S.O.S AYUDA SIN FRONTERAS

Herrería Oña, Asier
Oviedo Ramírez, Alexis
Paradas Borrego, Álvaro
Viera Chaves, Fco Javier

ÍNDICE

INTRODUCCIÓN AL PROBLEMA

Este es un proyecto donde procederemos a colaborar con la ONG S.O.S Ayuda Sin Fronteras para el proyecto de la asignatura de IISSI (Introducción a la Ingeniería del Software y a los Sistemas de Información). Este proyecto surgió como alternativa diferente a la idea normal de colaborar con una empresa o comercio. Queremos compartir este pequeño apartado sobre las obras realizadas por parte de esta ONG, llevada a cabo por parte del Parque de Bomberos Escultor Sebastián Santos (Parque de Bomberos nº5, Sevilla):

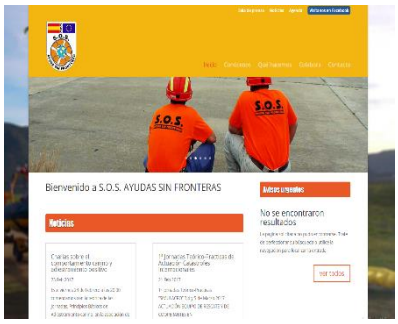
Dirección Parque de Bomberos(Sevilla)

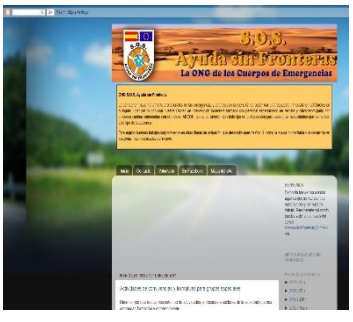
ONG S.O.S. AYUDA SIN FRONTERAS

La componen mayoritariamente profesionales de las emergencias y se crea con la intención de poder dar una respuesta inmediata en catástrofes en cualquier lugar del mundo con nuestra Unidad de Intervención Inmediata formada con personal especializado en rescate y salvamento junto con unidades caninas entrenadas con el método ARCON, contando también con todo tipo de profesionales para esas otras necesidades que demandan este tipo de situaciones.

Este equipo humano trabaja conjuntamente en otras líneas de actuación que desarrolla nuestra O.N.G. como la ayuda humanitaria y al desarrollo en las zonas más necesitadas del mundo.

Nos pusimos en contacto con el administrador de la ONG y nos informó de la necesidad de la informatización de algunos de sus cometidos, la creación de un sistema por medio el cual sus voluntarios pudieran registrarse en la web, gestionar el sistema de donaciones y pagos de cuota.





Página Web: S.O.S. Ayuda Sin Fronteras |La O.N.G. de los Cuerpos de Emergencias

Blogspot: S.O.S Ayuda Sin Fronteras

CATÁLOGO DE REQUISITOS

Se va a implementar un sistema informático en web que permita:

SISTEMA DE REGISTRO PARA USUARIOS

Como administrador de la ONG

Quiero obtener una plataforma donde aquellas personas que quieran pertenecer a la ONG puedan rellenar un registro de información con un mínimo de campos obligatorios.

Para poder obtener información sobre aquellas personas que van a proceder a colaborar, tanto si son voluntarios o donantes y poder pagar las cuotas de voluntario.

INFORMACIÓN SOBRE DONANTES

Como voluntario

Quiero tener la información necesaria sobre todos los voluntarios de la ONG: NIF, nombre, apellidos, dirección, email y teléfono.

Para poder tener un documento con todos los integrantes que componen el voluntariado de la ONG.

INFORMACIÓN SOBRE DONACIONES

Como voluntario

Quiero tener toda la información posible sobre los recursos aportados a la causa, ya sea por parte anónima o particular: a qué campaña va destinada, el tipo de donación realizada ya sea capital o material, la cantidad donada y la información sobre el donante.

Para poder llevar un mayor control sobre los ingresos y gastos realizados, además de una buena contabilización, que irá ligado al sistema de donaciones.

INFORMACIÓN SOBRE CAMPAÑAS

Como voluntario

Quiero ver información sobre las diferentes campañas activas y antiguas: nombre de la campaña, fecha de inicio, fecha de finalización, presupuesto y el sistema de voluntariado responsable de tal campaña.

Para poder tener toda la información sobre todas las campañas realizadas y las que se están realizando actualmente.

INFORMACIÓN SOBRE SISTEMA DE DONACIONES

Como voluntario

Quiero disponer de un formato donde obtenga toda la información sobre los ingresos que se hayan recaudado y en los que se haya invertido.

Para que los integrantes de la ONG o personas ajenas estén informados y se vuelquen por la causa.

GLOSARIO DE TÉRMINOS

Acción humanitaria: actividad desarrollada por una ONG para ayudar a las víctimas de desastres.

Administrador Web: persona especificada con poder para modificar la página Web de la ONG además de la información de los usuarios registrados.

Blog: medio utilizado por personas para expresar públicamente una opinión o tratar de varios temas de interés.

Campaña: objetivo dentro de una acción humanitaria que se realiza durante un determinado período de tiempo mediante unos recursos obtenidos mediante donaciones.

Catástrofe: Suceso desdichado en el que se produce gran destrucción y muchas desgracias con grave alteración del desarrollo normal de las cosas.

Cuerpos de emergencia/Profesionales de emergencia: una brigada/cuerpo de emergencia es un cuerpo de élite que dispone de la preparación y los recursos para participar ante una situación caótica o catastrófica.

Donación: acto por el cual una persona transmite gratuitamente una cosa que le pertenece a otra que la acepta.

Donante: persona o institución que hace alguna donación a una campaña. Donación: aportación económica o de medios materiales realizadas por un donante a una campaña. Se admiten donaciones anónimas.

Modelo conceptual: es la descripción de cómo se relacionan los conceptos en un problema. El modelo conceptual sirve para representar un problema de manera gráfica a través de diagramas entidad relación, diccionarios/glosarios y diagrama de clases.

ONG: organización no gubernamental que se dedica a actividades humanitarias, sin fines lucrativos.

Página Web: es un documento o información electrónica capaz de contener texto, sonido, vídeo, programas, enlaces, imágenes, y muchas otras cosas. Para que pueda ser accedida a ella es necesario mediante un navegador web, además de conexión a Internet.

Presupuesto: necesidades económicas que se consideran necesarias para una campaña, expresadas en euros, y que deben obtenerse mediante donaciones anónimas, de personas o de instituciones.

Pruebas de aceptación: se trata de ensayos realizados para que el cliente verifique que el sistema es válido para él.

Puestos de la organización: cargos en la estructura organizativa de la ONG que son ocupados por voluntarios.

Recurso: medio necesario para llevar a cabo una campaña. Los recursos pueden ser aportaciones económicas (dinero expresado en euros), o bien medios materiales (vehículos, material sanitario, medicamentos, alimentos no perecederos, etc.) que tienen una valoración económica definida, o bien medios materiales que no tienen una valoración económica (ropa usada, calzado, juguetes, etc.).

Reglas de negocios: describe las políticas, normas, operaciones, definiciones y restricciones presentes en una organización y que son de vital importancia para alcanzar los objetivos misionales.

Usuario: persona registrada en la página Web con capacidad para convertirse en voluntario, donante o ambas.

Voluntario: persona que dedica parte de su tiempo al trabajo no remunerado en una ONG.

HISTORIAS DE USUARIO

Gestión de matriculación del voluntario	Gestión de información sobre donantes	Gestión de información sobre las donaciones		Gestión de campañas		
RI-005 Información sobre voluntarios	**RI-004 Información sobre donantes**	**RI-001 Información sobre acciones humanitarias**	**RI-003 Información sobre donaciones**	**RI-002 Información sobre campañas**	**RI-006 Información sobre asignación de voluntarios a puestos**	**RI-007 Información sobre participación de voluntarios en campañas**
–	**–**	**RN-001 Partición de las donaciones**	**RN-003 Cantidad mínima de donación**	**–**	**RN-002 Voluntario antes que director**	
RF-002 Informes sobre voluntarios en campaña	**RF-001 Informes de donaciones por campañas, donantes y tipos de recursos**	**–**	**RF-001 Informes de donaciones por campañas, donantes y tipos de recursos**	**RF-001 Informes de donaciones por campañas, donantes y tipos de recursos**	**RF-003 Informes sobre voluntarios en la organización**	**RF-003 Informes sobre voluntarios en la organización**
RNF-001 - Visualización y funcionamiento correcto **RNF-002 - Cumplimiento de normativa** **RNF-003 - Tiempo de respuesta** **RNF-004 - Disponibilidad** **RNF-005 - Fácil de usar**						

REQUISITOS GENERALES/OBJETIVOS

Gestión de matriculación de voluntario: se necesita una aplicación que permita llevar a cabo la matriculación de un voluntario y la asignación de puestos de la organización correspondiente.

Gestión de información sobre donantes: se necesita una aplicación que permita obtener cierta información sobre los donantes a la ONG, como el nombre de dicho donante, sus apellidos, su DNI, su dirección, su email y su teléfono.

Gestión de información sobre donaciones: se necesita una aplicación que permita obtener las donaciones realizadas por parte de los voluntarios/usuarios, tales como la cantidad de donación realizada, qué tipo de donación, ya sea material o capital, a qué campaña va destinada, y sobre todo, la información del donante.

Gestión de campañas: se necesita una aplicación que permita visualizar tanto las campañas activas como las pasadas: nombre de la campaña, fecha de inicio, fecha de finalización, presupuesto y el personal dedicado a esa campaña.

Gestión del sistema de donaciones: se necesita una aplicación que permita disponer de toda la información sobre los ingresos (donaciones de cualquier tipo) que han sido recaudados, además de aquellos recursos o capitales que se hayan invertido.

REQUISITOS DE INFORMACIÓN

RI-001 Información sobre acciones humanitarias:
Como **voluntario**
Quiero disponer de la información correspondiente a las acciones humanitarias desarrolladas por SOS Ayuda Sin Fronteras: denominación de la acción humanitaria, el país donde se desarrolla, y las campañas que componen la acción
Para poder tener un control sobre las acciones humanitarias

RI-002 Información sobre campañas:
Como voluntario
Quiero disponer de la información correspondiente a las campañas de las acciones humanitarias: la denominación de la campaña, el intervalo de tiempo para su realización, su presupuesto y el voluntario que dirige la campaña.
Para poder tener un control sobre las campañas

RI-003 Información sobre donaciones:
Como voluntario
Quiero disponer de la información correspondiente a las donaciones de recursos efectuadas por instituciones, por particulares o anónimas: la campaña a la que va destinada la donación, el tipo de recurso que se dona, la unidad en que se mide el recurso donado, el valor unitario de la unidad del recurso donado, la cantidad de unidades donada y el donante. Debe tenerse en cuenta que los tipos de recursos son muy diversos y que pueden definirse nuevos tipos para cada campaña.
Para poder llevar una contabilidad de recursos y llevar un control mejor de gastos e ingresos

RI-004 Información sobre donantes:
Como voluntario
Quiero disponer la información correspondiente a los donantes cuando éstos no sean anónimos: nif y nombre
Para poder relacionar los ingresos con las personas o instituciones que realizan las donaciones

RI-005 Información sobre voluntarios:
Como voluntario
Quiero tener disponible la información correspondiente a todos los voluntarios de SOS Ayuda Sin Fronteras: En concreto: nif, nombre, dirección, email y teléfono.
Para poder tener un inventario de las personas que colaboran con la ONG

RI-006 Información sobre asignación de voluntarios a puestos
Como voluntario
Quiero disponer de la información correspondiente a la asignación de los voluntarios a los distintos puestos de SOS Ayuda Sin Fronteras: el puesto ocupado, el voluntario que lo ocupa y el intervalo de tiempo de asignación del voluntario al puesto
Para poder hacer una mejor asignación y control de voluntarios a puestos en la ONG

RI-007 Información sobre participación de voluntarios en campañas
Como voluntario
Quiero disponer de la información correspondiente a la participación de voluntarios en las campañas de SOS Ayuda Sin Fronteras: el voluntario que participa, la campaña en la que participa y el intervalo de tiempo de adscripción del voluntario a la campaña
Para poder hacer nuevas asignaciones de voluntarios a campañas cuando éstas salgan, o bien hacer un catálogo de qué voluntarios colaboran en que campañas

REQUISITOS DE REGLAS DE NEGOCIOS

RN-001 Partición de las donaciones:
Como gerente de la ONG
Quiero que se cumpla la siguiente regla de negocio: todas las donaciones realizadas a la ONG deben ser repartidas equitativamente sobre las diferentes campañas activas, o si existe una campaña principal la cual necesita recursos o capital inmediatamente, todas estas donaciones irán destinadas a la misma.
Para poder tener un control sobre las donaciones realizadas, y que su reparto por las distintas campañas se lleven acabo correctamente.

RN-002 Voluntario antes que director:
Como gerente de la ONG,
Quiero que se cumpla la siguiente regla de negocio: para que un voluntario pueda dirigir una campaña, debe haber sido voluntario previamente en otras campañas.
Para que los directores de campaña conozcan lo que significa ser voluntario cuando tomen decisiones.

RN-003 Cantidad mínima de donación:
Como gerente de la ONG
Quiero que se cumpla la siguiente regla de negocio: La donación mínima para ser voluntario es de 25€ al año, si la donación es menor, el usuario es donante pero no voluntario
Para poder tener un control sobre las donaciones, donantes, y nuevos usuarios/voluntarios.

RN-004 Director de campaña veterano.
Como gerente de la ONG
Quiero que se cumpla la siguiente regla de negocio: un Director de campaña debe de tener privilegios a la hora de publicar noticias en la web, mientras que un Director de campaña veterano puede editarlas. Para ser director de campaña veterano, la persona debe de haber dirigido 2 o más campañas.
Para poder corregir posibles errores sin tener que recurrir al gerente o al administrador web.

RN-005 Exento de donaciones.
Como gerente de la ONG
Quiero que los directores de campaña estén exentos de la donación anual de 25€ y sea algo opcional para ellos.
Para recompensar de alguna forma el trabajo realizado y que puedan seguir aportando.

REQUISITOS FUNCIONALES

RF-001 Informes de donaciones por campañas, donantes y tipos de recursos:
Como usuario
Quiero me gustaría poder obtener informes de las donaciones a cada campaña por diferentes criterios (donantes, tipos de recursos) .
Para poder conocer las campañas cuyo presupuesto no ha sido cubierto.

RF-002 Informes sobre voluntarios en campaña:
Como usuario
Quiero poder obtener informes de voluntarios en cada campaña (nombre, apellidos, DNI, etc.), en una fecha dada y por intervalo de tiempo.
Para poder realizar estudios sobre determinadas campañas y poder tener una mejor organización de todas la campañas activas, además de las pasadas, obteniendo informes por separado de cada una de ellas.

RF-003 Informes sobre voluntarios en la organización:
Como usuario
Quiero poder obtener informes de voluntarios que ocupen puestos en la estructura de la organización, en una fecha determinada y por intervalo de tiempo. Así mismo quisiera poder obtener informes de los diferentes puestos ocupados por una misma persona.
Para poder saber la organización de puestos de trabajo y tener una mejor organización del personal dentro de la ONG.

REQUISITOS NO FUNCIONALES

RNF-001 - Visualización y funcionamiento correcto:
Como usuario,
Quiero que el sistema permita: visualizarse y funcionar correctamente en cualquier navegador,
Para que todo usuario pueda hacer disfrute del mismo sistema.

RNF-002 - Cumplimiento de normativa:
Como usuario,
Quiero que el sistema permita: cumplir las disposiciones recogidas en la Ley Orgánica de Datos Personales y en el Reglamento de medidas de seguridad,
Para que no haya ningún problema legal.

RNF-003 - Tiempo de respuesta:
Como usuario,
Quiero que el sistema permita: no tardar más de cinco segundos en mostrar los resultados de una búsqueda, en el caso de que se supere este plazo, el sistema detiene la búsqueda y muestra los resultados encontrados.
Para acelerar el proceso de interacción entre usuario y sistema.

RNF-004 - Disponibilidad:
Como usuario,
Quiero que el sistema permita: estar disponible las 24 horas del día, intentando que los mantenimientos duren lo menos posible y sean cuando menos interacción de usuarios haya.
Para no retrasar a un usuario que quiera interactuar con el sistema.

RNF-005 - Fácil de usar:
Como usuario,
Quiero que el sistema permita: un fácil y sencillo manejo para cualquier tipo de usuario con o sin conocimientos técnicos.
Para una comodidad de entendimiento del sistema para el usuario.

PRUEBAS DE ACEPTACIÓN

RN-001 PARTICIÓN DE LAS DONACIONES

Como gerente de la ONG
Quiero que se cumpla la siguiente regla de negocio: todas las donaciones realizadas a la ONG deben ser repartidas equitativamente sobre las diferentes campañas activas, o si existe una campaña principal la cual necesita recursos o capital inmediatamente, todas estas donaciones irán destinadas a la misma.
Para poder tener un control sobre las donaciones realizadas, y que su reparto por las distintas campañas se lleven acabo correctamente.

Pruebas de aceptación:

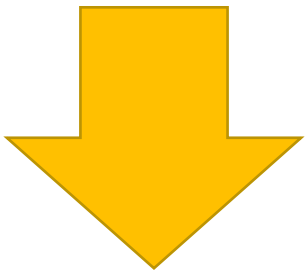
☑Todas las donaciones recibidas son exactamente repartidas entre todas las campañas activas disponibles y no se recibe ningún mensaje de error.

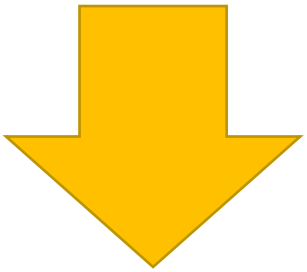
☑Al existir una campaña activa principal, es decir, aquella que necesita recursos y capital inmediatamente, las donaciones realizadas se traspasarán a esta misma directamente.

