



Introducción a la Ingeniería del Software y los Sistemas de Información Departamento de Lenguajes y Sistemas Informáticos Grado en Ingeniería Informática del Software

> Juan Carlos Cortés Muñoz María Elena Molino Peña Alejandro José Muñoz Aranda Mario Ruano Fernández

https://projetsii.informatica.us.es/projects/zl39gm6lqwghg63nzf3

## HISTORIAL DE VERSIONES

Versión	Contenido	Participantes
Primer entregable	Primer entregable completo	Juan Carlos Cortés Muñoz María Elena Molino Peña Alejandro Muñoz Aranda Mario Ruano Fernández
Revisión de primer entregable	<ul> <li>Requisitos generales: Desde el punto de vista del responsable.</li> <li>Número de página modificado a la derecha.</li> <li>Incluida información sobre expectativas del sistema.</li> <li>Anexo añadido.</li> <li>Enlaces externos en el pie de página.</li> <li>Página web de la fundación añadida.</li> <li>Índice actualizado.</li> <li>Acta de reunión con cliente.</li> </ul>	Juan Carlos Cortés Muñoz María Elena Molino Peña Alejandro Muñoz Aranda Mario Ruano Fernández
Segundo entregable	Segundo entregable completo	Juan Carlos Cortés Muñoz María Elena Molino Peña Alejandro Muñoz Aranda Mario Ruano Fernández
Revisión de segundo entregable	<ul> <li>Requisitos generales: generalización de los procesos de gestión de proyectos, emisión de mensajes y gestión de cuentas.</li> <li>Requisitos de información: faltaban referencias a entidades de importancia.</li> <li>Pruebas de aceptación: se añade identificadores unívocos para cada una.</li> <li>Modelado conceptual: mejora de algunas asociaciones, inclusión de algunas restricciones, especificación del tipo de herencia.</li> </ul>	Juan Carlos Cortés Muñoz María Elena Molino Peña Alejandro Muñoz Aranda Mario Ruano Fernández
Tercer entregable	Tercer entregable completo	Juan Carlos Cortés Muñoz María Elena Molino Peña Alejandro Muñoz Aranda Mario Ruano Fernández

# ÍNDICE

1.	INTRODUCCIÓN AL PROBLEMA	5
	Sobre el cliente	5
	Sobre los usuarios	5
	Estructura organizacional	6
	Gestión de proyectos	6
	Medición de resultados y evaluación de proyectos	6
	Dominio del problema	7
2.	GLOSARIO DE TÉRMINOS	8
3.	VISIÓN GENERAL DEL SISTEMA	10
	Expectativas del sistema	10
	Tipos de usuarios	10
4.	CATÁLOGO DE REQUISITOS	11
	Requisitos generales	11
	Requisitos de información	12
	Reglas de negocio y pruebas de aceptación	15
	Requisitos funcionales	20
	Requisitos no funcionales	23
5.	MODELO CONCEPTUAL	24
	Modelado UML	24
	Escenarios de prueba	27
	Escenario de prueba 1. Gestión de proyectos	
	Escenario de prueba 2. Gestión de cuentas Escenario de prueba 3. Gestión de comunicaciones	
6.		
	Matriz de clases de entidad y requisitos de información	
	Matriz de clases de entidad y reglas de negocio	34
	Matriz de asociaciones y requisitos de información	
	Matriz de asociaciones y reglas de negocios	36
	Matriz de restricciones y requisitos de información	
	Matriz de restricciones y reglas de negocio	
7.	MODELO RELACIONAL	39
	Justificación de estrategias de transformación de jerarquías	40

8.	MODELO TECNOLÓGICO	42
	Tablas, secuencias y triggers de secuencias	42
	Funciones y procedimientos	53
	Cursores y listados de consultas	63
	Triggers	73
	Pruebas	79
9.	ANEXOS	130
	Anexo I: Prototipo de newsletter	130
	Anexo II: Acta de reunión	131

## 1. INTRODUCCIÓN AL PROBLEMA

#### Sobre el cliente

Deporte y Desafío<sup>1</sup> es una fundación española dependiente del Ministerio de Educación y Ciencia, con sede general en Madrid, que lleva 20 años trabajando por la inclusión social de las personas con discapacidades a través del deporte.

Fundada en 1998, Deporte y Desafío fue una de las principales entidades en introducir en España el deporte y el material adaptado para personas con discapacidades.

Según explican en su página web: "Mediante la promoción del deporte, aumentamos la libertad de las personas que tienen una discapacidad, promovemos la independencia y ayudamos a descubrir, tanto a ellos como a sus familias, su potencial social, mental y deportivo".

Esto es posible gracias a una amplia variedad de actividades deportivas de carácter trimestral, jornadas puntuales y actividades de larga duración, en las que los participantes están continuamente acompañados de voluntarios y asistidos por especialistas.

Cada año incorporan nuevos deportes y actividades a sus proyectos, al mismo tiempo que amplían su presencia geográfica por todo el territorio español.

Deporte y Desafío también contribuye a la formación e inserción laboral de las personas con discapacidad.

### Sobre los usuarios

Los principales usuarios a los que va dirigida la labor que se realiza desde Deporte y Desafío son personas con algún tipo de discapacidad, sin importar el tipo de esta, ya que todos los programas deportivos de la fundación son de carácter inclusivo.

Para diferenciar bien los perfiles y roles a la hora de llevar a cabo su labor, Deporte y Desafío identifica a este conjunto de personas como participantes, ya que son los protagonistas de los diferentes programas y actividades.

También los tutores legales (familiares, responsables de centro, etc.) tienen un importante papel dentro del grupo de los usuarios, ya que en el caso de que el participante sea menor de edad o tenga un alto grado de discapacidad, son los que están en contacto directo con la fundación para realizar los trámites de autorización, inscripción y representación del participante.

<sup>&</sup>lt;sup>1</sup> Página web de la fundación: <a href="https://deporteydesafio.com">https://deporteydesafio.com</a>

## Estructura organizacional

La fundación está presidida por Jorge Pérez de Leza, cargo de carácter institucional, quien encabeza el patronato que administra la fundación. Además, existe un patronato de honor con mero carácter institucional.

La actividad íntegra de la fundación está en manos de su directora general, Carmen Pardo, quien dirige a un equipo de cuatro coordinadoras de programas deportivos, a la responsable de integración laboral y al director técnico del programa de esquí alpino adaptado (Sierra Nevada, Granada).

La fundación puede llevar a cabo sus actividades gracias a la financiación a través de subvenciones de entidades públicas y privadas, así como de donaciones particulares de carácter monetario o material.

## Gestión de proyectos

Deporte y Desafío presenta a sus distintos patrocinadores proyectos por cada evento o programa deportivo que pretende realizar, con el fin de obtener la financiación necesaria para cubrir las actividades que los componen.

A la hora de llevar a cabo cada uno de los proyectos, la labor de gestión, ejecución, seguimiento y evaluación recae en el equipo de cuatro coordinadoras, con la directora general al frente del mismo.

Este trabajo integral comprende desde la búsqueda y selección de los participantes con discapacidad que formarán parte de las actividades, la formación y dirección del grupo de voluntarios, la localización de los espacios para la realización de actividades, hacer presupuestos, receptar facturas, etc. así como las correspondientes labores de gestión diaria de oficina.

Deporte y Desafío también cuenta con la figura de un director técnico en Sierra Nevada, siendo él el enlace entre la sede de Madrid y las actividades en la nieve que se realizan en Granada.

También existe una persona encargada de la contabilidad de la fundación.

## Medición de resultados y evaluación de proyectos

El método de medición a través del cual el equipo de cuatro coordinadoras de programas deportivos lleva a cabo la evaluación de proyectos es la observación participante.

También realizan encuestas a los participantes y voluntarios para obtener las valoraciones sobre las diferentes actividades ejecutadas.

Todos estos datos, en su mayoría de carácter cualitativo, son transmitidos a las empresas patrocinadoras partícipes en el proyecto en cuestión en forma de informes o memorias.

Deporte y Desafío no cuenta con ninguna herramienta para gestionar todos los datos generados en el proceso de evaluación de proyectos.

## Dominio del problema

A grandes rasgos, desde Deporte y Desafío consideran que tienen problemas de tiempos a la hora de gestionar sus proyectos, principalmente derivados de la falta de clasificación y mal almacenamiento de datos.

Aseguran que su gestión podría ser mucho más fácil economizando los tiempos a través de nuevos flujos de trabajo mejorados, suponiendo algunos de estos incluso la eliminación de tareas que, hoy en día, suponen un trabajo mecánico bastante costoso en lo que a tiempo se refiere.

Cuentan con unos recursos humanos limitados. Principalmente es un equipo de cuatro personas el encargado de gestionar y ejecutar por completo los proyectos.

Además, los recursos técnicos son escasos. Utilizan el paquete ofimático básico para el desarrollo de todas sus labores de planificación y gestión. No cuentan con una base de datos como tal, ni con copias de seguridad de archivos de gran valor.

De manera esquemática, el dominio del problema lo componen:

- Datos desactualizados o ingresados fuera de tiempo.
- Labores de búsqueda y selección de participantes para las actividades.
- Labores de búsqueda y selección de los voluntarios más indicados para cada actividad.
- Recopilación de actividades en archivo de texto, lo que impide la búsqueda rápida y por diferentes criterios de selección, así como la manipulación atómica de los resultados.
- Almacenamiento de datos (cuestionarios de satisfacción y evaluación de actividades) en forma de documento escaneado.
- Informes y memorias de actividades bastante limitadas en datos cuantitativos.
- Limitaciones de comunicación masiva. No cuentan con otro canal de comunicación más allá del email y el teléfono (salvo algunas comunicaciones vía Twitter y la versión free del servicio de mailing MailChimp).

## 2. GLOSARIO DE TÉRMINOS

- **Actividad**<sup>2</sup>: Ejercicio o juego que desempeñan los participantes (en eventos o programas deportivos) para disfrutar o entretenerse con otras personas.
- **Boletín**<sup>3</sup>: Medio de comunicación para informar de noticias o temas de interés de la fundación, que se publican con una cierta periodicidad.
- Coordinador: Miembro del equipo de la organización que lleva a cabo la gestión de los proyectos, su evaluación, y todas las tareas que sean necesarias en el desarrollo de los mismos.
- **Donación**: Aportación económica o de medios materiales realizada por un donante a la organización.
- **Donante**: Persona o institución que contribuye económicamente, o con material, con la fundación.
- Encuesta: Cuestionario que completan los participantes y voluntarios para evaluar las actividades llevadas a cabo y los servicios prestados por la fundación, con el objetivo de obtener información para poder mejorarlos.
- **Especialista**: Persona que cuenta con la formación necesaria para ayudar, socorrer o incluso enseñar la forma de ejecutar una actividad determinada (personal sanitario, monitores, etc.).
- Evaluación del proyecto: Valoración de una actividad finalizada, realizada a partir de encuestas a los participantes y voluntarios, que se envía a los patrocinadores en modo de informe.
- **Evento**<sup>4</sup>: Acontecimiento lúdico-deportivo, con el fin de promover el deporte en general, y dar a conocer el deporte adaptado para personas con discapacidad (fiestas, mercadillos, subastas, teatros, conciertos, carreras, torneos, etc.).
- **Factura**: Documento en el que se recoge la información y el costo de un producto o servicio, su cantidad y la fecha de pago del mismo.
- Familiar: Persona que pertenece a la familia de un participante o que mantiene una relación muy estrecha con este, que puede acompañarlo durante los eventos o representarlo administrativamente.
- **Fundación**: Sociedad u organización cuyos miembros se dedican a obras sociales o humanitarias sin ánimo de lucro.
- **Informe**: Documentación que recoge la evaluación de un proyecto que se envía a los patrocinadores.
- **Material adaptado**: Equipamiento deportivo acondicionado para que los participantes puedan realizar las actividades.
- **Memoria**<sup>5</sup>: Documento donde quedan recogidos los resultados y el desarrollo de los proyectos, junto con pequeños informes de cada actividad, efectuados durante el año.
- **Newsletter**<sup>6</sup>: Información que se remite de forma breve y resumida al correo electrónico de voluntarios, participantes y tutores legales subscritos a la fundación, para darles a conocer nuevas actividades y la posibilidad de inscribirse a estas.

<sup>&</sup>lt;sup>2</sup> Esquí alpino: https://deporteydesafio.com/programas-deportivos/actividades-squi-alpino.html

<sup>&</sup>lt;sup>3</sup> Sección de noticias: <a href="https://www.deporteydesafio.com/noticias/page/2/">https://www.deporteydesafio.com/noticias/page/2/</a>

<sup>&</sup>lt;sup>4</sup> Sección de eventos: <a href="https://www.deporteydesafio.com/noticias/eventos/">https://www.deporteydesafio.com/noticias/eventos/</a>

<sup>&</sup>lt;sup>5</sup> Memoria anual 2017: https://www.deporteydesafio.com/newsletter/newsletter-2017.html

<sup>&</sup>lt;sup>6</sup> Newsletter: <a href="https://mailchi.mp/31acfe3133bc/jornadas-8-de-junio-senderismo-y-geocaching-1659937?e=41becb9e91">https://mailchi.mp/31acfe3133bc/jornadas-8-de-junio-senderismo-y-geocaching-1659937?e=41becb9e91</a>

- **Participante**: Persona con discapacidad física, intelectual y/o sensorial, de cualquier edad, que es beneficiario de los programas deportivos y actividades que se realizan en la fundación.
- Patrocinador<sup>7</sup>: Entidad pública o privada, que subvenciona actividades con fines publicitarios. Según la aportación realizada, se clasifican en oro, plata, bronce o colaborador.
- **Programa deportivo**<sup>8</sup>: Conjunto de actividades desarrolladas por la fundación para la inclusión social de las personas con discapacidad a través del deporte que, a su vez, potencia la libertad e independencia para ellos y sus familiares.
- **Proyecto**: Idea propuesta por el equipo de coordinación, con la directora general al frente, que comprende la búsqueda de participantes, el lugar de realización y los voluntarios implicados, la elaboración de presupuestos, la recepción de facturas, etc. para plantear un programa deportivo a los organismos públicos y privados con el fin de conseguir los fondos necesarios para su ejecución.
- **Subvención**: Contribución de dinero otorgada a la organización por parte del Estado, una administración pública u organismo oficial, así como por entidades privadas, que patrocinan y cubren el coste total de un proyecto deportivo.
- **Tutor legal**: Representante legal de un participante menor de edad o con un alto grado de discapacidad.
- **Voluntario**: Persona comprometida con la fundación, asignada a un participante durante una actividad, cuya labor es acompañarlo, asistirlo y fomentar su integración en el grupo.

<sup>&</sup>lt;sup>7</sup> Directorio de patrocinadores: <a href="https://deporteydesafio.com/patrocinadores.html">https://deporteydesafio.com/patrocinadores.html</a>

<sup>&</sup>lt;sup>8</sup> Oferta de programas deportivos: <a href="https://deporteydesafio.com/programas-deportivos.html">https://deporteydesafio.com/programas-deportivos.html</a>

## 3. VISIÓN GENERAL DEL SISTEMA

## Expectativas del sistema

El objetivo principal de este proyecto se centra en el desarrollo de una intranet de gestión basada en un sistema de información que permita mejorar, desde un punto de vista administrativo, los proyectos que se llevan a cabo en Deporte y Desafío.

Principalmente, desde la fundación esperan reducir, de manera notable, el tiempo requerido por cada programa deportivo o evento en su fase de planificación. Confían en que el nuevo sistema supondrá un cambio en su dinámica de trabajo, agilizando tareas de selección de participantes, logrando cerrar con mayor brevedad la fase de inscripción en actividades y formalizar mejor los equipos de voluntariado.

Del mismo modo, consideran obtener mejores valoraciones y mejores calificaciones de sus actividades, haciendo crecer la confianza de sus patrocinadores en ellos.

En definitiva, el sistema de información debe canalizar todos los datos que derivan de los diferentes procesos que comprende cualquier proyecto de la fundación, ya sean propios de la fase de planificación y gestión, ejecución, o evaluación y memoria.

Esto permitirá optimizar los flujos de trabajo, principalmente economizando tiempos, almacenando de forma eficaz los datos generados para su futuro uso aplicado.

## Tipos de usuarios

- Coordinador (Admin): usuario que cuenta con todos los privilegios de la plataforma.
  Pertenece al equipo de coordinación de programas deportivos de la fundación. Puede
  introducir, modificar y eliminar datos de participantes, voluntarios, tutores legales,
  actividades, eventos y proyectos, así como patrocinadores y donantes. También puede
  acceder a toda la relación de datos y realizar consultas. Tiene la opción de generar
  informes con los datos de los cuestionarios de las actividades.
- Participante: usuario al que van dirigidas las actividades que se realizan en la asociación. Tiene acceso a sus datos, donde puede consultar un historial de participación. Puede ver un listado de las próximas actividades en el que puede indicar su disponibilidad para participar en cada una de ellas. Tiene acceso a un cuestionario de evaluación de las actividades realizadas. Su perfil puede estar asociado al de un tutor legal, el cual tiene privilegios manipulación de dicho perfil.
- Voluntario: este tipo de usuario tiene acceso a sus datos, donde puede obtener un historial de voluntariado. Puede ver un listado de las próximas actividades en el que puede indicar su disponibilidad para participar como voluntario. Tiene acceso a un cuestionario de evaluación de las actividades realizadas.
- Tutor legal: este usuario está siempre relacionado con el perfil de un participante. En
  el caso de que el participante sea menor de edad o tenga un grado de discapacidad
  superior al 50% es obligatorio la presencia de este usuario. Tiene acceso a sus datos
  personales y a los del participante al que representa, donde puede consultar el
  historial de participación de este, modificar su estado de disponibilidad, así como
  enviar información médica sobre el participante y sus principales necesidades.

# 4. CATÁLOGO DE REQUISITOS

## Requisitos generales

## RG - 1 Gestión de proyectos

Como coordinador,

quiero gestionar correctamente y en tiempo los proyectos,

**para** dar a conocer de forma eficaz la oferta de actividades, ofrecer mejores servicios a los participantes y voluntarios, y cumplir las expectativas de los patrocinadores para obtener financiación para proyectos futuros.

#### RG – 2 Gestión de comunicaciones

Como coordinador,

quiero controlar el envío de mensajes,

para hacer llegar de forma rápida, masiva y selectiva, en función del tipo de destinatario, las diferentes comunicaciones que se inician desde la fundación, agilizando el proceso de promoción de actividades, tanto entre participantes y voluntarios, como para la búsqueda de financiación entre patrocinadores.

#### RG - 3 Gestión de cuentas

Como coordinador,

quiero conocer el estado de las cuentas de proyectos,

para controlar el rendimiento de los activos y de la financiación recibida por parte de los patrocinadores, además de llevar una contabilidad del pago de las inscripciones de los participantes y los recibos.

## Requisitos de información

#### RI - 1 Información de coordinadores

Como coordinador,

quiero disponer de la siguiente información sobre los coordinadores:

- Datos personales: DNI (obligatorio), nombre (obligatorio), apellidos, fecha de nacimiento (obligatorio), dirección, localidad, provincia, código postal, email y teléfono (obligatorio).
- Eventos y programas deportivos en los que son los responsables de coordinación.

**para** contar con la información completa del equipo de coordinación de la fundación, conociendo la relación de proyectos en los que están trabajando.

## RI - 2 Información de participantes

Como coordinador,

quiero disponer de la siguiente información de los participantes:

- Datos personales: DNI (obligatorio), nombre (obligatorio), apellidos, fecha de nacimiento (obligatorio), dirección, localidad, provincia, código postal, email y teléfono (obligatorio).
- Grado de discapacidad (obligatorio).
- Tutor legal o persona que le representa (obligatorio).
- Disponibilidad para las actividades programadas.
- Prioridad de participación (alta, media, baja).
- Informes médicos del participante.

**para** recopilar una memoria de todos los participantes y ofrecerles las actividades que mejor se adapten a cada uno.

## RI – 3 Información de voluntarios

Como coordinador,

quiero disponer de la siguiente información sobre los voluntarios:

- Datos personales: DNI (obligatorio), nombre (obligatorio), apellidos, fecha de nacimiento (obligatorio), dirección, localidad, provincia, código postal, email y teléfono (obligatorio).
- Disponibilidad para las actividades programadas.
- Prioridad de participación (alta, media, baja).
- Participantes a los que ha acompañado en actividades.

para tener una agenda de los colaboradores y realizar una mejor asignación de las tareas en las que tengan un mayor grado de compatibilidad.

#### RI – 4 Información de proyectos

Como coordinador,

quiero disponer de la siguiente información sobre los proyectos:

- Identificador, fecha de inicio y fin del proyecto, ubicación y nombre (todos los atributos son obligatorios).
- Responsable de coordinación (obligatorio).
- Actividades que lo componen (obligatorio).

para realizar un seguimiento de los proyectos (eventos y programas deportivos) que se realizan a lo largo del año y disponer de toda la información necesaria da cara a organizar futuras ediciones.

#### RI - 5 Información de actividades

Como coordinador,

quiero disponer de la siguiente información sobre las actividades:

- Identificador, nombre, objetivos, número de plazas para participantes, número de voluntarios necesarios, tipo de actividad (deportiva, formativa o social), coste total y coste de inscripción (todos los atributos son obligatorios).

para tener los datos de cada una de las actividades que forman parte de los eventos y programas deportivos que lleva a cabo la fundación, permitiendo la gestión de personal y la contabilidad de los costes de estas.

#### RI – 6 Información de tutores legales

Como coordinador,

quiero disponer de la siguiente información sobre los tutores legales:

- Datos personales: DNI (obligatorio), nombre (obligatorio), apellidos, fecha de nacimiento (obligatorio), dirección, localidad, provincia, código postal, email y teléfono (obligatorio).
- Participante al que representan (obligatorio).

para tener una agenda de los representantes legales de los participantes.

#### RI – 7 Información de donaciones

Como coordinador,

quiero disponer de la siguiente información sobre las donaciones:

- Cantidad donada, valor unitario de la misma y fecha (todos los atributos son obligatorios).
- Tipo de donación (obligatorio).
- Donante (persona o institución que realiza la donación) (obligatorio).

para tener un inventario de las donaciones que se realizan a la fundación.

#### RI – 8 Información de patrocinadores

Como coordinador,

quiero disponer de la siguiente información sobre los patrocinadores:

- Datos de la empresa: CIF (obligatorio), nombre (obligatorio), dirección, localidad, provincia, código postal, email, teléfono y tipo de patrocinador (oro, plata o bronce) (obligatorio).
- Financiaciones que realiza.
- Actividades que financia.

para tener una agenda de los patrocinadores que financian las actividades.

#### RI – 9 Información de patrocinios

Como coordinador,

quiero disponer de la siguiente información sobre los patrocinios obtenidos:

- Cantidad (obligatorio).

**para** llevar la contabilidad de los recursos monetarios con los que cuenta cada proyecto, valorar costes y plantear las tasas de inscripción óptimas para cada actividad.

### RI - 10 Información de recibos

Como coordinador,

quiero disponer de la siguiente información sobre los recibos:

- Estado de la factura (pagada, pendiente o anulada) (obligatorio), fecha de emisión (obligatorio), fecha de pago, fecha de vencimiento (obligatorio), concepto (obligatorio) e importe (obligatorio).

para llevar la contabilidad de los gastos que generan las actividades, el nivel de cobertura de las diferentes financiaciones, así como tener las cuentas de inscripciones de participación actualizadas.

## RI – 11 Información sobre mensajes

Como coordinador,

**quiero** disponer de la siguiente información sobre los mensajes que se emiten:

- Identificador, tipo de mensaje (email, newsletter o informe), fecha de envío, asunto, contenido (todos los atributos son obligatorios).

para contar con una relación completa de todas las comunicaciones que inicia la fundación.

## RI – 12 Información sobre cuestionarios

Como coordinador,

quiero disponer de la siguiente información sobre los cuestionarios:

- Identificador (obligatorio), fecha.
- Actividad que valora.
- Coordinador que lo elabora.
- Participantes y voluntarios que lo responden.
- Preguntas y respuestas que lo componen.

para tener datos suficientes con los que valorar el desarrollo y la ejecución de las actividades, mejorando la calidad de los correspondientes informes que se le envían a los patrocinadores y permitiendo reflejar la realidad de cada actividad en la memoria anual.

