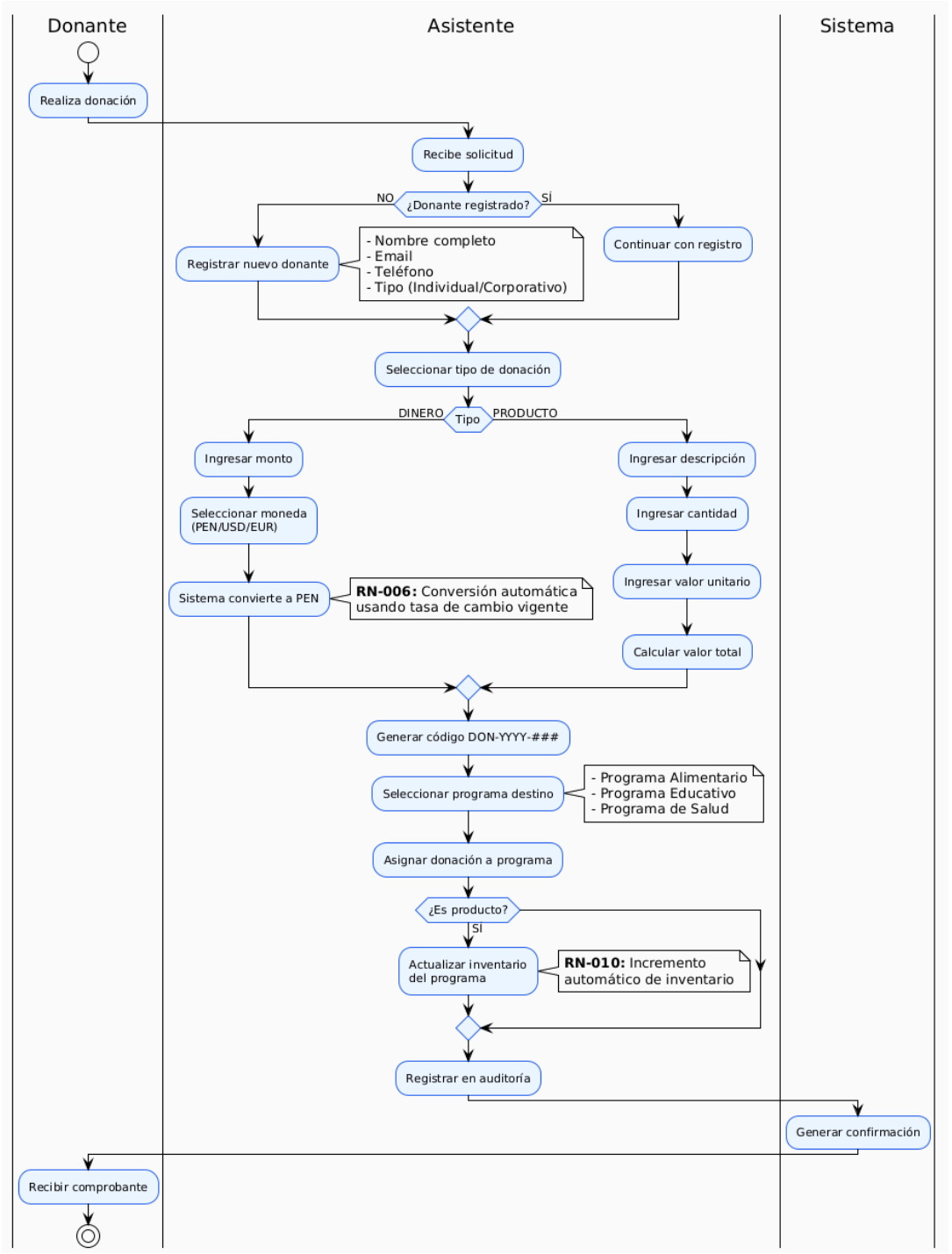
vw_recent_activity

- Actividad reciente (últimos 30 días) de donaciones y entregas

vw_dashboard_metrics

- KPIs consolidados para dashboard ejecutivo

8. PROCESOS DE NEGOCIO



8.1 Proceso: Gestión de donaciones

Descripción: Proceso completo desde registro de donación hasta asignación a programa.