☒Se realiza una partición no equitativa de los recursos o capitales, por lo que se produce un mensaje de error porque todas las donaciones deben repartirse de manera equitativa.

RN-002 VOLUNTARIO ANTES QUE DIRECTOR

Como gerente de la ONG,
Quiero que se cumpla la siguiente regla de negocio: para que un voluntario pueda dirigir una campaña, debe haber sido voluntario previamente en otras campañas.
Para que los directores de campaña conozcan lo que significa ser voluntario cuando tomen decisiones.





Pruebas de aceptación:

☑El voluntario puede dirigir una campaña y no se recibe ningún mensaje de error, porque previamente este voluntario ha estado implicado en otras campañas como voluntario.

☒Se asigna a un voluntario como director de una campaña, pero éste nunca ha sido voluntario en otras campañas anteriormente, por lo que se produce un mensaje de error.

RN-003 CANTIDAD MÍNIMA DE DONACIÓN

Como gerente de la ONG
Quiero que se cumpla la siguiente regla de negocio: La donación mínima para ser voluntario es de 25€ al año, si la donación es menor, el usuario es donante pero no voluntario
Para poder tener un control sobre las donaciones, donantes, y nuevos usuarios/voluntarios.

Pruebas de aceptación:

☑Todas las donaciones son recibidas sin ningún problema y no se recibe ningún mensaje de error.

☑La donación que se realiza es de 25€,se recibe, por lo que el donante se convierte en voluntario y no se recibe ningún mensaje de error.

☒Las donaciones que se intentan realizar no llegan a ser recibidas por el sistema, por lo que se produce un mensaje de error.

☒Las donaciones que se intentan realizar no son posibles, debido a que sea 0 o una cifra menor a 0.

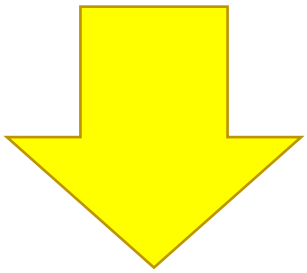
RN-004 - DIRECTOR DE CAMPAÑA VETERANO

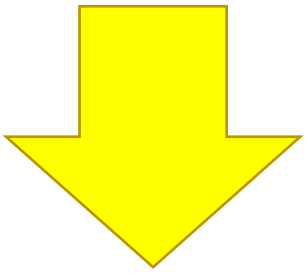
Como gerente de la ONG
Quiero que se cumpla la siguiente regla de negocio: un Director de campaña debe de tener privilegios a la hora de publicar noticias en la web, mientras que un Director de campaña veterano puede editarlas. Para poder ser director de campaña veterano, la persona debe de haber dirigido 2 o más campañas.
Para poder corregir posibles errores sin tener que recurrir al gerente o al administrador web.

Pruebas de aceptación:
☑Tras dirigir 2 campañas el director de campaña pasa automáticamente a ser director de campaña veterano.
☑El director de campaña veterano puede editar en las noticias de los demás.
☒Un director de campaña que ha dirigido 2 o más campañas sigue siendo director de campaña y no veterano.
☒El director de campaña veterano tiene permisos insuficientes para editar noticias

RN-005 -EXENTO DE DONACIONES

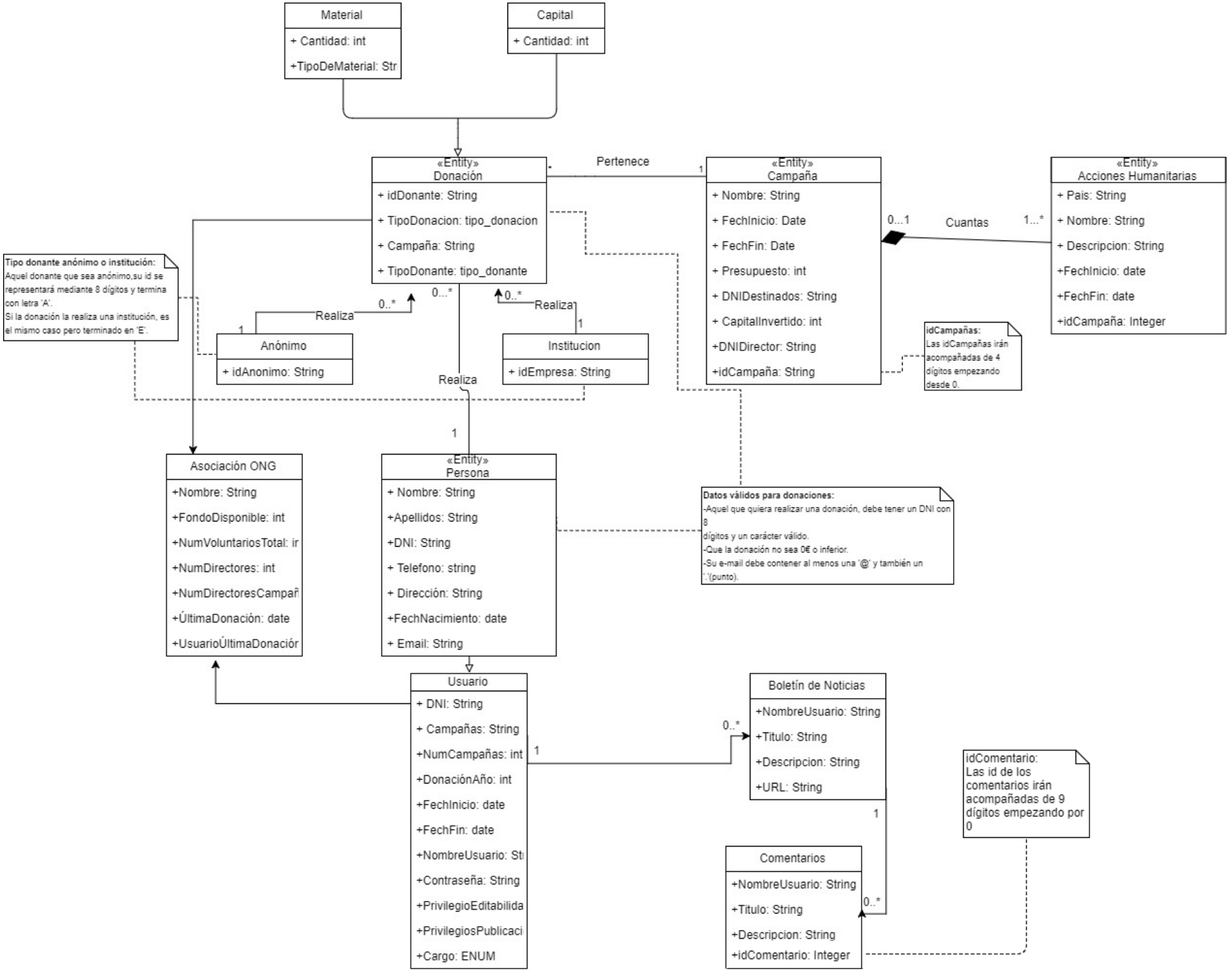
Como gerente de la ONG
Quiero que los directores de campaña estén exentos de la donación anual de 25€ y sea algo opcional para ellos.
Para recompensar de alguna forma el trabajo realizado y que puedan seguir aportando.





Pruebas de aceptación:
☑Un director de campaña o un director de campaña veterano tienen en las donaciones el valor 0 y sigue manteniendo su rango.
☒Un director de campaña o director de campaña veterano tiene en las donaciones el valor 0 y es degradado a usuario.

MODELO CONCEPTUAL



CASO DE PRUEBA 1

En este caso de prueba, realizamos un diagrama UML sobre la regla de negocio la cual todas las donaciones deben repartirse de manera equitativa (RN-001 - Partición de las donaciones). En esta prueba, hemos diseñado un escenario donde se realizan dos donaciones de diferentes usuarios (u1 y u2) y un anónimo(a1), donde podemos ver cuáles son los datos de cada uno de los de los usuarios (DNI, cargo, nombre de usuario, número de campañas a la que pertenece, etc.) relacionado con la donación que realiza (qué tipo de donación, la cantidad, etc.).

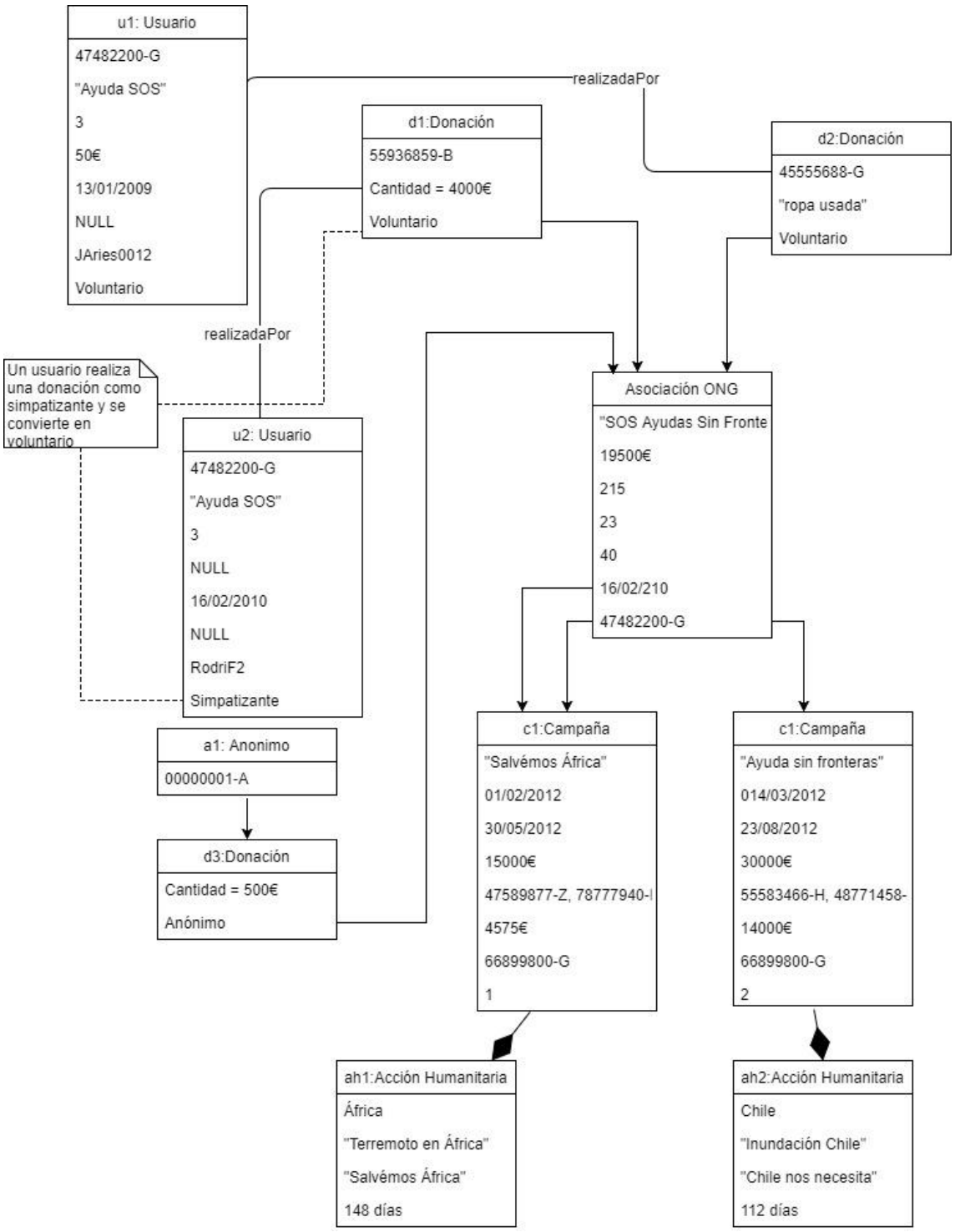
El usuario u1 (voluntario por su matriculación en la ONG y su donación anual) realiza una donación d2. Dicha donación es de tipo material, “ropa usada”, dirigida a la asociación de manera intermediaria antes de ir a la campaña.

El usuario u2(simpatizante, no está matriculado en la ONG) realiza una donación d1. Dicha donación es de tipo monetaria, con una cantidad de 4000€, dirigida a la asociación de manera intermediaria antes de ir a la campaña.

El anónimo a1(no tiene ninguna documentación) realiza una donación d3. Dicha donación es monetaria, con una cantidad de 500€, dirigida a la asociación de manera intermediaria antes de ir a la campaña.

Esas donaciones realizadas, según nuestra primera regla de negocio, deben ser admitidas por la asociación de la ONG, y es ésta la que debe repartir esas donaciones de manera equitativa entre todas las campañas existentes (en el ejemplo, entre “Salvemos África” y “Ayudas sin fronteras”), o en su defecto, irán todas destinadas a una campaña principal. En el caso de que la donación sea realizada por un anónimo o institución, solo aparecerá un id con ciertos dígitos y una letra mayúscula.

Esas donaciones a campañas están relacionadas con ciertas acciones humanitarias, las cuales engloban la catástrofe ocurrida, a la que van destinadas todas las donaciones.

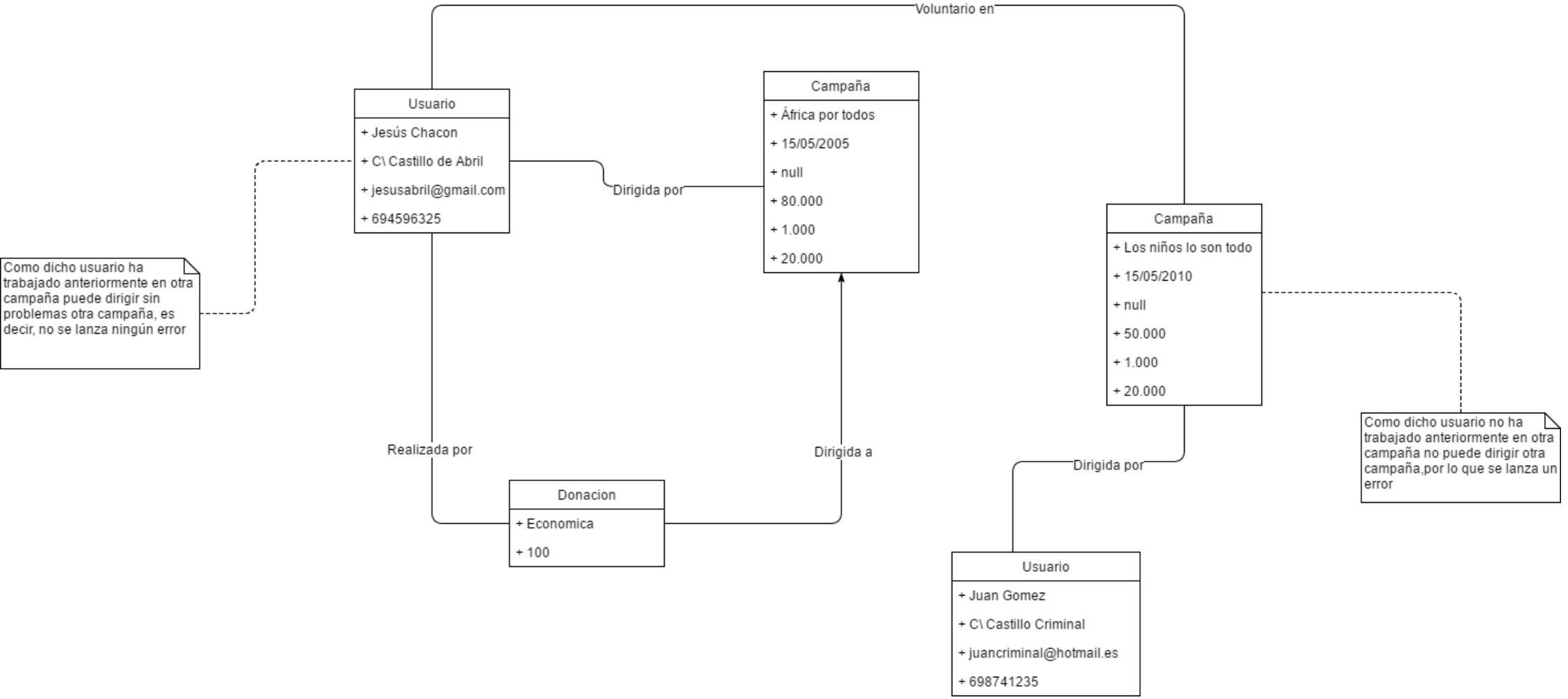

u1: Usuario
47482200-G
"Ayuda SOS"
3
50€
13/01/2009
NULL
JAries0012
Voluntario
realizadaPor
d1:Donación
55936859-B
Cantidad = 4000€
Voluntario
d2:Donación
45555688-G
"ropa usada"
Voluntario
realizadaPor
Un usuario realiza una donación como simpatizante y se convierte en voluntario
u2: Usuario
47482200-G
"Ayuda SOS"
3
NULL
16/02/2010
NULL
RodriF2
Simpatizante
Asociación ONG
"SOS Ayudas Sin Fronte
19500€
215
23
40
16/02/210
47482200-G
a1: Anonimo
00000001-A
d3:Donación
Cantidad = 500€
Anónimo
c1:Campaña
"Salvémos África"
01/02/2012
30/05/2012
15000€
47589877-Z, 78777940-I
4575€
66899800-G
1
c1:Campaña
"Ayuda sin fronteras"
014/03/2012
23/08/2012
30000€
55583466-H, 48771458-
14000€
66899800-G
2
ah1:Acción Humanitaria
África
"Terremoto en África"
"Salvémos África"
148 días
ah2:Acción Humanitaria
Chile
"Inundación Chile"
"Chile nos necesita"
112 días

CASO DE PRUEBA 2

En este caso de prueba, realizamos un diagrama UML sobre la regla de negocio la cual describe que antes de poder ser director de una campaña hay que ser voluntario en varias campañas para poder llegar a tomar el cargo de director de campaña (RN-002 - Voluntario antes que director).Hemos diseñado un escenario donde existen de usuarios u1 y u2, con su respectiva documentación.