## Reglas de negocio y pruebas de aceptación

## RN - 1 Prioridad de participación 1

Como coordinador,

**quiero** que los participantes que no han participado nunca en una actividad determinada tengan la mayor prioridad a la hora de ser seleccionados en esta,

para dar la oportunidad de participación a todos los miembros de la fundación que lo deseen.

#### PA - 1

- PA 1.1: se inscribe a un participante que ya ha participado en una edición anterior de la actividad propuesta y se le asigna prioridad de participación baja.
- **PA 1.2**: se inscribe a un participante que no ha participado nunca en la actividad propuesta y se le asigna prioridad de participación alta.

## RN – 2 Prioridad de participación 2

Como coordinador,

**quiero** que los participantes con mayor tiempo sin participar en ediciones pasadas de una actividad determinada tengan prioridad media para ser seleccionados como participantes en una nueva edición de esta,

para dar la oportunidad de participación a todos los miembros de la fundación que lo deseen.

## PA - 2

- **PA 2.1**: se inscribe a un participante que lleva mucho tiempo sin participar en una determinada actividad y se le asigna prioridad de participación media.
- **PA 2.2**: se inscribe a un participante que ha participado recientemente en una determinada actividad y se le asigna prioridad de participación baja.

## RN - 3 Pagos pendientes

Como coordinador,

**quiero** que un participante no sea seleccionado para realizar una actividad futura si tiene pendiente de pago algún recibo,

para evitar el impago de actividades y cumplir con los objetivos económicos.

## PA - 3

- **PA 3.1**: se intenta inscribir en una actividad a un participante que tiene algún recibo pendiente de pago y el sistema no permite la inscripción.
- **PA 3.2**: se intenta inscribir en una actividad a un participante que no tiene ningún recibo pendiente de pago y el sistema permite la inscripción.

## RN - 4 Tutor legal

Como coordinador,

**quiero** que todos los participantes menores de edad o con un grado de discapacidad superior al 50% tengan asignado un tutor legal,

para cumplir con la normativa vigente de la fundación.

#### PA - 4

- **PA 4.1**: se intenta inscribir a un participante menor de edad sin especificar los datos del representante legal y el sistema no permite la inscripción.
- **PA 4.2**: se intenta inscribir a un participante menor de edad especificando los datos del representante legal y el sistema permite la inscripción.
- **PA 4.3**: se intenta inscribir a un participante con un grado de discapacidad mayor al 50% sin especificar los datos del representante legal y el sistema no permite la inscripción.
- **PA 4.4**: se intenta inscribir a un participante con un grado de discapacidad mayor al 50% especificando los datos del tutor legal y el sistema permite la inscripción.

## RN - 5 Información médica

Como coordinador,

**quiero** que no se pueda inscribir a un participante en una actividad sin tener asociado, al menos, un informe médico y se especifique su grado de discapacidad,

**para** tener la posibilidad de conocer correctamente su diagnóstico médico y conocer sus principales necesidades asistenciales.

#### PA - 5

- **PA 5.1**: se intenta inscribir a un participante que no facilita informe médico alguno ni especifica su grado de discapacidad y el sistema no permite la inscripción.
- **PA 5.2**: se intenta inscribir a un participante que no facilita informe médico alguno, pero especifica su grado de discapacidad y el sistema no permite la inscripción.
- **PA 5.3**: se intenta inscribir a un participante que facilita informe médico, pero no especifica su grado de discapacidad y el sistema no permite la inscripción.
- **PA 5.4**: se intenta inscribir a un participante que facilita informe médico y especifica su grado de discapacidad y el sistema permite la inscripción.

#### RN - 6 Prioridad de voluntariado

Como coordinador,

**quiero** que los voluntarios que han participado en alguna edición anterior de una actividad determinada tengan prioridad de participación alta a la hora de ser seleccionados,

para contar con personas preparadas y con experiencia previa que ofrezcan el mejor servicio.

## PA - 6

- **PA 6.1**: se inscribe a un voluntario que ya ha participado en una edición anterior de la actividad propuesta y se le asigna prioridad de participación alta.
- PA 6.2: se inscribe a un voluntario que no ha participado en una edición anterior de la actividad propuesta pero sí en otras actividades y se le asigna prioridad de participación media.
- **PA 6.3**: se inscribe a un voluntario que no ha participado en ninguna actividad anteriormente y se le asigna prioridad de participación baja.

#### RN - 7 Estado de interés en actividades

Como coordinador,

**quiero** que el estado de interés por defecto de los usuarios en una nueva actividad sea cero, **para** que puedan, de manera personal, cambiar e indicar su disposición a participar.

## PA - 7

- **PA 7.1**: en el sistema se inserta una nueva actividad. El sistema registra que los usuarios no tienen interés por participar en dicha actividad.
- **PA 7.2**: en el sistema se inserta una nueva actividad. El sistema no registra que los usuarios no tienen interés por participar en dicha actividad.

## RN - 8 Envío de mensajes

Como coordinador,

quiero que sólo los patrocinadores puedan recibir informes,

**para** mejorar la seguridad de los datos de mayor grado de confidencialidad, canalizando de forma efectiva las comunicaciones entre los diferentes tipos de usuarios.

#### PA - 8

- PA 8.1: un coordinador envía un mensaje de tipo newsletter a un grupo de correos de participantes, voluntarios, tutores legales o patrocinadores y el sistema permite el envío.
- **PA 8.2**: un coordinador envía un mensaje de tipo informe a un participante, voluntario o tutor legal y el sistema no permite el envío.
- PA − 8.3: un coordinador envía un mensaje de tipo informe a un patrocinador y el sistema permite el envío.

## RN – 9 Tipo de patrocinador

Como coordinador,

quiero que se clasifique a los patrocinadores según la financiación que aportan:

- Si sus aportaciones cubren el 100% de los costes de un programa deportivo o un evento, será patrocinador de tipo ORO.
- Si sus aportaciones cubren parcialmente los costes de un programa deportivo, será patrocinador de tipo PLATA.
- Si sus aportaciones cubren parcialmente los costes de un evento, será patrocinador de tipo BRONCE.

**para** llevar a cabo una diferenciación entre los patrocinadores y promover entre estos la actitud solidaria a través del reconocimiento.

#### PA - 9

- PA 9.1: un patrocinador cubre el 100% de los costes de un programa deportivo o evento y el sistema lo clasifica como patrocinador ORO.
- PA 9.2: un patrocinador cubre parcialmente los costes de un programa deportivo y el sistema lo clasifica como patrocinador PLATA.
- **PA 9.3**: un patrocinador cubre parcialmente los costes de un evento y el sistema lo clasifica como patrocinador BRONCE.

#### RN – 10 Información obligatoria sobre donantes

Como coordinador,

**quiero** que las donaciones sólo se puedan registrar si están relacionadas con una persona o institución, quien figurará como donante,

para tener una relación exacta de donantes y donaciones.

#### PA - 10

- **PA 10.1**: se intenta registrar una donación sin especificar a una persona o institución como donante y el sistema no permite el registro.
- **PA 10.2**: se intenta registrar una donación especificando a una persona o institución como donante y el sistema permite el registro.

#### RN – 11 Coste de inscripciones

Como coordinador,

**quiero** que el sistema especifique el coste de inscripción de una actividad si la financiación de los patrocinios no alcanza a cubrir el 100% de los costes totales de esta,

para asegurar que se pueden llevar a cabo todas las actividades que se planifican.

## **PA - 11**

- **PA 12.1**: se registra una actividad cuyos costes totales no se cubren con la financiación de los patrocinios y el sistema especifica el coste de la inscripción.
- **PA 12.2**: se registra una actividad cuyos costes totales se cubren en su totalidad con la financiación de los patrocinios y el sistema especifica el coste cero de la inscripción.

### RN – 12 Plazas de inscripciones

Como coordinador,

**quiero** que se impida registrar en una actividad a más participantes de los que permite el número de plazas de la actividad,

**para** asegurar que se registra al número correcto de participantes y que se podrá realizar la actividad de manera correcta, contando con la cobertura asistencial correcta de profesionales y de voluntarios.

## PA - 12

- PA 13.1: se intenta registrar a un participante en una actividad que ya cuenta con tantos participantes registrados como plazas disponibles, y el sistema no permite el registro.
- PA 13.2: se intenta registrar a un participante en una actividad que cuenta con menos participantes registrados que plazas disponibles, y el sistema permite el registro.

## RN – 13 Plazo de respuesta de cuestionarios

Como coordinador,

**quiero** que las respuestas de los cuestionarios de una actividad sólo puedan registrarse durante los 15 días posteriores a la fecha de creación del cuestionario,

**para** poder realizar la evaluación de los proyectos correctamente y elaborar informes, así como contar con datos para la memoria anual.

## PA - 13

- **PA 14.1:** se intenta registrar un cuestionario de una actividad por un voluntario o participante pasados 15 días de la fecha de finalización de esta y el sistema no permite el registro.
- **PA 14.2:** se intenta registrar un cuestionario de una actividad por un voluntario o participante antes de los 15 días de la fecha de finalización de esta y el sistema permite el registro.

## RN – 14 Grado de discapacidad

Como coordinador,

quiero que el grado de discapacidad esté representado por un real entre 0 y 1.

## **PA - 14**

- **PA 15.1**: se inserta un número real entre 0 y 1, como grado de discapacidad, y el sistema permite el registro de dicho dato.
- **PA 15.2**: se inserta un número real menor que 0 o mayor que 1, como grado de discapacidad, y el sistema no permite el registro de dicho dato.

## RN – 15 Voluntarios requeridos

Como coordinador,

**quiero** que el sistema calcule el número de voluntarios requeridos para una actividad conforme se van inscribiendo participantes en esta,

para contar con el número exacto de voluntarios.

## **PA - 15**

- **PA 15.1**: se inscribe a un participante en una actividad y el número de voluntarios requeridos de la misma se incrementa en uno.
- **PA 15.2**: se inscribe a un participante en una actividad y el número de voluntarios requeridos de la misma no se incrementa.

## Requisitos funcionales

#### RF - 1 Fichas de usuarios

Como coordinador,

**quiero** obtener fichas con los datos de los participantes, voluntarios y patrocinadores, **para** estudiar y analizar sus datos.

#### RF - 2 Valoración de actividades

Como coordinador,

**quiero** recibir los cuestionarios realizados por voluntarios y participantes de una actividad, **para** poder realizar la evaluación de los proyectos correspondientes.

#### RF - 3 Gestión de mensajes

Como coordinador,

**quiero** gestionar el envío de diferentes tipos de comunicaciones según las condiciones o características de los receptores, ya sean participantes, voluntarios, tutores legales y/o patrocinadores,

para que cada usuario reciba notificaciones personalizadas y acordes a su perfil.

## RF – 4 Registro de informes médicos

Como coordinador,

**quiero** gestionar la inserción de informes médicos en el sistema, asociados a un participante, **para** disponer de información de primera necesidad sobre la discapacidad del participante.

## RF - 5 Generación de recibos

Como coordinador,

**quiero** que el sistema genere los recibos correspondientes a las inscripciones de los participantes,

para gestionar de forma correcta los pagos de las inscripciones de las actividades.

#### RF - 6 Lista de donantes

Como coordinador,

**quiero** que el sistema genere una lista de donantes, tanto particulares como institucionales, **para** saber quiénes son los usuarios del sistema que han realizado donaciones y cuántas.

## RF - 7 Listas de distribución de correo

Como coordinador,

**quiero** obtener una lista de distribución de correos electrónicos en función del grupo de destinatarios, el cual pueda ser 'voluntarios', 'participantes', 'tutores' o 'patrocinadores', **para** mandar mensajes de manera selectiva.

#### RF - 8 Informe de voluntarios

Como coordinador,

**quiero** tener una lista con todos los voluntarios que participan en una actividad, **para** controlar el voluntariado.

#### RF - 9 Historial de voluntariado

Como coordinador,

**quiero** tener un historial de las actividades en las que ha participado cada voluntario, **para** controlar el voluntariado.

#### RF – 10 Informe de participantes

Como coordinador,

**quiero** tener una lista con todos los participantes que intervienen en una actividad, **para** saber quiénes son los inscritos en cada actividad.

## RF - 11 Historial de participantes

Como coordinador,

**quiero** tener un historial de las actividades en las que ha intervenido cada participante, **para** controlar su participación.

## RF – 12 Informe de patrocinios

Como coordinador,

**quiero** tener una lista con todos los patrocinadores que financian una actividad, **para** controlar las financiaciones.

## RF - 13 Informe de donaciones

Como coordinador,

**quiero** tener una lista con todas las donaciones recibidas durante un periodo de tiempo, **para** controlar los bienes materiales y no materiales que recibe la fundación.

#### RF - 14 Informe de actividades

Como coordinador,

**quiero** obtener un informe detallado con todas las actividades que se han llevado a cabo en la fundación durante un periodo de tiempo,

para llevar un control sobre todas ellas.

## RF – 15 Disponibilidad en actividades

Como usuario,

**quiero** que el sistema me permita determinar mi disponibilidad de participación en las próximas actividades ofertadas,

para mostrar cuáles son de mi interés.

## RF - 16 Registro de usuarios en el sistema

Como coordinador,

**quiero** gestionar la inserción de participantes, voluntarios, tutores legales e instituciones en el sistema de información,

para contar con una relación completa de los datos de los usuarios.

#### RF - 17 Actualización de usuarios en el sistema

Como coordinador,

**quiero** gestionar la actualización de participantes, voluntarios, tutores legales e instituciones en el sistema de información,

para contar con una relación completa y actualizada de los datos de los usuarios.

## RF - 18 Eliminación de usuarios en el sistema

Como coordinador,

**quiero** gestionar la eliminación de participantes, voluntarios, tutores legales e instituciones en el sistema de información,

para contar con una relación completa y exacta de los datos de los usuarios.

## RF – 19 Registro de proyectos y actividades

Como coordinador,

**quiero** que el sistema permita la gestión de la inserción de proyectos, así como añadirles actividades a estos,

para poder gestionar de forma correcta la oferta de actividades entre los usuarios.

## Requisitos no funcionales

## RNF - 1 Temporización

El sistema, con una concurrencia máxima de 100 usuarios, debe dar una respuesta en un tiempo menor a 2 segundos.

## RNF - 2 Disponibilidad

El sistema debe estar disponible para su uso las 24 horas del día durante todo el año.

## RNF - 3 Compatibilidad

El sistema debe ser compatible con los navegadores Mozilla Firefox hasta la versión 63.0.1, Google Chrome hasta la versión 70.0.3538.77 y Microsoft Edge hasta la versión 42.17134.1.0.

## RNF – 4 Seguridad

El sistema debe contar con un mecanismo de autenticación de usuarios por nombre de usuario y contraseña para asegurar el acceso y la seguridad a los datos.

## RNF - 5 Usabilidad I

El sistema debe ser accesible para los usuarios desde navegadores web tanto de dispositivos móviles como de ordenadores portátiles y de sobremesa.

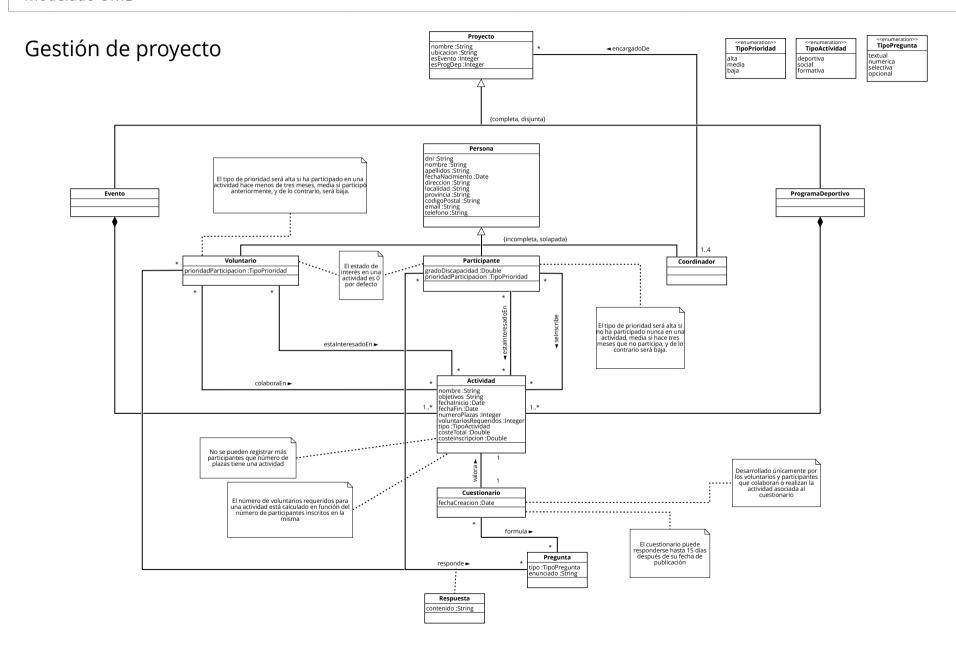
## RNF - 6 Usabilidad II

El sistema debe dar acceso a cualquier punto de la aplicación en un máximo de tres clics.

## 5. MODELO CONCEPTUAL

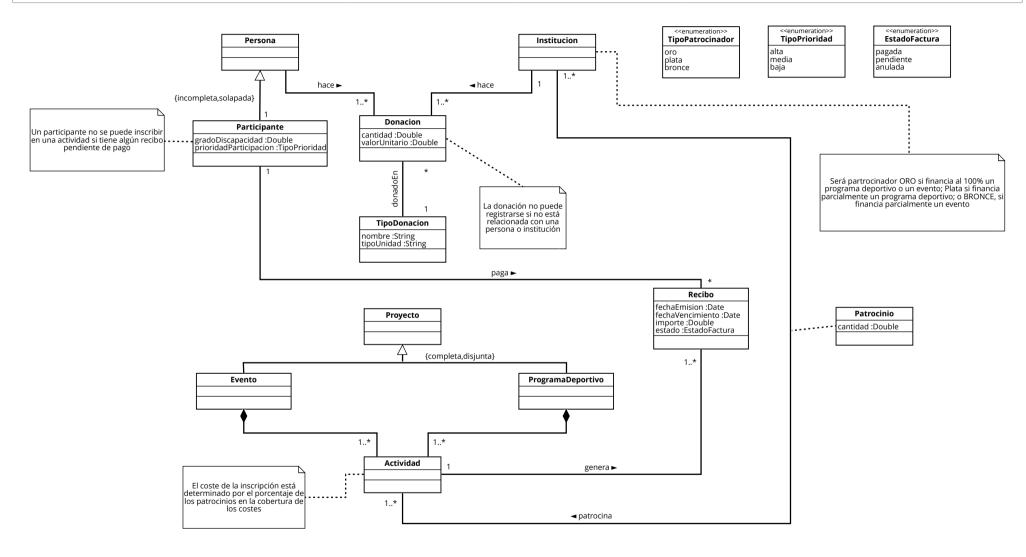
Modelado UML





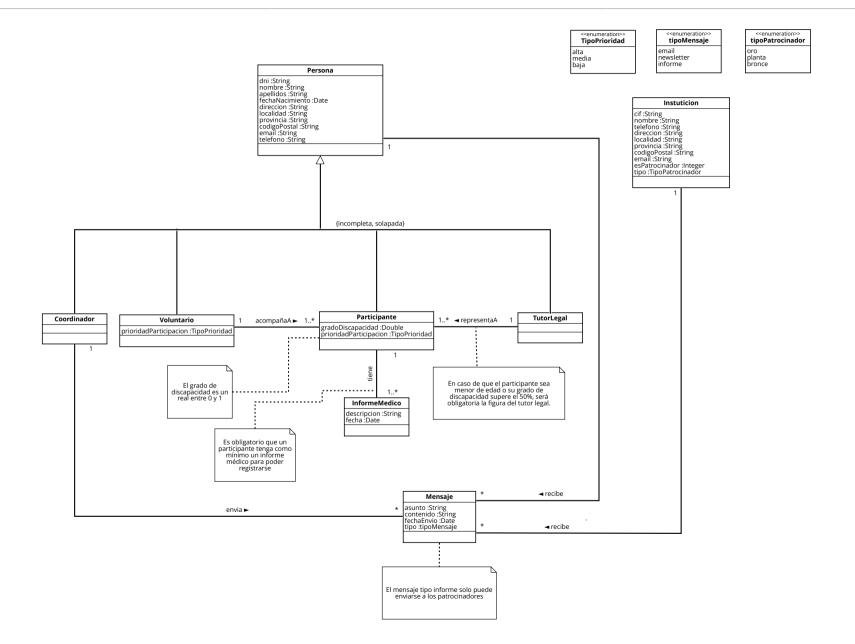
# Gestión de cuentas





# Gestión de comunicaciones

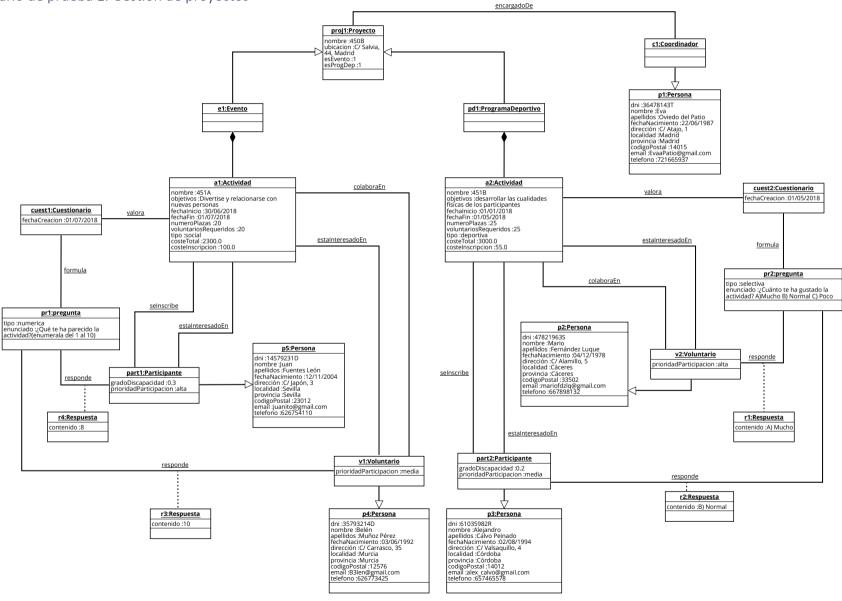




## Escenarios de prueba







## Descripción textual

En este escenario de prueba definimos 5 personas (p1, p2, p3, p4, p5), siendo p1, coordinador (c1), p4 y p2, voluntarios (v1, v2) y p5 y p3, participantes (part1, part2).

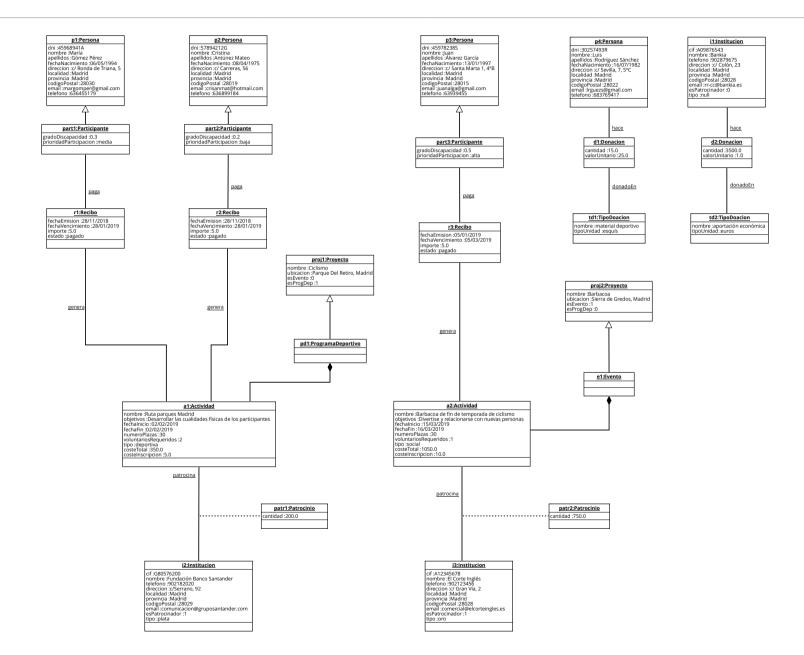
Definimos también un proyecto (proj1), el cual consiste en un evento, formado por una actividad (a1), y un programa deportivo (pd1), formado por otra actividad (a2).

