Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería de Sistemas e Informatica

Escuela Académico Profesional de Ingeniería de Software

“Entrega parcial 2”

Docente: Jorge Luis Chávez Soto

Curso: Base de datos II

Integrantes:

- Zafra Venegas, Andrey Hegel

- Ramirez Garcia, Estefano Alexis

- Ramirez Alvarez Maleck Adriano

- Llontop Falcon ,Henry Alessandro

- Quispe Alvarado Luis Guillermo

LIMA-PERÚ

2025

Epígrafe

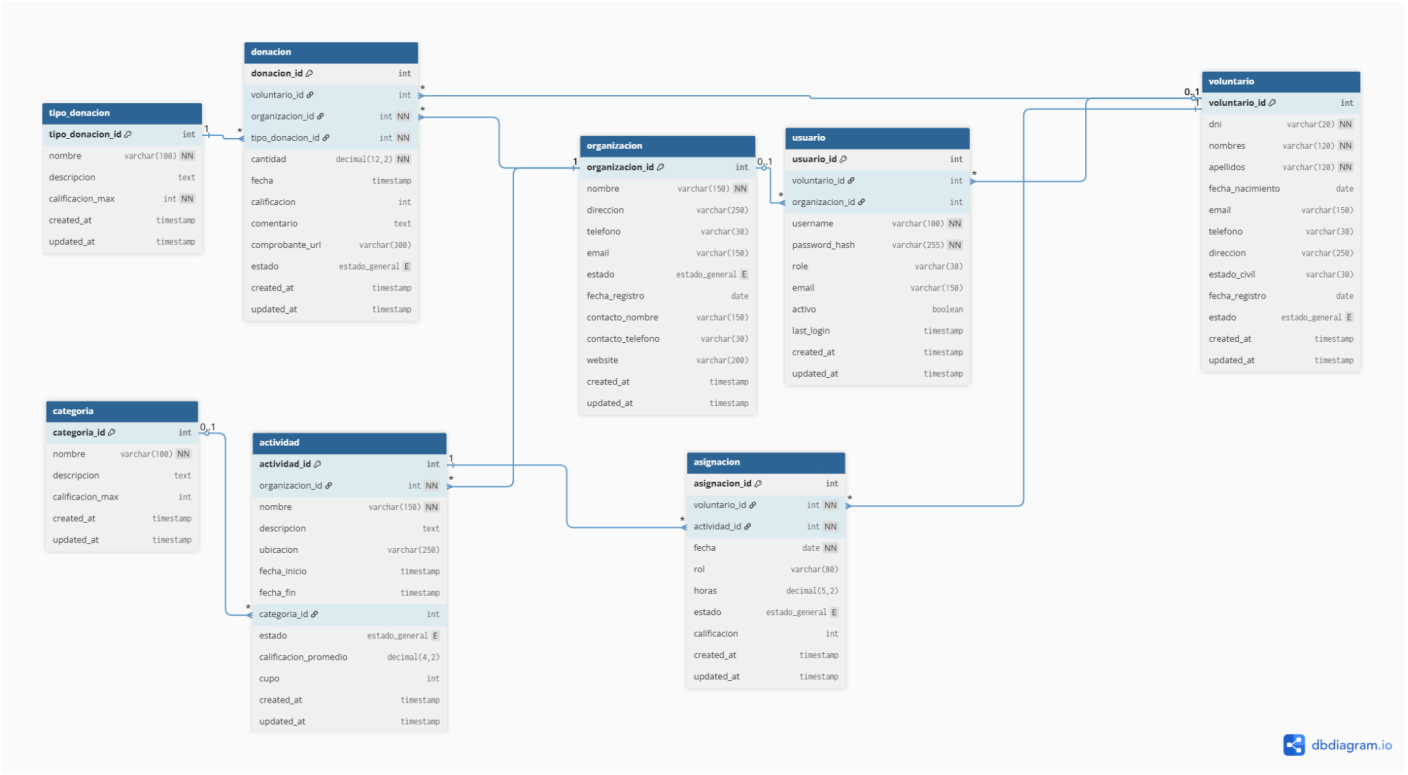
"El éxito es la suma de pequeños esfuerzos repetidos día tras día."