El usuario u1, según constata en su documentación, tiene cargo de director de campaña, por lo que éste usuario puede dirigir campañas (en este caso en la campaña c1), ya que ha sido voluntario en 3 de las campañas habilitadas por la Asociación ONG. Además realiza una donación y se envía a la propia Asociación (RN-001). Este director de campaña además es voluntario a su vez en otra campaña c2.

El usuario u2, según constata su documentación, es un usuario voluntario, por lo que no tiene la disponibilidad de dirigir alguna campaña habilitada. Para poder llegar a serlo, debe ser voluntario en más campañas.

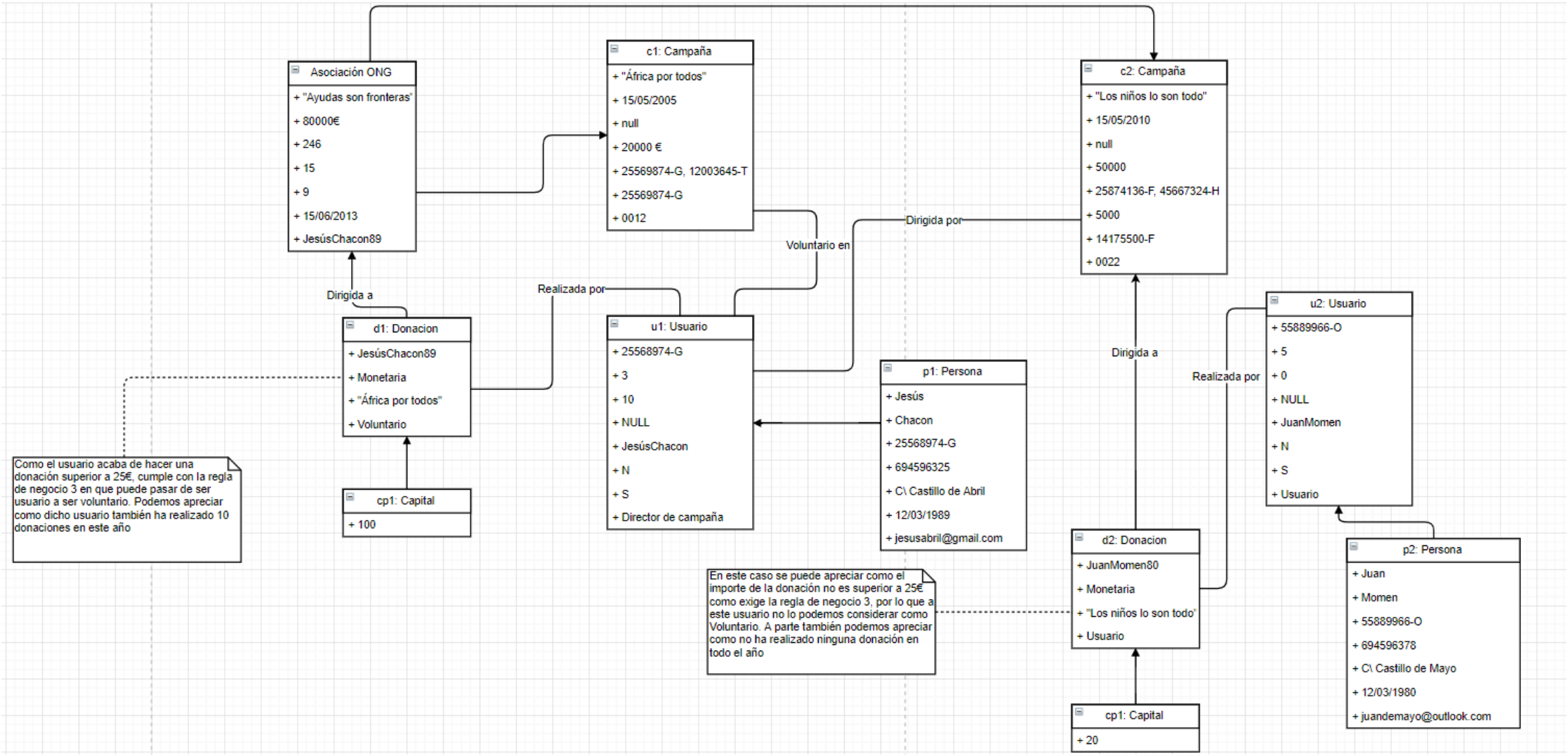

Voluntario en
Usuario
+ Jesús Chacon
+ C\ Castillo de Abril
+ jesusabril@gmail.com
+ 694596325
Dirigida por
Campaña
+ África por todos
+ 15/05/2005
+ null
+ 80.000
+ 1.000
+ 20.000
Como dicho usuario ha trabajado anteriormente en otra campaña puede dirigir sin problemas otra campaña, es decir, no se lanza ningún error
Realizada por
Donacion
+ Economica
+ 100
Dirigida a
Campaña
+ Los niños lo son todo
+ 15/05/2010
+ null
+ 50.000
+ 1.000
+ 20.000
Como dicho usuario no ha trabajado anteriormente en otra campaña no puede dirigir otra campaña,por lo que se lanza un error
Dirigida por
Usuario
+ Juan Gomez
+ C\ Castillo Criminal
+ juancriminal@hotmail.es
+ 698741235

CASO DE PRUEBA 3

El caso de prueba que vamos a describir es un diagrama de UML sobre la regla de negocio en la que se restringe que si un voluntario no ha realizar una donación anual de 25€ pasa de ser voluntario a ser un usuario. (RN-003 Cantidad mínima de donación). En dicha prueba tenemos descritos los usuarios u1 y u2 los cuales uno si cumple la condición y otro no por lo que este último lanzara un error.

El usuario u1 podemos ver como desde que se registro tiene registradas 10 donaciones, de las cuales la última donación es de 100€, por lo tanto dicho usuario que tiene de cargo voluntario seguirá siendo voluntario en dicha campaña a la que está dirigida la donación.

El usuario u2 podemos apreciar cómo a diferencia del u1 solo ha realizado 5 donaciones y de las cuales la última solo tiene un importe monetario de 20€ por lo que se elevara el error RN-003 el cual dice que un voluntario solo puede seguir siendo voluntario si anualmente como mínimo tiene una donación de 25€, es decir u2 pasara de ser voluntario a ser usuario.


Asociación ONG
+ "Ayudas son fronteras"
+ 80000€
+ 246
+ 15
+ 9
+ 15/06/2013
+ JesúsChacon89
c1: Campaña
+ "África por todos"
+ 15/05/2005
+ null
+ 20000 €
+ 25569874-G, 12003645-T
+ 25569874-G
+ 0012
c2: Campaña
+ "Los niños lo son todo"
+ 15/05/2010
+ null
+ 50000
+ 25874136-F, 45667324-H
+ 5000
+ 14175500-F
+ 0022
Dirigida por
Voluntario en
Dirigida a
Realizada por
d1: Donacion
+ JesúsChacon89
+ Monetaria
+ "África por todos"
+ Voluntario
u1: Usuario
+ 25568974-G
+ 3
+ 10
+ NULL
+ JesúsChacon
+ N
+ S
+ Director de campaña
p1: Persona
+ Jesús
+ Chacon
+ 25568974-G
+ 694596325
+ C\ Castillo de Abril
+ 12/03/1989
+ jesusabril@gmail.com
u2: Usuario
+ 55889966-O
+ 5
+ 0
+ NULL
+ JuanMomen
+ N
+ S
+ Usuario
Dirigida a
Realizada por
Como el usuario acaba de hacer una donación superior a 25€, cumple con la regla de negocio 3 en que puede pasar de ser usuario a ser voluntario. Podemos apreciar como dicho usuario también ha realizado 10 donaciones en este año
cp1: Capital
+ 100
En este caso se puede apreciar como el importe de la donación no es superior a 25€ como exige la regla de negocio 3, por lo que a este usuario no lo podemos considerar como Voluntario. A parte también podemos apreciar como no ha realizado ninguna donación en todo el año
d2: Donacion
+ JuanMomen80
+ Monetaria
+ "Los niños lo son todo"
+ Usuario
p2: Persona
+ Juan
+ Momen
+ 55889966-O
+ 694596378
+ C\ Castillo de Mayo
+ 12/03/1980
+ juandemayo@outlook.com
cp1: Capital
+ 20

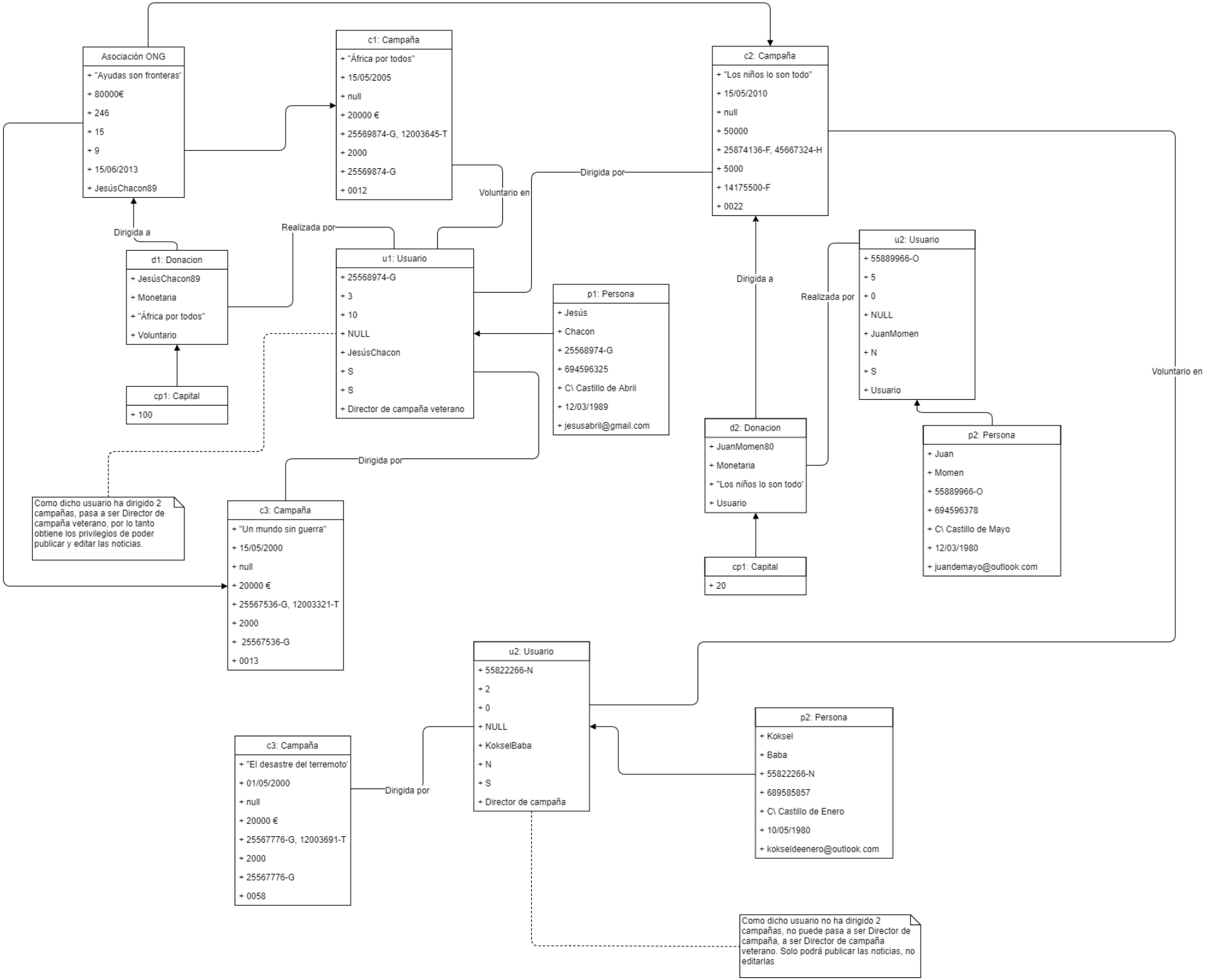
CASO DE PRUEBA 4

En este caso de prueba, realizamos un diagrama UML sobre la regla de negocio la cual describe que un director de campaña debe tener privilegios a la hora de publicar noticias en la web, mientras que un director de campaña veterano puede editarlas, para ello, este último debe haber dirigido dos o más campañas(RN-004 - Director de campaña veterano). Hemos diseñado un escenario con tres usuarios diferentes, u1, u2 y u3.

El usuario u1 tiene relacionada a su persona dos campañas, es decir, en ese momento está dirigiendo dos campañas, c2 y c3. También es voluntaria en otra, c1.Como este usuario dirige dos o más campañas, es considerado como un director de campaña veterano, lo que tiene el privilegio de editabilidad.

El usuario u2 no tiene ninguna campaña asociada a su persona, es decir, no dirige ninguna campaña, únicamente realiza una donación, es decir, es solo un usuario voluntario.

El usuario u3 tiene relacionada una sola campaña c4 a su persona, es decir, en ese momento está dirigiendo una sola campaña, por lo que solo es considerado director de campaña, pero no veterano. Ser director de campaña le da privilegio de publicación, pero no de editabilidad.


Asociación ONG
+ "Ayudas son fronteras"
+ 80000€
+ 246
+ 15
+ 9
+ 15/06/2013
+ JesúsChacon89
c1: Campaña
+ "África por todos"
+ 15/05/2005
+ null
+ 20000 €
+ 25569874-G, 12003645-T
+ 2000
+ 25569874-G
+ 0012
c2: Campaña
+ "Los niños lo son todo"
+ 15/05/2010
+ null
+ 50000
+ 25874136-F, 45667324-H
+ 5000
+ 14175500-F
+ 0022
Dirigida a
Realizada por
Voluntario en
Dirigida por
d1: Donacion
+ JesúsChacon89
+ Monetaria
+ "África por todos"
+ Voluntario
u1: Usuario
+ 25568974-G
+ 3
+ 10
+ NULL
+ JesúsChacon
+ S
+ S
+ Director de campaña veterano
p1: Persona
+ Jesús
+ Chacon
+ 25568974-G
+ 694596325
+ C\ Castillo de Abril
+ 12/03/1989
+ jesusabril@gmail.com
cp1: Capital
+ 100
Como dicho usuario ha dirigido 2 campañas, pasa a ser Director de campaña veterano, por lo tanto obtiene los privilegios de poder publicar y editar las noticias.
c3: Campaña
+ "Un mundo sin guerra"
+ 15/05/2000
+ null
+ 20000 €
+ 25567536-G, 12003321-T
+ 2000
+ 25567536-G
+ 0013
Dirigida a
Realizada por
u2: Usuario
+ 55889966-O
+ 5
+ 0
+ NULL
+ JuanMomen
+ N
+ S
+ Usuario
d2: Donacion
+ JuanMomen80
+ Monetaria
+ "Los niños lo son todo"
+ Usuario
p2: Persona
+ Juan
+ Momen
+ 55889966-O
+ 694596378
+ C\ Castillo de Mayo
+ 12/03/1980
+ juandemayo@outlook.com
cp1: Capital
+ 20
Voluntario en
u2: Usuario
+ 55822266-N
+ 2
+ 0
+ NULL
+ KokselBaba
+ N
+ S
+ Director de campaña
p2: Persona
+ Koksel
+ Baba
+ 55822266-N
+ 689585857
+ C\ Castillo de Enero
+ 10/05/1980
+ kokseldeenero@outlook.com
c3: Campaña
+ "El desastre del terremoto"
+ 01/05/2000
+ null
+ 20000 €
+ 25567776-G, 12003691-T
+ 2000
+ 25567776-G
+ 0058
Dirigida por
Como dicho usuario no ha dirigido 2 campañas, no puede pasa a ser Director de campaña, a ser Director de campaña veterano. Solo podrá publicar las noticias, no editarlas

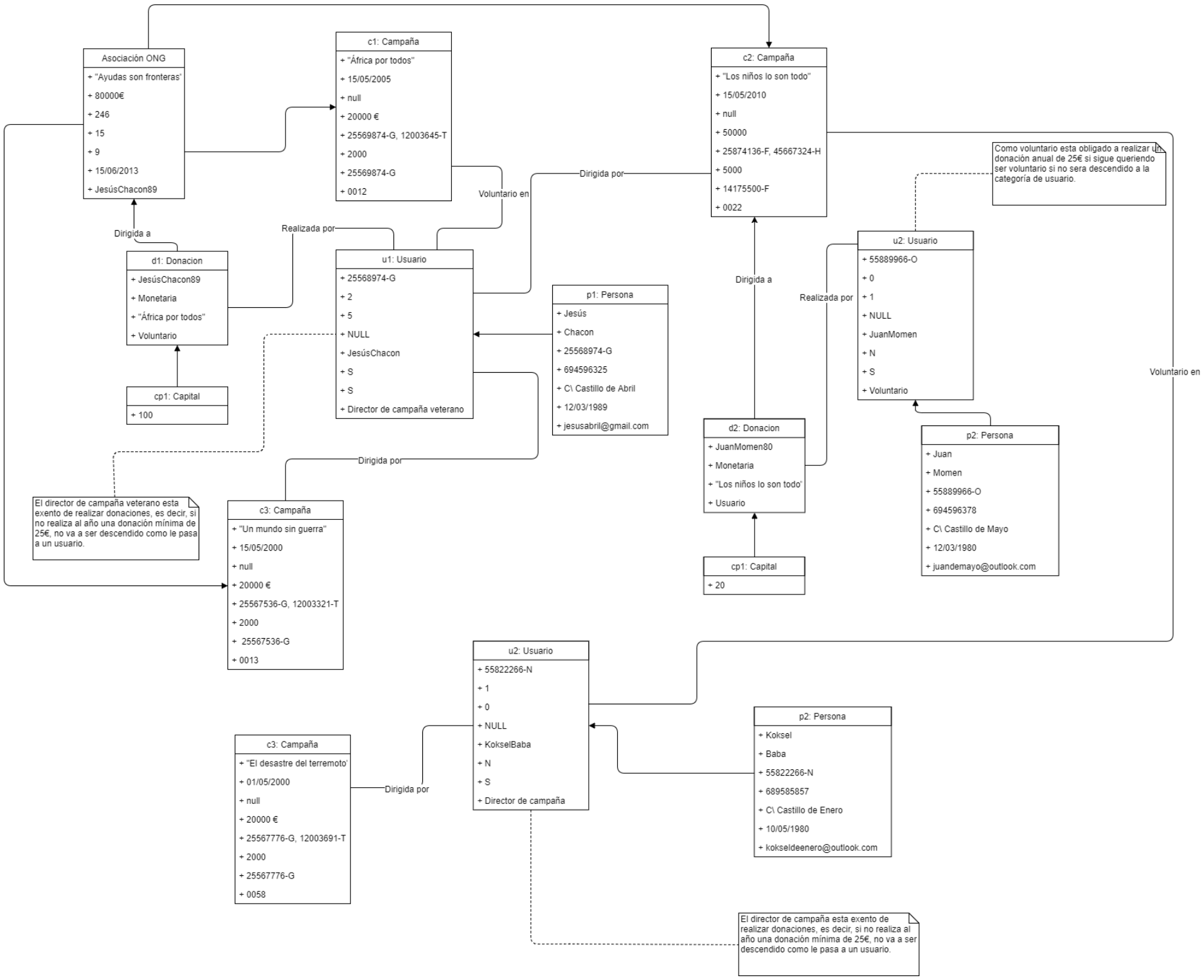
CASO DE PRUEBA 5

El caso de prueba que vamos a describir es un diagrama de UML sobre la regla de negocio en la que se restringe que si un director de campaña no ha realizar una donación anual de 25€ no pasa a ser usuario. (RN-005 -Exento de donaciones). En dicha prueba tenemos descritos los usuario u1 y u2 los cuales son directores de campaña uno siendo veterano y el otro no, luego tenemos u3 que es un voluntario el cual si que tiene la restricción de los 25€.

u1 es un director de campaña veterano al cual no tiene porque tener al final de año 25€ en donaciones para seguir manteniendo su cargo en la ONG.

u2 es director de campaña el cual aun no siendo veterano con respecto a esta restricción, tiene los mismos derecho que un director que no sea veterano, es decir que no tiene porque tener a final de año la donación de 25€ para seguir manteniendo su cargo en la ONG.