El participante 1 y el voluntario 1 participan y colaboran respectivamente en la actividad 1, y cada uno la valora, respondiendo un cuestionario (cuest1), que contiene la pregunta pr1 de tipo numérica.

De la misma forma, el participante 2 y el voluntario 2 participan y colaboran respectivamente en la actividad 2, y cada uno la valora, respondiendo un cuestionario (cuest2), que contiene la pregunta pr2 de tipo selectiva.

## Escenario de prueba 2. Gestión de cuentas





## Descripción textual

En este escenario de prueba definimos 4 personas (p1, p2, p3, p4), siendo 3 de ellas participantes (part1, part2, part3).

Los participantes pagan los recibos (r1, r2, r3) de las actividades (a1, a2) que van a realizar respectivamente.

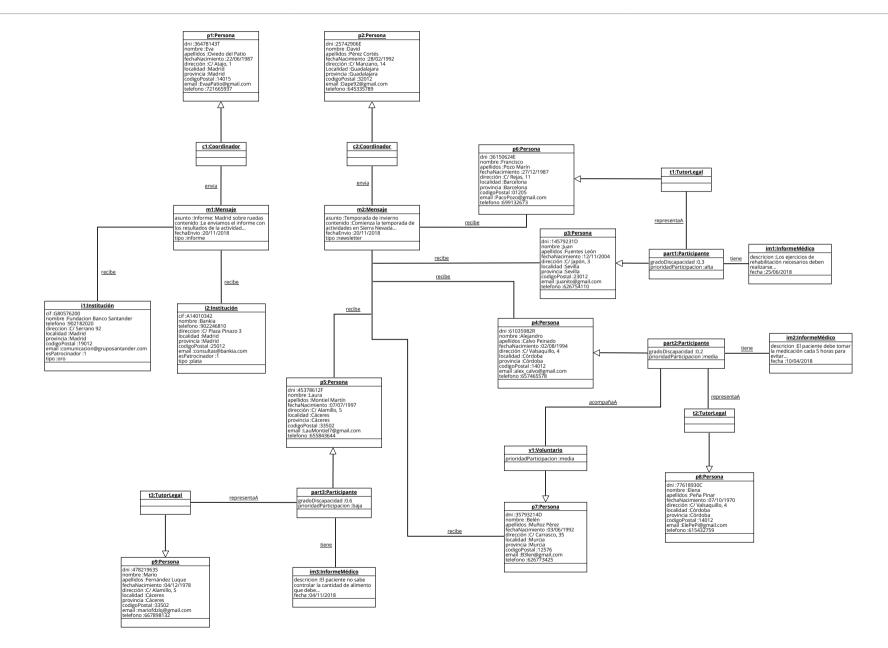
La actividad 1, (a1) forma parte de un programa deportivo (pd1), el cuál hereda del proyecto 1 (proj1), y del mismo modo, la actividad 2 (a2) forma parte de un evento (e1), el cuál hereda del proyecto 2 (proj2).

La persona 4 (p4) realiza una donación (d1) de material deportivo (td1) a la fundación.

Definimos también 3 instituciones (i1, i2, i3), de las cuáles 2 (i2, i3) van a ser patrocinadores de las actividades, de forma que la institución 2 (i2) va a financiar la actividad 1, mientras que la institución 3 (i3) va a financiar la actividad 2. La institución restante (i1) en este caso actúa como donante (correspondiente a la donación d2), la cual realizará una aportación económica (td2).

## Escenario de prueba 3. Gestión de comunicaciones





## Descripción textual

En este escenario de prueba definimos 2 mensajes (m1, m2).

Un coordinador (c1), que hereda de p1, envía m1, este mensaje es recibido por dos instituciones (i1, i2).

El m2 es enviado por otro coordinador (c2), que hereda de p2, este mensaje es recibido por todas las personas excepto p1 y p2, (p3, p4, p5) son participantes (part1, part2, part3) respectivamente,

(p6, p8, p9) son respectivamente los siguientes tutores legales (t1, t2, t3) y por último también recibe el mensaje p7 que se corresponde con el voluntario v1.

Los tutores legales (t1, t2, t3) representan respectivamente a los participantes (part1, part2, part3).

El voluntario (v1) acompaña al participante (part2).

Los participantes (part1, part2, part3) tienen un informe médico que, respectivamente, corresponden a (im1, im2, im3).

# 6. MATRICES DE TRAZABILIDAD

# Matriz de clases de entidad y requisitos de información

C RI	RI-1	RI-2	RI-3	RI-4	RI-5	RI-6	RI-7	RI-8	RI-9	RI-10	RI-11	RI-12
Actividad					<b>~</b>							
Coordinador	<b>&gt;</b>											
Cuestionario												~
Donación							<b>✓</b>					
Evento				~								
Informe Médico		~										
Institución								~				
Mensaje											~	
Participante		~										
Patrocinador								<b>✓</b>				
Patrocinio									~			
Persona	<b>&gt;</b>	~	~			~	~					
Pregunta												~
Programa Deportivo				<b>✓</b>								
Proyecto				~								
Recibo										~		
Respuesta												~
Tutor Legal						~						
Voluntario	_		~									

# Matriz de clases de entidad y reglas de negocio

C RN	RN-1	RN-2	RN-3	RN-4	RN-5	RN-6	RN-7	RN-8	RN-9	RN-10	RN-11	RN-12	RN-13	RN-14	RN-15
Actividad	<b>*</b>	~					~		<b>~</b>		~	~			~
Coordinador								~							
Cuestionario													~		
Donación										<b>~</b>					
Evento	<b>~</b>	~					~								
Informe Médico					~										
Institución								~		<b>~</b>					
Mensaje								~							
Participante	<b>✓</b>	~	~	~	~		~							<b>✓</b>	
Patrocinador									~						
Patrocinio									~						
Persona								~		<b>✓</b>					
Pregunta													~		
Programa Deportivo	<b>✓</b>	~					~								
Proyecto									~						
Recibo			~								~				
Respuesta													~		
Tutor Legal				~											
Voluntario						<b>✓</b>	~								~

# Matriz de asociaciones y requisitos de información

A RI	RI-1	RI-2	RI-3	RI-4	RI-5	RI-6	RI-7	RI-8	RI-9	RI-10	RI-11	RI-12
acompañaA		<b>~</b>	<b>✓</b>									
colaboraEn			<b>✓</b>		<b>✓</b>							
donadoEn							~					
encargadoDe	<b>~</b>			~								
envía	<b>~</b>										~	
estaInteresadoEn		~	~		~							
formula												~
genera					<b>✓</b>					~		
hace							~					
paga		<b>✓</b>								~		
patrocina					<b>✓</b>			~	<b>~</b>			
recibe		~	~			~	~	<b>✓</b>			~	
representaA		<b>✓</b>				<b>✓</b>						
responde		<b>✓</b>	~									~
seInscribe		~			~							
tiene		<b>✓</b>										
valora					~							~

# Matriz de asociaciones y reglas de negocios

A RN	RN-1	RN-2	RN-3	RN-4	RN-5	RN-6	RN-7	RN-8	RN-9	RN-10	RN-11	RN-12	RN-13	RN-14	RN-15
acompañaA						<b>~</b>									
colaboraEn						~									<b>~</b>
donadoEn										~					
encargadoDe								~							
envía								~							
estaInteresadoEn							~								
formula													<b>✓</b>		
genera			~								<b>✓</b>				
hace										~					
paga			~												
patrocina									<b>~</b>						
recibe								~							
representaA				<b>~</b>											
responde													<b>~</b>		
seInscribe	<b>~</b>	<b>✓</b>										<b>✓</b>			
tiene					<b>✓</b>									<b>✓</b>	
valora													>		

# Matriz de restricciones y requisitos de información

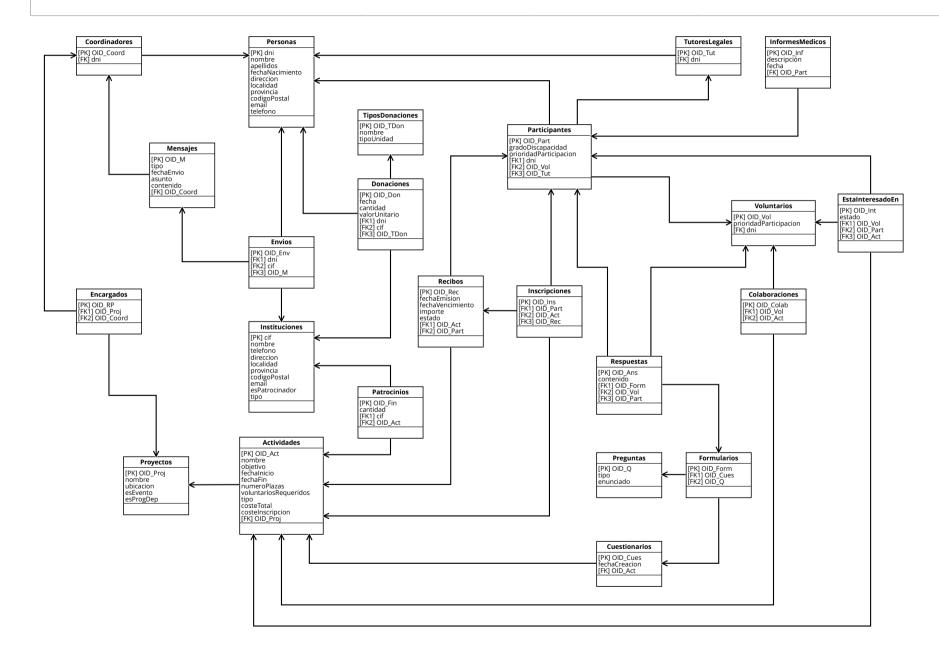
R RI	RI-1	RI-2	RI-3	RI-4	RI-5	RI-6	RI-7	RI-8	RI-9	RI-10	RI-11	RI-12
El mensaje tipo informe solo puede enviarse a los patrocinadores.	~										~	
Si el participante es menor de edad o tiene más de 0.5 de grado de discapacidad debe tener un tutor legal.		~				~						
No puede registrarse a un participante que no tenga al menos un informe médico.		~										
El grado de discapacidad es un real entre 0 y 1.		~										
Un participante no se puede inscribir en una actividad si tiene alguna factura pendiente de pago.		~								~		
La donación no puede registrarse si no está relacionada con una persona o una institución.							<b>&gt;</b>					
El coste de la inscripción está determinado por el porcentaje de los patrocinios en la cobertura de los costes.				<b>*</b>	<b>~</b>				>			
Un patrocinador ORO es aquel que financia al 100% un programa deportivo o un evento.								<b>~</b>				
Un patrocinador PLATA es aquel que financia parcialmente un programa deportivo.								>				
Un patrocinador BRONCE es aquel que financia parcialmente un evento.								<b>&gt;</b>				
No se pueden registrar más participantes que número de plazas tiene una actividad.				<b>*</b>	~							
Los cuestionarios solo los pueden responder participantes y voluntarios.												~
El cuestionario puede responderse hasta 15 días después de su fecha de publicación.												<b>~</b>
Prioridad de participación será alta si nunca ha participado antes, media se hace tres meses que no participa, de lo contrario será baja.		<b>*</b>										
Prioridad de voluntariado será alta si ha participado en menos de tres meses, media si ha participado anteriormente, de lo contrario será baja.			<b>~</b>									
El número de voluntarios requeridos para una actividad está calculado en función del número de participantes inscritos en la misma.				<b>*</b>	~		_					
El estado de interés en una actividad es 0 por defecto.		~	~									

# Matriz de restricciones y reglas de negocio

R RN	RN-1	RN-2	RN-3	RN-4	RN-5	RN-6	RN-7	RN-8	RN-9	RN-10	RN-11	RN-12	RN-13	RN-14	RN-15
El mensaje tipo informe solo puede enviarse a los patrocinadores.								~							
Si el participante es menor de edad o tiene más de 0.5 de grado de discapacidad debe tener un tutor legal.				~											
No puede registrarse a un participante que no tenga al menos un informe médico.					>										
El grado de discapacidad es un real entre 0 y 1.														~	
Un participante no se puede inscribir en una actividad si tiene alguna factura pendiente de pago.			>												
La donación no puede registrarse si no está relacionada con una persona o una institución.										~					
El coste de la inscripción está determinado por el porcentaje de los patrocinios en la cobertura de los costes.											~				
Un patrocinador ORO es aquel que financia al 100% un programa deportivo o un evento.									<b>&gt;</b>						
Un patrocinador PLATA es aquel que financia parcialmente un programa deportivo.									<b>&gt;</b>						
Un patrocinador BRONCE es aquel que financia parcialmente un evento.									~						
No se pueden registrar más participantes que número de plazas tiene una actividad.												<b>~</b>			
Los cuestionarios solo los pueden responder participantes y voluntarios.													<b>&gt;</b>		
El cuestionario puede responderse hasta 15 días después de su fecha de publicación.		>											>		
Prioridad de participación será alta si nunca ha participado antes, media se hace tres meses que no participa, de lo contrario será baja.	<b>*</b>														
Prioridad de voluntariado será alta si ha participado en menos de tres meses, media si ha participado anteriormente, de lo contrario será baja.						>									
El número de voluntarios requeridos para una actividad está calculado en función del número de participantes inscritos en la misma.															<b>*</b>
El estado de interés en una actividad es 0 por defecto.							~								

## 7. MODELO RELACIONAL





Tablas	Atributos	Primary Keys	Foreign Keys
Actividades	OID_Act, nombre, objetivo, fechalnicio, fechaFin, numeroPlazas, Voluntarios requeridos, tipo, costeTotal, costeInscripcion, OID_Proj	OID_Act	OID_Proj / Proyectos
Colaboraciones	OID_Colab, OID_VoI, OID_Act	OID_Colab	OID_Vol / Voluntarios, OID_Act / Actividades
Coordinadores	OID_Coord, dni	OID_Coord	dni / Personas
Cuestionarios	OID_Cues, fechaCreacion, OID_Act	OID_Cues	OID_Act / Actividades
Donaciones	OID_Don, fecha, cantidad, valorUnitario, dni, cif, OID_TDon	OID_Don	dni / Personas, cif / Instituciones, OID_TDon / TiposDonaciones
Encargados	OID_RP, OID_Proj, OID_Coord	OID_RP	OID_Proj / Proyectos, OID_Coord / Coordinadores
Envios	OID_Env, dni, cif, OID_M	OID_Env	dni / Personas, cif / Instituciones, OID_M / Mensajes
EstaInteresadoEn	OID_Int, estado, OID_Vol, OID_Part, OID_Act	OID_Int	OID_Vol / Voluntarios, OID_Part / Participantes, OID_Act / Actividades
Formularios	OID_Form, OID_Cues, OID_Q	OID_Form	OID_Cues / Cuestionarios, OID_Q / Preguntas
InformesMedicos	OID_Inf, descripcion, fecha, OID_Part	OID_Inf	OID_Part / Participantes
Inscripciones	OID_Ins, OID_Part, OID_Act, OID_Rec	OID_Ins	OID_Part / Participantes, OID_Act / Actividades, OID_Rec / Recibos
Instituciones	cif, nombre, telefono, direccion, localidad, provincia, codigoPostal, email	cif	
Mensajes	OID_M, tipo, fechalnicio, asunto, contenido, OID_Coord	OID_M	OID_Coord / Coordinadores
Participantes	OID_Part, gradoDiscapacidad, prioridadParticipacion, dni, OID_Vol, OID_Tut	OID_Part	dni / Personas, OID_Vol / Voluntarios, OID_Tut / TutoresLegales
Patrocinios	OID_Fin, cantidad, OID_Patr, OID_Act	OID_Fin	cif / Instituciones, OID_Act / Actividades
Personas	dni, nombre, apellidos, fechaNacimiento, direccion, localidad, provincia, codigoPostal, email, telefono	dni	
Preguntas	OID_Q, tipo, enunciado	OID_Q	
Proyectos	OID_Proj, nombre, ubicación, esEvento, esProgDep	OID_Proj	
Recibos	OID_Rec, fechaEmision, fechaVencimiento, importe, estado, OID_Act, OID_Part	OID_Rec	OID_Act / Actividades, OID_Part / Participantes
Respuestas	OID_Ans, contenido, OID_Form, OID_Vol, OID_Part	OID_Ans	OID_Form / Formularios, OID_Vol / Voluntarios, OID_Part / Participantes
TiposDonaciones	OID_TDon, nombre, tipoUnidad	OID_TDon	
TutoresLegales	OID_Tut, dni	OID_Tut	dni / Personas
Voluntarios	OID_Vol, prioridadParticipacion, dni	OID_Vol	dni / Personas

## Justificación de estrategias de transformación de jerarquías

### Jerarquía de clasificación de Persona.

De la entidad Persona heredan Participante, Voluntario, Coordinador y Tutor Legal. Además, el sistema de información almacena datos de personas que no se clasifican en ninguna de las subclases anteriores, luego se considera una clasificación incompleta y solapada. De esta manera, y debido al amplio número de entidades en esta jerarquía, se ha optado por considerar una relación por cada entidad.

### • Jerarquía de clasificación de Institución.

De la entidad Institución sólo hereda Patrocinador. Además, el sistema de información almacena datos de instituciones que no se clasifican como patrocinadoras, por lo que se considera una clasificación incompleta y solapada. Debido a que la subclase Patrocinador sólo pretende identificar qué instituciones registradas en el sistema de información han patrocinado alguna actividad de la fundación, así como el tipo de patrocinador con el que se le identifica, se ha optado por considerar una única relación para toda la jerarquía con indicador booleano (0 como false, 1 como true) para la subclase.

### • Jerarquía de clasificación de Proyecto.

De la entidad Proyecto heredan Evento y Programa Deportivo. Al ser sólo dos subclases que no cuentan con atributos propios, sino que simplemente diferencian el tipo de proyecto, se ha optado por una relación única para toda la jerarquía con indicador booleano (0 como false, 1 como true) para la subclase.