Actores:

- Donante (externo)
- Asistente (registra)
- Administrador (supervisa)

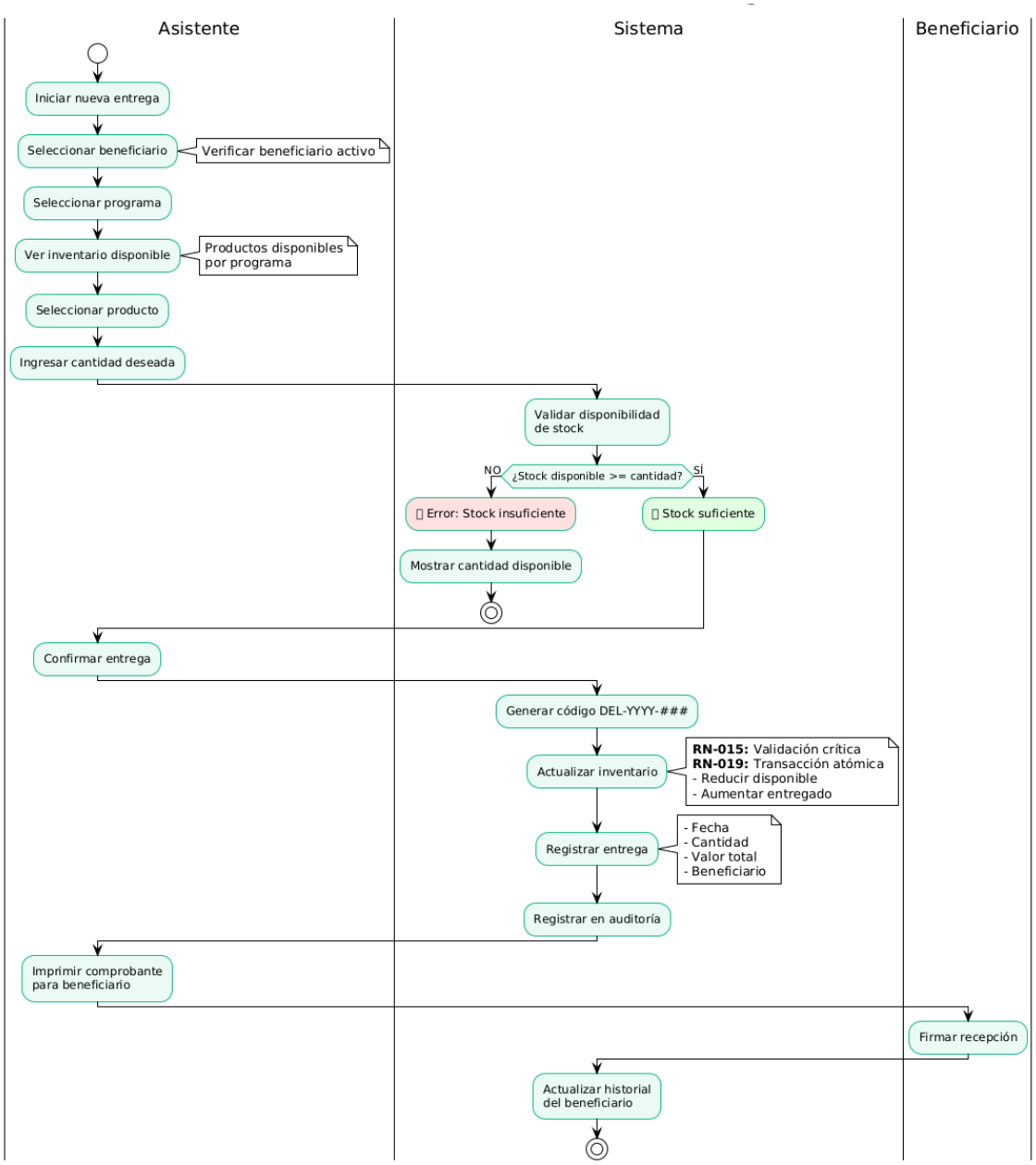
Flujo:

1. Donante contacta a HopeCare
2. Asistente registra datos del donante (si es nuevo)
3. Asistente registra donación:
    - Si es monetaria: monto + moneda
    - Si es producto: descripción + cantidad + valor unitario
4. Sistema convierte automáticamente a PEN (si aplica)
5. Asistente asigna donación a programa
6. Sistema actualiza inventario (si es producto)
7. Sistema genera comprobante
8. Trigger registra en auditoría

Validaciones:

- Donante debe existir y estar activo
- Programa debe existir y estar activo
- Montos/cantidades deben ser > 0
- Moneda obligatoria para donaciones monetarias

8.2 Proceso: Gestión de Entregas



Descripción: Distribución de recursos desde inventario hasta beneficiario.