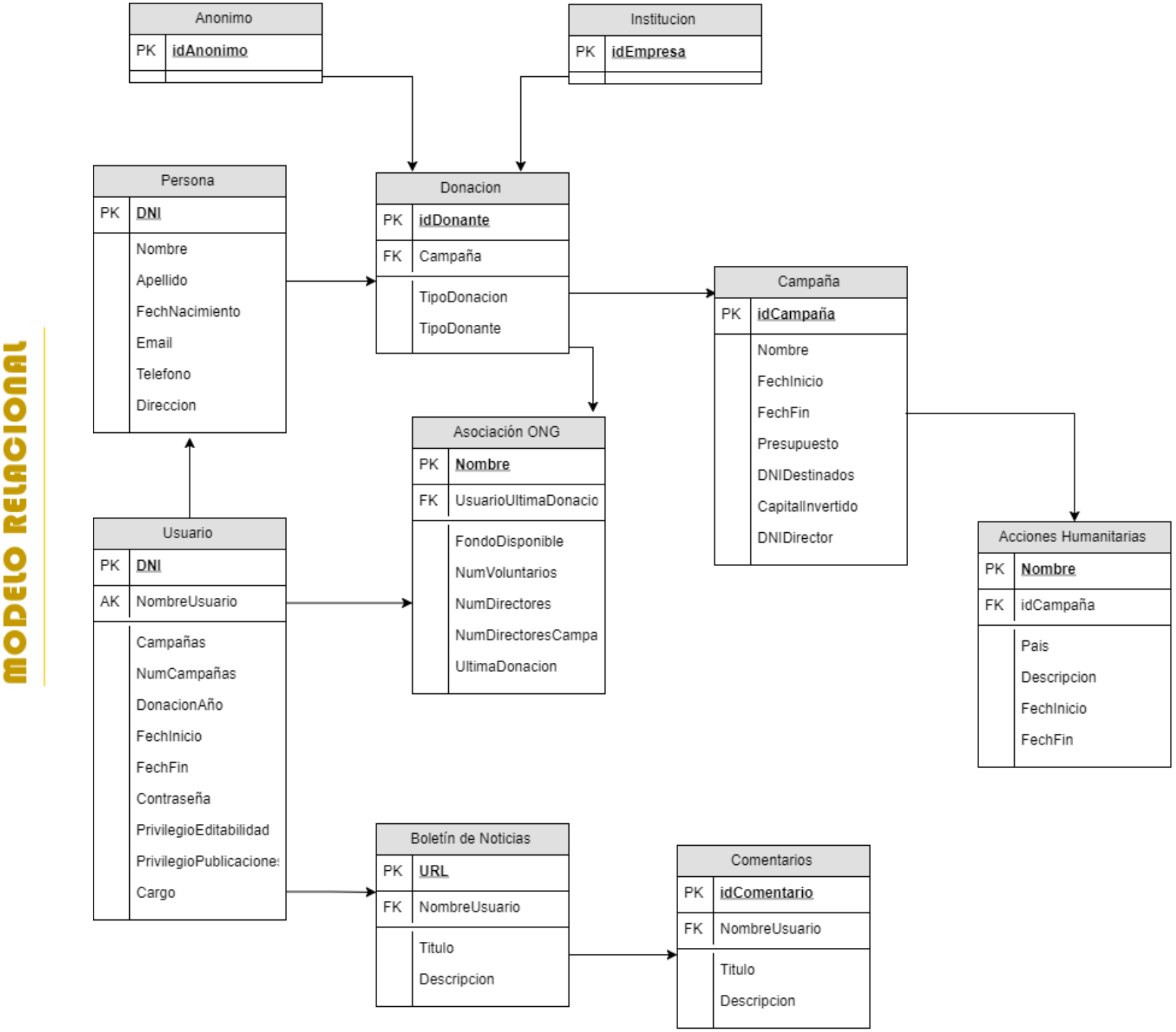
u3 sin embargo con su cargo en la ONG es voluntario si esta sujeto a la condición anterior, es decir, necesita tener al final de año una donación mínima de 25€ para seguir manteniendo su cargo de voluntario.


Asociación ONG
+ "Ayudas son fronteras"
+ 80000€
+ 246
+ 15
+ 9
+ 15/06/2013
+ JesúsChacon89
c1: Campaña
+ "África por todos"
+ 15/05/2005
+ null
+ 20000 €
+ 25569874-G, 12003645-T
+ 2000
+ 25569874-G
+ 0012
c2: Campaña
+ "Los niños lo son todo"
+ 15/05/2010
+ null
+ 50000
+ 25874136-F, 45667324-H
+ 5000
+ 14175500-F
+ 0022
Como voluntario esta obligado a realizar una donación anual de 25€ si sigue queriendo ser voluntario si no sera descendido a la categoría de usuario.
Dirigida a
Realizada por
Voluntario en
Dirigida por
d1: Donacion
+ JesúsChacon89
+ Monetaria
+ "África por todos"
+ Voluntario
u1: Usuario
+ 25568974-G
+ 2
+ 5
+ NULL
+ JesúsChacon
+ S
+ S
+ Director de campaña veterano
p1: Persona
+ Jesús
+ Chacon
+ 25568974-G
+ 694596325
+ C\ Castillo de Abril
+ 12/03/1989
+ jesusabril@gmail.com
u2: Usuario
+ 55889966-O
+ 0
+ 1
+ NULL
+ JuanMomen
+ N
+ S
+ Voluntario
p2: Persona
+ Juan
+ Momen
+ 55889966-O
+ 694596378
+ C\ Castillo de Mayo
+ 12/03/1980
+ juandemayo@outlook.com
cp1: Capital
+ 100
El director de campaña veterano esta exento de realizar donaciones, es decir, si no realiza al año una donación mínima de 25€, no va a ser descendido como le pasa a un usuario.
c3: Campaña
+ "Un mundo sin guerra"
+ 15/05/2000
+ null
+ 20000 €
+ 25567536-G, 12003321-T
+ 2000
+ 25567536-G
+ 0013
d2: Donacion
+ JuanMomen80
+ Monetaria
+ "Los niños lo son todo"
+ Usuario
cp1: Capital
+ 20
u2: Usuario
+ 55822266-N
+ 1
+ 0
+ NULL
+ KokselBaba
+ N
+ S
+ Director de campaña
p2: Persona
+ Koksel
+ Baba
+ 55822266-N
+ 689585857
+ C\ Castillo de Enero
+ 10/05/1980
+ kokseldeenero@outlook.com
c3: Campaña
+ "El desastre del terremoto"
+ 01/05/2000
+ null
+ 20000 €
+ 25567776-G, 12003691-T
+ 2000
+ 25567776-G
+ 0058
El director de campaña esta exento de realizar donaciones, es decir, si no realiza al año una donación mínima de 25€, no va a ser descendido como le pasa a un usuario.

MATRIZ DE TRAZABILIDAD

	RI-001	*RI-002*	*RI-003*	*RI-004*	*RI-005*	*RI-006*	*RI-007*	*RF-001*	*RF-002*	*RF-003*
Donación			X	x				x		
Campaña		x	X				x	x	x	x
Acc.humanitaria	x						x			x
Persona				x	x	x				x
Institución			x	x						x
Voluntario	x	x	x	x	x	x	x		x	x
Usuario								x	x	x
Material			x				x	x		
Capital			x				x	x		

MODELO RELACIONAL



JUSTIFICACION ESTRATEGIAS DE MODIFICACION JERARQUIAS

TRANSFORMACIÓN DE ENTIDADES:

Al realizar la transformación del modelo conceptual al modelo relacional, para cada entidad se ha trasformado en una relación cuyo nombre ha pasado a estar en plural, con sus correspondientes atributos necesarios.

En aquellas relaciones donde hubiese atributos que sean claves semánticas sencillas, se han usado para definir las claves primarias.

En aquellas relaciones donde hubiese atributos que sean claves semánticas y no se hayan definido como claves primarias, se han definidos claves alternativas.

Para el caso de los enumerados, se creará una restricción asociado al atributo de su correspondiente tabla.

TRANSFORMACIÓN DE ASOCIACIONES:

En el caso de las asociaciones con cardinalidad 1:N, se ha creado un atributo en la tabla con participación [N] que será una clave ajena apuntando al atributo con clave primaria de la tabla con participación [1].

En el caso de las asociaciones con cardinalidad 1:1, se ha creado un atributo en una de las tablas que será una clave ajena apuntando al atributo con clave primaria de la otra tabla.

En el caso de las asociaciones con cardinalidad N:M, se ha creado una tabla auxiliar que relacione ambas entidades, la cual tendrán atributos que serán una clave primaria compuesta y ajenas que apuntan al atributo con clave primaria de cada tabla de cada lado de la asociación.

TRANSFORMACIÓN DE CLASIFICACIONES:

El modelo conceptual cuenta con entidades que son heredadas de otras entidades jerárquicamente superiores:

- En el caso de Donacion, que hereda a Anonimo,Institucion y Persona se trata de una clasificación disjunta y completa, luego la estrategia más viable es crear una tabla Persona, otra tabla Anonimo y otra Institucion.

NORMALIZACIÓN DEL MODELO RELACIONAL:

Cumpliendo todas las transformaciones anteriores se obtiene el modelo relacional en 3FN. La justificación es:

- Está en 1FN porque los atributos son monovaluados.
- Está en 2FN porque está en 1FN y todos los atributos que no forman parte de ninguna clave candidata son completamente dependientes de las claves candidatas de la relación.
- Está en 3FN porque está en 2FN y no existen dependencias transitivas.