— Robert Collier

Índice

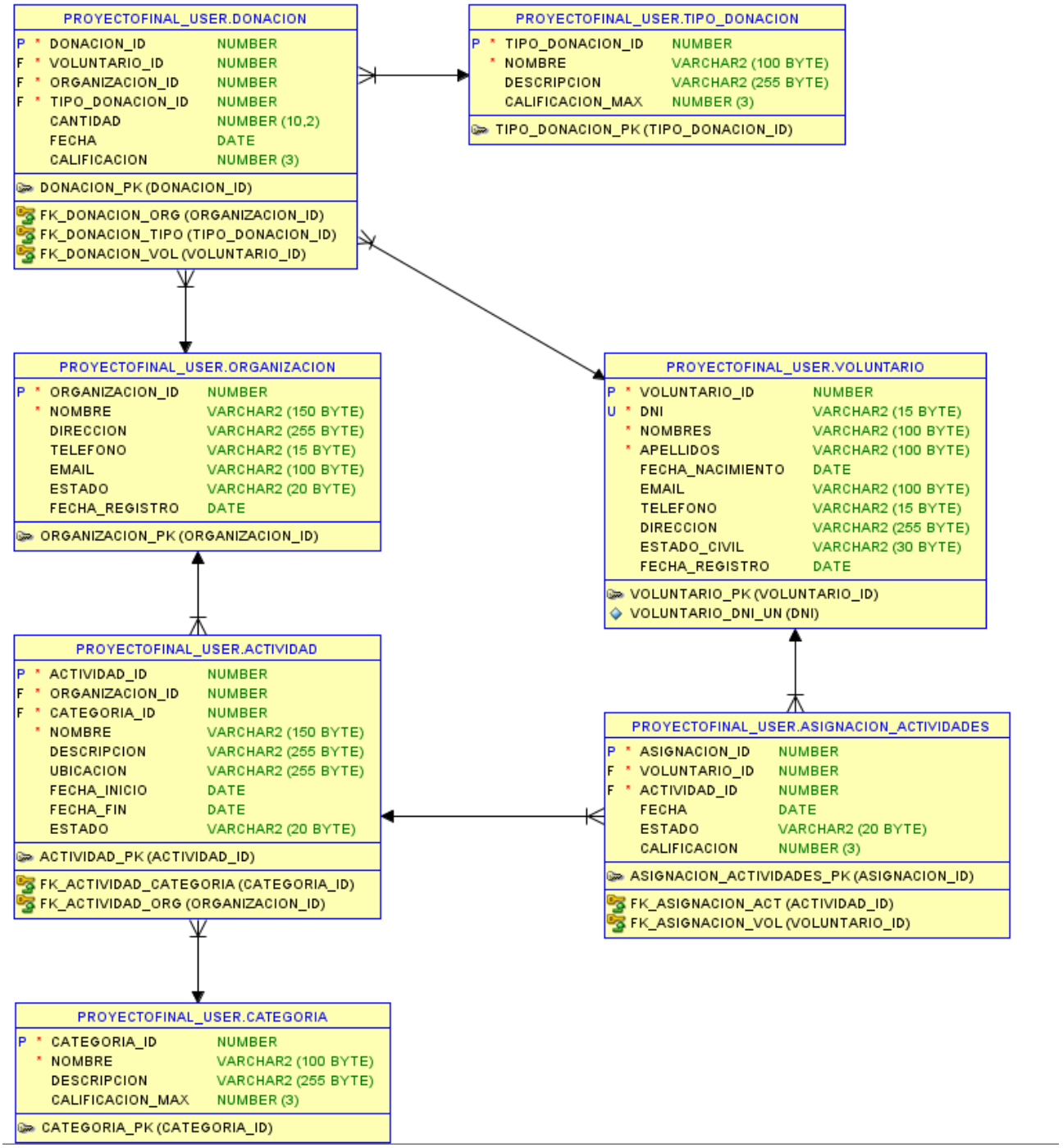
Modelo de Datos Lógico.....4
Modelo de datos físico..... 5
Esquema de base de datos.....6