Actores:

- Asistente (ejecuta entrega)
- Beneficiario (recibe)
- Sistema (valida stock)

Flujo:

1. Asistente selecciona beneficiario
2. Asistente selecciona programa y producto
3. Asistente ingresa cantidad a entregar
4. Sistema valida stock disponible CRÍTICO
5. Si hay stock:

    - Sistema registra entrega
    - Sistema reduce stock disponible
    - Sistema incrementa cantidad entregada
    - Trigger registra en auditoría
6. Si NO hay stock:
    - Sistema rechaza operación
    - Muestra mensaje de error

Validaciones:

- Beneficiario debe existir y estar activo
- Programa debe existir y estar activo
- Producto debe existir en inventario del programa

9. REGLAS DE NEGOCIO

9.1 Módulo: Donantes

ID	Regla	Descripción	Implementación
RN-001	Donante único	Todo donante debe tener email o código único	CONSTRAINT UNIQUE
RN-002	Tipo de donante	Solo puede ser: INDIVIDUAL, CORPORATE, GOVERNMENT	CHECK constraint
RN-003	Donante activo	Solo donantes con is_active='Y' pueden recibir donaciones	Validación en package

9.2 Módulo: Donaciones

ID	Regla	Descripción	Implementación
RN-004	Donación con donante	Toda donación DEBE tener un donante válido	FK + validación
RN-005	Moneda obligatoria	Donaciones monetarias DEBEN especificar moneda	PL/SQL validation
RN-006	Conversión automática	Sistema convierte todas las donaciones a PEN usando tasa de cambio vigente	Function convert_to_base_currency
RN-007	Valor mínimo	Donaciones monetarias: monto > 0	CHECK (amount > 0)
RN-008	Productos con cantidad	Donaciones de productos DEBEN tener cantidad > 0	CHECK (quantity > 0)
RN-009	Asignación a programa	Toda donación debe asignarse a un programa existente	FK + trigger

RN-010	Actualización de inventario	Donaciones de productos incrementan automáticamente el inventario del programa	Trigger/Package

9.3 Módulo: Beneficiarios

ID	Regla	Descripción	Implementación
RN-011	Beneficiario único	Beneficiario identificado por código único	CONSTRAINT UNIQUE
RN-012	Familia mínima	family_size >= 1 (al menos la persona registrada)	CHECK (family_size >= 1)
RN-013	Dirección obligatoria	Todo beneficiario DEBE tener dirección registrada	NOT NULL + validación
RN-014	Beneficiario activo	Solo beneficiarios activos pueden recibir entregas	WHERE is_active='Y'

9.4 Módulo: Entregas

ID	Regla	Descripción	Implementación
RN-015	Validación de stock	**NO SE PUEDE entregar más de lo disponible en inventario**	**PL/SQL Package validation**
RN-016	Beneficiario válido	Toda entrega debe ser para un beneficiario activo y registrado	FK + validation
RN-017	Programa activo	Solo se puede entregar de programas activos	Validation
RN-018	Cantidad positiva	Cantidad entregada > 0	CHECK (quantity_delivered > 0)
RN-019	Actualización atómica	Al confirmar entrega, se debe: reducir stock disponible + aumentar entregado	Transaction + Package
RN-020	Estado de entrega	Solo puede ser: PENDING, COMPLETED, CANCELLED	CHECK constraint
RN-021	Entrega cancelada	Si se cancela una entrega COMPLETED, se devuelve el stock	Procedure cancel_delivery

9.5 Módulo: Programas

ID	Regla	Descripción	Implementación

RN-022	Programa único	Cada programa tiene código único PROG###	UNIQUE constraint
RN-023	Fechas coherentes	end_date >= start_date (si existe end_date)	CHECK constraint
RN-024	Tipo de programa	Debe ser: FOOD, EDUCATION, HEALTH, EMERGENCY	CHECK o tabla catálogo
RN-025	No borrar con entregas	No se puede desactivar un programa con entregas pendientes	Business logic