MODELO TECNÓLOGICO

```
--Creacion Usuario
CREATE USER ProyectoIISSI IDENTIFIED BY 1234;
GRANT "CONNECT" TO ProyectoIISSI;
GRANT "RESOURCE" TO ProyectoIISSI;
--Borrado de tablas con restricciones
DROP TABLE Donacion CASCADE CONSTRAINTS;
DROP TABLE Material CASCADE CONSTRAINTS;
DROP TABLE Capital CASCADE CONSTRAINTS;
DROP TABLE Campanya CASCADE CONSTRAINTS;
DROP TABLE AccionesHumanitarias CASCADE CONSTRAINTS;
DROP TABLE Noticias CASCADE CONSTRAINTS;
DROP TABLE Comentarios CASCADE CONSTRAINTS;
DROP TABLE Usuario CASCADE CONSTRAINTS;
DROP TABLE AsociacionONG CASCADE CONSTRAINTS;
DROP TABLE Persona CASCADE CONSTRAINTS;
DROP TABLE Institucion CASCADE CONSTRAINTS;
DROP TABLE Anonimo CASCADE CONSTRAINTS;
--Borrado de Secuencias
DROP SEQUENCE SEC_idCampanya;
DROP SEQUENCE SEC_idComentario;
DROP SEQUENCE SEC_idAnonimo;
DROP SEQUENCE SEC_idEmpresa;
```

```
--Creación de tablas
CREATE TABLE Persona(
DNI CHAR(9),
Nombre VARCHAR2(25),
Apellidos VARCHAR2(30),
FechNacimiento DATE,
Email VARCHAR2(40),
Telefono CHAR(9),
Direccion VARCHAR2(50)
);
CREATE TABLE Donacion(
idDonante CHAR(9),
TipoDonacion VARCHAR2(10),
Campanya VARCHAR2(40),
TipoDonante VARCHAR2(15)
);
CREATE TABLE Campanya(
Nombre VARCHAR2(40),
FechInicio DATE,
FechFin DATE,
Presupuesto INT,
DNIDestinados CHAR(9),
DNIDirector CHAR(9),
CapitalInvertido INT
);
CREATE TABLE Material(
Cantidad INT(3),
TipoDeMaterial VARCHAR2(20)
);
CREATE TABLE Capital(
Cantidad INT(10));
CREATE TABLE Usuario(
NombreUsuario VARCHAR2(15),
Contrasenya VARCHAR2(20),
DNI CHAR(9),
Campanyas VARCHAR2(2000),
NumCampanyas INT(5),
DonacionAnyo INT(5),
FechInicio DATE,
FechFin DATE,
PrivilegioEdit CHAR(1),
PrivilegioPub CHAR(1),
Cargo VARCHAR2(17)
);
CREATE TABLE Noticias(
NombreUsuario VARCHAR2(15),
Titulo VARCHAR2(100),
Descripcion VARCHAR2(10000),
Url VARCHAR2(100)
);
CREATE TABLE Comentario(
NombreUsuario VARCHAR2(5),
Titulo VARCHAR2(100),
Descripcion VARCHAR2(10000),
idComentario INT(9)
);
CREATE TABLE AccionesHumanitarias(
Pais VARCHAR2(15),
```

```
IdCampanya INTEGER,
Nombre VARCHAR2(15),
Descripcion VARCHAR2(200),
FechInicio DATE,
FechFin DATE
);
CREATE TABLE Anonimo(
idAnonimo CHAR(9)
);
CREATE TABLE Institucion(
IdEmpresa CHAR(9)
);
CREATE TABLE AsociacionONG(
Nombre VARCHAR2(40),
FondoDisponible INT(6),
NumVoluntarios INT(5),
NumDirectores INT(3),
NumDirectoresVet INT(3),
UltimaDonacion DATE,
UsuarioUltimaDonacion VARCHAR2(15)
);
```

```
--Restricciones NOT NULL
ALTER TABLE Material MODIFY (Cantidad NOT NULL);
ALTER TABLE Material MODIFY (TipoDeMaterial NOT NULL);
ALTER TABLE Capital MODIFY (Cantidad NOT NULL);
ALTER TABLE Donacion MODIFY (idDonante NOT NULL);
ALTER TABLE Donacion MODIFY (TipoDonante NOT NULL);
ALTER TABLE Donacion MODIFY (TipoDonacion NOT NULL);
ALTER TABLE Institucion MODIFY (idEmpresa NOT NULL);
ALTER TABLE Anonimo MODIFY (idAnonimo NOT NULL);
ALTER TABLE AsociacionONG MODIFY (Nombre NOT NULL);
ALTER TABLE AsociacionONG MODIFY (FondoDisponible NOT NULL);
ALTER TABLE AsociacionONG MODIFY (NumVoluntariosTotal NOT NULL);
ALTER TABLE AsociacionONG MODIFY (NumDirectores NOT NULL);
ALTER TABLE AsociacionONG MODIFY (NumDirectoresVet NOT NULL);
ALTER TABLE AsociacionONG MODIFY (UltimaDonacion NOT NULL);
ALTER TABLE AsociacionONG MODIFY (UsuarioUltimaDonacion NOT NULL);
ALTER TABLE Persona MODIFY (Nombre NOT NULL);
ALTER TABLE Persona MODIFY (Apellidos NOT NULL);
ALTER TABLE Persona MODIFY (DNI NOT NULL);
ALTER TABLE Persona MODIFY (Telefono NOT NULL);
ALTER TABLE Persona MODIFY (Direccion NOT NULL);
ALTER TABLE Persona MODIFY (FechNacimiento NOT NULL);
ALTER TABLE Persona MODIFY (Email NOT NULL);
ALTER TABLE Usuario MODIFY (DNI NOT NULL);
ALTER TABLE Usuario MODIFY (NumCampañas NOT NULL);
ALTER TABLE Usuario MODIFY (DonacionAnyo NOT NULL);
ALTER TABLE Usuario MODIFY (FechInicio NOT NULL);
ALTER TABLE Usuario MODIFY (NombreUsuario NOT NULL);
ALTER TABLE Usuario MODIFY (Contrasenya NOT NULL);
ALTER TABLE Usuario MODIFY (PrivilegioEdit NOT NULL);
ALTER TABLE Usuario MODIFY (PrivilegioPub NOT NULL);
ALTER TABLE Usuario MODIFY (Cargo NOT NULL);
ALTER TABLE Noticias MODIFY (NombreUsuario NOT NULL);
ALTER TABLE Noticias MODIFY (Titulo NOT NULL);
ALTER TABLE Noticias MODIFY (Descripcion NOT NULL);
ALTER TABLE Noticias MODIFY (URL NOT NULL);
ALTER TABLE Comentarios MODIFY (NombreUsuario NOT NULL);
ALTER TABLE Comentarios MODIFY (Titulo NOT NULL);
ALTER TABLE Comentarios MODIFY (Descripcion NOT NULL);
ALTER TABLE Comentarios MODIFY (idComentario NOT NULL);
ALTER TABLE Campanya MODIFY (Nombre NOT NULL);
ALTER TABLE Campanya MODIFY (FechInicio NOT NULL);
ALTER TABLE Campanya MODIFY (Presupuesto NOT NULL);
ALTER TABLE Campanya MODIFY (CapitalInvertido NOT NULL);
ALTER TABLE Campanya MODIFY (idCampanya NOT NULL);
ALTER TABLE AccionesHumanitarias MODIFY (Nombre NOT NULL);
ALTER TABLE AccionesHumanitarias MODIFY (Descripcion NOT NULL);
ALTER TABLE AccionesHumanitarias MODIFY (FechInicio NOT NULL);
ALTER TABLE AccionesHumanitarias MODIFY (idCampanya NOT NULL);
```

```
--Primary Keys
ALTER TABLE Donacion ADD CONSTRAINT PK_Donacion PRIMARY KEY
(idDonante);
ALTER TABLE Anonimo ADD CONSTRAINT PK_Anonimo PRIMARY KEY (idAnonimo);
ALTER TABLE Institucion ADD CONSTRAINT PK_Institucion PRIMARY KEY
(idEmpresa);
ALTER TABLE AsociacionONG ADD CONSTRAINT PK_AsociacionONG PRIMARY KEY
(Nombre);
ALTER TABLE Persona ADD CONSTRAINT PK_Persona PRIMARY KEY (DNI);
ALTER TABLE Usuario ADD CONSTRAINT PK_Usuario PRIMARY KEY
(DNI,NombreUsuario);
ALTER TABLE Noticias ADD CONSTRAINT PK_Noticias PRIMARY KEY (URL);
ALTER TABLE Comentarios ADD CONSTRAINT PK_Comentarios PRIMARY KEY
(idComentario);
ALTER TABLE AccionesHumanitarias ADD CONSTRAINT
PK_AccionesHumanitarias PRIMARY KEY (Nombre);
--Foreign Keys
ALTER TABLE Anonimo ADD CONSTRAINT FK_Anonimo FOREIGN KEY (idAnonimo)
REFERENCES Donacion (idDonante) ON DELETE CASCADE;
ALTER TABLE Institucion ADD CONSTRAINT FK_Institucion FOREIGN KEY
(idEmpresa) REFERENCES Donacion (idDonante) ON DELETE CASCADE;
ALTER TABLE Persona ADD CONSTRAINT FK_Persona FOREIGN KEY (DNI)
REFERENCES Donacion (idDonante) ON DELETE CASCADE;
ALTER TABLE Usuario ADD CONSTRAINT FK_Usuario FOREIGN KEY (DNI)
REFERENCES Persona (DNI) ON DELETE CASCADE;
ALTER TABLE Noticias ADD CONSTRAINT FK_Noticias FOREIGN KEY
(NombreUsuario) REFERENCES Usuario (NombreUsuario) ON DELETE CASCADE;
ALTER TABLE Comentarios ADD CONSTRAINT FK_Comentarios FOREIGN KEY
(NombreUsuario) REFERENCES Usuario (NombreUsuario) ON DELETE CASCADE;
ALTER TABLE AccionesHumanitarias ADD CONSTRAINT
FK_AccionesHumanitarias FOREIGN KEY (idCampanya) REFERENCES Campanya
(idCampanya) ON DELETE CASCADE;
ALTER TABLE Donacion ADD CONSTRAINT FK_Donacion FOREIGN KEY (Campanya)
REFERENCES Campanya (idCampanya) ON DELETE CASCADE;
```

```
--Restricciones
ALTER TABLE Donacion ADD CONSTRAINT CK_idDonante CHECK
(REGEXP_LIKE(idDonante, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-
Z]'));
ALTER TABLE Persona ADD CONSTRAINT CK_DNI CHECK (REGEXP_LIKE(DNI, '[0-
9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Anonimo ADD CONSTRAINT CK_idAnonimo CHECK
(REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A]'));
ALTER TABLE Institucion ADD CONSTRAINT CK_idEmpresa CHECK
(REGEXP_LIKE(idEmpresa, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9][E]'));
ALTER TABLE Usuario ADD CONSTRAINT CK_DNI CHECK (REGEXP_LIKE(DNI, '[0-
9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Usuario ADD CONSTRAINT CK_Cargo CHECK (Cargo
IN('Simpatizante','Voluntario','Director','Director
Veterano','Administrador'));
ALTER TABLE Persona ADD CONSTRAINT CK_Telefono CHECK
(REGEXP_LIKE(Telefono, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9]'));
ALTER TABLE Usuario ADD CONSTRAINT CK_DonacionAnyo CHECK
(DonacionAnyo>0);
ALTER TABLE Campanya ADD CONSTRAINT CK_Presupuesto CHECK
(Presupuesto>=0);
ALTER TABLE Campanya ADD CONSTRAINT CK_CapitalInvertido CHECK
(CapitalInvertido>=0);

--Secuencias
CREATE SEQUENCE SEC_idCampanya
INCREMENT BY 1
START WITH 0
MAXVALUE 9999;
CREATE SEQUENCE SEC_idComentario
INCREMENT BY 1
START WITH 0
MAXVALUE 999999999;
CREATE SEQUENCE SEC_idAnonimo
INCREMENT BY 1
START WITH 0A
MAXVALUE 99999999A;
CREATE SEQUENCE SEC_idEmpresa
INCREMENT BY 1
START WITH 0E
MAXVALUE 99999999E;
```

```
--Triggers asociados a secuencias
CREATE OR REPLACE TRIGGER TR_SEC_idCampanya
BEFORE INSERT ON Campanya
FOR EACH ROW
DECLARE
valorSec NUMBER := 0;
BEGIN
SELECT SEC_idCampanya.NEXTVAL INTO valorSec FROM DUAL;
:NEW.OID_C := valorSec;
END;
/
CREATE OR REPLACE TRIGGER TR_SEC_idAnonimo
BEFORE INSERT ON Anonimo
FOR EACH ROW
DECLARE
valorSec NUMBER := 0;
BEGIN
SELECT SEC_idAnonimo.NEXTVAL INTO valorSec FROM DUAL;
:NEW.OID_A := valorSec;
END;
/
CREATE OR REPLACE TRIGGER TR_SEC_idEmpresa
BEFORE INSERT ON Institucion
FOR EACH ROW
DECLARE
valorSec NUMBER := 0;
BEGIN
SELECT SEC_idEmpresa.NEXTVAL INTO valorSec FROM DUAL;
:NEW.OID_E := valorSec;
END;
/
CREATE OR REPLACE TRIGGER TR_SEC_idComentario
BEFORE INSERT ON Comentario
FOR EACH ROW
DECLARE
valorSec NUMBER := 0;
BEGIN
SELECT SEC_idComentario.NEXTVAL INTO valorSec FROM DUAL;
:NEW.OID_C := valorSec;
END;
/
```

```
--Procedimientos y funciones asociadas a las reglas funcionales
--RF-001

CREATE OR REPLACE PROCEDURE PR_informe_donacion(v_campanya IN
Campanya.Nombre%TYPE) IS
    BEGIN
        SELECT (idDonante, TipoDonacion) FROM Donacion WHERE
Campanya=v_campanya;
    END;

--RF-002

CREATE OR REPLACE PROCEDURE PR_informe_voluntarios_campanya(v_fecha IN
Usuario.FechInicio%TYPE)IS
    BEGIN
        SELECT (*) FROM Usuario WHERE Usuario.Cargo=voluntario AND
Usuario.FechInicio=v_fecha;
    END;

--RF-003

CREATE OR REPLACE PROCEDURE
PR_informe_voluntarios_organizacion(v_fecha IN
Usuario.FechInicio%TYPE) IS
    BEGIN
        SELECT(Campanya,Cargo) FROM Usuario WHERE Usuario.FechInicio=
v_fecha;
    END;
```

```
--Triggers asociados a las reglas de negocio
--RN-002

CREATE OR REPLACE TRIGGER TR_voluntario_antes_director
AFTER INSERT OR UPDATE ON Usuario
FOR EACH ROW
DECLARE
        v_NumCampanyas NUMBER := :OLD.NumCampanyas;
        v_Cargo VARCHAR2(20) := :NEW.Cargo;
BEGIN
    IF v_NumCampanyas <2 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Para ser director, se debe
haber participado en 2 o más campañas.');
    END IF;
END;
/
--RN-003
CREATE OR REPLACE TRIGGER TR_cantidad_minima_donacion
AFTER INSERT OR UPDATE OF Cantidad ON Donacion
FOR EACH ROW
DECLARE
        v_Cantidad NUMBER := :NEW.Cantidad;
BEGIN
    IF v_Cantidad <=0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'La cantidad mínima de
donación debe ser un número positivo distinto de 0.');
    END IF;
END;
/

--RN-005
CREATE OR REPLACE TRIGGER TR_cantidad_minima_donacion_voluntario
AFTER INSERT OR UPDATE OF DonacionAnyo ON Usuario
FOR EACH ROW
DECLARE
        v_Usuario VARCHAR2 (15):= :NEW.Usuario;
        v_DonacionAnyo NUMBER := :NEW.DonacionAnyo;
BEGIN
    SELECT (*) INTO v_Usuario FROM Usuario WHERE Usuario = v_Usuario;
    IF DonacionAnyo<25 THEN
        RAISE_APPLICATION_ERROR(-2003, 'La donación para ser
voluntario debe superar los 25€ anuales.');
    END IF;
END;
/
```

```
--Función Auxiliar
CREATE OR REPLACE FUNCTION ASSERT_EQUALS (v_Salida BOOLEAN,
salidaEsperada BOOLEAN)
RETURN VARCHAR2
IS
BEGIN
    IF v_Salida = salidaEsperada THEN
        RETURN 'EXITO';
    ELSE
        RETURN 'FALLO';
    END IF;
END;
/
--Cabeceras de paquetes
--Tabla Persona
CREATE OR REPLACE PACKAGE PCK_Persona
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Persona.DNI%TYPE,
v_Nombre IN Persona.Nombre%TYPE, v_Apellidos IN
Persona.Apellidos%TYPE, v_FechNacimiento IN
Persona.FechNacimiento%TYPE, v_Email IN Persona.Email%TYPE,v_Telefono
IN Persona.Telefono%TYPE,v_Direccion IN Persona.Direccion%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN
Persona.DNI%TYPE, v_Nombre IN Persona.Nombre%TYPE, v_Apellidos IN
Persona.Apellidos%TYPE, v_FechNacimiento IN
Persona.FechNacimiento%TYPE, v_Email IN Persona.Email%TYPE,v_Telefono
IN Persona.Telefono%TYPE,v_Direccion IN Persona.Direccion%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Persona.DNI%TYPE,
salidaEsperada BOOLEAN);
END;
/

--Tabla Donacion
CREATE OR REPLACE PACKAGE PCK_Donacion
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_idDonante IN
Donacion.idDonante%TYPE, v_TipoDonacion IN Donacion.TipoDonacion%TYPE,
v_Campanya IN Donacion.Campanya%TYPE, v_TipoDonante IN
Donacion.TipoDonante%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_idDonante IN
Donacion.idDonante%TYPE, v_TipoDonacion IN Donacion.TipoDonacion%TYPE,
v_Campanya IN Donacion.Campanya%TYPE, v_TipoDonante IN
Donacion.TipoDonante%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2,v_idDonante IN
Donacion.idDonante%TYPE , salidaEsperada BOOLEAN);
END;
/
```

```
--Tabla Campanya
CREATE OR REPLACE PACKAGE PCK_Campanya
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN
Campanya.Nombre%TYPE,v_FechInicio IN Campanya.FechInicio%TYPE
,v_FechFin IN Campanya.FechFin%TYPE, v_Presupuesto IN
Campanya.Presupuesto%TYPE, v_DNIDestinados IN
Campanya.DNIDestinados%TYPE, v_DNIDirector IN
Campanya.DNIDirector%TYPE, v_CapitalInvertido IN
Campanya.CapitalInvertido%TYPE salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Nombre IN
Campanya.Nombre%TYPE,v_FechInicio IN Campanya.FechInicio%TYPE
,v_FechFin IN Campanya.FechFin%TYPE, v_Presupuesto IN
Campanya.Presupuesto%TYPE, v_DNIDestinados IN
Campanya.DNIDestinados%TYPE, v_DNIDirector IN
Campanya.DNIDirector%TYPE, v_CapitalInvertido IN
Campanya.CapitalInvertido%TYPE salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2,v_Nombre IN
Campanya.Nombre%TYPE , salidaEsperada BOOLEAN);
END;
/

--Tabla Material
CREATE OR REPLACE PACKAGE PCK_Material
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Cantidad IN
Material.Cantidad%TYPE, v_TipoDeMaterial IN
Material.TipoDeMaterial%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Cantidad IN
Material.Cantidad%TYPE, v_TipoDeMaterial IN
Material.TipoDeMaterial%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2,v_TipoDeMaterial IN
Material.TipoDeMaterial%TYPE , salidaEsperada BOOLEAN);
END;
/

--Tabla Capital
CREATE OR REPLACE PACKAGE PCK_Capital
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Cantidad IN
Capital.Cantidad%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Cantidad IN
Capital.Cantidad%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Cantidad IN
Capital.Cantidad%TYPE, salidaEsperada BOOLEAN);
END;
/
```

```
--Tabla Usuario
CREATE OR REPLACE PACKAGE PCK_Usuario
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Usuario.NombreUsuario%TYPE,v_Contrasenya IN
Usuario.Contrasenya%TYPE,v_DNI IN Usuario.DNI%TYPE,v_Campanyas IN
Usuario.Campanyas%TYPE,v_NumCampanyas IN
Usuario.NumCampanyas%TYPE,v_DonacionAnyo IN
Usuario.DonacionAnyo%TYPE,v_FechInicio IN
Usuario.FechInicio%TYPE,v_FechFin IN
Usuario.FechFin%TYPE,v_PrivilegioEdit IN
Usuario.PrivilegioEdit%TYPE,v_PrivilegioPub IN
Usuario.PrivilegioPub%TYPE,v_Cargo IN Usuario.Cargo%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Usuario.NombreUsuario%TYPE,v_Contrasenya IN
Usuario.Contrasenya%TYPE,v_DNI IN Usuario.DNI%TYPE,v_Campanyas IN
Usuario.Campanyas%TYPE,v_NumCampanyas IN
Usuario.NumCampanyas%TYPE,v_DonacionAnyo IN
Usuario.DonacionAnyo%TYPE,v_FechInicio IN
Usuario.FechInicio%TYPE,v_FechFin IN
Usuario.FechFin%TYPE,v_PrivilegioEdit IN
Usuario.PrivilegioEdit%TYPE,v_PrivilegioPub IN
Usuario.PrivilegioPub%TYPE,v_Cargo IN Usuario.Cargo%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Usuario.NombreUsuario%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Comentario
CREATE OR REPLACE PACKAGE PCK_Comentario