Modelo de Datos Lógico



Para más detalle: https://dbdiagram.io/d/68c985361ff9c616bdf6eb08

Modelo de datos físico



Esquema de base de datos

Scripts de generación de esquemas de Base de Datos

```
CREATE TABLESPACE ProyectoBD2_Final
DATAFILE 'C:\APP\ANDRE\PRODUCT\21C\ORADATA\XE\XEPDB1\ProyectoBD2_Final.DBF'
SIZE 128M
AUTOEXTEND ON NEXT 64M MAXSIZE 2048M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

CREATE TEMPORARY TABLESPACE TempProyectoBD2_Final
TEMPFILE 'C:\APP\ANDRE\PRODUCT\21C\ORADATA\XE\XEPDB1\TempProyectoBD2_Final.DBF'
SIZE 128M

CREATE USER proyectoFinal_user IDENTIFIED BY proyectoFinal
DEFAULT TABLESPACE ProyectoBD2_Final
TEMPORARY TABLESPACE TempProyectoBD2_Final
QUOTA UNLIMITED ON ProyectoBD2_Final;
```

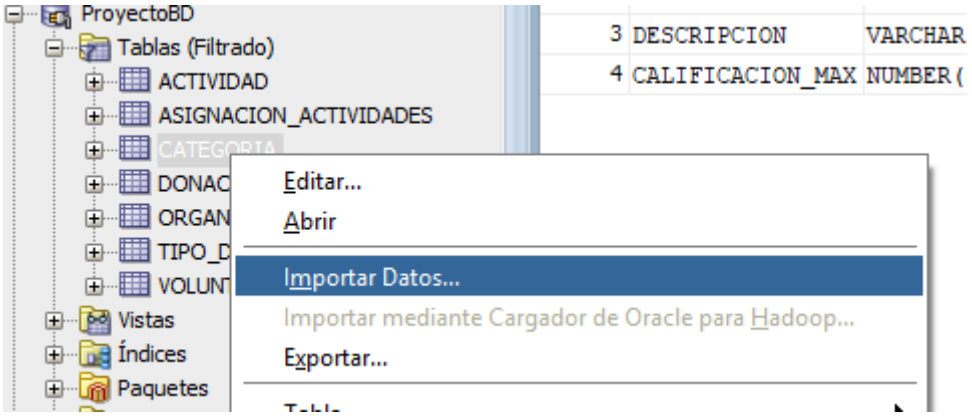
Scripts de generación de objetos de Base de Datos