# 8. MODELO TECNOLÓGICO

# Tablas, secuencias y triggers de secuencias

```
Archivo: 1_tablas_IS-G1-SSR_dep&des.sql
DROP TABLE RESPUESTAS;
DROP TABLE FORMULARIOS;
DROP TABLE CUESTIONARIOS;
DROP TABLE PREGUNTAS;
DROP TABLE ENVIOS;
DROP TABLE MENSAJES;
DROP TABLE DONACIONES;
DROP TABLE TIPODONACIONES;
DROP TABLE PATROCINIOS;
DROP TABLE INSCRIPCIONES;
DROP TABLE RECIBOS;
DROP TABLE COLABORACIONES;
DROP TABLE ENCARGADOS;
DROP TABLE ESTAINTERESADOEN;
DROP TABLE ACTIVIDADES;
DROP TABLE PROYECTOS;
DROP TABLE INSTITUCIONES;
DROP TABLE INFORMESMEDICOS;
DROP TABLE PARTICIPANTES;
DROP TABLE VOLUNTARIOS;
DROP TABLE TUTORESLEGALES;
DROP TABLE COORDINADORES;
DROP TABLE PERSONAS;
-- Tabla de PERSONAS y su clasificación de herencia
CREATE TABLE PERSONAS (
   dni CHAR(9) NOT NULL
       nombre VARCHAR2(50) NOT NULL,
   apellidos VARCHAR2(50),
   fechaNacimiento DATE NOT NULL,
   direccion VARCHAR2(50),
   localidad VARCHAR2(50),
   provincia VARCHAR2(50),
   codigoPostal CHAR(5),
   email VARCHAR2(50),
   telefono CHAR(9) NOT NULL,
   PRIMARY KEY (dni)
);
```

```
CREATE TABLE COORDINADORES (
   OID Coord INTEGER PRIMARY KEY NOT NULL,
   dni CHAR(9) NOT NULL,
   FOREIGN KEY (dni) REFERENCES PERSONAS ON DELETE CASCADE
);
CREATE TABLE TUTORESLEGALES (
   OID Tut INTEGER PRIMARY KEY NOT NULL,
   dni CHAR(9) NOT NULL,
   FOREIGN KEY (dni) REFERENCES PERSONAS ON DELETE CASCADE
);
CREATE TABLE VOLUNTARIOS (
   OID Vol INTEGER PRIMARY KEY NOT NULL,
   dni CHAR(9) NOT NULL,
   prioridadParticipacion VARCHAR2(50)
       CONSTRAINT e_prioridad_Vol CHECK (prioridadParticipacion IN ('alta', 'media', 'baja')),
   FOREIGN KEY (dni) REFERENCES PERSONAS ON DELETE CASCADE
);
CREATE TABLE PARTICIPANTES (
   OID Part INTEGER PRIMARY KEY NOT NULL.
   dni CHAR(9) NOT NULL,
   gradoDiscapacidad NUMBER NOT NULL
       CONSTRAINT ck_gradoDiscapacidad CHECK (gradoDiscapacidad BETWEEN 0.0 AND 1.0),
   prioridadParticipacion VARCHAR2(50)
        CONSTRAINT e prioridad Part CHECK (prioridadParticipacion IN ('alta', 'media', 'baja')),
   OID Tut INTEGER,
   OID Vol INTEGER,
   FOREIGN KEY (dni) REFERENCES PERSONAS ON DELETE CASCADE,
   FOREIGN KEY (OID Vol) REFERENCES VOLUNTARIOS,
   FOREIGN KEY (OID Tut) REFERENCES TUTORESLEGALES
);
-- Tablas de información de referencia de PARTICIPANTE y VOLUNTARIOS
CREATE TABLE INFORMESMEDICOS (
   OID Inf INTEGER PRIMARY KEY NOT NULL,
   OID Part INTEGER NOT NULL,
   descripcion LONG NOT NULL,
   fecha DATE NOT NULL,
   FOREIGN KEY (OID_Part) REFERENCES PARTICIPANTES ON DELETE CASCADE
);
```

```
-- Tabla de INSTITUCIONES
CREATE TABLE INSTITUCIONES (
   cif CHAR(9) PRIMARY KEY NOT NULL
       nombre VARCHAR2(50) NOT NULL,
   telefono CHAR(9) NOT NULL,
   direccion VARCHAR2(50),
   localidad VARCHAR2(50),
   provincia VARCHAR2(50),
   codigoPostal CHAR(5),
   email VARCHAR2(50),
   esPatrocinador SMALLINT NOT NULL
       CONSTRAINT ck_esPatrocinador CHECK (esPatrocinador IN (0,1)),
   tipo VARCHAR2(50)
       CONSTRAINT enum tipoPatrocinador CHECK (tipo IN ('oro', 'plata', 'bronce'))
);
-- Tabla de PROYECTOS y ACTIVIDADES
CREATE TABLE PROYECTOS (
   OID_Proj INTEGER PRIMARY KEY NOT NULL,
   nombre VARCHAR2(50) NOT NULL.
   ubicacion VARCHAR2(50) NOT NULL,
   esEvento SMALLINT NOT NULL
       CONSTRAINT ck_esEvento CHECK (esEvento IN (0,1)),
   esProgDep SMALLINT NOT NULL
       CONSTRAINT ck esProgDep CHECK (esProgDep IN (0,1))
);
CREATE TABLE ACTIVIDADES (
   OID Act INTEGER PRIMARY KEY NOT NULL,
   OID Proj INTEGER NOT NULL,
   nombre VARCHAR2(50) NOT NULL.
   objetivo VARCHAR2(255) NOT NULL,
   fechaInicio DATE NOT NULL,
   fechaFin DATE NOT NULL,
   numeroPlazas INTEGER NOT NULL,
   voluntariosRequeridos INTEGER NOT NULL,
   tipo VARCHAR2(50) NOT NULL
       CONSTRAINT enum tipoActividad CHECK (tipo IN ('deportiva', 'formativa', 'social')),
   costeTotal NUMBER(*,2) NOT NULL,
   costeInscripcion NUMBER(*,2) NOT NULL,
   UNIQUE (OID Proj, nombre),
   FOREIGN KEY (OID_Proj) REFERENCES PROYECTOS ON DELETE CASCADE
);
```

```
CREATE TABLE ESTAINTERESADOEN (
   OID Int INTEGER PRIMARY KEY NOT NULL,
   OID Part INTEGER.
   OID Vol INTEGER,
   OID Act INTEGER NOT NULL.
   estado SMALLINT NOT NULL
       CONSTRAINT intSelector CHECK (estado IN (0,1)),
   FOREIGN KEY (OID Part) REFERENCES PARTICIPANTES ON DELETE CASCADE,
   FOREIGN KEY (OID Vol) REFERENCES VOLUNTARIOS ON DELETE CASCADE.
   FOREIGN KEY (OID Act) REFERENCES ACTIVIDADES ON DELETE CASCADE
);
CREATE TABLE ENCARGADOS (
   OID RP INTEGER PRIMARY KEY NOT NULL,
   OID Proj INTEGER NOT NULL,
   OID Coord INTEGER NOT NULL,
   FOREIGN KEY (OID Proj) REFERENCES PROYECTOS ON DELETE CASCADE,
   FOREIGN KEY (OID Coord) REFERENCES COORDINADORES
);
CREATE TABLE COLABORACIONES (
   OID_Colab INTEGER PRIMARY KEY NOT NULL,
   OID Vol INTEGER NOT NULL.
   OID Act INTEGER NOT NULL,
   FOREIGN KEY (OID Vol) REFERENCES VOLUNTARIOS ON DELETE CASCADE,
   FOREIGN KEY (OID Act) REFERENCES ACTIVIDADES ON DELETE CASCADE
);
CREATE TABLE RECIBOS (
   OID Rec INTEGER PRIMARY KEY NOT NULL,
   OID Act INTEGER NOT NULL,
   OID Part INTEGER,
   fechaEmision DATE NOT NULL,
   fechaVencimiento DATE NOT NULL.
   importe NUMBER(*,2) NOT NULL,
   estado VARCHAR2(50) NOT NULL
       CONSTRAINT enum estadoRecibo CHECK (estado IN ('pagado', 'pendiente', 'anulado')),
   FOREIGN KEY (OID Act) REFERENCES ACTIVIDADES,
   FOREIGN KEY (OID Part) REFERENCES PARTICIPANTES
);
CREATE TABLE INSCRIPCIONES (
   OID_Ins INTEGER PRIMARY KEY NOT NULL,
   OID Part INTEGER NOT NULL,
   OID Act INTEGER NOT NULL,
   OID Rec INTEGER,
   FOREIGN KEY (OID_Part) REFERENCES PARTICIPANTES ON DELETE CASCADE,
   FOREIGN KEY (OID Act) REFERENCES ACTIVIDADES ON DELETE CASCADE,
   FOREIGN KEY (OID Rec) REFERENCES RECIBOS
);
```

```
-- Tablas de PATROCINIOS y DONACIONES
CREATE TABLE PATROCINIOS (
   OID Fin INTEGER PRIMARY KEY NOT NULL.
   cif CHAR(9) NOT NULL,
   cantidad NUMBER(*,2) NOT NULL,
   OID Act INTEGER NOT NULL,
   FOREIGN KEY (cif) REFERENCES INSTITUCIONES,
   FOREIGN KEY (OID Act) REFERENCES ACTIVIDADES
);
CREATE TABLE TIPODONACIONES (
   OID_TDon INTEGER PRIMARY KEY NOT NULL,
   nombre VARCHAR2(50) NOT NULL,
   tipoUnidad VARCHAR2(50) NOT NULL
);
CREATE TABLE DONACIONES (
   OID Don INTEGER PRIMARY KEY NOT NULL,
   cif CHAR(9),
   dni CHAR(9),
   OID TDon INTEGER NOT NULL,
   fecha DATE NOT NULL,
   cantidad NUMBER(*,2) NOT NULL,
   valorUnitario NUMBER(*,2) NOT NULL,
   FOREIGN KEY (cif) REFERENCES INSTITUCIONES,
   FOREIGN KEY (dni) REFERENCES PERSONAS,
   FOREIGN KEY (OID TDon) REFERENCES TIPODONACIONES
);
-- Tablas de MENSAJES y ENVIOS
CREATE TABLE MENSAJES (
   OID_M INTEGER PRIMARY KEY NOT NULL,
   OID Coord INTEGER NOT NULL,
   tipo VARCHAR2(50) NOT NULL
        CONSTRAINT enum tipoMensaje CHECK (tipo IN ('email', 'newsletter', 'informe')),
   fechaEnvio DATE NOT NULL,
   asunto VARCHAR2(255) NOT NULL,
   contenido LONG NOT NULL,
   FOREIGN KEY (OID Coord) REFERENCES COORDINADORES
);
```

```
CREATE TABLE ENVIOS (
   OID Env INTEGER PRIMARY KEY NOT NULL,
   OID M INTEGER NOT NULL,
   dni CHAR(9),
   cif CHAR(9).
   FOREIGN KEY (OID M) REFERENCES MENSAJES,
   FOREIGN KEY (dni) REFERENCES PERSONAS,
   FOREIGN KEY (cif) REFERENCES INSTITUCIONES
);
-- Tablas de CUESTIONARIOS
CREATE TABLE PREGUNTAS (
   OID_Q INTEGER PRIMARY KEY NOT NULL,
   tipo VARCHAR2(50) NOT NULL
        CONSTRAINT enum_tipoPregunta CHECK (tipo IN ('textual', 'numerica', 'selectiva', 'opcional')),
   enunciado VARCHAR(255) NOT NULL
);
CREATE TABLE CUESTIONARIOS (
   OID Cues INTEGER PRIMARY KEY NOT NULL,
   OID_Act INTEGER NOT NULL,
   fechaCreacion DATE NOT NULL.
   FOREIGN KEY (OID_Act) REFERENCES ACTIVIDADES
);
CREATE TABLE FORMULARIOS (
   OID Form INTEGER PRIMARY KEY NOT NULL,
   OID Cues INTEGER NOT NULL,
   OID Q INTEGER NOT NULL,
   FOREIGN KEY (OID Cues) REFERENCES CUESTIONARIOS ON DELETE CASCADE,
   FOREIGN KEY (OID Q) REFERENCES PREGUNTAS
);
CREATE TABLE RESPUESTAS (
   OID Ans INTEGER PRIMARY KEY NOT NULL,
   OID Form INTEGER NOT NULL,
   OID Part INTEGER,
   OID Vol INTEGER,
   contenido VARCHAR2(1000) NOT NULL,
   FOREIGN KEY (OID Form) REFERENCES FORMULARIOS ON DELETE CASCADE,
   FOREIGN KEY (OID Part) REFERENCES PARTICIPANTES,
   FOREIGN KEY (OID_Vol) REFERENCES VOLUNTARIOS
);
```

-----

#### -- SECUENCIAS

-----

DROP SEQUENCE SEC Coord; CREATE SEQUENCE SEC Coord INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Tut; CREATE SEQUENCE SEC\_Tut INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Vol: CREATE SEQUENCE SEC Vol INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Part: CREATE SEQUENCE SEC Part INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Inf: CREATE SEQUENCE SEC Inf INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Proj; CREATE SEQUENCE SEC Proj INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Act; CREATE SEQUENCE SEC Act INCREMENT BY 1 START WITH 1; DROP SEOUENCE SEC Int: CREATE SEQUENCE SEC Int INCREMENT BY 1 START WITH 1; DROP SEOUENCE SEC RP: CREATE SEQUENCE SEC RP INCREMENT BY 1 START WITH 1; DROP SEOUENCE SEC Colab: CREATE SEQUENCE SEC Colab INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Rec; CREATE SEQUENCE SEC Rec INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Ins; CREATE SEQUENCE SEC Ins INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Fin; CREATE SEQUENCE SEC Fin INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC TDon; CREATE SEQUENCE SEC TDon INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Don; CREATE SEQUENCE SEC Don INCREMENT BY 1 START WITH 1: DROP SEQUENCE SEC M; CREATE SEQUENCE SEC M INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Env; CREATE SEQUENCE SEC Env INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC 0; CREATE SEQUENCE SEC Q INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC Cues; CREATE SEQUENCE SEC Cues INCREMENT BY 1 START WITH 1; DROP SEQUENCE SEC\_Form; CREATE SEQUENCE SEC Form INCREMENT BY 1 START WITH 1: DROP SEQUENCE SEC Ans; CREATE SEQUENCE SEC Ans INCREMENT BY 1 START WITH 1;

```
-- TRIGGERS ASOCIADOS A SECUENCIAS
______
CREATE OR REPLACE TRIGGER SECUENCIA COORDINADORES
BEFORE INSERT ON COORDINADORES
FOR EACH ROW
BEGIN
   SELECT SEC Coord.NEXTVAL INTO :NEW.OID Coord FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA TUTORESLEGALES
BEFORE INSERT ON TUTORESLEGALES
FOR EACH ROW
BEGIN
   SELECT SEC Tut.NEXTVAL INTO :NEW.OID Tut FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA VOLUNTARIOS
BEFORE INSERT ON VOLUNTARIOS
FOR EACH ROW
BEGIN
   SELECT SEC Vol.NEXTVAL INTO :NEW.OID Vol FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_PARTICIPANTES
BEFORE INSERT ON PARTICIPANTES
FOR EACH ROW
BEGIN
   SELECT SEC_Part.NEXTVAL INTO :NEW.OID_Part FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_INFORMESMEDICOS
BEFORE INSERT ON INFORMESMEDICOS
FOR EACH ROW
BEGIN
   SELECT SEC_Inf.NEXTVAL INTO :NEW.OID_Inf FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_PROYECTOS
BEFORE INSERT ON PROYECTOS
FOR EACH ROW
BEGIN
   SELECT SEC_Proj.NEXTVAL INTO : NEW.OID_Proj FROM DUAL;
END;
```

```
CREATE OR REPLACE TRIGGER SECUENCIA ACTIVIDADES
BEFORE INSERT ON ACTIVIDADES
FOR EACH ROW
BEGIN
   SELECT SEC_Act.NEXTVAL INTO :NEW.OID_Act FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA ESTAINTERESADOEN
BEFORE INSERT ON ESTAINTERESADOEN
FOR EACH ROW
BEGIN
   SELECT SEC_Int.NEXTVAL INTO :NEW.OID_Int FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_ENCARGADOS
BEFORE INSERT ON ENCARGADOS
FOR EACH ROW
BEGIN
   SELECT SEC_RP.NEXTVAL INTO :NEW.OID_RP FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_COLABORACIONES
BEFORE INSERT ON COLABORACIONES
FOR EACH ROW
BEGIN
   SELECT SEC Colab.NEXTVAL INTO : NEW.OID Colab FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA RECIBOS
BEFORE INSERT ON RECIBOS
FOR EACH ROW
BEGIN
   SELECT SEC Rec.NEXTVAL INTO :NEW.OID Rec FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_INSCRIPCIONES
BEFORE INSERT ON INSCRIPCIONES
FOR EACH ROW
BEGIN
   SELECT SEC Ins.NEXTVAL INTO :NEW.OID Ins FROM DUAL;
END;
```

```
CREATE OR REPLACE TRIGGER SECUENCIA_PATROCINIOS
BEFORE INSERT ON PATROCINIOS
FOR EACH ROW
BEGIN
   SELECT SEC_Fin.NEXTVAL INTO :NEW.OID_Fin FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA TIPODONACIONES
BEFORE INSERT ON TIPODONACIONES
FOR EACH ROW
BEGIN
   SELECT SEC TDon.NEXTVAL INTO : NEW.OID TDon FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA DONACIONES
BEFORE INSERT ON DONACIONES
FOR EACH ROW
BEGIN
   SELECT SEC_Don.NEXTVAL INTO :NEW.OID_Don FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA MENSAJES
BEFORE INSERT ON MENSAJES
FOR EACH ROW
BEGIN
   SELECT SEC_M.NEXTVAL INTO :NEW.OID_M FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_ENVIOS
BEFORE INSERT ON ENVIOS
FOR EACH ROW
BEGIN
   SELECT SEC_Env.NEXTVAL INTO :NEW.OID_Env FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_PREGUNTAS
BEFORE INSERT ON PREGUNTAS
FOR EACH ROW
BEGIN
   SELECT SEC_Q.NEXTVAL INTO :NEW.OID_Q FROM DUAL;
END;
```

```
CREATE OR REPLACE TRIGGER SECUENCIA_CUESTIONARIOS
BEFORE INSERT ON CUESTIONARIOS
FOR EACH ROW
BEGIN
   SELECT SEC_Cues.NEXTVAL INTO :NEW.OID_Cues FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_FORMULARIOS
BEFORE INSERT ON FORMULARIOS
FOR EACH ROW
BEGIN
   SELECT SEC_Form.NEXTVAL INTO :NEW.OID_Form FROM DUAL;
END;
CREATE OR REPLACE TRIGGER SECUENCIA_RESPUESTAS
BEFORE INSERT ON RESPUESTAS
FOR EACH ROW
BEGIN
   SELECT SEC_Ans.NEXTVAL INTO :NEW.OID_Ans FROM DUAL;
END;
```