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Comentario.NombreUsuario%TYPE, v_Titulo IN Comentario.Titulo%TYPE,
v_Descripcion IN Comentario.Descripcion%TYPE, v_idComentario IN
Comentario.idComentario%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Comentario.NombreUsuario%TYPE, v_Titulo IN Comentario.Titulo%TYPE,
v_Descripcion IN Comentario.Descripcion%TYPE, v_idComentario IN
Comentario.idComentario%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_idComentario IN
Comentario.idComentario%TYPE, salidaEsperada BOOLEAN);
END;
/
```

```
--Tabla Noticias
CREATE OR REPLACE PACKAGE PCK_Noticias
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Noticias.NombreUsuario%TYPE, v_Titulo IN Noticias.Titulo%TYPE,
v_Descripcion IN Noticias.Descripcion%TYPE, v_Url IN
Noticias.Url%TYPE,salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Noticias.NombreUsuario%TYPE, v_Titulo IN Noticias.Titulo%TYPE,
v_Descripcion IN Noticias.Descripcion%TYPE, v_Url IN
Noticias.Url%TYPE,salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Titulo IN
Noticias.Titulo%TYPE , salidaEsperada BOOLEAN);
END;
/

--Tabla AccionesHumanitarias
CREATE OR REPLACE PACKAGE PCK_AccionesHumanitarias
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Pais IN
AccionesHumanitarias.Pais%TYPE,v_IdCampanya IN
AccionesHumanitarias.IdCampanya%TYPE,v_Nombre IN
AccionesHumanitarias.Nombre%TYPE,v_Descripcion IN
AccionesHumanitarias.Descripcion%TYPE,v_FechInicio IN
AccionesHumanitarias.FechInicio%TYPE,v_FechFin IN
AccionesHumanitarias.FechFin%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Pais IN
AccionesHumanitarias.Pais%TYPE,v_IdCampanya IN
AccionesHumanitarias.IdCampanya%TYPE,v_Nombre IN
AccionesHumanitarias.Nombre%TYPE,v_Descripcion IN
AccionesHumanitarias.Descripcion%TYPE,v_FechInicio IN
AccionesHumanitarias.FechInicio%TYPE,v_FechFin IN
AccionesHumanitarias.FechFin%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_IdCampanya IN
AccionesHumanitarias.IdCampanya%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Anonimo
CREATE OR REPLACE PACKAGE PCK_Anonimo
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_idAnonimo IN
Anonimo.idAnonimo%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_idAnonimo IN
Anonimo.idAnonimo%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_idAnonimo IN
Anonimo.idAnonimo%TYPE, salidaEsperada BOOLEAN);
END;
/
```

```
--Tabla Institucion
CREATE OR REPLACE PACKAGE PCK_Institucion
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_IdEmpresa IN
Institucion.IdEmpresa%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_IdEmpresa IN
Institucion.IdEmpresa%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2,v_IdEmpresa IN
Institucion.IdEmpresa%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla AsociacionONG
CREATE OR REPLACE PACKAGE PCK_AsociacionONG
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN
AsociacionONG.Nombre%TYPE, v_FondosDisponibles IN
AsociacionONG.FondosDisponibles%TYPE, v_NumVoluntarios IN
AsociacionONG.NumVoluntarios%TYPE, v_NumDirectores IN
AsociacionONG.NumDirectores%TYPE, v_NumDirectoresVet IN
AsociacionONG.NumDirectoresVet%TYPE, v_UltimaDonacion IN
AsociacionONG.UltimaDonacion%TYPE, v_UsuarioUltimaDonacion IN
AsociacionONG.UsuarioUltimaDonacion%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar ();
PROCEDURE Eliminar (nombrePrueba VARCHAR2, , salidaEsperada BOOLEAN);
END;
/
```

```
--Cuerpos de paquetes
--Tabla Persona
CREATE OR REPLACE PACKAGE BODY PCK_Persona
IS
    CURSOR C IS
        SELECT * FROM Persona;
    v_Salida BOOLEAN := TRUE;
    v_Persona Persona%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Persona;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Persona;
        DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25)
|| RPAD('Apellidos:', 25) || RPAD('FechNacimiento:', 25) ||
RPAD('Email:', 25))|| RPAD('Telefono:', 25))|| RPAD('Direccion:',
25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Persona.DNI, 25) ||
RPAD(v_Alumnos.Persona, 25) || RPAD(v_Persona.Apellidos, 25) ||
RPAD(v_Persona.Fecha_Nacimiento, 25) || RPAD(v_Persona.Email, 25)||
RPAD(v_Persona.Telefono, 25);|| RPAD(v_Persona.Direccion, 25));
        FETCH C INTO v_Persona;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Persona.DNI%TYPE,
v_Nombre IN Persona.Nombre%TYPE, v_Apellidos IN
Persona.Apellidos%TYPE, v_FechNacimiento IN
Persona.FechNacimiento%TYPE, v_Email IN Persona.Email%TYPE,v_Telefono
IN Persona.Telefono%TYPE,v_Direccion IN Persona.Direccion%TYPE,
salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Persona (DNI, Nombre, Apellidos, FechNacimiento,
Email, Telefono, Direccion) VALUES (v_DNI, v_Nombre, v_Apellidos,
v_FechNacimiento, v_Email, v_Telefono, v_Direccion);
        SELECT * INTO v_Persona FROM Persona WHERE DNI = v_DNI;
        IF v_Persona.Nombre != v_Nombre AND v_Persona.Apellidos !=
v_Apellidos AND v_Persona.FechNacimiento != v_FechNacimiento AND
v_Persona.Email != v_Email AND v_Persona.Telefono != v_Telefono  AND
v_Persona.Direccion != v_Direccion THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
```

```
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN
Persona.DNI%TYPE, v_Nombre IN Persona.Nombre%TYPE, v_Apellidos IN
Persona.Apellidos%TYPE, v_FechNacimiento IN
Persona.FechNacimiento%TYPE, v_Email IN Persona.Email%TYPE,v_Telefono
IN Persona.Telefono%TYPE,v_Direccion IN Persona.Direccion%TYPE,
salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Persona SET Nombre = v_Nombre, Apellidos = v_Apellidos,
FechNacimiento = v_FechNacimiento, Email = v_Email, Telefono=
v_Telefono, Direccion= v_Direccion WHERE DNI = v_DNI;
        SELECT * INTO v_Persona FROM Persona WHERE DNI = v_DNI;
        IF v_Persona.Nombre != v_Nombre AND v_Persona.Apellidos !=
v_Apellidos AND v_Persona.FechNacimiento != v_FechNacimiento AND
v_Persona.Email != v_Email AND v_Persona.Telefono != v_Telefono  AND
v_Persona.Direccion != v_Direccion THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Persona.DNI%TYPE,
salidaEsperada BOOLEAN)
    IS
        v_NumPersona NUMBER := 0;
    BEGIN
        DELETE FROM persona WHERE DNI = v_DNI;
        SELECT COUNT(*) INTO v_NumPersona FROM Persona WHERE DNI =
v_DNI;
        IF v_NumPersona != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Donacion
CREATE OR REPLACE PACKAGE BODY PCK_Donacion
IS
    CURSOR C IS
        SELECT * FROM Donacion;
    v_Salida BOOLEAN := TRUE;
    v_Donacion Donacion%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Donacion;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Donacion;
        DBMS_OUTPUT.PUT_LINE(RPAD('idDonante:', 25) ||
RPAD('TipoDonacion:', 25) || RPAD('Campanya:', 25) ||
RPAD('TipoDonante:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Donacion.idDonante, 25) ||
RPAD(v_Donacion.TipoDonacion, 25) || RPAD(v_Donacion.Campanya, 25) ||
RPAD(v_Donacion.TipoDonante, 25));
        FETCH C INTO v_Donacion;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_idDonante IN
Donacion.idDonanteI%TYPE, v_TipoDonacion IN
Donacion.TipoDonacion%TYPE, v_Campanya IN Donacion.Campanya%TYPE,
v_TipoDonante IN Donacion.TipoDonante%TYPE,salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Donacion (idDonante, TipoDonacion,
Campanya,TipoDonante) VALUES (v_idDonante, v_TipoDonante, v_Campanya,
v_TipoDonantel);
        SELECT * INTO v_Donacion FROM Donacion WHERE idDonante =
v_idDonante;
        IF v_Donacion.TipoDonacion != v_TipoDonacion AND
v_Donacion.Campanya != v_Campanya AND v_Donacion.TipoDonante !=
v_TipoDonante THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_idDonante IN
Donacion.idDonante%TYPE, v_TipoDonacion IN Donacion.TipoDonacion%TYPE,
v_Campanya IN Donacion.Campanya%TYPE, v_TipoDonante IN
Donacion.TipoDonante%TYPE, salidaEsperada BOOLEAN)
```

```
    IS
    BEGIN
        UPDATE Donacion SET idDonante= v_idDonante, TipoDonacion =
v_TipoDonacion, Campanya= v_Campanya, TipoDonante = v_TipoDonante
WHERE idDonante = v_idDonante;
        SELECT * INTO v_Donacion FROM Donacion WHERE idDonante =
v_idDonante;
        IF v_Donacion.idDonante != v_idDonante AND
v_Donacion.TipoDonacion != v_TipoDonacion AND v_Donacion.Campanya !=
v_Campanya AND v_Donacion.TipoDonante != v_TipoDonante THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_idDonante IN
Donacion.idDonante%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumDonantes NUMBER := 0;
    BEGIN
        DELETE FROM DonacionWHERE idDonante = v_idDonante;
        SELECT COUNT(*) INTO v_NumDonantes FROM Donacion WHERE
idDonante = v_idDonante;
        IF v_NumDonante != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Campanya
CREATE OR REPLACE PACKAGE BODY PCK_Campanya
IS
    CURSOR C IS
        SELECT * FROM Campanya;
    v_Salida BOOLEAN := TRUE;
    v_Campanya Campanya%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Campanya;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Campanya;
        DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) ||
RPAD('FechInicio:', 25) || RPAD('FechFin:', 25) ||
RPAD('Presupuesto:', 25) || RPAD('DNIDestinados:', 25) ||
RPAD('DNIDirector:', 25) || RPAD('CapitalInvertido:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Campanya.Nombre, 25) ||
RPAD(v_Campanya.FechInicio, 25) || RPAD(v_Campanya.FechFin, 25) ||
RPAD(v_Campanya.Presupuesto, 25) || RPAD(v_Campanya.DNIDestinados, 25)
|| RPAD(v_Campanya.DNIDirector, 25) ||
RPAD(v_Campanya.CapitalInvertido, 25));
        FETCH C INTO v_Campanya;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN
Campanya.Nombre%TYPE, v_FechInicio IN Campanya.FechInicio%TYPE,
v_FechFin IN Campanya.FechFin%TYPE, v_Presupuesto IN
Campanya.Presupuesto%TYPE, v_DNIDestinatario IN
Campanya.DNIDestinatario%TYPE, v_DNIDirector IN
Campanya.DNIDirector%TYPE, v_CapitalInvertido IN
Campanya.CapitalInvertido%TYPE salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Campanya (Nombre, FechInicio, FechFin,
Presupuesto, DNIDestinados, DNIDirector, CapitalInvertido) VALUES
(v_Nombre, v_FechInicio, v_FechFin, v_Presupuesto, v_DNIDestinatario,
v_DNIDirector, v_CapitalInvertido);
        SELECT * INTO v_Campanya FROM Campanya WHERE Nombre =
v_Nombre;
        IF v_Campanya.Nombre != v_Nombre AND v_Campanya.FechInicio !=
v_FechInicio AND v_Campanya.FechFin != v_FechFin AND
v_Campanya.Presupuesto != v_Presupuesto THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
```

```
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Nombre IN
Campanya.Nombre%TYPE, v_FechInicio IN Campanya.FechInicio%TYPE,
v_FechFin IN Campanya.FechFin%TYPE, v_Presupuesto IN
Campanya.Presupuesto%TYPE, v_DNIDestinados IN
Campanya.DNIDestinados%TYPE, v_DNIDirector IN
Campanya.DNIDirector%TYPE, v_CapitalInvertido IN
Campanya.CapitalInvertido%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Campanya SET Nombre = v_Nombre, FechInicio =
v_FechInicio, FechFin = v_FechFin, Presupuesto = v_Presupuesto WHERE
DNIDestinados = v_DNIDestinados;
        SELECT * INTO v_Campanya FROM Campanya WHERE DNIDestinados =
v_DNIDestinados;
        IF v_Campanya.Nombre != v_Nombre AND v_Campanya.FechInicio !=
v_FechInicio AND v_Campanya.FechFin != v_FechFin AND
v_Campanya.Presupuesto != v_Presupuesto THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN
Campanya.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumCampanya := 0;
    BEGIN
        DELETE FROM Campanya WHERE Nombre = v_Nombre;
        SELECT COUNT(*) INTO v_NumCampanya FROM Campanya WHERE Nombre
= v_Nombre;
        IF v_NumCampanya != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Material
CREATE OR REPLACE PACKAGE BODY PCK_Material
IS
    CURSOR C IS
        SELECT * FROM Material;
    v_Salida BOOLEAN := TRUE;
    v_Material Material%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Material;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Material;
        DBMS_OUTPUT.PUT_LINE(RPAD('Cantidad:',
25)||('TipoDeMaterial:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE( RPAD(v_Material.Cantidad,
25)||RPAD(v_Material.TipoDeMaterial, 25));
        FETCH C INTO v_Material;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Cantidad IN
Material.TipoDeMaterial%TYPE, v_Cantidad IN Material.Cantidad%TYPE,
salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Capital (Cantidad, TipoDeMaterial) VALUES
(v_Cantidad, v_TipoDeMaterial);
        SELECT * INTO v_Material FROM Material WHERE TipoDeMaterial =
v_TipoDeMaterial;
        IF v_Material.TipoDeMaterial != v_TipoDeMaterial
v_Material.Cantidad != v_Cantidad THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Cantidad IN
Material.TipoDeMaterial%TYPE, v_Cantidad IN Material.Cantidad%TYPE,
salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Material SET TipoMaterial = v_TipoMaterial;
        SELECT * INTO v_Material FROM Material WHERE TipoDeMaterial =
v_TipoDeMaterial;
```

```
        IF v_Material.TipoDeMaterial != v_TipoDeMaterial
v_Material.Cantidad != v_Cantidad THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2,v_Cantidad IN
Material.TipoDeMaterial%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumMaterial NUMBER := 0;
    BEGIN
        DELETE FROM Material WHERE Material = v_TipoDeMaterial;
        SELECT COUNT(*) INTO v_NumMaterial FROM Material WHERE
Material = v_TipoDeMaterial;
        IF v_NumMaterial != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Capital
CREATE OR REPLACE PACKAGE BODY PCK_Capital
IS
    CURSOR C IS
        SELECT * FROM Capital;
    v_Salida BOOLEAN := TRUE;
    v_Capital Capital%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Capital;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Capital;
        DBMS_OUTPUT.PUT_LINE(RPAD('Cantidad:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE( RPAD(v_Capital.Cantidad, 25));
        FETCH C INTO v_Alumnos;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Cantidad IN
Capital.Cantidad%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Capital (Cantidad) VALUES (v_Cantidad);
        SELECT * INTO v_Capital FROM Capital WHERE Cantidad =
v_Cantidad;
        IF v_Capital.Cantidad != v_Cantidad THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Cantidad IN
Capital.Cantidad%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Capital SET Cantidad = v_Cantidad;
        SELECT * INTO v_Capital FROM Capital WHERE Cantidad =
v_Cantidad;
        IF v_Capital.Cantidad != v_Cantidad THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
```

```
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Cantidad IN
Capital.Cantidad%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumCapital NUMBER := 0;
    BEGIN
        DELETE FROM Capital WHERE Cantidad = v_Cantidad;
        SELECT COUNT(*) INTO v_NumCapital FROM Capital WHERE Cantidad
= v_Cantidad;
        IF v_NumCapital != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Usuario
CREATE OR REPLACE PACKAGE BODY PCK_Usuario
IS
    CURSOR C IS
        SELECT * FROM Usuario;
    v_Salida BOOLEAN := TRUE;
    v_Usuario Persona%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Usuario;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Usuario;
        DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) ||
RPAD('NombreUsuario:', 25) || RPAD('Contrasenya:', 25) ||
RPAD('Campanyas:', 25) || RPAD('NumCampanyas:', 25 ||
RPAD('DonacionAnyo:', 25) || RPAD('DonacionAnyo:', 25) ||
RPAD('FechInicio:', 25) || RPAD('FechFin:', 25) ||
RPAD('PrivilegioEdit:', 25) || RPAD('PrivilegioPub:', 25) ||
RPAD('Cargo:', 25)));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Usuario.DNI, 25) ||
RPAD(v_Usuario.NombreUsuario, 25) || RPAD(v_Usuario.Contrasenya, 25)
|| RPAD(v_Usuario.Campanyas, 25) || RPAD(v_Usuario.NumCampanyas, 25)||