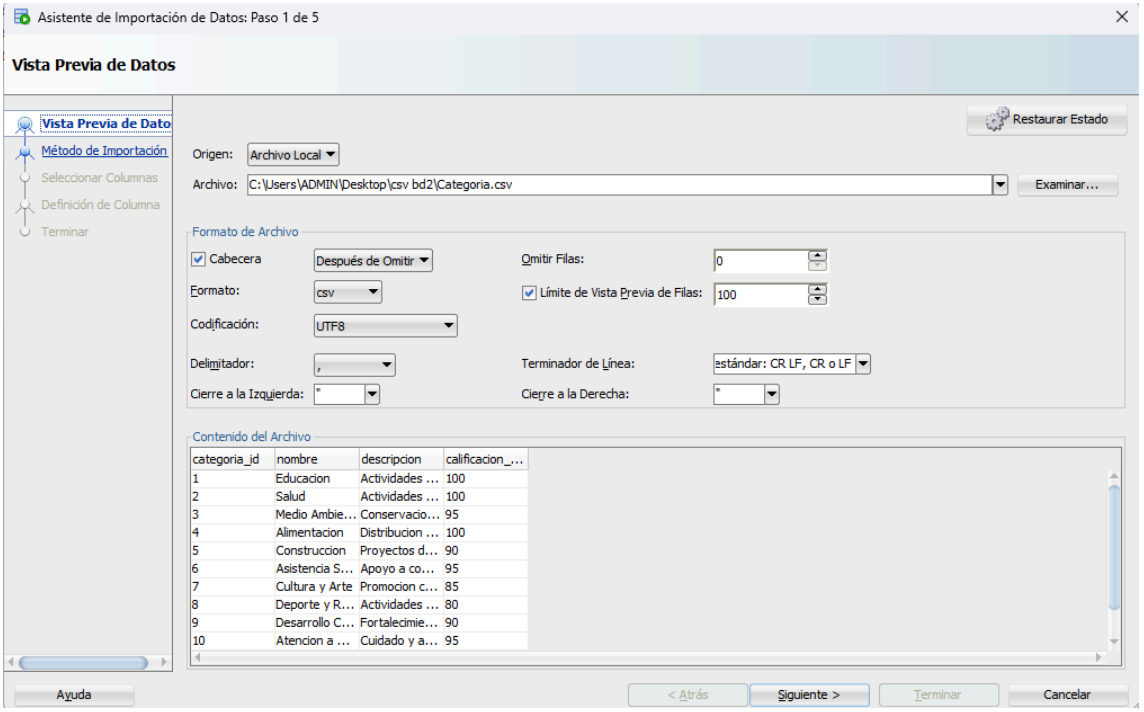
```
CREATE TABLE Tipo_donacion (
    tipo_donacion_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nombre VARCHAR2(100) NOT NULL,
    descripcion VARCHAR2(255),
    calificacion_max NUMBER(3)
);

CREATE TABLE Organizacion (
    organizacion_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nombre VARCHAR2(150) NOT NULL,
    direccion VARCHAR2(255),
    telefono VARCHAR2(15),
    email VARCHAR2(100),
    estado VARCHAR2(20),
    fecha_registro DATE DEFAULT SYSDATE
);

CREATE TABLE Voluntario (
    voluntario_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    dni VARCHAR2(15) UNIQUE NOT NULL,
    nombres VARCHAR2(100) NOT NULL,
    apellidos VARCHAR2(100) NOT NULL,
    fecha_nacimiento DATE,
    email VARCHAR2(100),
    telefono VARCHAR2(15),
    direccion VARCHAR2(255),
    estado_civil VARCHAR2(30),
    fecha_registro DATE DEFAULT SYSDATE
);

CREATE TABLE Categoria (
    categoria_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nombre VARCHAR2(100) NOT NULL,
    descripcion VARCHAR2(255),
    calificacion_max NUMBER(3)
);
```