## Funciones y procedimientos

### Archivo: 2 procedimientos IS-G1-SSR dep&des.sql

```
-- FUNCIONES
-- Función auxiliar para pruebas de paquetes
CREATE OR REPLACE FUNCTION ASSERT EOUALS (salida BOOLEAN, salida esperada BOOLEAN)
RETURN VARCHAR2 AS
BEGIN
   IF (salida = salida esperada) THEN
        RETURN 'EXITO';
   ELSE
        RETURN 'FALLO';
   END IF;
END ASSERT EQUALS;
-- Función auxiliar para calcular precio de inscripción a actividad
CREATE OR REPLACE FUNCTION calcularCosteInscripcion (w costeTotal IN ACTIVIDADES.costeTotal%TYPE, w numeroPlazas IN ACTIVIDADES.numeroPlazas%TYPE)
RETURN NUMBER IS w costeInscripcion ACTIVIDADES.costeInscripcion%TYPE;
    w costeInscripcion := w costeTotal / w numeroPlazas;
RETURN (w_costeInscripcion);
END;
-- Función auxiliar para calcular la fecha de vencimiento de pago de un recibo.
CREATE OR REPLACE FUNCTION calcularFechaVencimiento (w fechaEmision IN RECIBOS.fechaEmision%TYPE)
RETURN DATE IS w fechaVencimiento RECIBOS.fechaVencimiento%TYPE;
BEGIN
   SELECT ADD MONTHS(w fechaEmision, 2) INTO w fechaVencimiento FROM DUAL;
RETURN (w fechaVencimiento);
END;
-- Función auxiliar para determinar si una persona es mayor de edad.
CREATE OR REPLACE FUNCTION esMayorDeEdad (w fechaNacimiento DATE)
RETURN BOOLEAN IS
w edad NUMBER;
BEGIN
   SELECT FLOOR(MONTHS BETWEEN(SYSDATE, w fechaNacimiento) / 12) INTO w edad FROM DUAL;
   RETURN (w edad >= 18);
END esMayorDeEdad;
```

```
-- PROCEDIMIENTOS
-- 1. Asociados a personas
-- 2. Asociados a patrocinadores, patrocinios y donaciones
-- 3. Asociados a proyectos y actividades
-- 4. Asociados a inscripciones y colaboraciones
-- 5. Asociados a mensaies v envíos
-- 6. Asociados a informes médicos de participantes
-- 7. Asociados a recibos
-- 8. Asociados a cuestionarios
______
-- 1. Asociados a personas
_____
-- Registrar PERSONA en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Persona (w dni IN PERSONAS.dni%TYPE, w nombre IN PERSONAS.nombre%TYPE,
   w apellidos IN PERSONAS.apellidos%TYPE, w fechaNacimiento IN PERSONAS.fechaNacimiento%TYPE, w direccion IN
   PERSONAS.direccion%TYPE, w localidad IN PERSONAS.localidad%TYPE, w provincia IN PERSONAS.provincia%TYPE,
   w_codigoPostal IN PERSONAS.codigoPostal%TYPE, w_email IN PERSONAS.email%TYPE, w_telefono IN PERSONAS.telefono%TYPE) IS
BEGIN
   INSERT INTO PERSONAS (dni, nombre, apellidos, fechaNacimiento, direccion, localidad, provincia, codigoPostal,
       email, telefono) VALUES (w dni, w nombre, w apellidos, w fechaNacimiento, w direccion, w localidad,
       w provincia, w codigoPostal, w email, w telefono);
   COMMIT WORK;
END Registrar Persona;
-- Actualizar PERSONA en el sistema de información
CREATE OR REPLACE PROCEDURE Act Persona (w dni IN PERSONAS.dni%TYPE, w nombre IN PERSONAS.nombre%TYPE,
   w apellidos IN PERSONAS.apellidos%TYPE, w fechaNacimiento IN PERSONAS.fechaNacimiento%TYPE, w direccion IN
   PERSONAS.direccion%TYPE, w localidad IN PERSONAS.localidad%TYPE, w provincia IN PERSONAS.provincia%TYPE.
   w_codigoPostal IN PERSONAS.codigoPostal%TYPE, w_email IN PERSONAS.email%TYPE, w_telefono IN PERSONAS.telefono%TYPE) IS
BEGIN
   UPDATE PERSONAS SET nombre=w nombre, apellidos=w apellidos, fechaNacimiento=w fechaNacimiento, direccion=w direccion,
       localidad=w localidad, provincia=w provincia, codigoPostal=w codigoPostal,email=w email, telefono=w telefono
   WHERE dni=w dni;
   COMMIT WORK:
END Act_Persona;
```

```
-- Eliminar PERSONA (con efecto cascada) en el sistema de información
CREATE OR REPLACE PROCEDURE Eliminar Persona (w dni IN PERSONAS.dni%TYPE) IS
   dniPersona PERSONAS.dni%TYPE:
BEGIN
   SELECT dni INTO dniPersona FROM PERSONAS WHERE dni=w dni:
   IF (dniPersona=w dni) THEN
       DELETE FROM PERSONAS WHERE dni=w dni;
   END IF:
   COMMIT WORK:
END Eliminar Persona;
-- Registrar COORDINADOR en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Coordinador (w dni IN PERSONAS.dni%TYPE, w nombre IN PERSONAS.nombre%TYPE,
   w apellidos IN PERSONAS.apellidos%TYPE, w fechaNacimiento IN PERSONAS.fechaNacimiento%TYPE, w direccion IN
   PERSONAS.direccion%TYPE, w localidad IN PERSONAS.localidad%TYPE, w provincia IN PERSONAS.provincia%TYPE,
   w codigoPostal IN PERSONAS.codigoPostal%TYPE, w email IN PERSONAS.email%TYPE, w telefono IN PERSONAS.telefono%TYPE) IS
BEGIN
   Registrar Persona(w dni, w nombre, w apellidos, w fechaNacimiento, w direccion, w localidad, w provincia, w codigoPostal, w email, w telefono);
   INSERT INTO COORDINADORES (dni) VALUES (w dni);
   COMMIT WORK:
END Registrar_Coordinador;
-- Registrar VOLUNTARIO en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Voluntario (w dni IN PERSONAS.dni%TYPE, w nombre IN PERSONAS.nombre%TYPE,
   w apellidos IN PERSONAS.apellidos%TYPE, w fechaNacimiento IN PERSONAS.fechaNacimiento%TYPE, w direccion IN
   PERSONAS.direccion%TYPE, w localidad IN PERSONAS.localidad%TYPE, w provincia IN PERSONAS.provincia%TYPE,
   w codigoPostal IN PERSONAS.codigoPostal%TYPE, w email IN PERSONAS.email%TYPE, w telefono IN PERSONAS.telefono%TYPE) IS
BEGIN
   Registrar Persona(w dni, w nombre, w apellidos, w fechaNacimiento, w direccion, w localidad, w provincia, w codigoPostal, w email, w telefono);
   INSERT INTO VOLUNTARIOS (dni, prioridadParticipacion) VALUES (w dni, 'baja');
   COMMIT WORK:
END Registrar Voluntario;
-- Registrar TUTOR LEGAL en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar TutorLegal (w dni IN PERSONAS.dni%TYPE, w nombre IN PERSONAS.nombre%TYPE,
   w apellidos IN PERSONAS.apellidos%TYPE, w fechaNacimiento IN PERSONAS.fechaNacimiento%TYPE, w direccion IN
   PERSONAS.direccion%TYPE, w localidad IN PERSONAS.localidad%TYPE, w provincia IN PERSONAS.provincia%TYPE,
   w codigoPostal IN PERSONAS.codigoPostal%TYPE, w email IN PERSONAS.email%TYPE, w telefono IN PERSONAS.telefono%TYPE) IS
BEGIN
   Registrar_Persona(w_dni, w_nombre, w_apellidos, w_fechaNacimiento, w_direccion, w_localidad, w_provincia, w_codigoPostal, w_email, w_telefono);
   INSERT INTO TUTORESLEGALES (dni) VALUES (w dni);
   COMMIT WORK;
END Registrar TutorLegal;
```

```
-- Registrar PARTICIPANTE en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Participante (w dni IN PERSONAS.dni%TYPE, w nombre IN PERSONAS.nombre%TYPE,
   w apellidos IN PERSONAS.apellidos%TYPE, w fechaNacimiento IN PERSONAS.fechaNacimiento%TYPE, w direccion IN
   PERSONAS.direccion%TYPE, w_localidad IN PERSONAS.localidad%TYPE, w_provincia IN PERSONAS.provincia%TYPE,
   w codigoPostal IN PERSONAS.codigoPostal%TYPE, w email IN PERSONAS.email%TYPE, w telefono IN PERSONAS.telefono%TYPE.
   w gradoDiscapacidad IN PARTICIPANTES.gradoDiscapacidad%TYPE, w dniVol IN VOLUNTARIOS.dni%TYPE, w dniTut IN TUTORESLEGALES.dni%TYPE) IS
   w OID Vol INTEGER:
   w OID Tut INTEGER;
BEGIN
   IF w dniVol IS NOT NULL AND w dniTut IS NOT NULL THEN
       SELECT OID Tut INTO w OID Tut FROM TUTORESLEGALES WHERE dni=w dniTut;
       SELECT OID Vol INTO w OID Vol FROM VOLUNTARIOS WHERE dni=w dniVol;
   ELSIF w dniVol IS NULL THEN
       w OID Vol:=NULL;
       SELECT OID Tut INTO w OID Tut FROM TUTORESLEGALES WHERE dni=w dniTut;
   ELSIF w dniTut IS NULL THEN
       w OID Tut:=NULL;
       SELECT OID Vol INTO w OID Vol FROM VOLUNTARIOS WHERE dni=w dniVol;
       w OID Vol:=NULL;
       w OID Tut:=NULL;
   END IF;
   Registrar Persona(w dni, w nombre, w apellidos, w fechaNacimiento, w direccion, w localidad, w provincia, w codigoPostal, w email, w telefono):
   INSERT INTO PARTICIPANTES (dni, gradoDiscapacidad, prioridadParticipacion, OID Vol, OID Tut)
       VALUES (w dni, w gradoDiscapacidad, 'alta', w OID Vol, w OID Tut);
   COMMIT WORK:
END Registrar_Participante;
-- 2. Asociados a patrocinadores, patrocinios y donaciones
-- Registrar INSTITUCION en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar_Institucion (w_cif IN INSTITUCIONES.cif%TYPE, w_nombre IN INSTITUCIONES.nombre%TYPE,
   w telefono IN INSTITUCIONES.telefono%TYPE, w direccion IN INSTITUCIONES.direccion%TYPE, w localidad IN INSTITUCIONES.localidad%TYPE,
   w provincia IN INSTITUCIONES.provincia%TYPE, w codigoPostal IN INSTITUCIONES.codigoPostal%TYPE, w email IN INSTITUCIONES.email%TYPE) IS
BEGIN
   INSERT INTO INSTITUCIONES VALUES (w_cif, w_nombre, w_telefono, w_direccion, w_localidad, w_provincia, w_codigoPostal, w_email, 0, null);
   COMMIT WORK:
END Registrar_Institucion;
```

```
-- Actualizar INSTITUCION en el sistema de información
CREATE OR REPLACE PROCEDURE Act Institucion (w cif IN INSTITUCIONES.cif%TYPE, w nombre IN INSTITUCIONES.nombre%TYPE,
   w telefono IN INSTITUCIONES.telefono%TYPE, w direccion IN INSTITUCIONES.direccion%TYPE, w localidad IN INSTITUCIONES.localidad%TYPE.
   w provincia IN INSTITUCIONES.provincia%TYPE, w codigoPostal IN INSTITUCIONES.codigoPostal%TYPE, w email IN INSTITUCIONES.email%TYPE) IS
   cifPatrocinador INSTITUCIONES.cif%TYPE:
BEGIN
   SELECT cif INTO cifPatrocinador FROM INSTITUCIONES WHERE cif=w cif;
   IF (cifPatrocinador=w cif) THEN
       UPDATE INSTITUCIONES SET nombre=w nombre, telefono=w telefono, direccion=w direccion, localidad=w localidad, provincia=w provincia,
            codigoPostal=w codigoPostal, email=w email WHERE cif=w cif;
   END IF:
   COMMIT WORK;
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.PUT LINE('No existe la institución con CIF: ' | w cif);
   ROLLBACK:
END Act Institucion;
-- Eliminar INSTITUCION en el sistema de información
CREATE OR REPLACE PROCEDURE Eliminar_Institucion (w_cif IN INSTITUCIONES.cif%TYPE) IS
   cifPatrocinador INSTITUCIONES.cif%TYPE:
BEGIN
   SELECT cif INTO cifPatrocinador FROM INSTITUCIONES WHERE cif=w cif;
   IF (cifPatrocinador=w cif) THEN
       DELETE FROM INSTITUCIONES WHERE cif=w cif;
   END IF;
   COMMIT WORK;
   EXCEPTION
   WHEN OTHERS THEN
        DBMS OUTPUT.PUT LINE('No existe la institución con CIF: ' | | w cif);
   ROLLBACK:
END Eliminar_Institucion;
-- Registrar PATROCINADOR en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar_Patrocinador (w_cif IN INSTITUCIONES.cif%TYPE, w_nombre IN INSTITUCIONES.nombre%TYPE,
   w telefono IN INSTITUCIONES.telefono%TYPE, w direccion IN INSTITUCIONES.direccion%TYPE, w localidad IN INSTITUCIONES.localidad%TYPE,
   w_provincia IN INSTITUCIONES.provincia%TYPE, w_codigoPostal IN INSTITUCIONES.codigoPostal%TYPE, w_email IN INSTITUCIONES.email%TYPE) IS
BEGIN
   INSERT INTO INSTITUCIONES VALUES (w_cif, w_nombre, w_telefono, w_direccion, w_localidad, w_provincia, w_codigoPostal, w_email, 1, null);
   COMMIT WORK:
END Registrar_Patrocinador;
```

```
-- Añadir PATROCINIO a ACTIVIDAD en el sistema de información
CREATE OR REPLACE PROCEDURE Add Patrocinio (w cif IN INSTITUCIONES.cif%TYPE, w OID Act IN ACTIVIDADES.OID Act%TYPE,
   w cantidad IN PATROCINIOS.cantidad%TYPE) IS
BEGIN
   INSERT INTO PATROCINIOS(cif. OID Act. cantidad) VALUES (w cif. w OID Act. w cantidad):
   COMMIT WORK:
END Add Patrocinio;
-- Registrar DONACION en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Donacion (w dni IN PERSONAS.dni%TYPE, w cif IN INSTITUCIONES.cif%TYPE, w nombre IN TIPODONACIONES.nombre%TYPE,
   w tipoUnidad IN TIPODONACIONES.tipoUnidad%TYPE, w cantidad IN DONACIONES.cantidad%TYPE, w valorUnitario IN DONACIONES.valorUnitario%TYPE) IS
   tipoDonacion TIPODONACIONES%ROWTYPE:
   w OID TDon INTEGER;
BEGIN
   SELECT OID TDon INTO w OID TDon FROM TIPODONACIONES WHERE nombre=w nombre AND tipoUnidad=w tipoUnidad:
   IF (w OID TDon>0) THEN
        INSERT INTO DONACIONES(fecha, cantidad, valorUnitario, dni, cif, OID TDon) VALUES (SYSDATE, w cantidad, w valorUnitario, w dni, w cif, w OID TDon);
   FND TF:
   COMMIT WORK;
   EXCEPTION
   WHEN OTHERS THEN
        INSERT INTO TIPODONACIONES(nombre, tipoUnidad) VALUES (w nombre, w tipoUnidad);
       SELECT OID TDon INTO w OID TDon FROM TIPODONACIONES WHERE nombre=w nombre AND tipoUnidad=w tipoUnidad;
        INSERT INTO DONACIONES(fecha, cantidad, valorUnitario, dni, cif, OID TDon) VALUES (SYSDATE, w cantidad, w valorUnitario, w dni, w cif, w OID TDon);
   COMMIT WORK;
END Registrar Donacion;
-- 3. Asociados a proyectos y actividades
-- Registrar PROYECTO por un COORDINADOR en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Proyecto (w dni IN PERSONAS.dni%TYPE, w nombre IN PROYECTOS.nombre%TYPE, w ubicacion IN PROYECTOS.ubicacion%TYPE,
   w esEvento IN PROYECTOS.esEvento%TYPE, w esProgDep IN PROYECTOS.esProgDep%TYPE) IS
   w OID Coord INTEGER;
   w OID Proj INTEGER;
BEGIN
   INSERT INTO PROYECTOS (nombre, ubicacion, esEvento, esProgDep) VALUES (w nombre, w ubicacion, w esEvento, w esProgDep);
   w OID Proj := SEC Proj.CURRVAL;
   w_OID_Coord := SEC_Coord.CURRVAL;
   INSERT INTO ENCARGADOS(OID Proj, OID Coord) VALUES (w OID Proj, w OID Coord);
   COMMIT WORK;
END Registrar Proyecto;
```

```
-- AÑADIR ACTIVIDAD a PROYECTO en el sistema de información
CREATE OR REPLACE PROCEDURE Add Actividad (w nombre IN ACTIVIDADES.nombre%TYPE, w objetivo IN ACTIVIDADES.objetivo%TYPE,
   w fechaInicio IN ACTIVIDADES.fechaInicio%TYPE, w fechaFin IN ACTIVIDADES.fechaFin%TYPE, w numeroPlazas IN ACTIVIDADES.numeroPlazas%TYPE.
   w tipo IN ACTIVIDADES.tipo%TYPE, w costeTotal IN ACTIVIDADES.costeTotal%TYPE, w OID Proj IN ACTIVIDADES.OID Proj%TYPE) IS
   w costeInscripcion ACTIVIDADES.costeInscripcion%TYPE:
BEGIN
   w costeInscripcion := calcularCosteInscripcion(w costeTotal, w numeroPlazas);
   INSERT INTO ACTIVIDADES(OID Proj, nombre, objetivo, fechaInicio, fechaFin, numeroPlazas, voluntariosRequeridos, tipo, costeTotal, costeInscripcion)
   VALUES (w OID Proj, w nombre, w objetivo, w fechaInicio, w fechaFin, w numeroPlazas, 0, w tipo, w costeTotal, w costeInscripcion);
   COMMIT WORK;
END Add Actividad;
-- 4. Asociados a inscripciones y colaboraciones
-- Inscribir PARTICIPANTE en ACTIVIDAD en el sistema de información
CREATE OR REPLACE PROCEDURE Inscribir Participante (w dni IN PERSONAS.dni%TYPE, w OID Act IN ACTIVIDADES.OID Act%TYPE) IS
w OID Part PARTICIPANTES.OID Part%TYPE;
w importe RECIBOS.importe%TYPE:
w_OID_Rec RECIBOS.OID_Rec%TYPE;
w OID Ins INSCRIPCIONES.OID Ins%TYPE:
BEGIN
   SELECT OID Part INTO w OID Part FROM PARTICIPANTES WHERE dni=w dni;
   SELECT costeInscripcion INTO w importe FROM ACTIVIDADES WHERE OID Act=w OID Act;
   IF (w importe<>0) THEN
       INSERT INTO INSCRIPCIONES(OID Part, OID Act, OID Rec) VALUES (w OID Part, w OID Act, null);
       Add ReciboInscripcion(w OID Act,w OID Part, SYSDATE, w importe, 'pendiente');
       SELECT OID Rec INTO w OID Rec FROM RECIBOS WHERE OID Act=w OID Act AND OID Part=w OID Part;
       w OID Ins:=SEC Ins.CURRVAL;
       UPDATE INSCRIPCIONES SET OID Rec = w OID Rec WHERE OID Ins=w OID Ins;
       INSERT INTO INSCRIPCIONES(OID Part, OID Act, OID Rec) VALUES (w OID Part, w OID Act, null);
   END IF;
   COMMIT WORK;
END Inscribir_Participante;
-- Inscribir VOLUNTARIO en ACTIVIDAD en el sistema de información
CREATE OR REPLACE PROCEDURE Inscribir Voluntario (w dni IN PERSONAS.dni%TYPE, w OID Act IN ACTIVIDADES.OID Act%TYPE) IS
w OID Vol VOLUNTARIOS.OID Vol%TYPE;
BEGIN
   SELECT OID Vol INTO w OID Vol FROM VOLUNTARIOS WHERE dni=w dni:
   INSERT INTO COLABORACIONES(OID Vol, OID Act) VALUES (w OID Vol, w OID Act);
   COMMIT WORK;
END Inscribir_Voluntario;
```

```
-- Actualizar ESTADO DE INTERES de un VOLUNTARIO en una ACTIVIDAD en el sistema de información
CREATE OR REPLACE PROCEDURE Act InteresVoluntariado (w dni IN VOLUNTARIOS.dni%TYPE, w OID Act IN ACTIVIDADES.OID Act%TYPE,
w estado IN ESTAINTERESADOEN.estado%TYPE) IS
w OID Vol VOLUNTARIOS.OID Vol%TYPE;
BEGIN
   SELECT OID Vol INTO w OID Vol FROM VOLUNTARIOS WHERE dni=w dni;
   UPDATE ESTAINTERESADOEN SET estado = w estado WHERE OID Vol=w OID Vol AND OID Act=w OID Act;
   COMMIT WORK;
END Act InteresVoluntariado;
-- Actualizar ESTADO DE INTERES de un PARTICIPANTE en una ACTIVIDAD en el sistema de información
CREATE OR REPLACE PROCEDURE Act InteresParticipante (w dni IN PARTICIPANTES.dni%TYPE, w OID Act IN ACTIVIDADES.OID Act%TYPE.
w estado IN ESTAINTERESADOEN.estado%TYPE) IS
w OID Part PARTICIPANTES.OID Part%TYPE;
BEGIN
   SELECT OID Part INTO w OID Part FROM PARTICIPANTES WHERE dni=w dni;
   UPDATE ESTAINTERESADOEN SET estado = w estado WHERE OID Part=w OID Part AND OID Act=w OID Act;
   COMMIT WORK:
END Act InteresParticipante;
-- 5. Asociados a mensajes y envíos
-- Registrar MENSAJE en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Mensaje (w tipo IN MENSAJES.tipo%TYPE, w fechaEnvio IN MENSAJES.fechaEnvio%TYPE,
   w asunto IN MENSAJES.asunto%TYPE, w contenido IN MENSAJES.contenido%TYPE, w OID Coord IN COORDINADORES.OID Coord%TYPE) IS
BEGIN
   INSERT INTO MENSAJES (tipo, fechaEnvio, asunto, contenido, OID Coord) VALUES (w tipo, w fechaEnvio, w asunto, w contenido, w OID Coord);
   COMMIT WORK:
END Registrar_Mensaje;
-- Registrar ENVIO de MENSAJE en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Envio (w dni IN PERSONAS.dni%TYPE, w cif IN INSTITUCIONES.cif%TYPE,
   w OID M IN MENSAJES.OID M%TYPE) IS
BEGIN
   INSERT INTO ENVIOS (OID M, dni, cif) VALUES (w OID M, w dni, w cif);
   COMMIT WORK;
END Registrar Envio;
```

```
-- 6. Asociados a informes médicos de participantes
-- Añadir INFORME MEDICO a un PARTICIPANTE en el sistema de información
CREATE OR REPLACE PROCEDURE Add InformeMedico (w OID Part IN INFORMESMEDICOS.OID Part%TYPE, w descripcion IN INFORMESMEDICOS.descripcion%TYPE) IS
BEGIN
   INSERT INTO INFORMESMEDICOS (descripcion, fecha, OID Part) VALUES (w descripcion, SYSDATE, w OID Part);
   COMMIT WORK:
END Add InformeMedico;
-- 7. Asociados a recibos
-- Registrar RECIBO de INSCRIPCION en el sistema de información
CREATE OR REPLACE PROCEDURE Add ReciboInscripcion (w OID Act IN ACTIVIDADES.OID Act%TYPE, w OID Part IN PARTICIPANTES.OID Part%TYPE,
   w fechaEmision IN RECIBOS.fechaEmision%TYPE, w importe IN RECIBOS.importe%TYPE, w estado IN RECIBOS.estado%TYPE) IS
   w fechaVencimiento RECIBOS.fechaVencimiento%TYPE;
BEGIN
   w_fechaVencimiento := calcularFechaVencimiento(w_fechaEmision);
   INSERT INTO RECIBOS (fechaEmision, fechaVencimiento, importe, estado, OID Act, OID Part)
   VALUES (w fechaEmision, w fechaVencimiento, w importe, w estado, w OID Act, w OID Part);
   COMMIT WORK;
END Add ReciboInscripcion;
-- Actualizar RECIBO en el sistema de información
CREATE OR REPLACE PROCEDURE Act Recibo (w OID Rec IN RECIBOS.OID Rec%TYPE, w fechaVencimiento IN RECIBOS.fechaVencimiento%TYPE,
   w importe IN RECIBOS.importe%TYPE, w estado IN RECIBOS.estado%TYPE) IS
BEGIN
   UPDATE RECIBOS SET fechaVencimiento=w fechaVencimiento, importe=w importe, estado=w estado WHERE OID Rec=w OID Rec;
   COMMIT WORK:
END Act_Recibo;
-- Cambiar estado de RECIBO en el sistema de información
CREATE OR REPLACE PROCEDURE Act_EstadoRecibo (w_OID_Rec IN RECIBOS.OID_Rec%TYPE, w_estado IN RECIBOS.estado%TYPE) IS
BEGIN
   UPDATE RECIBOS SET estado=w estado WHERE OID Rec=w OID Rec;
   COMMIT WORK;
END Act_EstadoRecibo;
```

```
-- 8. Asociados a cuestionarios
-- Registrar PREGUNTA en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar Pregunta (w tipo IN PREGUNTAS.tipo%TYPE, w enunciado IN PREGUNTAS.enunciado%TYPE) IS
BEGIN
   INSERT INTO PREGUNTAS (tipo, enunciado) VALUES (w tipo, w enunciado);
   COMMIT WORK:
END Registrar Pregunta;
-- Registrar CUESTIONARIO en el sistema de información
CREATE OR REPLACE PROCEDURE Registrar_Cuestionario (w_OID_Act IN CUESTIONARIOS.OID_Act%TYPE) IS
BEGIN
   INSERT INTO CUESTIONARIOS (fechaCreacion, OID Act) VALUES (SYSDATE, w OID Act);
   COMMIT WORK:
END Registrar_Cuestionario;
-- Añadir PREGUNTA a CUESTIONARIO en el sistema de información
CREATE OR REPLACE PROCEDURE Add Formulario (w_OID_Q IN PREGUNTAS.OID_Q%TYPE, w_OID_Cues IN CUESTIONARIOS.OID_Cues%TYPE) IS
BEGIN
   INSERT INTO FORMULARIOS (OID_Cues, OID_Q) VALUES (w_OID_Cues, w_OID_Q);
   COMMIT WORK;
END Add Formulario;
-- Registrar RESPUESTA a PREGUNTA de un CUESTIONARIO
CREATE OR REPLACE PROCEDURE Registrar Respuesta (w OID Vol IN VOLUNTARIOS.OID Vol%TYPE, w OID Part IN PARTICIPANTES.OID Part%TYPE,
   w OID Form IN FORMULARIOS.OID Form%TYPE, w contenido IN RESPUESTAS.contenido%TYPE) IS
BEGIN
   INSERT INTO RESPUESTAS (OID Vol, OID Part, OID Form, contenido) VALUES (w OID Vol, w OID Part, w OID Form, w contenido);
   COMMIT WORK:
END Registrar_Respuesta;
-- Actualizar PREGUNTA de un CUESTIONARIO
CREATE OR REPLACE PROCEDURE Act_Pregunta (w_OID Cues IN CUESTIONARIOS.OID Cues%TYPE, w_OID_Q IN PREGUNTAS.OID_Q%TYPE,
   w tipo IN PREGUNTAS.tipo%TYPE, w enunciado IN PREGUNTAS.enunciado%TYPE) IS
   w OID Form FORMULARIOS.OID Form%TYPE;
   n OID Q PREGUNTAS.OID Q%TYPE;
BEGIN
   SELECT OID Form INTO w OID Form FROM FORMULARIOS WHERE OID Cues=w OID Cues AND OID Q=w OID Q;
   Registrar_Pregunta(w_tipo, w_enunciado);
   n OID Q := SEC Q.CURRVAL;
   UPDATE FORMULARIOS SET OID Q=n OID Q WHERE OID Form=w OID Form AND OID Cues=w OID Cues;
   COMMIT WORK:
END Act_Pregunta;
```