9.6 Módulo: Usuarios y seguridad

ID	Regla	Descripción	Implementación
RN-026	Usuario único	Username debe ser único en el sistema	UNIQUE constraint
RN-027	Contraseña hasheada	Las contraseñas se almacenan cifradas (MD5 mínimo)	Function hash_password
RN-028	Roles definidos	Solo existen 2 roles: ADMIN y ASSISTANT	FK a tbl_roles
RN-029	Permisos por rol	- ADMIN: acceso total - ASSISTANT: sólo operaciones	Interceptor (Spring)
RN-030	Auditoría completa	Todas las operaciones críticas se registran	Triggers de auditoría

9.7 Módulo: Auditoría

ID	Regla	Descripción	Implementación
RN-031	Registro inmutable	Los registros de auditoría NO se pueden modificar ni eliminar	Trigger de protección
RN-032	Auditoría automática	Toda operación INSERT/UPDATE/DELETE en donaciones y entregas se registra automáticamente	AFTER trigger
RN-033	Información completa	Se registra: acción, valores anteriores, valores nuevos, usuario, timestamp	Trigger implementation

9.8 Módulo: Donaciones

RN-004: Toda donación DEBE tener un donante válido
Implementación: FK + validación en package

RN-006: Sistema convierte todas las donaciones a PEN usando tasa de cambio vigente
Implementación: `function convert_to_base_currency`

RN-007: Donaciones monetarias: monto > 0
Implementación: `CHECK (amount > 0)`

RN-010: Donaciones de productos incrementan automáticamente el inventario del programa
Implementación: Trigger en `PKG_DONATIONS.register_product_donation`

9.9 Módulo: Entregas

RN-015: NO SE PUEDE entregar más de lo disponible en inventario
Implementación: `PKG_DELIVERIES.validate_stock` (PL/SQL validation)

```
-- Ejemplo de validación de stock

IF NOT validate_stock(p_program_id, p_product_description, p_quantity) THEN

   v_available_stock := get_available_stock(p_program_id, p_product_description);

   RAISE_APPLICATION_ERROR(-20104,

      'Insufficient stock. Available: ' || v_available_stock ||

      ', Requested: ' || p_quantity);

END IF;
```

RN-019: Al confirmar entrega, se debe reducir stock disponible + aumentar entregado
Implementación: Transaction en package

```
UPDATE tbl_program_inventory

SET available_quantity = available_quantity - p_quantity,

   delivered_quantity = delivered_quantity + p_quantity,

   last_updated = CURRENT_TIMESTAMP

WHERE inventory_id = v_inventory_id;
```

10. IMPLEMENTACIÓN DE SEGURIDAD

10.1 Arquitectura de seguridad

Niveles de Seguridad:

Nivel 1: Autenticación

├─ Hash MD5 de contraseñas

├─ Validación en login

└─ Sesión HTTP

Nivel 2: Autorización

├─ Roles: ADMIN, ASSISTANT

├─ Permisos por módulo

└─ Interceptor en Spring Boot

Nivel 3: Auditoría

├─ Triggers automáticos

├─ Registros inmutables

└─ Timestamp + Usuario

Nivel 4: Protección de Datos

├─ Bind variables (SQL Injection)

├─ Validación de entrada

└─ DBMS_ASSERT

10.2 Funciones de seguridad

fn_hash_password

```
CREATE OR REPLACE FUNCTION fn_hash_password(p_password VARCHAR2)
RETURN VARCHAR2 IS
BEGIN
   RETURN LOWER(RAWTOHEX(DBMS_CRYPTO.HASH(
      UTL_RAW.CAST_TO_RAW(p_password),
      DBMS_CRYPTO.HASH_MD5
   )));
END;
```

fn_validate_password

```
CREATE OR REPLACE FUNCTION fn_validate_password(
    p_username VARCHAR2,
    p_password VARCHAR2
) RETURN BOOLEAN IS
    v_stored_hash VARCHAR2(200);
    v_is_active CHAR(1);
BEGIN
    SELECT password_hash, is_active INTO v_stored_hash, v_is_active
    FROM tbl_users
    WHERE UPPER(username) = UPPER(p_username);

    IF v_is_active = 'N' THEN RETURN FALSE; END IF;

    IF fn_hash_password(p_password) = v_stored_hash THEN
        UPDATE tbl_users SET last_login = CURRENT_TIMESTAMP
        WHERE UPPER(username) = UPPER(p_username);
        COMMIT;
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN RETURN FALSE;
END;
```