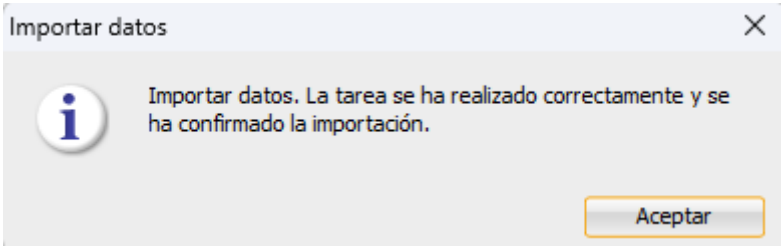
```
CREATE TABLE Actividad (
    actividad_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    organizacion_id NUMBER NOT NULL,
    categoria_id NUMBER NOT NULL,
    nombre VARCHAR2(150) NOT NULL,
    descripcion VARCHAR2(255),
    ubicacion VARCHAR2(255),
    fecha_inicio DATE,
    fecha_fin DATE,
    estado VARCHAR2(20),
    CONSTRAINT fk_actividad_org FOREIGN KEY (organizacion_id)
        REFERENCES Organizacion (organizacion_id),
    CONSTRAINT fk_actividad_categoria FOREIGN KEY (categoria_id)
        REFERENCES Categoria (categoria_id)
);

CREATE TABLE Donacion (
    donacion_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    voluntario_id NUMBER NOT NULL,
    organizacion_id NUMBER NOT NULL,
    tipo_donacion_id NUMBER NOT NULL,
    cantidad NUMBER(10,2),
    fecha DATE DEFAULT SYSDATE,
    calificacion NUMBER(3),
    CONSTRAINT fk_donacion_vol FOREIGN KEY (voluntario_id)
        REFERENCES Voluntario (voluntario_id),
    CONSTRAINT fk_donacion_org FOREIGN KEY (organizacion_id)
        REFERENCES Organizacion (organizacion_id),
    CONSTRAINT fk_donacion_tipo FOREIGN KEY (tipo_donacion_id)
        REFERENCES Tipo_donacion (tipo_donacion_id)
);

CREATE TABLE Asignacion_actividades (
    asignacion_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    voluntario_id NUMBER NOT NULL,
    actividad_id NUMBER NOT NULL,
    fecha DATE DEFAULT SYSDATE,
    estado VARCHAR2(20),
    calificacion NUMBER(3),
    CONSTRAINT fk_asignacion_vol FOREIGN KEY (voluntario_id)
        REFERENCES Voluntario (voluntario_id),
    CONSTRAINT fk_asignacion_act FOREIGN KEY (actividad_id)
        REFERENCES Actividad (actividad_id)
);
```

Scripts de carga de Datos

Se usa la herramienta de importar datos de un csv que viene en sql developer







Scripts de Creación de Objetos de Programación almacenados

Procedimiento almacenado: Registrar nueva donación

Automatiza el proceso de insertar donaciones nuevas, asegurando que solo se necesiten los parámetros principales (el sistema añade fecha y otros datos automáticamente).

```
CREATE OR REPLACE PROCEDURE RegistrarDonacion (
    p_voluntario_id     IN NUMBER,
    p_organizacion_id   IN NUMBER,
    p_tipo_donacion_id  IN NUMBER,
    p_cantidad          IN NUMBER,
    p_calificacion      IN NUMBER DEFAULT NULL
)
AS
    v_exists NUMBER;
BEGIN
    -- Verificar existencia del voluntario
    SELECT COUNT(*) INTO v_exists FROM Voluntario WHERE voluntario_id = p_voluntario_id;
    IF v_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'El voluntario no existe.');
    END IF;

    -- Verificar existencia de la organización
    SELECT COUNT(*) INTO v_exists FROM Organizacion WHERE organizacion_id = p_organizacion_id;
    IF v_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'La organización no existe.');
    END IF;

    -- Verificar existencia del tipo de donación
    SELECT COUNT(*) INTO v_exists FROM Tipo_donacion WHERE tipo_donacion_id = p_tipo_donacion_id;
    IF v_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'El tipo de donación no existe.');
    END IF;
```

```
    -- Inserción de la donación
    INSERT INTO Donacion (
        voluntario_id,
        organizacion_id,
        tipo_donacion_id,
        cantidad,
        fecha,
        calificacion
    ) VALUES (
        p_voluntario_id,
        p_organizacion_id,
        p_tipo_donacion_id,
        p_cantidad,
        SYSDATE,
        p_calificacion
    );

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Donación registrada correctamente.');
END;
/
```