### Cursores y listados de consultas

#### Archivo: 3 cursores IS-G1-SSR dep&des.sql

```
-- RF-1. Fichas de PARTICIPANTES
CREATE OR REPLACE PROCEDURE FichaParticipante(w_OID_Part PARTICIPANTES.OID_Part%TYPE) IS
   CURSOR C Participante IS
       SELECT * FROM PARTICIPANTES NATURAL JOIN PERSONAS WHERE OID Part=w OID Part;
   CURSOR C Intereses IS
       SELECT nombre FROM ACTIVIDADES NATURAL JOIN ESTAINTERESADOEN WHERE OID Part=w OID Part AND estado=1
          ORDER BY fechaInicio ASC:
   CURSOR C InformesMedicos IS
       SELECT * FROM INFORMESMEDICOS WHERE OID Part=w OID Part ORDER BY fecha DESC;
   CURSOR C Voluntario IS
       SELECT P.dni, P.nombre, P.apellidos FROM PARTICIPANTES PART
          LEFT JOIN VOLUNTARIOS VOL ON PART.OID Vol=VOL.OID Vol
          LEFT JOIN PERSONAS P ON P.dni=VOL.dni WHERE OID Part=w OID Part;
   CURSOR C TutorLegal IS
       SELECT P.dni, P.nombre, P.apellidos, P.telefono FROM PARTICIPANTES PART
          LEFT JOIN TUTORESLEGALES TUT ON PART.OID Tut=TUT.OID Tut
          LEFT JOIN PERSONAS P ON P.dni=TUT.dni WHERE OID Part=w OID Part:
   w gradoDiscapacidad PARTICIPANTES.gradoDiscapacidad%TYPE;
BEGTN
   FOR REG Part IN C Participante LOOP
       DBMS OUTPUT.PUT LINE('========:');
       DBMS OUTPUT.PUT LINE(' FICHA DE PARTICIPANTE');
      DBMS_OUTPUT.PUT_LINE('=========');
       DBMS OUTPUT.PUT LINE('- Participante: ' | REG Part.nombre | | ' ' | REG Part.apellidos);
       DBMS OUTPUT.PUT LINE('- DNI: ' | REG Part.dni);
      DBMS OUTPUT.PUT_LINE('- Fecha de nacimiento: ' || REG_Part.fechaNacimiento);
      DBMS OUTPUT.PUT LINE('- Email: ' | REG Part.email);
      DBMS_OUTPUT.PUT_LINE('- Teléfono: ' || REG_Part.telefono);
      DBMS OUTPUT.PUT LINE('- Dirección: ' | REG Part.dirección | ' , ' | REG Part.localidad | ' , ' |
          REG_Part.codigoPostal | ', ' | REG_Part.provincia);
      DBMS OUTPUT.PUT LINE('- Prioridad de participación: ' | REG Part.prioridadParticipación);
       w gradoDiscapacidad:=REG Part.gradoDiscapacidad * 100;
   END LOOP:
   FOR REG Tut IN C TutorLegal LOOP
       DBMS OUTPUT.PUT LINE('- Tutor legal: ' || REG Tut.nombre || ' ' || REG Tut.apellidos || ' ' ||
           '(' || REG_Tut.dni || ')' || ' - Teléfono: ' || REG_Tut.telefono);
   FOR REG Vol IN C Voluntario LOOP
      DBMS OUTPUT.PUT LINE('- Voluntario asignado: ' || REG_Vol.nombre || ' ' || REG_Vol.apellidos || ' ' ||
           '(' || REG Vol.dni || ')');
   END LOOP;
   DBMS OUTPUT.PUT LINE('-----');
   DBMS_OUTPUT.PUT_LINE('Informes médicos');
   DBMS OUTPUT.PUT LINE('------'):
```

```
DBMS OUTPUT.PUT LINE('- GRADO DE DISCAPACIDAD: ' || w gradoDiscapacidad || '%');
  FOR REG Inf IN C InformesMedicos LOOP
      DBMS OUTPUT.PUT LINE('----');
      DBMS_OUTPUT.PUT_LINE('- Nº Informe: ' || REG_Inf.OID_Inf);
      DBMS OUTPUT.PUT LINE('- Fecha: ' | REG Inf.fecha);
      DBMS OUTPUT.PUT LINE('- Descripción: ' | REG Inf.descripcion);
  FND LOOP:
  DBMS OUTPUT.PUT LINE('----'):
  DBMS OUTPUT.PUT LINE('Actividades de interés');
  DBMS_OUTPUT.PUT_LINE('-----'):
  FOR REG Int IN C Intereses LOOP
      DBMS OUTPUT.PUT LINE('- ' | REG Int.nombre);
  END LOOP:
END FichaParticipante;
-- RF-1. Fichas de VOLUNTARIOS
CREATE OR REPLACE PROCEDURE FichaVoluntario(w OID Vol VOLUNTARIOS.OID Vol%TYPE) IS
  CURSOR C Voluntario IS
      SELECT * FROM VOLUNTARIOS NATURAL JOIN PERSONAS WHERE OID Vol=w OID Vol;
  CURSOR C Intereses IS
      SELECT nombre FROM ACTIVIDADES NATURAL JOIN ESTAINTERESADOEN WHERE OID Vol=w OID Vol AND estado=1
         ORDER BY fechaInicio ASC:
BEGIN
  FOR REG Vol IN C Voluntario LOOP
      DBMS OUTPUT.PUT LINE('========'):
      DBMS OUTPUT.PUT LINE(' FICHA DE VOLUNTARIO');
      DBMS OUTPUT_LINE('========'):
      DBMS_OUTPUT.PUT_LINE('- Voluntario: ' || REG_Vol.nombre || ' ' || REG_Vol.apellidos);
      DBMS OUTPUT.PUT LINE('- DNI: ' | REG Vol.dni);
      DBMS OUTPUT.PUT LINE('- Fecha de nacimiento: ' | REG Vol.fechaNacimiento);
      DBMS OUTPUT.PUT LINE('- Email: ' | REG Vol.email);
      DBMS_OUTPUT.PUT_LINE('- Teléfono: ' | REG_Vol.telefono);
      DBMS OUTPUT.PUT LINE('- Dirección: ' || REG Vol.dirección || ', ' || REG Vol.localidad || ', ' ||
         REG_Vol.codigoPostal || ', ' || REG_Vol.provincia);
      DBMS OUTPUT.PUT LINE('- Prioridad de voluntariado: ' | REG Vol.prioridadParticipacion);
  END LOOP:
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS OUTPUT.PUT LINE('Actividades de interés');
  DBMS OUTPUT.PUT LINE('-----');
  FOR REG Int IN C Intereses LOOP
      DBMS OUTPUT.PUT LINE('- ' | REG Int.nombre);
  END LOOP:
END FichaVoluntario:
```

```
-- RF-1. Fichas de PATROCINADORES
CREATE OR REPLACE PROCEDURE FichaPatrocinador(w cif INSTITUCIONES.cif%TYPE) IS
   CURSOR C Patrocinador IS
      SELECT * FROM INSTITUCIONES WHERE cif=w cif AND esPatrocinador=1;
   CURSOR C Patrocinios IS
      SELECT cantidad, nombre FROM ACTIVIDADES NATURAL JOIN PATROCINIOS WHERE cif=w cif ORDER BY cantidad DESC;
BFGTN
   FOR REG Ins IN C Patrocinador LOOP
      DBMS OUTPUT.PUT LINE('========');
      DBMS OUTPUT.PUT LINE(' FICHA DE PATROCINADOR');
      DBMS OUTPUT_LINE('========'):
      DBMS OUTPUT.PUT LINE('- Patrocinador: ' | REG Ins.nombre);
      DBMS OUTPUT.PUT LINE('- CIF: ' | REG Ins.cif);
      DBMS OUTPUT.PUT LINE('- Tipo de patrocinador: ' | REG Ins.tipo);
      DBMS OUTPUT.PUT LINE('- Email: ' | REG Ins.email);
      DBMS_OUTPUT.PUT_LINE('- Teléfono: ' || REG_Ins.telefono);
      DBMS OUTPUT.PUT LINE('- Dirección: ' || REG Ins.dirección || ', ' || REG Ins.localidad || ', ' ||
          REG Ins.codigoPostal | ', ' | REG Ins.provincia);
   FND LOOP:
   DBMS_OUTPUT.PUT_LINE('----');
   DBMS OUTPUT.PUT LINE('Patrocinios');
   DBMS OUTPUT.PUT LINE('-----'):
   FOR REG Fin IN C Patrocinios LOOP
      DBMS_OUTPUT.PUT_LINE('- ' || RPAD(' (' || REG_Fin.cantidad || ' euros)',20) || REG_Fin.nombre);
   END LOOP;
END FichaPatrocinador;
-- RF-2. Valoración de ACTIVIDADES mediante CUESTIONARIOS contestados por PARTICIPANTES
-- Procedimiento auxiliar que obtiene las valoraciones de un CUESTIONARIO contestado por un PARTICIPANTE
CREATE OR REPLACE PROCEDURE GET ValoracionesPart(w OID Cues CUESTIONARIOS.OID Cues%TYPE, w OID Part PARTICIPANTES.OID Part%TYPE) IS
   CURSOR C Formularios IS
      SELECT P.enunciado AS enunciado, R.contenido AS respuesta FROM FORMULARIOS F
          LEFT JOIN RESPUESTAS R ON F.OID Form=R.OID Form
          LEFT JOIN PREGUNTAS P ON F.OID_Q=P.OID_Q
          WHERE F.OID Cues=w OID Cues AND R.OID Part=w OID Part;
   CURSOR C Participante IS
      SELECT * FROM PERSONAS NATURAL JOIN PARTICIPANTES WHERE OID Part=w OID Part;
BEGIN
   FOR REG Part IN C Participante LOOP
      DBMS OUTPUT.PUT LINE('- PARTICIPANTE: ' | REG Part.nombre | | ' ' | REG Part.apellidos | | ' (' | REG Part.dni | | ')');
   END LOOP:
   FOR REG_Form IN C_Formularios LOOP
      DBMS_OUTPUT.PUT_LINE('- ' || REG_Form.enunciado);
      DBMS OUTPUT.PUT LINE('---> ' || REG Form.respuesta);
   DBMS OUTPUT.PUT LINE('----'):
END GET ValoracionesPart;
```

```
CREATE OR REPLACE PROCEDURE GET CuestionariosPart(w OID Act ACTIVIDADES.OID Act%TYPE) IS
   CURSOR C Cuestionarios IS
       SELECT DISTINCT C.OID Cues AS OID Cues, I.OID Part AS OID Part FROM ACTIVIDADES A LEFT JOIN CUESTIONARIOS C ON A.OID Act=C.OID Act
          LEFT JOIN INSCRIPCIONES I ON A.OID Act=I.OID Act LEFT JOIN RESPUESTAS R ON I.OID Part=R.OID Part WHERE A.OID Act=w OID Act
           AND EXISTS (SELECT * FROM RESPUESTAS WHERE R.OID Part=I.OID Part):
   actividad ACTIVIDADES%ROWTYPE;
BFGTN
   SELECT * INTO actividad FROM ACTIVIDADES WHERE OID Act=w OID Act;
   DBMS OUTPUT.PUT LINE('========='):
   DBMS OUTPUT.PUT LINE(' CUESTIONARIOS DE ACTIVIDAD - VALORACIONES DE PARTICIPANTES');
   DBMS OUTPUT.PUT LINE('========'):
   DBMS OUTPUT.PUT LINE('- Actividad: ' | actividad.nombre);
   DBMS_OUTPUT.PUT_LINE('- Objetivo: ' || actividad.objetivo);
   DBMS OUTPUT.PUT LINE('-----'):
   FOR REG Cues IN C Cuestionarios LOOP
       GET ValoracionesPart(REG Cues.OID Cues, REG Cues.OID Part);
   END LOOP:
END GET CuestionariosPart;
-- RF-2. Valoración de ACTIVIDADES mediante CUESTIONARIOS contestados por VOLUNTARIOS
-- Procedimiento auxiliar que obtiene las valoraciones de un CUESTIONARIO contestado por un VOLUNTARIO
CREATE OR REPLACE PROCEDURE GET ValoracionesVol(w OID Cues CUESTIONARIOS.OID Cues%TYPE, w OID Vol VolUNTARIOS.OID Vol%TYPE) IS
   CURSOR C Formularios IS
       SELECT P.enunciado AS enunciado, R.contenido AS respuesta FROM FORMULARIOS F
          LEFT JOIN RESPUESTAS R ON F.OID Form=R.OID Form
          LEFT JOIN PREGUNTAS P ON F.OID O=P.OID O
          WHERE F.OID Cues=w OID Cues AND R.OID Vol=w OID Vol;
   CURSOR C Voluntario IS
       SELECT * FROM PERSONAS NATURAL JOIN VOLUNTARIOS WHERE OID Vol=w OID Vol;
BEGIN
   FOR REG Vol IN C Voluntario LOOP
       DBMS_OUTPUT.PUT_LINE('- VOLUNTARIO: ' || REG_Vol.nombre || ' ' || REG_Vol.apellidos || ' (' || REG_Vol.dni || ')');
   END LOOP:
   FOR REG Form IN C Formularios LOOP
       DBMS OUTPUT.PUT LINE('- ' | REG Form.enunciado);
       DBMS OUTPUT.PUT LINE('---> ' || REG Form.respuesta);
   DBMS OUTPUT.PUT LINE('----'):
END GET ValoracionesVol;
CREATE OR REPLACE PROCEDURE GET_CuestionariosVol(w_OID_Act ACTIVIDADES.OID_Act%TYPE) IS
   CURSOR C Cuestionarios IS
       SELECT DISTINCT C.OID Cues AS OID Cues, CO.OID Vol AS OID Vol FROM ACTIVIDADES A LEFT JOIN CUESTIONARIOS C ON A.OID Act=C.OID Act
          LEFT JOIN COLABORACIONES CO ON A.OID Act=CO.OID Act LEFT JOIN RESPUESTAS R ON CO.OID Vol=R.OID Vol WHERE A.OID Act=w OID Act
          AND EXISTS (SELECT * FROM RESPUESTAS WHERE R.OID Vol=CO.OID Vol);
   actividad ACTIVIDADES%ROWTYPE;
BEGIN
   SELECT * INTO actividad FROM ACTIVIDADES WHERE OID Act=w OID Act;
```

```
DBMS OUTPUT.PUT LINE('=========');
   DBMS OUTPUT.PUT LINE(' CUESTIONARIOS DE ACTIVIDAD - VALORACIONES DE VOLUNTARIOS');
   DBMS OUTPUT.PUT LINE('=======');
   DBMS OUTPUT.PUT LINE('- Actividad: ' | actividad.nombre);
  DBMS_OUTPUT.PUT_LINE('- Objetivo: ' || actividad.objetivo);
   DBMS_OUTPUT.PUT_LINE('-----'):
   FOR REG Cues IN C Cuestionarios LOOP
      GET ValoracionesVol(REG Cues.OID Cues, REG_Cues.OID_Vol);
END GET CuestionariosVol;
-- RF-6. Lista de DONANTES
CREATE OR REPLACE PROCEDURE ListaDonantes IS
   CURSOR C DonantesP IS
      SELECT dni. nombre. apellidos. direccion. localidad, codigoPostal, provincia, telefono, COUNT(*) AS n_donaciones
         FROM PERSONAS NATURAL JOIN DONACIONES
         GROUP BY dni, nombre, apellidos, direccion, localidad, codigoPostal, provincia, telefono
         ORDER BY n donaciones DESC:
   CURSOR C DonantesI IS
      SELECT cif, nombre, direccion, localidad, codigoPostal, provincia, telefono, COUNT(*) AS n donaciones
         FROM INSTITUCIONES NATURAL JOIN DONACIONES
         GROUP BY cif, nombre, direccion, localidad, codigoPostal, provincia, telefono
         ORDER BY n donaciones DESC;
BEGIN
   DBMS OUTPUT.PUT LINE('========'):
   DBMS_OUTPUT.PUT_LINE(' LISTA DE DONANTES'):
   DBMS OUTPUT.PUT LINE('========');
   DBMS_OUTPUT.PUT_LINE('-----');
   DBMS OUTPUT.PUT LINE('-- Donantes particulares'):
  DBMS_OUTPUT.PUT_LINE( -- Domaintes particulares );
   DBMS OUTPUT.PUT LINE(RPAD('DNI',15) | RPAD('Nº DONACIONES',16) | RPAD('NOMBRE',35) | RPAD('DIRECCION',60) | RPAD('TELEFONO',15));
   FOR REG DonP IN C DonantesP LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_DonP.dni.15) || RPAD(REG_DonP.n donaciones.16) || RPAD(REG_DonP.nombre || ' ' || REG_DonP.apellidos.35) ||
         RPAD(REG_DonP.direccion | | ', ' | REG_DonP.localidad | | ', ' | REG_DonP.codigoPostal | | ', ' | REG_DonP.provincia,60) |
         RPAD(REG DonP.telefono,15));
   END LOOP:
   DBMS OUTPUT.PUT LINE('-----'):
   DBMS_OUTPUT.PUT_LINE('-- Donantes institucionales');
  DBMS OUTPUT.PUT LINE('-----'):
   DBMS_OUTPUT_PUT_LINE(RPAD('CIF',15) || RPAD('Nº DONACIONES',16) || RPAD('NOMBRE',35) || RPAD('DIRECCION',60) || RPAD('TELEFONO',15));
   FOR REG DonI IN C DonantesI LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_DonI.cif,15) || RPAD(REG_DonI.n_donaciones,16) || RPAD(REG_DonI.nombre,35) ||
         RPAD(REG_DonI.dirección | ', ' | REG_DonI.localidad | ', ' | REG_DonI.codigoPostal | ', ' | REG_DonI.provincia,60) |
         RPAD(REG DonI.telefono,15));
   END LOOP:
END ListaDonantes;
```

```
-- RF-7. Lista de correos electrónicos según grupo de destinatarios ('participantes', 'voluntarios' 'tutores', 'patrocinadores')
CREATE OR REPLACE PROCEDURE Lista Email(grupo VARCHAR2) IS
   CURSOR C Participantes IS
       SELECT email, nombre, apellidos FROM PERSONAS NATURAL JOIN PARTICIPANTES WHERE email IS NOT NULL ORDER BY apellidos;
   CURSOR C Voluntarios IS
       SELECT email. nombre. apellidos FROM PERSONAS NATURAL JOIN VOLUNTARIOS WHERE email IS NOT NULL ORDER BY apellidos;
   CURSOR C TutoresLegales IS
       SELECT email, nombre, apellidos FROM PERSONAS NATURAL JOIN TUTORESLEGALES WHERE email IS NOT NULL ORDER BY apellidos;
   CURSOR C Patrocinadores IS
       SELECT email, nombre FROM INSTITUCIONES WHERE email IS NOT NULL AND esPatrocinador=1;
BEGIN
   DBMS_OUTPUT.PUT_LINE('========='):
   DBMS OUTPUT.PUT LINE(' LISTA DE CORREO DE ' | UPPER(grupo));
   DBMS OUTPUT.PUT LINE('========');
   DBMS OUTPUT.PUT LINE(RPAD('EMAIL',35) | RPAD('NOMBRE',50));
   IF (LOWER(grupo)='participantes') THEN
       FOR REG Part IN C Participantes LOOP
           DBMS_OUTPUT.PUT_LINE(RPAD(REG_Part.email,35) || RPAD(REG_Part.apellidos || ', ' || REG_Part.nombre,50));
       FND LOOP:
   ELSIF (LOWER(grupo)='voluntarios') THEN
       FOR REG Vol IN C Voluntarios LOOP
           DBMS_OUTPUT.PUT_LINE(RPAD(REG_Vol.email,35) || RPAD(REG_Vol.apellidos || ', ' || REG_Vol.nombre,50));
       END LOOP:
   ELSIF (LOWER(grupo)='tutores') THEN
       FOR REG Tut IN C TutoresLegales LOOP
           DBMS OUTPUT.PUT LINE(RPAD(REG Tut.email, 35) | RPAD(REG Tut.apellidos | ' ', ' | REG Tut.nombre, 50));
       END LOOP:
   ELSIF (LOWER(grupo)='patrocinadores') THEN
       FOR REG Ins IN C Patrocinadores LOOP
           DBMS OUTPUT.PUT LINE(RPAD(REG Ins.email, 35) || RPAD(REG Ins.nombre, 50));
       END LOOP;
   ELSE
       raise application error(-20600, 'No existe ningún grupo de destinatarios con ese nombre.');
   END IF:
END Lista Email;
-- RF-8. Lista de VOLUNTARIOS en ACTIVIDAD
CREATE OR REPLACE PROCEDURE Lista VolAct(w OID Act ACTIVIDADES.OID Act%TYPE) IS
   CURSOR C Actividad IS
       SELECT * FROM ACTIVIDADES WHERE OID Act=w OID Act;
   CURSOR C Voluntarios IS
       SELECT * FROM PERSONAS NATURAL JOIN VOLUNTARIOS V LEFT JOIN COLABORACIONES C ON V.OID Vol=C.OID Vol
           WHERE OID_Act=w_OID_Act ORDER BY apellidos;
BEGIN
   DBMS OUTPUT.PUT LINE(RPAD('OID ACT',10) | RPAD('ACTIVIDAD',35) | RPAD('Nº VOL REQUERIDOS',20) |
       RPAD('TIPO',20) || RPAD('COSTE TOTAL',20) || RPAD('COSTE INSCRIPCIÓN',20));
   FOR REG Act IN C Actividad LOOP
       DBMS_OUTPUT.PUT_LINE(RPAD(REG_Act.OID_Act,10) || RPAD(REG_Act.nombre,35) || RPAD(REG_Act.voluntariosRequeridos,20) ||
           RPAD(REG Act.tipo,20) | RPAD(REG Act.costeTotal | ' €',20) | RPAD(REG Act.costeInscripcion | ' €',20));
   END LOOP;
```

```
DBMS_OUTPUT.PUT_LINE('============:=:');
   DBMS OUTPUT.PUT LINE(' VOLUNTARIOS DE LA ACTIVIDAD');
   DBMS OUTPUT.PUT LINE(RPAD('DNI',15) | RPAD('Nombre',30) | RPAD('Nacimiento',12) |
      RPAD('Email'.25) || RPAD('Teléfono'.15));
   FOR REG Vol IN C Voluntarios LOOP
      DBMS OUTPUT.PUT LINE(RPAD(REG Vol.dni,15) | RPAD(REG Vol.nombre | ' ' | REG Vol.apellidos,30) | RPAD(REG Vol.fechaNacimiento,12) |
          RPAD(REG Vol.email,25) | RPAD(REG Vol.telefono,15));
   FND LOOP:
END Lista VolAct;
-- RF-9. Historial de VOLUNTARIADO
CREATE OR REPLACE PROCEDURE Lista HistVol(w OID Vol VOLUNTARIOS.OID Vol%TYPE) IS
   CURSOR C Voluntario IS
      SELECT * FROM PERSONAS NATURAL JOIN VOLUNTARIOS WHERE OID Vol=w OID Vol;
   CURSOR C Colaboraciones IS
      SELECT P.nombre AS Proj nombre, A.OID Act AS OID Act, A.nombre AS Act nombre, A.fechaInicio AS Act fInicio, A.fechaFin AS Act fFin
          FROM PERSONAS NATURAL JOIN VOLUNTARIOS NATURAL JOIN COLABORACIONES C
          LEFT JOIN ACTIVIDADES A ON A.OID Act=C.OID Act
         LEFT JOIN PROYECTOS P ON P.OID Proj=A.OID Proj
          WHERE OID Vol=w OID Vol ORDER BY A.fechaInicio DESC;
BEGIN
   DBMS_OUTPUT.PUT_LINE(RPAD('OID_VOL'.10) || RPAD('DNI'.15) || RPAD('NOMBRE'.35) || RPAD('EMAIL'.35) || RPAD('TELEFONO'.15));
   FOR REG Vol IN C Voluntario LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_Vol.OID_Vol,10) || RPAD(REG_Vol.dni,15) || RPAD(REG_Vol.nombre || ' ' || REG_Vol.apellidos,35) ||
          RPAD(REG Vol.email.35) | RPAD(REG Vol.telefono.15)):
   END LOOP:
   DBMS_OUTPUT.PUT_LINE('===============:=::);
   DBMS OUTPUT.PUT LINE(' HISTORIAL DE VOLUNTARIADO');
   DBMS_OUTPUT_PUT_LINE('==============:=');
   DBMS_OUTPUT.PUT_LINE(RPAD('Proyecto',40) || RPAD('OID_Act',10) || RPAD('Actividad',45) || RPAD('Inicio',15) || RPAD('Fin',15));
   FOR REG Colab IN C Colaboraciones LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_Colab.Proj_nombre,40) || RPAD(REG_Colab.OID_Act,10) || RPAD(REG_Colab.Act nombre,45) ||
          RPAD(REG Colab.Act fInicio.15) | RPAD(REG Colab.Act fFin.15));
   END LOOP:
END Lista HistVol;
-- RF-10. Lista de PARTICIPANTES en ACTIVIDAD
CREATE OR REPLACE PROCEDURE Lista PartAct(w OID Act ACTIVIDADES.OID Act%TYPE) IS
   CURSOR C Actividad IS
      SELECT * FROM ACTIVIDADES WHERE OID Act=w OID Act;
   CURSOR C Participantes IS
      SELECT PER1.dni AS Part_dni, PER1.nombre AS Part_nombre, PER1.apellidos AS Part_apellidos, PER1.fechaNacimiento AS Part_fechaNacimiento,
          PART.gradoDiscapacidad AS gradoDiscapacidad, PER2.nombre AS Tut nombre, PER2.apellidos AS Tut apellidos, PER2.email AS Tut email,
          PER2.telefono AS Tut telefono FROM PERSONAS PER1
         LEFT JOIN PARTICIPANTES PART ON PER1.dni=PART.dni
          LEFT JOIN INSCRIPCIONES I ON PART.OID Part=I.OID Part
          LEFT JOIN TUTORESLEGALES T ON T.OID Tut=PART.OID Tut
          LEFT JOIN PERSONAS PER2 ON T.dni=PER2.dni
          WHERE OID Act=w OID Act ORDER BY PER1.apellidos;
```

```
BEGIN
   DBMS OUTPUT.PUT LINE(RPAD('OID ACT',10) | RPAD('ACTIVIDAD',40) | RPAD('Nº VOL REOUERIDOS',20) |
      RPAD('TIPO'.20) || RPAD('COSTE TOTAL'.20) || RPAD('COSTE INSCRIPCIÓN'.20));
   FOR REG Act IN C Actividad LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_Act.OID_Act.10) || RPAD(REG_Act.nombre.40) || RPAD(REG_Act.voluntariosRegueridos.20) ||
          RPAD(REG Act.tipo,20) | RPAD(REG Act.costeTotal | ' €',20) | RPAD(REG Act.costeInscripcion | ' €',20));
   FND LOOP:
   DBMS OUTPUT.PUT LINE(' PARTICIPANTES DE LA ACTIVIDAD');
   DBMS_OUTPUT.PUT_LINE('===============:=:);
   DBMS OUTPUT.PUT LINE(RPAD('DNI',15) || RPAD('Nombre',45) || RPAD('Nacimiento',12) || RPAD('% discapacidad', 15) || RPAD('|',2) ||
      RPAD('Tutor Legal', 35) || RPAD('Teléfono', 12) || RPAD('Email', 25));
   FOR REG Part IN C Participantes LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_Part.Part_dni,15) || RPAD(REG_Part.Part_nombre || ' ' || REG_Part.Part_apellidos,45) ||
          RPAD(REG Part.Part fechaNacimiento,12) || RPAD(REG Part.gradoDiscapacidad * 100 || ' %', 15) || RPAD('|',2) ||
          RPAD(REG Part.Tut nombre | ' ' | REG Part.Tut apellidos, 35) | RPAD(REG Part.Tut telefono, 12) | RPAD(REG Part.Tut email, 25));
   END LOOP:
END Lista PartAct;
-- RF-11. Historial de PARTICIPACION
CREATE OR REPLACE PROCEDURE Lista HistPart(w OID Part PARTICIPANTES.OID Part%TYPE) IS
   CURSOR C Participante IS
      SELECT * FROM PERSONAS NATURAL JOIN PARTICIPANTES WHERE OID Part=w OID Part;
   CURSOR C Inscripciones IS
      SELECT P.nombre AS Proj nombre, A.OID Act AS OID Act, A.nombre AS Act nombre, A.fechaInicio AS Act fInicio, A.fechaFin AS Act fFin
          FROM PERSONAS NATURAL JOIN PARTICIPANTES NATURAL JOIN INSCRIPCIONES I
          LEFT JOIN ACTIVIDADES A ON A.OID Act=I.OID Act
          LEFT JOIN PROYECTOS P ON P.OID Proj=A.OID Proj
          WHERE OID Part=w OID Part ORDER BY A.fechaInicio DESC;
BEGIN
   DBMS OUTPUT.PUT LINE(RPAD('OID PART',10) || RPAD('DNI',15) || RPAD('NOMBRE',35) || RPAD('TELEFONO',15) || RPAD('EMAIL',35));
   FOR REG Vol IN C Participante LOOP
      DBMS_OUTPUT_PUT_LINE(RPAD(REG_Vol.OID_Vol.10) || RPAD(REG_Vol.dni.15) || RPAD(REG_Vol.nombre || ' ' || REG_Vol.apellidos.35) ||
          RPAD(REG Vol.telefono,15) | RPAD(REG Vol.email,35));
   FND LOOP:
   DBMS_OUTPUT_PUT_LINE('==============:=:'):
   DBMS OUTPUT.PUT LINE(' HISTORIAL DE PARTICIPACIÓN');
   DBMS_OUTPUT.PUT_LINE('=============:=:');
   DBMS OUTPUT.PUT LINE(RPAD('Proyecto',40) || RPAD('OID Act',10) || RPAD('Actividad',45) || RPAD('Inicio',15) || RPAD('Fin',15));
   FOR REG Ins IN C Inscripciones LOOP
      DBMS OUTPUT.PUT LINE(RPAD(REG Ins.Proj nombre,40) | RPAD(REG Ins.OID Act,10) | RPAD(REG Ins.Act nombre,45) |
          RPAD(REG_Ins.Act_fInicio,15) || RPAD(REG_Ins.Act_fFin,15));
   END LOOP:
END Lista HistPart;
```

```
-- RF-12. Lista de PATROCINIOS de ACTIVIDAD
CREATE OR REPLACE PROCEDURE Lista PatrociniosAct(w OID Act ACTIVIDADES.OID Act%TYPE) IS
   CURSOR C Actividad IS
      SELECT * FROM ACTIVIDADES WHERE OID Act=w OID Act;
   CURSOR C Patrocinios IS
      SELECT * FROM PATROCINIOS NATURAL JOIN INSTITUCIONES WHERE OID Act=w OID Act ORDER BY cantidad DESC:
BFGTN
   DBMS OUTPUT.PUT LINE(RPAD('OID ACT',10) | RPAD('ACTIVIDAD',35) | RPAD('Nº VOL REOUERIDOS',20) |
      RPAD('TIPO',20) | RPAD('COSTE TOTAL',20) | RPAD('COSTE INSCRIPCIÓN',20));
   FOR REG Act IN C Actividad LOOP
      DBMS_OUTPUT.PUT_LINE(RPAD(REG_Act.OID_Act,10) || RPAD(REG_Act.nombre,35) || RPAD(REG_Act.voluntariosRequeridos,20) ||
          RPAD(REG Act.tipo,20) | RPAD(REG Act.costeTotal | ' €',20) | RPAD(REG Act.costeInscripcion | ' €',20));
   END LOOP:
   DBMS_OUTPUT_PUT_LINE('============:'):
   DBMS OUTPUT.PUT LINE(' PATROCINIOS DE LA ACTIVIDAD');
   DBMS_OUTPUT.PUT_LINE('=============:=::'):
   DBMS OUTPUT.PUT LINE(RPAD('OID Fin',8) | RPAD('Cantidad',15) | RPAD('Patrocinador',50) | RPAD('Tipo',15) | RPAD('Email', 35) | RPAD('Teléfono',15));
   FOR REG Fin IN C Patrocinios LOOP
      DBMS OUTPUT.PUT LINE(RPAD(REG Fin.OID Fin,8) | RPAD(REG Fin.cantidad | ' €',15) | RPAD(REG Fin.nombre,50) | RPAD(REG Fin.tipo,15) | RPAD(REG Fin.tipo,15) |
         RPAD(REG Fin.email, 35) | RPAD(REG Fin.telefono,15));
   END LOOP:
END Lista PatrociniosAct;
-- RF-13. Lista de DONACIONES en intervalo temporal
CREATE OR REPLACE PROCEDURE Lista DonTemp(f1 DATE, f2 DATE) IS
   CURSOR C Donaciones IS
      SELECT * FROM DONACIONES NATURAL JOIN TIPODONACIONES
         WHERE fecha BETWEEN f1 AND f2
         ORDER BY fecha ASC:
BEGIN
   IF (f1 < f2) THEN
      DBMS_OUTPUT_PUT_LINE('==============:='):
      DBMS OUTPUT.PUT LINE(' DONACIONES REALIZADAS ENTRE EL ' | f1 | f1 | f2 | f2):
      DBMS_OUTPUT_PUT_LINE('===========');
      DBMS OUTPUT.PUT LINE(RPAD('OID DON',10) || RPAD('DONACIÓN',30) || RPAD('CANTIDAD DONADA',25) || RPAD('VALOR UNITARIO',20) ||
         RPAD('FECHA',15) | RPAD('ID DONANTE',20));
      FOR REG Don IN C Donaciones LOOP
          DBMS_OUTPUT.PUT_LINE(RPAD(REG_Don.OID_Don,10) || RPAD(REG_Don.nombre,30) || RPAD(REG_Don.cantidad || ' ' || REG_Don.tipoUnidad,25)||
             RPAD(REG Don.valorUnitario | | ' €', 20) | RPAD(REG Don.fecha, 15) | RPAD(REG Don.dni | REG Don.cif, 20));
      END LOOP;
   ELSE
      raise application error(-20600, 'La fecha f1 debe ser anterior a la fecha f2.');
END Lista DonTemp;
```

```
-- RF-14. Lista de ACTIVIDADES en intervalo temporal
CREATE OR REPLACE PROCEDURE Lista ActTemp(f1 DATE, f2 DATE) IS
   CURSOR C Actividades IS
          SELECT P.nombre AS Proj_nombre, A.* FROM PROYECTOS P LEFT JOIN ACTIVIDADES A ON P.OID_Proj=A.OID_Proj
          WHERE A.fechaInicio BETWEEN f1 AND f2 ORDER BY A.fechaInicio ASC:
BEGIN
   IF (f1 < f2) THEN
      DBMS OUTPUT.PUT LINE('==============:=');
      DBMS OUTPUT.PUT LINE(' ACTIVIDADES REALIZADAS ENTRE EL ' | f1 | ' Y EL ' | f2);
      DBMS_OUTPUT.PUT_LINE('===========:');
      DBMS OUTPUT.PUT LINE(RPAD('PROYECTO',37) || RPAD('OID ACT',10) || RPAD('ACTIVIDAD',40) || RPAD('TIPO ACTIVIDAD',18) ||
          RPAD('Nº MAX. PART',15) || RPAD('Nº VOL REOU.',15) || RPAD('COSTE TOTAL',20) || RPAD('COSTE INSCRIPCIÓN',20));
      FOR REG Act IN C Actividades LOOP
         DBMS_OUTPUT.PUT_LINE(RPAD(REG_Act.Proj_nombre,37) || RPAD(REG_Act.OID_Act,10) || RPAD(REG_Act.nombre,40) ||
             RPAD(REG Act.tipo,18) || RPAD(REG Act.numeroPlazas,15) || RPAD(REG Act.voluntariosRequeridos.15) ||
             RPAD(REG Act.costeTotal | ' €',20) | RPAD(REG Act.costeInscripcion | ' €',20));
      END LOOP;
   ELSE
      raise application error(-20600, 'La fecha f1 debe ser anterior a la fecha f2.');
   END IF;
END Lista ActTemp;
```