10.3 Control de acceso por roles

Matriz de permisos:

Módulo	Admin	Assistant
Dashboard	Ver	Ver
Donors	CRUD	x
Beneficiaries	CRUD	x
Programs	CRUD	x
Donations	CRUD	Crear/Ver
Deliveries	CRUD	CRUD
Reports	Todos	Básicos

Implementación en Spring Boot:

```
@Component
public class SecurityInterceptor implements HandlerInterceptor {
@Override
    public boolean preHandle(HttpServletRequest request,
                             HttpServletResponse response,
                             Object handler) {
        User user = (User) session.getAttribute("user");
        String requestPath = request.getRequestURI();


        // URLs solo para ADMIN
        String[] adminOnlyPaths = {"/donors", "/beneficiaries", "/programs"};
        for (String adminPath : adminOnlyPaths) {
            if (requestPath.contains(adminPath)) {
                if (!"Administrator".equals(user.getRoleName())) {
                    response.sendRedirect("/");
                    return false;
                }
```

```
            }
        }
        return true;
    }
}
```

11. SCRIPTS Y PROCEDIMIENTOS

11.1 Orden de ejecución

1. 01_SETUP.sql - Creación de usuario y permisos
2. 02_TABLES.sql - Tablas y secuencias
3. 03_DATA_CATALOGS.sql - Datos de catálogos
4. 03_DATA_TEST.sql - Datos de prueba
5. 04_PKG_DONATIONS.sql - Package de donaciones
6. 05_PKG_DELIVERIES.sql - Package de entregas
7. 06_PKG_REPORTS.sql - Package de reportes
8. 07_PKG_DONORS.sql - Package de donantes
9. 08_PKG_BENEFICIARIES.sql - Package de beneficiarios
10. 09_PKG_PROGRAMS.sql - Package de programas
11. 10_TRIGGERS.sql - Triggers de auditoría
12. 11_VIEWS.sql - Vistas de negocio
13. 12_TEST_OPERATIONS.sql - Suite de pruebas
14. 13_SECURITY.sql - Configuración de seguridad
15. 14_BACKUP.sql - Procedimientos de backup
16. Script_Transacciones.sql - Pruebas de transacciones
17. Script_Concurrencia.sql - Control de concurrencia
18. Script_SQL_Injection.sql - Pruebas de seguridad
19. Script_Auditoria_BD.sql - Auditoría de sistema

11.2 Script de transacciones

Propósito: Demostrar flujos transaccionales completos con validaciones.

Casos de prueba:

1. Donación monetaria con conversión de moneda
2. Donación de productos con actualización de inventario
3. Entrega con validación de stock
4. Intento de entrega sin stock (debe fallar)
5. Flujo completo: Donación → Inventario → Entrega

Resultado esperado:

- Todas las transacciones válidas se completan
- Transacciones inválidas son rechazadas
- Inventario se mantiene consistente
- Auditoría registra todas las operaciones

12. CONTROL DE CONCURRENCIA

12.1 Mecanismos implementados

1. Bloqueos implícitos

- Oracle bloquea automáticamente en UPDATE/DELETE
- Nivel de fila (row-level locking)

2. SELECT FOR UPDATE

```
SELECT available_quantity INTO v_stock

FROM tbl_program_inventory

WHERE program_id = p_program_id

AND product_description = p_product_description

FOR UPDATE;  -- Bloqueo explícito
```

3. NOWAIT y WAIT n

```
-- No esperar si recurso está bloqueado

SELECT * FROM tbl_inventory

FOR UPDATE NOWAIT;

-- Esperar máximo 5 segundos
```

SELECT * FROM tbl_inventory

FOR UPDATE WAIT 5;

4. Niveles de Aislamiento

- Oracle por defecto: READ COMMITTED
- Evita dirty reads
- Permite lecturas no repetibles (acceptable para este caso)

12.2 Escenarios de Concurrencia