Función almacenada: Calcular total donado por voluntario

Devuelve el monto total donado por un voluntario. Puede usarse en reportes o consultas de desempeño de voluntarios.

```
CREATE OR REPLACE FUNCTION CalcularTotalDonado (
    p_voluntario_id IN NUMBER
) RETURN NUMBER
IS
    v_total NUMBER := 0;
BEGIN
    -- Calcula la suma total de donaciones realizadas por el voluntario
    SELECT NVL(SUM(cantidad), 0)
    INTO v_total
    FROM Donacion
    WHERE voluntario_id = p_voluntario_id;

    RETURN v_total;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
END;
/
```

Trigger: Actualizar estado de actividad automáticamente

Actualiza automáticamente el estado de una actividad según sus fechas, evitando errores humanos y manteniendo la consistencia de datos.

```
CREATE OR REPLACE TRIGGER ActualizarEstadoActividad
BEFORE INSERT OR UPDATE ON Actividad
FOR EACH ROW
BEGIN
    IF :NEW.fecha_inicio > SYSDATE THEN
        :NEW.estado := 'Programada';
    ELSIF :NEW.fecha_inicio <= SYSDATE
       AND (:NEW.fecha_fin IS NULL OR :NEW.fecha_fin > SYSDATE) THEN
        :NEW.estado := 'En curso';
    ELSIF :NEW.fecha_fin <= SYSDATE THEN
        :NEW.estado := 'Finalizada';
    END IF;
END;
/
```

Vista: Vista de donaciones detalladas

Facilita las consultas y reportes mostrando toda la información de donaciones en una sola vista sin tener que hacer múltiples joins cada vez

```
CREATE OR REPLACE VIEW Donaciones_Detalladas AS
SELECT
    d.donacion_id,
    d.fecha AS fecha_donacion,
    d.cantidad,
    d.calificacion AS calificacion_donacion,

    v.voluntario_id,
    v.nombres || ' ' || v.apellidos AS nombre_voluntario,
    v.dni,
    v.email AS email_voluntario,

    o.organizacion_id,
    o.nombre AS nombre_organizacion,
    o.email AS email_organizacion,

    td.tipo_donacion_id,
    td.nombre AS tipo_donacion,
    td.descripcion AS descripcion_tipo_donacion
FROM Donacion d
JOIN Voluntario v ON d.voluntario_id = v.voluntario_id
JOIN Organizacion o ON d.organizacion_id = o.organizacion_id
JOIN Tipo_donacion td ON d.tipo_donacion_id = td.tipo_donacion_id;
```

Paquete PL/SQL: Gestión de voluntarios

Agrupa operaciones comunes relacionadas con voluntarios (registro y conteo), haciendo más organizada y reutilizable la lógica del sistema.

```
CREATE OR REPLACE PACKAGE pkg_voluntarios AS
    PROCEDURE registrar_voluntario(
        p_dni IN VARCHAR2,
        p_nombres IN VARCHAR2,
        p_apellidos IN VARCHAR2,
        p_email IN VARCHAR2,
        p_telefono IN VARCHAR2,
        p_direccion IN VARCHAR2
    );

    FUNCTION contar_voluntarios RETURN NUMBER;
END pkg_voluntarios;
/

CREATE OR REPLACE PACKAGE BODY pkg_voluntarios AS
    PROCEDURE registrar_voluntario(
        p_dni IN VARCHAR2,
        p_nombres IN VARCHAR2,
        p_apellidos IN VARCHAR2,
        p_email IN VARCHAR2,
        p_telefono IN VARCHAR2,
        p_direccion IN VARCHAR2
    )
    IS
    BEGIN
        INSERT INTO Voluntario (dni, nombres, apellidos, email, telefono, direccion)
        VALUES (p_dni, p_nombres, p_apellidos, p_email, p_telefono, p_direccion);
        DBMS_OUTPUT.PUT_LINE('Voluntario registrado correctamente.');
    END;

    FUNCTION contar_voluntarios RETURN NUMBER IS
        v_count NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_count FROM Voluntario;
        RETURN v_count;
    END;
END pkg_voluntarios;
/
```