## **Triggers**

```
Archivo: 4 triggers IS-G1-SSR dep&des.sql
-- RN-1 y RN-2. Prioridad de participación
CREATE OR REPLACE TRIGGER PRIORIDAD PARTICIPACION
BEFORE UPDATE ON ESTAINTERESADOEN
FOR EACH ROW
DECLARE
   w OID Part PARTICIPANTES.OID Part%TYPE;
   w OID Act ACTIVIDADES.OID Act%TYPE;
   n ins3Meses INTEGER;
   n ins INTEGER;
BEGIN
   w OID Part:=:OLD.OID Part;
   w OID Act:=:OLD.OID Act;
   SELECT COUNT(*) INTO n ins3Meses FROM INSCRIPCIONES NATURAL JOIN ACTIVIDADES WHERE OID Part = w OID Part AND fechaInicio BETWEEN ADD MONTHS(SYSDATE, -3) AND
SYSDATE;
   SELECT COUNT(*) INTO n ins FROM INSCRIPCIONES NATURAL JOIN ACTIVIDADES WHERE OID Part=w OID Part;
   IF (n ins3Meses > 0) THEN
        UPDATE PARTICIPANTES SET prioridadParticipacion='baja' WHERE OID Part=w OID Part;
   ELSIF (n ins > 0) THEN
       UPDATE PARTICIPANTES SET prioridadParticipacion='media' WHERE OID Part=w OID Part;
END PRIORIDAD PARTICIPACION;
CREATE OR REPLACE TRIGGER PARTICIPANTE INSCRITO
AFTER INSERT ON INSCRIPCIONES
FOR EACH ROW
BEGIN
   UPDATE PARTICIPANTES SET prioridadParticipacion='baja' WHERE OID Part=:NEW.OID Part;
END PARTICIPANTE_INSCRITO;
-- RN-3. Pagos pendientes
CREATE OR REPLACE TRIGGER PAGOS PENDIENTES
BEFORE INSERT ON INSCRIPCIONES
FOR EACH ROW
DECLARE
   CURSOR C_PAGOS IS SELECT * FROM RECIBOS WHERE OID_Part=: NEW.OID_Part;
   FOR recibo IN C PAGOS LOOP
        IF (recibo.estado = 'pendiente') THEN
            raise application error(-20600, 'El participante tiene pagos pendientes');
        END IF;
   END LOOP;
END PAGOS_PENDIENTES;
```

```
-- RN-4. Representante legal
CREATE OR REPLACE TRIGGER REPRESENTANTE LEGAL
BEFORE INSERT ON PARTICIPANTES FOR EACH ROW
DECLARE
   w fechaNacimiento PERSONAS.fechaNacimiento%TYPE:
BEGIN
   SELECT fechaNacimiento INTO w fechaNacimiento FROM PERSONAS WHERE dni=:NEW.dni;
   IF (:NEW.OID Tut IS NULL AND (NOT esMayorDeEdad(w fechaNacimiento) OR :NEW.gradoDiscapacidad > 0.5)) THEN
       raise application error(-20600, 'Si el participante es menor de edad o tiene un grado de discapacidad superior al 50% debe tener asignado un tutor
legal.');
   END IF:
END REPRESENTANTE LEGAL;
-- RN-5. Información médica
CREATE OR REPLACE TRIGGER INFORMACION MEDICA
BEFORE INSERT ON INSCRIPCIONES
FOR EACH ROW
DECLARE.
   n informes INTEGER;
BEGIN
   SELECT COUNT(*) INTO n_informes FROM INFORMESMEDICOS WHERE OID Part=:NEW.OID Part;
   IF (n informes=0) THEN
       raise_application_error(-20600, 'El participante debe tener asociado al menos un informe médico.');
   END IF;
END INFORMACION MEDICA;
-- RN-6. Prioridad de voluntariado
CREATE OR REPLACE TRIGGER PRIORIDAD VOLUNTARIADO
BEFORE UPDATE ON ESTAINTERESADOEN
FOR EACH ROW
DECLARE
   w OID Vol VOLUNTARIOS.OID Vol%TYPE;
   w_OID_Act ACTIVIDADES.OID_Act%TYPE;
   n ins3Meses INTEGER;
   n ins INTEGER;
BEGIN
   w OID Vol:=:OLD.OID Vol;
   w OID Act:=:OLD.OID Act;
   SELECT COUNT(*) INTO n ins3Meses FROM COLABORACIONES NATURAL JOIN ACTIVIDADES WHERE OID Vol=w OID Vol AND fechaInicio BETWEEN ADD MONTHS(SYSDATE, -3) AND
SYSDATE;
   SELECT COUNT(*) INTO n_ins FROM COLABORACIONES NATURAL JOIN ACTIVIDADES WHERE OID Vol=w_OID Vol;
   IF (n ins3Meses > 0) THEN
       UPDATE VOLUNTARIOS SET prioridadParticipacion='alta' WHERE OID Vol=w_OID Vol;
   ELSIF (n ins > 0) THEN
       UPDATE VOLUNTARIOS SET prioridadParticipacion='media' WHERE OID_Vol=w_OID_Vol;
   END IF:
END PRIORIDAD_VOLUNTARIADO;
```

```
CREATE OR REPLACE TRIGGER VOLUNTARIO INSCRITO
BEFORE INSERT ON COLABORACIONES
FOR EACH ROW
BEGIN
    UPDATE VOLUNTARIOS SET prioridadParticipacion='alta' WHERE OID Vol=:NEW.OID Vol;
END VOLUNTARIO INSCRITO;
-- RN-7. Estado de interés en actividades
CREATE OR REPLACE TRIGGER INTERES ACT PART
AFTER INSERT ON ACTIVIDADES
FOR EACH ROW
DECLARE
   w OID Part PARTICIPANTES.OID Part%TYPE:
   w OID Act ACTIVIDADES.OID Act%TYPE;
   CURSOR C PARTICIPANTES IS SELECT OID Part FROM PARTICIPANTES;
BEGIN
   w OID Act:=:NEW.OID Act;
   FOR participante IN C PARTICIPANTES LOOP
        w OID Part:=participante.OID Part;
        INSERT INTO ESTAINTERESADOEN(OID Vol, OID Part, OID Act, estado) VALUES (null, w OID Part, w OID Act, 0);
   END LOOP:
END ESTADO_INTERES_ACT_PART;
CREATE OR REPLACE TRIGGER INTERES ACT VOL
AFTER INSERT ON ACTIVIDADES
FOR EACH ROW
DECLARE
   w OID Vol VOLUNTARIOS.OID Vol%TYPE;
   w OID Act ACTIVIDADES.OID Act%TYPE;
   CURSOR C VOLUNTARIOS IS SELECT OID Vol FROM VOLUNTARIOS;
BEGIN
   w OID Act:=:NEW.OID Act;
   FOR voluntario IN C VOLUNTARIOS LOOP
        w OID Vol:=voluntario.OID Vol:
        INSERT INTO ESTAINTERESADOEN(OID_Vol, OID_Part, OID_Act, estado) VALUES (w_OID_Vol, null, w_OID_Act, 0);
   END LOOP;
END ESTADO_INTERES_ACT_VOL;
CREATE OR REPLACE TRIGGER INTERES INICIAL PART
AFTER INSERT ON PARTICIPANTES
FOR EACH ROW
DECLARE
   w_OID_Part PARTICIPANTES.OID_Part%TYPE;
   CURSOR C ACTIVIDADES IS SELECT * FROM ACTIVIDADES WHERE fechaInicio > SYSDATE;
BEGIN
   SELECT : NEW.OID Part INTO w OID Part FROM DUAL;
   FOR actividad IN C ACTIVIDADES LOOP
        INSERT INTO ESTAINTERESADOEN(OID Part, OID Act, estado) VALUES (w OID Part, actividad.OID Act, 0);
   END LOOP;
END INTERES_INICIAL_PART;
```

```
CREATE OR REPLACE TRIGGER INTERES INICIAL VOL
AFTER INSERT ON VOLUNTARIOS
FOR EACH ROW
DECLARE
   w OID Vol VOLUNTARIOS.OID Vol%TYPE;
   CURSOR C ACTIVIDADES IS SELECT * FROM ACTIVIDADES WHERE fechaInicio > SYSDATE;
BEGIN
   SELECT : NEW.OID Vol INTO w OID Vol FROM DUAL;
   FOR actividad IN C ACTIVIDADES LOOP
        INSERT INTO ESTAINTERESADOEN(OID Vol, OID Act, estado) VALUES (w OID Vol, actividad.OID Act, 0);
   END LOOP:
END INTERES INICIAL VOL;
-- RN-8. Envío de mensajes
CREATE OR REPLACE TRIGGER ENVIO MENSAJE
BEFORE INSERT ON ENVIOS
FOR EACH ROW
DECLARE
   mensaje MENSAJES%ROWTYPE;
BEGIN
   SELECT * INTO mensaje FROM MENSAJES WHERE OID M=:NEW.OID M;
   IF (mensaje.tipo='informe' AND :NEW.dni IS NOT NULL) THEN
        raise_application_error(-20600, 'Sólo los patrocinadores pueden recibir informes.');
   END IF:
END ENVIO MENSAJE;
-- RN-9. Tipo de patrocinador
CREATE OR REPLACE TRIGGER TIPO PATROCINADOR
AFTER INSERT ON PATROCINIOS
FOR EACH ROW
DECLARE
   w OID Proj INTEGER;
   w OID Act INTEGER:
   costeTotalProj ACTIVIDADES.costeTotal%TYPE;
   proyecto PROYECTOS%ROWTYPE;
BEGIN
   SELECT : NEW.OID Act INTO w OID Act FROM DUAL;
   SELECT OID Proj INTO w OID Proj FROM ACTIVIDADES WHERE OID Act=w OID Act;
   SELECT * INTO proyecto FROM PROYECTOS WHERE OID Proj=w OID Proj;
   SELECT SUM(costeTotal) INTO costeTotalProj FROM ACTIVIDADES WHERE OID Proj=w OID Proj;
   IF (:NEW.cantidad >= costeTotalProj) THEN
        UPDATE INSTITUCIONES SET tipo='oro' WHERE cif=:NEW.cif;
   ELSIF (:NEW.cantidad < costeTotalProj AND proyecto.esProgDep = 1) THEN
       UPDATE INSTITUCIONES SET tipo='plata' WHERE cif=:NEW.cif;
   ELSIF (:NEW.cantidad < costeTotalProj AND proyecto.esEvento = 1) THEN
       UPDATE INSTITUCIONES SET tipo='bronce' WHERE cif=:NEW.cif;
   END IF:
END TIPO_PATROCINADOR;
```

```
-- RN-11. Coste de inscripciones
CREATE OR REPLACE TRIGGER COSTE INSCRIPCION
AFTER INSERT ON PATROCINIOS
FOR EACH ROW
DECLARE
   w_OID_Act INTEGER;
   costesIns ACTIVIDADES.costeTotal%TYPE;
   actividad ACTIVIDADES%ROWTYPE;
BEGIN
   SELECT : NEW.OID Act INTO w OID Act FROM DUAL;
   SELECT * INTO actividad FROM ACTIVIDADES WHERE OID Act=w OID Act;
   IF (:NEW.cantidad > actividad.costeTotal) THEN
        UPDATE ACTIVIDADES SET costeInscripcion=0 WHERE OID_Act=w_OID_Act;
   ELSE
        costesIns:=actividad.costeTotal - :NEW.cantidad;
       UPDATE ACTIVIDADES SET costeInscripcion=calcularCosteInscripcion(costesIns, actividad.numeroPlazas) WHERE OID Act=w OID Act;
   END IF;
END COSTE INSCRIPCION;
-- RN-12. Plazas de inscripciones
CREATE OR REPLACE TRIGGER PLAZAS INSCRIPCIONES
BEFORE INSERT ON INSCRIPCIONES
FOR EACH ROW
DECLARE
   n inscritos INTEGER;
   w numeroPlazas ACTIVIDADES.numeroPlazas%TYPE;
BEGIN
   SELECT numeroPlazas INTO w numeroPlazas FROM ACTIVIDADES WHERE OID Act=:NEW.OID Act;
   SELECT COUNT(OID Part) INTO n inscritos FROM INSCRIPCIONES WHERE OID Act=:NEW.OID Act;
   IF (n inscritos = w numeroPlazas) THEN
        raise_application_error(-20600, 'No hay más plazas de inscripción disponibles');
   END IF:
END PLAZAS_INSCRIPCIONES;
-- RN-13. Plazo de respuesta de cuestionario
CREATE OR REPLACE TRIGGER PLAZO CUESTIONARIO
BEFORE INSERT ON RESPUESTAS
FOR EACH ROW
DECLARE
   w_fechaCreacion DATE;
BEGIN
   SELECT fechaCreacion INTO w fechaCreacion FROM FORMULARIOS NATURAL JOIN CUESTIONARIOS WHERE OID Form=:NEW.OID Form;
   IF (SYSDATE NOT BETWEEN w fechaCreacion AND w fechaCreacion+15) THEN
        raise application error(-20602, 'Sólo puede responderse al formulario durante los primeros 15 días desde su fecha de creación.');
END PLAZO CUESTIONARIO;
```

```
-- RN-15. Cálculo de voluntarios necesarios por actividad

CREATE OR REPLACE TRIGGER VOL_REQUERIDOS

AFTER INSERT ON INSCRIPCIONES

FOR EACH ROW

DECLARE

n_voluntarios ACTIVIDADES.voluntariosRequeridos%TYPE;

BEGIN

SELECT voluntariosRequeridos INTO n_voluntarios FROM ACTIVIDADES WHERE OID_Act=:NEW.OID_Act;
n_voluntarios:=n_voluntarios + 1;

UPDATE ACTIVIDADES SET voluntariosRequeridos=n_voluntarios WHERE OID_Act=:NEW.OID_Act;

END VOL_REQUERIDOS;
/
```