RPAD(v_Usuario.DonacionAnyo, 25)|| RPAD(v_Usuario.FechInicio, 25)||
RPAD(v_Usuario.FechFin, 25)|| RPAD(v_Usuario.PrivilegioEdit, 25)||
RPAD(v_Usuario.PrivilegioPub, 25)|| RPAD(v_Usuario.Cargo, 25));
        FETCH C INTO v_Usuario;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Usuario.DNI%TYPE,
v_NombreUsuario IN Usuario.NombreUsuario%TYPE, v_Contrasenya IN
Usuario.Contrasenya%TYPE, v_Campanyas IN Usuario.Campanyas%TYPE,
v_NumCampanyas IN Usuario.NumCampanyas%TYPE,v_DonacionAnyo IN
Usuario.DonacionAnyo%TYPE,v_FechInicio IN
Usuario.FechInicio%TYPE,v_PrivilegioEdit IN
Usuario.PrivilegioEdit%TYPE,v_PrivilegioPub IN
Usuario.PrivilegioPub%TYPE,v_FechFin IN Usuario.FechFin%TYPE,v_Cargo
IN Usuario.Cargo%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Usuario(DNI,
NombreUsuario,Contrasenya,Campanyas,NumCampanyas,DonacionAnyo,FechInic
io,FechFin,PrivilegioEdit,PrivilegioPub,Cargo) VALUES (v_DNI,
v_NombreUsuario, v_Contrasenya,
v_Campanyas,v_NumCampanyas,v_DonacionAnyo,v_FechInicio,v_FechFin,v_Pri
vilegioEdit,v_PrivilegioPub,v_Cargo);
        SELECT * INTO v_Usuario FROM Usuario WHERE DNI = v_DNI;
        IF v_Usuario.NombreUsuario != v_NombreUsuario AND
v_Usuario.Contrasenya!= v_Contrasenya AND v_Usuario.Campanyas!=
v_Campanyas AND v_Usuario.NumCampanyas!= v_NumCampanyas AND
v_Usuario.DonacionAnyo != v_DonacionAnyo AND v_Usuario.FechInicio !=
```

```
v_FechInicio AND v_Usuario.FechFin != v_FechFin AND
v_Usuario.PrivilegioEdit != v_ PrivilegioEdit AND
v_Usuario.PrivilegioPub != v_PrivilegioPub AND v_Usuario.Cargo !=
v_Cargo THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN
Usuario.DNI%TYPE, v_NombreUsuario IN Usuario.NombreUsuario%TYPE,
v_Contrasenya IN Usuario.Contrasenya%TYPE, v_Campanyas IN
Usuario.Campanyas%TYPE, v_NumCampanyas IN
Usuario.NumCampanyas%TYPE,v_DonacionAnyo IN
Usuario.DonacionAnyo%TYPE,v_FechInicio IN
Usuario.FechInicio%TYPE,v_PrivilegioEdit IN
Usuario.PrivilegioEdit%TYPE,v_PrivilegioPub IN
Usuario.PrivilegioPub%TYPE,v_FechFin IN Usuario.FechFin%TYPE,v_Cargo
IN Usuario.Cargo%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Usuario SET NombreUsuario =v_NombreUsuario,
Contrasenya= v_Contrasenya, Campanyas = v_Campanyas, Email = v_Email
WHERE DNI = v_DNI;
        SELECT * INTO v_Usuario FROM Usuario WHERE = v_DNI;
        IF v_Usuario.NombreUsuario != v_NombreUsuario AND
v_Usuario.Contrasenya!= v_Contrasenya AND v_Usuario.Campanyas!=
v_Campanyas AND v_Usuario.NumCampanyas!= v_NumCampanyas AND
v_Usuario.DonacionAnyo != v_DonacionAnyo AND v_Usuario.FechInicio !=
v_FechInicio AND v_Usuario.FechFin != v_FechFin AND
v_Usuario.PrivilegioEdit != v_ PrivilegioEdit AND
v_Usuario.PrivilegioPub != v_PrivilegioPub AND v_Usuario.Cargo !=
v_Cargo THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Usuario.DNI%TYPE,
salidaEsperada BOOLEAN)
    IS
        v_NumAlumnos NUMBER := 0;
    BEGIN
        DELETE FROM Usuario WHERE DNI = v_DNI;
        SELECT COUNT(*) INTO v_NumAlumnos FROM Alumnos WHERE DNI =
v_DNI;
```

```
        IF v_NumAlumnos != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Noticias
CREATE OR REPLACE PACKAGE BODY PCK_Noticias
IS
    CURSOR C IS
        SELECT * FROM Noticias;
    v_Salida BOOLEAN := TRUE;
    v_Noticias Noticias%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Noticias;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Noticias;
        DBMS_OUTPUT.PUT_LINE(RPAD('NombreUsuario:', 25) ||
RPAD('Titulo:', 25) || RPAD('Descripcion:', 25) || RPAD('Url:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Noticias.NombreUsuario, 25) ||
RPAD(v_Noticias.Titulo, 25) || RPAD(v_Noticias.Descripcion, 25) ||
RPAD(v_Noticias.Url, 25));
        FETCH C INTO v_Noticias;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Noticias.NombreUsuario%TYPE, v_Titulo IN Noticias.Titulo%TYPE,
v_Descripcion IN Noticias.Descripcion%TYPE, v_Url IN
Noticias.Url%TYPE,salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Noticias (NombreUsuario, Titulo, Descripcion, Url)
VALUES (v_NombreUsuario, v_Titulo, v_Descripcion, v_Url);
        SELECT * INTO v_Noticias FROM Noticias WHERE Titulo =
v_Titulo;
        IF v_Noticias.NombreUsuario != v_NombreUsuario AND
v_Noticias.Titulo != v_Titulo AND v_Noticias.Descripcion !=
v_Descripcion AND v_Noticias.Url != v_Url THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Noticias.NombreUsuario%TYPE, v_Titulo IN Noticias.Titulo%TYPE,
v_Descripcion IN Noticias.Descripcion%TYPE, v_Url IN
Noticias.Url%TYPE,salidaEsperada BOOLEAN)
    IS
    BEGIN
```

```
        UPDATE Noticias SET NombreUsuario = v_NombreUsuario,
Titulo=v_Titulo, Descripcion = v_Descripcion, Url=v_Url WHERE Titulo =
v_Titulo;
        SELECT * INTO v_Noticias FROM Noticias WHERE Titulo =
v_Titulo;
        IF v_Noticias.NombreUsuario != v_NombreUsuario AND
v_Noticias.Titulo != v_Titulo AND v_Noticias.Descripcion !=
v_Descripcion AND v_Noticias.Url != v_Url THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Titulo IN
Noticias.Titulo%TYPE , salidaEsperada BOOLEAN)
    IS
        v_NumNoticias NUMBER := 0;
    BEGIN
        DELETE FROM Noticias WHERE Titulo = v_Titulo;
        SELECT * INTO v_Noticias FROM Noticias WHERE Titulo =
v_Titulo;
        IF v_NumNoticias != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Comentario
CREATE OR REPLACE PACKAGE BODY PCK_Comentario
IS
    CURSOR C IS
        SELECT * FROM Material;
    v_Salida BOOLEAN := TRUE;
    v_Material Material%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Material;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Material;
        DBMS_OUTPUT.PUT_LINE(RPAD('Cantidad:', 25) ||
RPAD('TipoDeMaterial:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Material.Cantidad, 25) ||
RPAD(v_Material.TipoDeMaterial, 25));
        FETCH C INTO v_Material;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Cantidad IN
Material.Cantidad%TYPE, v_TipoDeMaterial IN
Material.TipoDeMaterial%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Material (Cantidad, TipoDeMaterial) VALUES
(v_Cantidad, v_TipoDeMaterial);
        SELECT * INTO v_Material FROM Material WHERE TipoDeMaterial=
v_TipoDeMaterial;
        IF v_Material.Cantidad != v_Cantidad THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Cantidad IN
Material.Cantidad%TYPE, v_TipoDeMaterial IN
Material.TipoDeMaterial%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Material SET Cantidad =Cantidad, WHERE TipoDeMaterial =
v_TipoDeMaterial;
        SELECT * INTO v_Material FROM Material WHERE TipoDeMaterial =
v_TipoDeMaterial;
        IF v_Material.Cantidad != v_Cantidad THEN
```

```
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_TipoDeMaterial IN
Material.TipoDeMaterial%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumMaterial NUMBER := 0;
    BEGIN
        DELETE FROM Material WHERE TipoDeMaterial = v_TipoDeMaterial;
        SELECT COUNT(*) INTO v_NumMaterial FROM Material WHERE
TipoDeMaterial = v_TipoDeMaterial;
        IF v_NumMaterial != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Comentario
CREATE OR REPLACE PACKAGE BODY PCK_Comentario
IS
    CURSOR C IS
        SELECT * FROM Comentario;
    v_Salida BOOLEAN := TRUE;
    v_Comentario Comentario%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Comentario;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Comentario;
        DBMS_OUTPUT.PUT_LINE(RPAD('NombreUsuario:', 25) ||
RPAD('Titulo:', 25)||RPAD('Descripcion:', 25)||RPAD('idComentario:',
25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Comentario.NombreUsuario, 25)
|| RPAD(v_Comentario.Titulo, 25)|| RPAD(v_Comentario.Descripcion,
25)|| RPAD(v_Comentario.idComentario, 25));
        FETCH C INTO v_Comentario;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Comentario.NombreUsuario%TYPE, v_Titulo IN Comentario.Titulo%TYPE,
v_Descripcion IN Comentario.Descripcion%TYPE, v_idComentario IN
Comentario.idComentario%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Comentario(NombreUsuario,Titulo, Descripcion,
idComentario) VALUES (v_NombreUsuario, v_Titulo, v_Descripcion,
v_idComentario);
        SELECT * INTO v_Comentario FROM Comentario WHERE NombreUsuario
= v_NombreUsuario;
        IF v_Comentario.Titulo != v_Titulo AND
v_Comentario.Descripcion != v_Descripcion AND
v_Comentario.idComentario != v_idComentario THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Comentario.NombreUsuario%TYPE, v_Titulo IN Comentario.Titulo%TYPE,
v_Descripcion IN Comentario.Descripcion%TYPE, v_idComentario IN
Comentario.idComentario%TYPE, salidaEsperada BOOLEAN)
```

```
    IS
    BEGIN
        UPDATE Comentario SET Titulo= v_Titulo, Descripcion =
v_Descripcion, idComentario = v_idComentario WHERE NombreUsuario =
v_NombreUsuario;
        SELECT * INTO v_Comentario FROM Comentario WHERE
NombreUsuario= v_NombreUsuario;
        IF v_Comentario.Titulo != v_Ttitulo AND
v_Comentario.Descripcion != v_Descripcion AND
v_Comentario.idComentario != v_idComentario THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_NombreUsuario IN
Comentario.NombreUsuario%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumComentarios NUMBER := 0;
    BEGIN
        DELETE FROM Comentario WHERE NombreUsuario= v_NombreUsuario;
        SELECT COUNT(*) INTO v_NumComentarios FROM Comentario WHERE
NombreUsuario = v_NombreUsuario;
        IF v_NumComentarios != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla AccionesHumanitarias
CREATE OR REPLACE PACKAGE BODY PCK_AccionesHumanitarias
IS
    CURSOR C IS
        SELECT * FROM AccionesHumanitarias;
    v_Salida BOOLEAN := TRUE;
    v_AccionesHumanitarias AccionesHumanitarias%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM AccionesHumanitarias;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_AccionesHumanitarias;
        DBMS_OUTPUT.PUT_LINE(RPAD('Pais:', 25) || RPAD('Campanya:',
25)||RPAD('Nombre:', 25)||RPAD('Descripcion:',
25)||RPAD('FechInicio:', 25)||RPAD('FechFin:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_AccionesHumanitarias.Pais, 25)
|| RPAD(v_AccionesHumanitarias.Campanya, 25)||
RPAD(v_AccionesHumanitarias.Nombre, 25)||
RPAD(v_AccionesHumanitarias.Descripcion, 25)||
RPAD(v_AccionesHumanitarias.FechInicio,25)||
RPAD(v_AccionesHumanitarias.FechFin,25));
        FETCH C INTO v_AccionesHumanitarias;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Pais IN
AccionesHumanitarias.Pais%TYPE, v_Campanya IN
AccionesHumanitarias.Campanya%TYPE, v_Nombre IN
AccionesHumanitarias.Nombre%TYPE, v_Descripcion IN
AccionesHumanitarias.Descripcion%TYPE,  v_FechInicio IN
AccionesHumanitarias.FechInicio%TYPE, v_FechFin IN
AccionesHumanitarias.FechFin%TYPE,salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO AccionesHumanitarias(Pais, Campanya, Nombre,
Descripcion,FechInicio,FechFin) VALUES (v_Pais, v_Campanya, v_Nombre,
v_Descripcion,v_FechInicio, v_FechFin);
        SELECT * INTO v_AccionesHumanitarias FROM AccionesHumanitarias
WHERE Pais = v_Pais;
        IF v_AccionesHumanitarias.Campanya != v_Campanya AND
v_AccionesHumanitarias.Nombre != v_Nombre AND
v_AccionesHumanitarias.Descripcion != v_Descripcion AND
v_AccionesHumanitarias.FechInicio != v_FechInicio AND
v_AccionesHumanitarias.FechFin != v_FechFin THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
```

```
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Pais IN
AccionesHumanitarias.Pais%TYPE, v_Campanya IN
AccionesHumanitarias.Campanya%TYPE, v_Nombre IN
AccionesHumanitarias.Nombre%TYPE, v_Descripcion IN
AccionesHumanitarias.Descripcion%TYPE, v_FechInicio IN
AccionesHumanitarias.FechInicio%TYPE, v_FechFin IN
AccionesHumanitarias.FechFin%TYPE,  salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE AccionesHumanitarias SET Campanya= v_Campanya, Nombre =
v_Nombre, Descripcion = v_Descripcion, FechInicio= v_FechInicio,
FechFin=v_FechFin, WHERE Pais = v_Pais;
        SELECT * INTO v_AccionesHumanitarias FROM AccionesHumanitarias
WHERE Pais = v_Pais;
        IF v_AccionesHumanitarias.Campanya != v_Campanya AND
v_AccionesHumanitarias.Nombre != v_Nombre AND
v_AccionesHumanitarias.Descripcion != v_Descripcion AND
v_AccionesHumanitarias.FechInicio != v_FechInicio AND
v_AccionesHumanitarias.FechFin != v_FechFin THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Pais IN
AccionesHumanitarias.Pais%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumAccionesHumanitarias NUMBER := 0;
    BEGIN
        DELETE FROM AccionesHumanitarias WHERE Pais= v_Pais;
        SELECT COUNT(*) INTO v_NumAccionesHumanitarias FROM
AccionesHumanitarias WHERE Pais = v_Pais;
        IF v_NumAccionesHumanitarias != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Anonimo
CREATE OR REPLACE PACKAGE BODY PCK_Anonimo
IS
    CURSOR C IS
        SELECT * FROM Anonimo;
    v_Salida BOOLEAN := TRUE;
    v_Anonimo Anonimo%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Anonimo;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO Anonimo;
        DBMS_OUTPUT.PUT_LINE(RPAD('idAnonimo:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Anonimo.idAnonimo, 25));
        FETCH C INTO v_Anonimo;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_idAnonimo IN
Anonimo.v_idAnonimo%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Anonimo (idAnonimo) VALUES (v_idAnonimo);
        SELECT * INTO v_Anonimo FROM Anonimo WHERE idAnonimo =
v_idAnonimo;
        IF v_Anonimo.idAnonimo != v_idAnonimo THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_idAnonimo IN
Anonimo.idAnonimo%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Anonimo SET idAnonimo = v_idAnonimo WHERE idAnonimo =
v_idAnonimo;
        SELECT * INTO v_idAnonimo FROM Anonimo WHERE idAnonimo =
v_idAnonimo;
        IF v_Anonimo.idAnonimo != v_idAnonimo THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
```

```
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_idAnonimo IN
Aninimo.idAnonimo%TYPE, salidaEsperada BOOLEAN)
    IS
        v_idAnonimo NUMBER := 0;
    BEGIN
        DELETE FROM Anonimo WHERE idAnonimo = v_idAnonimo;
        SELECT COUNT(*) INTO v_idAnonimo FROM Anonimo WHERE idAnonimo
= v_idAnonimo;
        IF v_idAnonimo != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Institucion
CREATE OR REPLACE PACKAGE BODY PCK_Institucion
IS
    CURSOR C IS
        SELECT * FROM Institucion;
    v_Salida BOOLEAN := TRUE;
    v_Institucion Institucion%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Institucion;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Institucion;
        DBMS_OUTPUT.PUT_LINE(RPAD('IdEmpresa:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Institucion.IdEmpresa, 25));
        FETCH C INTO v_Institucion;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_IdEmpresa IN
Institucion.IdEmpresa%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Institucion(IdEmpresa) VALUES (v_IdEmpresa);
        SELECT * INTO v_Institucion FROM Institucion WHERE IdEmpresa =
v_IdEmpresa;
        IF v_Institucion.IdEmpresa != v_IdEmpresa THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_IdEmpresa IN
Institucion.IdEmpresa%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Institucion SET IdEmpresa = v_IdEmpresa;
        SELECT * INTO v_Institucion FROM Institucion WHERE IdEmpresa =
v_IdEmpresa;
        IF v_Institucion.IdEmpresa != v_IdEmpresa THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
```

```
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_IdEmpresa IN
Institucion.IdEmpresa%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumInstituciones NUMBER := 0;
    BEGIN
        DELETE FROM Institucion WHERE IdEmpresa = v_IdEmpresa;
        SELECT COUNT(*) INTO v_NumInstituciones FROM Institucion WHERE
IdEmpresa = v_IdEmpresa;
        IF v_NumInstituciones != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla AsociacionesONG
CREATE OR REPLACE PACKAGE BODY PCK_AsociacionONG
IS
    CURSOR C IS
        SELECT * FROM AsociacionONG;
    v_Salida BOOLEAN := TRUE;
    v_AsociacionONG AsociacionONG%ROWTYPE;
PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM AsociacionONG;
    END Inicializar;
PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_AsociacionONG;
        DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) ||
RPAD('FondosDisponibles:', 25) || RPAD('NumVoluntarios:', 25) ||
RPAD('NumDirectores:', 25) || RPAD('NumDirectoresVet:', 25) ||
RPAD('UltimaDonacion:', 25) || RPAD('UsuarioUltimaDonacion:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_AsociacionONG.Nombre, 25) ||
RPAD(v_AsociacionONG.FondosDisponibles, 25) ||
RPAD(v_AsociacionONG.NumVoluntarios, 25) ||
RPAD(v_AsociacionONG.NumDirectores, 25) ||
RPAD(v_AsociacionONG.NumDirectoresVet, 25) ||
RPAD(v_AsociacionONG.UltimaDonacion, 25)||
RPAD(v_AsociacionONG.UsuarioUltimaDonacion, 25));
        FETCH C INTO v_AsociacionONG;
        END LOOP;
        CLOSE C;
    END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN
AsociacionONG.Nombre%TYPE, v_FondosDisponibles IN
AsociacionONG.FondosDisponibles%TYPE, v_NumVoluntarios IN
AsociacionONG.NumVoluntarios%TYPE, v_NumDirectores IN
AsociacionONG.NumDirectores%TYPE, v_NumDirectoresVet IN
AsociacionONG.NumDirectoresVet%TYPE, v_UltimaDonacion IN
AsociacionONG.UltimaDonacion%TYPE, v_UsuarioUltimaDonacion IN
AsociacionONG.UsuarioUltimaDonacion%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Alumnos (DNI, Nombre, Apellidos, Fecha_Nacimiento,
Email) VALUES (v_DNI, v_Nombre, v_Apellidos, v_Fecha_Nacimiento,
v_Email);
        SELECT * INTO v_AsociacionONG FROM AsociacionONG WHERE Nombre
= v_Nombre;
        IF v_AsociacionesONG.Nombre != v_Nombre AND
v_AsociacionONG.FondosDisponibles != v_FondosDisponibles AND
v_AsociacionONG.NumVoluntarios != v_NumVoluntarios AND
v_AsociacionONG.NumDirectores != v_NumDirectores
v_AsociacionONG.NumDirectoresVet != v_NumDirectoresVet
v_AsociacionONG.UltimaDonacion != v_UltimaDonacion
v_AsociacionONG.UsuarioUltimaDonacion != v_UsuarioUltimaDonacion THEN
            v_Salida := FALSE;
        END IF;
```

```
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Nombre IN
AsociacionONG.Nombre%TYPE, v_FondosDisponibles IN
AsociacionONG.FondosDisponibles%TYPE, v_NumVoluntarios IN
AsociacionONG.NumVoluntarios%TYPE, v_NumDirectores IN
AsociacionONG.NumDirectores%TYPE, v_NumDirectoresVet IN
AsociacionONG.NumDirectoresVet%TYPE, v_UltimaDonacion IN
AsociacionONG.UltimaDonacion%TYPE, v_UsuarioUltimaDonacion IN
AsociacionONG.UsuarioUltimaDonacion%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE AsociacionONG SET Nombre = v_Nombre, FondosDisponibles
= v_FondosDisponibles, NumVoluntarios = v_NumVoluntarios,
NumDirectores=v_NumDirectores, NumDirectoresVet= v_NumDirectoresVet,
UltimaDonacion= v_UltimaDonacion, UsuarioUltimaDonacion=
v_UsuarioUltimaDonacion  WHERE Nombre = v_Nombre;
        SELECT * INTO v_AsociacionONG FROM AsociacionONG WHERE Nombre
= v_Nombre;
        IF v_AsociacionONG.Nombre != v_Nombre AND
v_AsociacionONG.FondosDisponibles != v_FondosDisponibles AND
v_AsociacionONG.NumVoluntarios != v_NumVoluntarios AND
v_AsociacionONG.NumDirectores != v_NumDirectores
v_AsociacionONG.NumDirectoresVet != v_NumDirectoresVet
v_AsociacionONG.UltimaDonacion != v_UltimaDonacion
v_AsociacionONG.UsuarioUltimaDonacion != v_UsuarioUltimaDonacion THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE,
salidaEsperada BOOLEAN)
    IS
        v_NumAsociacionesONG NUMBER := 0;
    BEGIN
        DELETE FROM AsociacionONG WHERE Nombre = v_Nombre;
        SELECT COUNT(*) INTO v_NumAsociacion FROM AsociacionONG WHERE
Nombre = v_Nombre;
        IF v_NumAsociacionONG != 0 THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(v_Salida, salidaEsperada));
```

```
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' ||
ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Eliminar;
END;
/
```

```
--Tabla Usuario
BEGIN
PCK_Usuario.Inicializar;
PCK_Usuario.Insertar ('Insertar un Usuario','Paula99',
'contrasena123', '23801240N',NULL,1,30,  TO_DATE('08/02/2016',
'DD/MM/YYYY'),NULL, S,S,'Administrador', TRUE);
PCK_Usuario.Insertar ('Insertar un Usuario','Sergio17',
'contrasena123','47966225R', 'Salvar
Africa',1,50,TO_DATE('28/10/2012', 'DD/MM/YYYY'),TO_DATE('04/09/2014',
'DD/MM/YYYY'), N,N,'Voluntario', TRUE);
PCK_Usuario.Insertar ('Insertar un Usuario','Juan_7Antonio',
'contrasena123', '56336094D','Salvar Africa,Salvar Asia',2,50,
TO_DATE('03/02/2013', 'DD/MM/YYYY'),TO_DATE('04/09/2017',
'DD/MM/YYYY'), N,S,'Director', TRUE);
PCK_Usuario.Insertar ('Insertar un Usuario','Sebastián79',
'contrasena123', '36081385C','Salvar Africa,Recaudacion
Leche,Recaudacion Gambas,Incendio Polo Norte',4,5,
TO_DATE('22/01/2016', 'DD/MM/YYYY'),TO_DATE('04/09/2017',
'DD/MM/YYYY'),S,S,'Director Veterano', TRUE);
PCK_Usuario.Insertar ('Insertar un Usuario con DNI
NULL','Juan12Antonio', 'contrasena123','Salvar Africa',1,25, NULL,
TO_DATE('04/09/2017', 'DD/MM/YYYY'),TO_DATE('04/09/2015',
'DD/MM/YYYY'), N,N,'Voluntario', FALSE);
PCK_Usuario.Insertar ('Insertar un Usuario violando CK_DNI_Usuario',
'563360949','Rubén08', 'contrasena123','Salvar Africa',1,2,
TO_DATE('25/08/2012', 'DD/MM/YYYY'),TO_DATE('04/09/2017',
'DD/MM/YYYY'),N,N,'Simpatizante', FALSE);
PCK_Usuario.Actualizar ('Actualizar un Usuario', '77843402V',
'Rubén265', 'contrasena123','Salvar Africa',1,70,
TO_DATE('25/08/2012', 'DD/MM/YYYY'),TO_DATE('04/09/2016',
'DD/MM/YYYY'),N,N,'Voluntario', TRUE);
PCK_Usuario.Actualizar ('Actualizar un Usuario inexistente',
'Susana69', 'contrasena123', '99694117L','Salvar Africa',1,53,
TO_DATE('04/09/2012', 'DD/MM/YYYY'),TO_DATE('04/09/2017',
'DD/MM/YYYY'), N,N,'Voluntario', FALSE);
PCK_Usuario.Eliminar ('Eliminar un Usuario', '23801240N', TRUE);
PCK_Usuario.Consultar;
END;
/
--Tabla de Campaña
BEGIN
PCK_Campanya.Insertar('Insertar una campaña', 'Salvemos África',
TO_DATE('13/03/1993', 'DD/MM/YYYY'), TO_DATE('03/04/2003',
'DD/MM/YYYY'), 23500, '00548678J,98523686N,42312153L,70964804Y',8900,
'45789966M',0001, TRUE);
PCK_Campanya.Insertar('Insertar una campaña', 'Salvemos África',
TO_DATE('02/01/1997', 'DD/MM/YYYY'), TO_DATE('14/05/2001',
'DD/MM/YYYY'), 23500, '00548678J,98523686N,42312153L,70964804Y',8900,
'45789966M',0001, TRUE);
PCK_Campanya.Insertar('Insertar una campaña', 'Terremoto de Haiti',
TO_DATE('13/11/2000', 'DD/MM/YYYY'), TO_DATE('03/08/2007',
'DD/MM/YYYY'), 30000, '12135544J,88995674L,44771012L,45668899M',15200,
'74859612J',0002, TRUE);
PCK_Campanya.Insertar('Insertar una campaña', 'Terremoto de Haiti',
TO_DATE('01/01/2006', 'DD/MM/YYYY'), TO_DATE('25/07/2010',
'DD/MM/YYYY'), 30000, '12135544J,88995674L,44771012L,45668899M',15200,
'74859612J',0002, TRUE);
```

```
PCK_Campanya.Insertar('Insertar una campaña', 'Tsunami en Somalia',
TO_DATE('11/11/2010', 'DD/MM/YYYY'), TO_DATE('04/09/2012',
'DD/MM/YYYY'), 5000, '78979512J,44557879L,42200014L,44773063N',2200,
NULL,0003,FALSE);
PCK_Campanya.Insertar('Insertar una campaña', 'Maremoto en España',
TO_DATE('15/12/2012', 'DD/MM/YYYY'), TO_DATE('04/09/2013',
'DD/MM/YYYY'), NULL, '78979512J,44557879L,42200014L,44773063N',NULL,
'45889630',0003,FALSE);
PCK_Campanya.Actualizar('Actualizar una campaña', 'Salvemos África',
TO_DATE('13/03/1993', 'DD/MM/YYYY'), TO_DATE('03/04/2003',
'DD/MM/YYYY'), 23500, '00548678J,98523686N,42312153L,70964804Y',8900,
'56558974K',0001, TRUE);
PCK_Campanya.Actualizar('Actualizar una campaña', 'Terremoto de
Haiti', TO_DATE(NULL, 'DD/MM/YYYY'), TO_DATE('25/07/2010',
'DD/MM/YYYY'), 30000, NULL,15200, '74859612J',0002, FALSE);
PCK_Campanya.Eliminar ('Eliminar una campaña', 'Maremoto en España',
TRUE);
PCK_Campanya.Consultar;
END;
/
--Tabla Persona
BEGIN
PCK_Persona.Insertar('Insertar una persona', 'Daniel','Garzón
Montalbán','56895574M','658987981','C\ Ortega Valencia',
TO_DATE('16/03/1992', 'DD/MM/YYYY'),'danal@hotmail.com', TRUE);
PCK_Persona.Insertar('Insertar una persona', 'Manuela','Garzón
Montero','47588221J','658945871','C\ Pedrera Valle',
TO_DATE('10/06/1992', 'DD/MM/YYYY'),'MGM92@hotmail.com', TRUE);
PCK_Persona.Insertar('Insertar una persona', 'Martín','González
Montalbán','56895574M','658987981','C\ TiroLínea',
TO_DATE('01/05/1995', 'DD/MM/YYYY'),'MartinGM00@hotmail.es', TRUE);
PCK_Persona.Insertar('Insertar una persona', 'Álvaro', 'Alonso
Agenjo', '41234577B','622324920','C\Oña ',TO_DATE('17/10/1987',
'DD/MM/YYYY'), 'alvaroaa@hotmail.es', TRUE);
PCK_Persona.Insertar('Insertar una persona', 'Alba', 'Alonso Parejo',
'41234570C','622598003','C\San Martín ',TO_DATE('10/10/1989',
'DD/MM/YYYY'), 'aalbaahotmail.es', FALSE);
PCK_Persona.Insertar('Insertar una persona', 'Álvaro', 'Bermejo
Martínez', '41234577120B','620000369','C\Carreta
',TO_DATE('11/11/1991', 'DD/MM/YYYY'), 'BermejoAl@hotmail.com',
FALSE);
PCK_Persona.Actualizar('Actualizar una persona', 'Álvaro', 'Bermejo
Martínez', '41234577B','620000369','C\Carreta ',TO_DATE('11/11/1991',
'DD/MM/YYYY'), 'BermejoAl@hotmail.com', TRUE);
PCK_Persona.Actualizar('Actualizar una persona', 'Álvaro', 'Bermejo
Martínez', NULL,'620000369','C\Carreta ',TO_DATE('11/11/1991',
'DD/MM/YYYY'), 'BermejoAl@hotmail.com', FALSE);
PCK_Persona.Eliminar('Eliminar una persona','41234577B', TRUE);
PCK_Persona.Consultar;
END;
/
```

```
--Tabla Asociación ONG
BEGIN
PCK_AsociacionONG.Insertar('Insertar una asociación ONG', 'SOS Ayudas
Sin Fronteras', 15000, 215, 45, 20, TO_DATE('20/09/2014',
'dd//MM//YYYY'), 'jmartinez89', TRUE);
PCK_AsociacionONG.Insertar('Insertar una asociación ONG', 'SOS Ayudas
Sin Fronteras', 15000, 215, 45, 20, TO_DATE('10/12/2016',
'dd//MM//YYYY'), 'leoJimenez09', TRUE);
PCK_AsociacionONG.Insertar('Insertar una asociación ONG', 'Ayudas Por
El Mundo', 23500, 420, 123, 57, TO_DATE('23/08/2015', 'dd//MM//YYYY'),
'hhJKevin2', TRUE);
PCK_AsociacionONG.Insertar('Insertar una asociación ONG', 'AYUDA A LOS
DEMÁS', 23500,420, 123, 57, TO_DATE('08/02/2014', 'dd//MM//YYYY'),
'MaríaQ_N', TRUE);
PCK_AsociacionONG.Insertar('Insertar una asociación ONG', NULL, 1500,
95, 20, 15, TO_DATE('29/10/2009', 'dd//MM//YYYY'), 'PEDRO99', FALSE);
PCK_AsociacionONG.Insertar('Insertar una asociación ONG', 'Todos
Unidos ONG', NULL, 150, 82, 33, TO_DATE('19/03/2010', 'dd//MM//YYYY'),
NULL, FALSE);
PCK_AsociacionONG.Actualizar('Actualizar una asociación ONG', 'SOS
Ayudas Sin Fronteras', 15000, 215, 45, 20, TO_DATE('20/09/2014',
'dd//MM//YYYY'), NULL, FALSE);
PCK_AsociacionONG.Actualizar('Actualizar una asociación ONG', 'Todos
Unidos ONG', 4579, 150, 82, 33, TO_DATE('19/03/2010', 'dd//MM//YYYY'),
'jjSantos78', TRUE);
PCK_AsociacionONG.Eliminar('Eliminar una asociación ONG', 'AYUDA A LOS
DEMÁS', TRUE);
PCK_AsociacionONG.Consultar;
END;
/
--Tabla Donación
BEGIN
PCK_Donacion.Inicializar;
PCK_Donacion.Insertar ('Realizar una donacion', '77843402V',
'monetaria', 'Africa lo es todo', 'voluntario', TRUE);
PCK_Donacion.Insertar ('Realizar una donacion', '77843402T',
'material', 'Los niños los son todo', 'Director de campaña', TRUE);
PCK_Donacion.Insertar ('Realizar una donacion', '77843402B',
'material', 'Koksel Baba no tien la culpa', 'Director de campaña
vetereano', TRUE);
PCK_Donacion.Insertar ('Realizar una donacion', '77843402Z',
'monetaria', 'Africa lo es todo', 'usuario', TRUE);
PCK_Donacion.Insertar ('Realizar una donacion', '7784V', 'piedras',
'Africa lo es todo', 'voluntario', FALSE);
PCK_Donacion.Insertar ('Realizar una donacion', '77843402Q',
'ordenadores', 'Africa lo es todo', 'voluntario', FALSE);
PCK_Donacion.Insertar ('Realizar una donacion', '77843402Q', 'ropa',
'Africa lo es todo', 'persona', FALSE);
PCK_Donacion.Insertar ('Realizar una donacion', '7702Q', 'ropa',
'Africa lo es todo', 'persona', FALSE);
PCK_Donacion.Consultar;
END;
/
```

```
--Tabla Material
BEGIN
PCK_Material.Inicializar;
PCK_Material.Insertar ('Añadir un material', 80, 'comida', TRUE);
PCK_Material.Insertar ('Añadir un material', 20, 'Medicinas', TRUE);
PCK_Material.Insertar ('Añadir un material', -5, 'comida', FALSE);
PCK_Material.Insertar ('Añadir un material', 9, 'ordenadores', FALSE);
PCK_Material.Consultar;
END;
/

--Tabla Capital
BEGIN
PCK_Capital.Inicializar;
PCK_Capital.Insertar ('Insertar un capital', 100, TRUE);
PCK_Capital.Insertar ('Insertar un capital', 0, FALSE);
PCK_Capital.Consultar;
END;
/

--Tabla Anónimo
BEGIN
PCK_Anonimo.Inicializar;
PCK_Anonimo.Insertar ('Insertar un usuario anonimo', '66998855T',
TRUE);
PCK_Anonimo.Insertar ('Insertar un usuario anonimo', '6699T', FALSE);
PCK_Anonimo.Consultar;
END;
/

--Tabla Comentario
BEGIN
PCK_Comentario.Inicializar;
PCK_Comentario.Insertar ('Realizar un comentario', 'GuanMomen',
'Molestia', 'La pagina web no me carga las campañas correctamente',
000000001, TRUE);
PCK_Comentario.Insertar ('Realizar un comentario', 'kokselBaba',
'Satisfaccion', 'Me gustan las acciones humanas que se realizan en
esta ONG', 000000002, TRUE);
PCK_Comentario.Insertar ('Realizar un comentario', 'AsierRojo',
'Informacion', 'Me gustaria objtener mas informacion de como poder
colaborar con una campaña', 000000003, TRUE);
PCK_Comentario.Insertar ('Realizar un comentario', 'EnriqueRoque',
'Aportacion', 'Podria ayudar por la causa hablando de ella en mi canal
de youtube', 000000004, TRUE);
PCK_Comentario.Insertar ('Realizar un comentario', 123, 'Molestia',
'La pagina web no me carga las campañas correctamente', 000000001,
FALSE);
PCK_Comentario.Insertar ('Realizar un comentario', 'GuanMomen',
'Molestia', 123, 000000001, FALSE);
PCK_Comentario.Insertar ('Realizar un comentario', 'GuanMomen', 123,
'La pagina web no me carga las campañas correctamente', 000000001,
FALSE);
PCK_Comentario.Insertar ('Realizar un comentario', 'GuanMomen',
'Molestia', 'La pagina web no me carga las campañas correctamente',
'000000008', FALSE);
PCK_Comentario.Consultar;
END;
```

```
/

--Tabla Noticia
BEGIN
PCK_Noticia.Inicializar;
PCK_Noticia.Insertar ('Realizar un noticia', 'GuanMomen', 'Nueva
campaña', 'se ha creado una nueva campaña en Africa',
'http://www.sosayudassinfronteras/Africa/nuevacampaña', TRUE);
PCK_Noticia.Insertar ('Realizar un noticia', 'kokselBaba',
'Satisfaccion', 'Se ha llegado a la presupuesto esperado en una
campaña', 'http://www.sosayudassinfronteras/presupuestosalcanzados',
TRUE);
PCK_Noticia.Insertar ('Realizar un noticia', 'AsierRojo',
'Informacion', 'se ha mejorado la pagina principal de la pagina web',
'http://www.sosayudassinfronteras/', TRUE);
PCK_Noticia.Insertar ('Realizar un noticia', 'EnriqueRoque',
'Aportacion', 'un famoso youtube nos ayuda hablando por su canal',
'https://www.youtube.com/channel/UC3gaAKujjwdr2isZG6GiA1w', TRUE);
PCK_Noticia.Insertar ('Realizar un noticia', 123, 'Nueva campaña', 'se
ha creado una nueva campaña en Africa',
'http://www.sosayudassinfronteras/Africa/nuevacampaña', FALSE);
PCK_Noticia.Insertar ('Realizar un noticia', 'kokselBaba', 123, 'Se ha
llegado a la presupuesto esperado en una campaña',
'http://www.sosayudassinfronteras/presupuestosalcanzados', FALSE);
PCK_Noticia.Insertar ('Realizar un noticia', 'AsierRojo',
'Informacion', 123, 'http://www.sosayudassinfronteras/', FALSE);
PCK_Noticia.Insertar ('Realizar un noticia', 'EnriqueRoque',
'Aportacion', 'un famoso youtube nos ayuda hablando por su canal',
'youtube.com', FALSE);
PCK_Noticia.Consultar;
END;
/

--Tabla Institucion
BEGIN
PCK_Institucion.Inicializar;
PCK_Institucion.Insertar ('Insertar una institucion', '22665588A',
TRUE);
PCK_Institucion.Insertar ('Insertar una institucion', '66992369',
FALSE);
PCK_Institucion.Consultar;
END;
/
```