## Pruebas

```
Archivo: 5 pruebas IS-G1-SSR dep&des.sql
SET SERVEROUTPUT ON:
--ESPECIFICACIÓN DEL PAQUETE
-----
--PRUEBAS PERSONAS
create or replace
PACKAGE PRUEBAS PERSONAS AS
   PROCEDURE inicializar:
   PROCEDURE insertar
       (nombre prueba VARCHAR2, w dni CHAR, w nombre VARCHAR2, w apellidos VARCHAR2, w fechaNacimiento DATE, w direccion VARCHAR,
       w localidad VARCHAR2, w provincia VARCHAR2, w codigoPostal CHAR, w email VARCHAR, w telefono CHAR, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
       (nombre prueba VARCHAR2, w dni CHAR, w nombre VARCHAR2, w apellidos VARCHAR2, w fechaNacimiento DATE, w direccion VARCHAR,
       w localidad VARCHAR2, w provincia VARCHAR2, w codigoPostal CHAR, w email VARCHAR, w telefono CHAR, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
       (nombre prueba VARCHAR2, w dni CHAR, salidaEsperada BOOLEAN);
END PRUEBAS PERSONAS;
-- PRUEBAS COORDINADORES
create or replace
PACKAGE PRUEBAS COORDINADORES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
       (nombre prueba VARCHAR2, w dni CHAR, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Coord INTEGER, w dni CHAR, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
       (nombre_prueba VARCHAR2, w_OID_Coord INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS_COORDINADORES;
-- PRUEBAS TUTORESLEGALES
create or replace
PACKAGE PRUEBAS TUTORESLEGALES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
       (nombre_prueba VARCHAR2, w_dni CHAR, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
       (nombre prueba VARCHAR2, w OID Tut INTEGER, w dni CHAR, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
       (nombre_prueba VARCHAR2, w_OID_Tut INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS TUTORESLEGALES:
```

```
-- PRUEBAS VOLUNTARIOS
create or replace
PACKAGE PRUEBAS VOLUNTARIOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w dni CHAR, w prioridadParticipacion VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Vol INTEGER, w dni CHAR, w prioridadParticipacion VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Vol INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS VOLUNTARIOS:
-- PRUEBAS PARTICIPANTES
create or replace
PACKAGE PRUEBAS PARTICIPANTES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w dni CHAR, w gradoDiscapacidad VARCHAR2, w prioridadParticipacion VARCHAR2, w OID Tut INTEGER,
            w OID Vol INTEGER, salidaEsperada BOOLEAN):
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Part INTEGER, w dni CHAR, w gradoDiscapacidad VARCHAR2, w prioridadParticipacion VARCHAR2,
            w_OID_Tut INTEGER, w_OID_Vol INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Part INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS PARTICIPANTES;
-- PRUEBAS INFORMESMEDICOS
create or replace
PACKAGE PRUEBAS INFORMESMEDICOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Part INTEGER, w descripcion LONG, w fecha DATE, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Inf INTEGER, w OID Part INTEGER, w descripcion LONG, w fecha DATE, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Inf INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS INFORMESMEDICOS;
-- PRUEBAS ESTAINTERESADOEN
create or replace
PACKAGE PRUEBAS ESTAINTERESADOEN AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Part INTEGER, w OID Vol INTEGER, w OID Act INTEGER, w estado SMALLINT , salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Int INTEGER, w OID Part INTEGER, w OID Vol INTEGER, w OID Act INTEGER, w estado SMALLINT,
            salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Int INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS ESTAINTERESADOEN;
```

```
-- PRUEBAS INSTITUCIONES
create or replace
PACKAGE PRUEBAS INSTITUCIONES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
       (nombre prueba VARCHAR2, w cif CHAR, w nombre VARCHAR2, w telefono CHAR, w direccion VARCHAR2, w localidad VARCHAR2, w provincia VARCHAR2,
       w codigoPostal CHAR, w email VARCHAR2, w esPatrocinador SMALLINT, w tipo VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
       (nombre prueba VARCHAR2, w cif CHAR, w nombre VARCHAR2, w telefono CHAR, w direccion VARCHAR2, w localidad VARCHAR2, w provincia VARCHAR2.
       w codigoPostal CHAR, w email VARCHAR2, w esPatrocinador SMALLINT, w tipo VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w cif CHAR, salidaEsperada BOOLEAN);
END PRUEBAS INSTITUCIONES:
--PRUEBAS PROYECTOS
create or replace
PACKAGE PRUEBAS PROYECTOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w nombre VARCHAR2, w ubicacion CHAR, w esEvento SMALLINT, w esProgDep SMALLINT, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre_prueba VARCHAR2, w_OID_Proj INTEGER, w_nombre VARCHAR2, w_ubicacion CHAR, w_esEvento SMALLINT, w_esProgDep SMALLINT,
            salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Proj INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS PROYECTOS;
--PRUEBAS ACTIVIDADES
create or replace
PACKAGE PRUEBAS ACTIVIDADES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Proj INTEGER, w nombre VARCHAR2, w objetivo VARCHAR2, w fechaInicio DATE, w fechaFin DATE,
            w numeroPlazas INTEGER, w voluntariosRequeridos INTEGER, w tipo VARCHAR2, w costeTotal NUMBER, w costeInscripcion NUMBER,
            salidaEsperada BOOLEAN);
   PROCEDURE actualizar
       (nombre prueba VARCHAR2, w OID Act INTEGER, w OID Proj INTEGER, w nombre VARCHAR2, w objetivo VARCHAR2, w fechaInicio DATE,
            w fechaFin DATE, w numeroPlazas INTEGER, w voluntariosRequeridos INTEGER, w tipo VARCHAR2, w costeTotal NUMBER,
            w costeInscripcion NUMBER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Act INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS_ACTIVIDADES;
```

```
-- PRUEBAS ENCARGADOS
create or replace
PACKAGE PRUEBAS ENCARGADOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Proj INTEGER, w OID Coord INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID RP INTEGER, w OID Proj INTEGER, w OID Coord INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID RP INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS ENCARGADOS;
-- PRUEBAS COLABORACIONES
create or replace
PACKAGE PRUEBAS COLABORACIONES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Vol INTEGER, w OID Act INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Colab INTEGER, w OID Vol INTEGER, w OID Act INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre_prueba VARCHAR2, w_OID_Colab INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS COLABORACIONES:
--PRUEBAS RECIBOS
create or replace
PACKAGE PRUEBAS RECIBOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Act INTEGER, w OID Part INTEGER, w fechaEmision DATE, w fechaVencimiento DATE, w importe NUMBER,
            w estado VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Rec INTEGER, w OID Act INTEGER, w OID Part INTEGER, w fechaEmision DATE, w fechaVencimiento DATE,
            w importe NUMBER, w estado VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Rec INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS RECIBOS;
-- PRUEBAS INSCRIPCIONES
create or replace
PACKAGE PRUEBAS INSCRIPCIONES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Part INTEGER, w OID Act INTEGER, w OID Rec INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Ins INTEGER, w OID Part INTEGER, w OID Act INTEGER, w OID Rec INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Ins INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS INSCRIPCIONES;
```

```
-- PRUEBAS PATROCIONIOS
create or replace
PACKAGE PRUEBAS PATROCINIOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w cif CHAR, w cantidad NUMBER, w OID Act INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre_prueba VARCHAR2, w_OID_Fin INTEGER, w_cif CHAR , w_cantidad NUMBER, w_OID_Act INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Fin INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS PATROCINIOS;
-- PRUEBAS TIPOSDONACIONES
create or replace
PACKAGE PRUEBAS TIPODONACIONES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w nombre VARCHAR2, w tipoUnidad VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID TDon INTEGER, w nombre VARCHAR2, w tipoUnidad VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre_prueba VARCHAR2, w_OID_TDon INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS TIPODONACIONES:
--PRUEBAS DONACIONES
create or replace
PACKAGE PRUEBAS DONACIONES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w cif CHAR, w dni CHAR, w OID TDon INTEGER, w cantidad NUMBER, w valorUnitario NUMBER, w fecha DATE,
            salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Don INTEGER, w cif CHAR, w dni CHAR, w OID TDon INTEGER, w cantidad NUMBER, w valorUnitario NUMBER,
            w fecha DATE, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Don INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS DONACIONES;
-- PRUEBAS MENSAJES
create or replace
PACKAGE PRUEBAS MENSAJES AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Coord INTEGER, w tipo VARCHAR2, w fechaEnvio DATE, w asunto VARCHAR2, w contenido LONG,
            salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre_prueba VARCHAR2, w OID M INTEGER, w OID Coord INTEGER, w tipo VARCHAR2, w fechaEnvio DATE, w asunto VARCHAR2,
            w contenido LONG, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID M INTEGER, salidaEsperada BOOLEAN);
```

```
END PRUEBAS MENSAJES;
--PRUEBAS ENVIOS
create or replace
PACKAGE PRUEBAS ENVIOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID M INTEGER, W dni CHAR, w cif CHAR, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2,w OID Env INTEGER, w OID M INTEGER, w dni CHAR, w cif CHAR, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Env INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS ENVIOS:
-- PRUEBAS PREGUNTAS
create or replace
PACKAGE PRUEBAS PREGUNTAS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w tipo VARCHAR2, w enunciado VARCHAR, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID_Q INTEGER, w tipo VARCHAR2, w enunciado VARCHAR, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre_prueba VARCHAR2, w_OID_Q INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS PREGUNTAS;
-- PRUEBAS CUESTIONARIOS
create or replace
PACKAGE PRUEBAS_CUESTIONARIOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre_prueba VARCHAR2, w_OID_ACT INTEGER, w_fechaCreacion DATE, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2,w OID Cues INTEGER, w OID ACT INTEGER,w fechaCreacion DATE, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Cues INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS CUESTIONARIOS;
-- PRUEBAS FORMULARIOS
create or replace
PACKAGE PRUEBAS FORMULARIOS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
        (nombre prueba VARCHAR2, w OID Cues INTEGER, w OID Q INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
        (nombre prueba VARCHAR2, w OID Form INTEGER, w OID Cues INTEGER, w OID Q INTEGER, salidaEsperada BOOLEAN);
   PROCEDURE eliminar
        (nombre prueba VARCHAR2, w OID Form INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS_fORMULARIOS;
```

```
-- PRUEBAS RESPUESTAS
create or replace
PACKAGE PRUEBAS RESPUESTAS AS
   PROCEDURE inicializar;
   PROCEDURE insertar
       (nombre prueba VARCHAR2, w OID Form INTEGER, w OID Part INTEGER, w OID Vol INTEGER, w contenido VARCHAR2, salidaEsperada BOOLEAN);
   PROCEDURE actualizar
       (nombre prueba VARCHAR2, w OID Ans INTEGER, w OID Form INTEGER, w OID Part INTEGER, w OID Vol INTEGER, w contenido VARCHAR2,
           salidaEsperada BOOLEAN);
   PROCEDURE eliminar
       (nombre prueba VARCHAR2, w OID Ans INTEGER, salidaEsperada BOOLEAN);
END PRUEBAS RESPUESTAS;
-----
--CUERPO DEL PAOUETE
-----
--PRUEBAS PERSONAS
create or replace
PACKAGE BODY PRUEBAS PERSONAS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM PERSONAS;
   END inicializar:
   PROCEDURE insertar(nombre prueba VARCHAR2, w dni CHAR, w nombre VARCHAR2, w apellidos VARCHAR2, w fechaNacimiento DATE, w direccion VARCHAR,
       w localidad VARCHAR2, w provincia VARCHAR2, w codigoPostal CHAR, w email VARCHAR, w telefono CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   persona PERSONAS%ROWTYPE;
   BEGIN
       INSERT INTO PERSONAS VALUES( w dni, w nombre, w apellidos, w fechaNacimiento, w direccion,
       w localidad, w provincia, w codigoPostal, w email, w telefono);
       SELECT * INTO persona FROM PERSONAS WHERE dni=w dni;
       IF (persona.dni<>w dni AND persona.nombre<>w nombre AND persona.apellidos<>w apellidos AND persona.fechaNacimiento<>w fechaNacimiento
           AND persona direccion<>w direccion AND persona localidad<>w localidad AND persona provincia<>w provincia
           AND persona.codigoPostal<>w_codigoPostal AND persona.email<>w_email AND persona.telefono<>w_telefono) THEN
           salida:=false;
       END IF;
       COMMIT WORK;
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
           ROLLBACK;
   END insertar;
```

```
PROCEDURE actualizar(nombre prueba VARCHAR2, w dni CHAR, w nombre VARCHAR2, w apellidos VARCHAR2, w fechaNacimiento DATE, w direccion VARCHAR,
        w localidad VARCHAR2, w provincia VARCHAR2, w codigoPostal CHAR, w email VARCHAR, w telefono CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true:
   persona PERSONAS%ROWTYPE;
   BEGIN
   UPDATE PERSONAS SET nombre=w nombre, apellidos=w apellidos, fechaNacimiento=w fechaNacimiento, direccion=w direccion,
        localidad=w localidad,provincia=w provincia, codigoPostal=w codigoPostal, email=w email, telefono=w telefono WHERE dni=w dni;
   SELECT * INTO persona FROM PERSONAS WHERE dni= w dni;
   IF (persona.dni<>w dni AND persona.nombre<>w nombre AND persona.apellidos<>w apellidos AND persona.fechaNacimiento<>w fechaNacimiento
        AND persona.direccion<br/><br/>yw direccion AND persona.localidad<br/><br/>yw localidad AND persona.provincia<br/><br/>yw provincia
        AND persona.codigoPostal<>w codigoPostal AND persona.email<>w email AND persona.telefono<>w telefono) THEN
        salida:=false;
   END IF:
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(false, salidaEsperada));
      ROLLBACK:
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w dni CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n_personas INTEGER;
   BEGIN
   DELETE FROM PERSONAS WHERE dni=w dni;
   SELECT COUNT (*) INTO n personas FROM PERSONAS WHERE dni=w dni;
   IF(n personas<>0) THEN
        salida:=false;
   END IF;
   COMMIT WORK;
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
        ROLLBACK:
   END eliminar;
END PRUEBAS_PERSONAS;
```

```
-- PRUEBAS COORDINADORES
create or replace
PACKAGE BODY PRUEBAS_COORDINADORES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM COORDINADORES;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w dni CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   coordinador COORDINADORES%ROWTYPE;
   w OID Coord INTEGER;
   BEGIN
       INSERT INTO COORDINADORES(dni) VALUES(w_dni);
       w OID Coord := sec Coord.currval;
       SELECT * INTO coordinador FROM COORDINADORES WHERE OID Coord=w OID Coord;
       IF (coordinador.dni<>w dni) THEN
           salida:=false;
       END IF:
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
           ROLLBACK;
   END insertar;
    PROCEDURE actualizar (nombre prueba VARCHAR2, w OID Coord INTEGER, w dni CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN := true;
   coordinador COORDINADORES%ROWTYPE;
   BEGIN
       UPDATE COORDINADORES SET dni=w dni WHERE OID Coord=w OID Coord:
       SELECT * INTO coordinador FROM COORDINADORES WHERE OID Coord=w_OID Coord;
       IF (coordinador.dni<>w dni) THEN
            salida := false;
       END IF;
       COMMIT WORK;
           DBMS_OUTPUT.PUT_LINE(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.PUT_LINE(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
   ROLLBACK;
   END actualizar;
```

```
PROCEDURE eliminar(nombre prueba VARCHAR2,w OID Coord INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n coordinadores INTEGER;
   BEGIN
   DELETE FROM COORDINADORES WHERE OID Coord=w OID Coord;
   SELECT COUNT (*) INTO n coordinadores FROM COORDINADORES WHERE OID Coord=w OID Coord;
   IF(n coordinadores<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
       ROLLBACK;
   END eliminar;
END PRUEBAS COORDINADORES;
-- PRUEBAS TUTORESLEGALES
create or replace
PACKAGE BODY PRUEBAS TUTORESLEGALES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM TUTORESLEGALES;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w dni CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   tutorlegal TUTORESLEGALES%ROWTYPE;
   w OID Tut INTEGER;
   BEGIN
       INSERT INTO TUTORESLEGALES(dni) VALUES( w dni);
       w OID Tut:= sec Tut.currval;
       SELECT * INTO tutorlegal FROM TUTORESLEGALES WHERE dni=w dni;
       IF (tutorlegal.dni<>w_dni) THEN
            salida:=false;
        END IF;
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
            ROLLBACK;
   END insertar;
```

```
PROCEDURE actualizar (nombre prueba VARCHAR2, w OID Tut INTEGER, w dni CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN := true:
   tutorlegal TUTORESLEGALES%ROWTYPE;
   BEGIN
       UPDATE TUTORESLEGALES SET dni=w dni WHERE OID Tut=w OID Tut;
       SELECT * INTO tutorlegal FROM TUTORESLEGALES WHERE OID Tut=w OID Tut;
       IF (tutorlegal.dni<>w dni) THEN
            salida := false:
       END IF;
       COMMIT WORK;
       DBMS OUTPUT.PUT LINE(nombre prueba || ':' || ASSERT EQUALS(salida,salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
            DBMS OUTPUT.PUT LINE(nombre prueba || ':' || ASSERT EQUALS(false,salidaEsperada));
       ROLLBACK;
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Tut INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n tutoreslegales INTEGER:
   BEGIN
   DELETE FROM TUTORESLEGALES WHERE OID Tut=w OID Tut;
   SELECT COUNT (*) INTO n tutoreslegales FROM TUTORESLEGALES WHERE OID Tut=w OID Tut;
   IF(n tutoreslegales<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false,salidaEsperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS TUTORESLEGALES;
--PRUEBAS VOLUNTARIOS
create or replace
PACKAGE BODY PRUEBAS VOLUNTARIOS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM VOLUNTARIOS;
   END inicializar;
   PROCEDURE insertar(nombre_prueba VARCHAR2, w_dni CHAR, w_prioridadParticipacion VARCHAR2, salidaEsperada BOOLEAN) AS
```

```
salida BOOLEAN:=true:
voluntario VOLUNTARIOS%ROWTYPE;
w OID Vol INTEGER:
BEGIN
    INSERT INTO VOLUNTARIOS(dni, prioridadParticipacion) VALUES(w dni, w prioridadParticipacion):
    w OID Vol:=SEC VOL.CURRVAL;
    SELECT * INTO voluntario FROM VOLUNTARIOS WHERE OID Vol= w OID Vol;
    IF (voluntario.dni<>w dni or voluntario.prioridadParticipacion<>w prioridadParticipacion ) THEN
    END IF;
    COMMIT WORK:
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
        DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
        ROLLBACK;
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Vol INTEGER, w dni CHAR, w prioridadParticipacion VARCHAR2, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
voluntario VOLUNTARIOS%ROWTYPE:
BEGIN
UPDATE VOLUNTARIOS SET dni=w_dni, prioridadParticipacion=w_prioridadParticipacion WHERE OID_Vol= w_OID_Vol;
SELECT * INTO voluntario FROM VOLUNTARIOS WHERE OID Vol = w OID Vol :
IF (voluntario.dni<>w dni or voluntario.prioridadParticipacion<>w prioridadParticipacion) THEN
salida:=false;
END IF;
COMMIT WORK:
DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
   ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Vol INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n voluntarios INTEGER;
BEGIN
DELETE FROM VOLUNTARIOS WHERE OID Vol= w OID Vol :
SELECT COUNT (*) INTO n voluntarios FROM VOLUNTARIOS WHERE OID Vol= w OID Vol;
IF(n voluntarios<>0) THEN
    salida:=false;
END IF;
COMMIT WORK;
```

```
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false,salidaEsperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS VOLUNTARIOS;
--PRUEBAS_PARTICIPANTES
create or replace
PACKAGE BODY PRUEBAS PARTICIPANTES AS
   PROCEDURE inicializar AS
   BEGTN
       DELETE FROM PARTICIPANTES:
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w dni CHAR, w gradoDiscapacidad VARCHAR2, w prioridadParticipacion VARCHAR2, w OID Tut INTEGER,
       w OID Vol INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   participante PARTICIPANTES%ROWTYPE;
   w OID Part INTEGER:
   BEGIN
       INSERT INTO PARTICIPANTES(dni, gradoDiscapacidad, prioridadParticipacion, OID Tut, OID Vol) VALUES( w dni, w gradoDiscapacidad,
            w prioridadParticipacion, w OID Tut, w OID Vol);
       w OID Part:= SEC PART.CURRVAL;
       SELECT * INTO participante FROM PARTICIPANTES WHERE OID Part=w OID Part;
       IF (participante.dni<>w dni AND participante.gradoDiscapacidad<>w gradoDiscapacidad AND
            participante.prioridadParticipacion<>w prioridadParticipacion AND participante.OID Tut<>w OID Tut
            AND participante.OID Vol<>w OID Vol) THEN
            salida:=false;
        END IF;
       COMMIT WORK:
       DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
            ROLLBACK;
   END insertar;
   PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Part INTEGER, w dni CHAR, w gradoDiscapacidad VARCHAR2, w prioridadParticipacion VARCHAR2,
       w OID Tut INTEGER, w OID Vol INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   participante PARTICIPANTES%ROWTYPE;
       UPDATE PARTICIPANTES SET dni=w dni, gradoDiscapacidad=w gradoDiscapacidad, prioridadParticipacion=w prioridadParticipacion,
           OID Tut=w OID Tut, OID Vol=w OID Vol WHERE OID Part= w OID Part;
```

```
SELECT * INTO participante FROM PARTICIPANTES WHERE OID Part = w OID Part :
       IF (participante.dni<>w dni or participante.gradoDiscapacidad<>w gradoDiscapacidad
            or participante.prioridadParticipacion<>w prioridadParticipacion or participante.OID Tut<>w OID Tut or participante.OID Vol<>w OID Vol) THEN
            salida:=false;
        END IF:
       COMMIT WORK;
       DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(false, salidaEsperada));
       ROLLBACK:
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Part INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n participantes INTEGER;
   BEGTN
       DELETE FROM PARTICIPANTES WHERE OID Part= w OID Part;
       SELECT COUNT (*) INTO n participantes FROM PARTICIPANTES WHERE OID Part = w OID Part;
       IF(n_participantes<>0) THEN
            salida:=false:
       END IF;
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EOUALS(false, salida Esperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS PARTICIPANTES;
-- PRUEBAS INFORMESMEDICOS
create or replace
PACKAGE BODY PRUEBAS INFORMESMEDICOS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM INFORMESMEDICOS;
   END inicializar;
   PROCEDURE insertar(nombre_prueba VARCHAR2, w_OID_Part INTEGER, w_descripcion LONG, w_fecha DATE, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   informeMedico INFORMESMEDICOS%ROWTYPE;
   w OID Inf INTEGER;
   BEGIN
       INSERT INTO INFORMESMEDICOS(OID Part, descripcion, fecha) VALUES( w OID Part, w descripcion, w fecha);
```

```
w OID Inf:= SEC INF.CURRVAL:
    SELECT * INTO informeMedico FROM INFORMESMEDICOS WHERE OID Inf=w OID Inf;
    IF (informeMedico.OID Part<>w OID Part AND informeMedico.descripcion<>w descripcion AND informemedico.fecha<>w fecha) THEN
        salida:=false:
    END IF:
    COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
        ROLLBACK:
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Inf INTEGER, w OID Part INTEGER, w descripcion LONG, w fecha DATE, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
informeMedico INFORMESMEDICOS%ROWTYPE;
BEGTN
UPDATE INFORMESMEDICOS SET OID Part=w OID Part, descripcion=w descripcion, fecha=w fecha WHERE OID Inf= w OID Inf;
SELECT * INTO informeMedico FROM INFORMESMEDICOS WHERE OID Inf= w OID Inf:
IF (informeMedico.OID_Part<>w_OID_Part AND informeMedico.descripcion<>w_descripcion AND informemedico.fecha<>w_fecha) THEN
salida:=false:
END IF;
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
  DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(false, salidaEsperada));
   ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre_prueba VARCHAR2, w_OID_Inf INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n informesMedicos INTEGER;
BEGIN
DELETE FROM INFORMESMEDICOS WHERE OID Inf= w OID Inf;
SELECT COUNT (*) INTO n informesMedicos FROM INFORMESMEDICOS WHERE OID Inf= w OID Inf;
IF(n informesMedicos<>0) THEN
    salida:=false;
END IF;
COMMIT WORK:
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
```

```
ROLLBACK:
   END eliminar;
END PRUEBAS INFORMESMEDICOS;
-- PRUEBAS ESTAINTERESADOEN
create or replace
PACKAGE BODY PRUEBAS ESTAINTERESADOEN AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM ESTAINTERESADOEN;
   END inicializar;
   PROCEDURE insertar(nombre_prueba VARCHAR2, w_OID_Part INTEGER, w_OID_Vol INTEGER, w_OID_Act INTEGER, w_estado SMALLINT, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   estaInteresado ESTAINTERESADOEN%ROWTYPE;
   w OID Int INTEGER;
   BEGIN
       INSERT INTO ESTAINTERESADOEN(OID Part, OID Vol, OID Act, estado) VALUES(w OID Part, w OID Vol, w OID Act, w estado);
       w OID Int:= SEC INT.CURRVAL;
       SELECT * INTO estaInteresado FROM ESTAINTERESADOEN WHERE OID Int=w OID Int;
       IF (estaInteresado.OID Part<>w_OID Part AND estaInteresado.OID Vol<>w_OID Vol AND estaInteresado.OID Act<>w_OID Act
            AND estaInteresado.estado<>w estado) THEN
            salida:=false;
        END IF;
       COMMIT WORK;
       DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
            ROLLBACK;
   END insertar:
   PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Int INTEGER, w OID Part INTEGER, w OID Vol INTEGER, w OID Act INTEGER,
       w estado SMALLINT , salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   estaInteresado ESTAINTERESADOEN%ROWTYPE;
   BEGIN
   UPDATE ESTAINTERESADOEN SET OID Part=w OID Part, OID Vol=w OID Vol, OID Act=w OID Act, estado=w estado WHERE OID Int= w OID Int;
   SELECT * INTO estaInteresado FROM ESTAINTERESADOEN WHERE OID Int= w OID Int;
   IF (estaInteresado.OID_Part<>w_OID_Part AND estaInteresado.OID_Vol<>w_OID_Vol AND estaInteresado.OID_Act<>w_OID_Act
       AND estaInteresado.estado<>w estado) THEN
   salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
```

```
EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
      ROLLBACK:
   END actualizar:
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Int INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n estaInteresadoEn INTEGER;
   BEGIN
   DELETE FROM ESTAINTERESADOEN WHERE OID Int= w OID Int;
   SELECT COUNT (*) INTO n estaInteresadoEn FROM ESTAINTERESADOEN WHERE OID Int= w OID Int;
   IF(n estaInteresadoEn<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS ESTAINTERESADOEN;
--PRUEBAS INSTITUCIONES
create or replace
PACKAGE BODY PRUEBAS INSTITUCIONES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM INSTITUCIONES;
   END inicializar:
   PROCEDURE insertar(nombre prueba VARCHAR2, w cif CHAR, w nombre VARCHAR2, w telefono CHAR, w direccion VARCHAR2,
       w localidad VARCHAR2, w provincia VARCHAR2,w codigoPostal CHAR, w email VARCHAR2, w esPatrocinador SMALLINT,
       w tipo VARCHAR2, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   institucion INSTITUCIONES%ROWTYPE:
   BEGIN
       INSERT INTO INSTITUCIONES VALUES( w cif, w nombre, w telefono, w direccion, w localidad, w provincia, w codigoPostal,
            w_email, w_esPatrocinador, w_tipo);
       SELECT * INTO institucion FROM INSTITUCIONES WHERE cif=w cif:
       IF (institucion.cif<>w cif or institucion.nombre<>w nombre or institucion.telefono<>w telefono or institucion.direccion<>w direccion
            or institucion.localidad<>w localidad or institucion.provincia<>w provincia or institucion.codigoPostal<>w codigoPostal
            or institucion.email<>w email or institucion.esPatrocinador<>w esPatrocinador or institucion.tipo<>w tipo) THEN
            salida:=false;
       END IF;
       COMMIT WORK;
```

```
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
        DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
        ROLLBACK:
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w cif CHAR, w nombre VARCHAR2, w telefono CHAR, w direccion VARCHAR2,
    w localidad VARCHAR2, w provincia VARCHAR2, w codigoPostal CHAR, w email VARCHAR2, w esPatrocinador SMALLINT,
    w tipo VARCHAR2, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true:
institucion INSTITUCIONES%ROWTYPE;
BEGIN
UPDATE INSTITUCIONES SET nombre=w nombre, telefono=w telefono, direccion=w direccion, localidad=w localidad,
provincia=w provincia, codigoPostal=w codigoPostal, email=w email, esPatrocinador=w esPatrocinador, tipo=w tipo WHERE cif=w cif;
SELECT * INTO institucion FROM INSTITUCIONES WHERE cif= w cif;
IF (institucion.cif<>w cif or institucion.nombre<>w nombre or institucion.telefono<>w telefono or institucion.direccion<>w
    or institucion.localidad<>w localidad or institucion.provincia<>w provincia or institucion.codigoPostal<>w codigoPostal
    or institucion.email<>w email or institucion.esPatrocinador<>w esPatrocinador or institucion.tipo<>w tipo) THEN
    salida:=false;
END IF:
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
   ROLLBACK;
END actualizar;
PROCEDURE eliminar(nombre prueba VARCHAR2, w cif CHAR, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n instituciones INTEGER;
BEGIN
DELETE FROM INSTITUCIONES WHERE cif=w cif;
SELECT COUNT (*) INTO n_instituciones FROM INSTITUCIONES WHERE cif=w_cif;
IF(n instituciones<>0) THEN
    salida:=false;
END IF;
COMMIT WORK;
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
    DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
    ROLLBACK;
```

```
END eliminar:
END PRUEBAS INSTITUCIONES;
-- PRUEBAS PROYECTOS
create or replace
PACKAGE BODY PRUEBAS PROYECTOS AS
PROCEDURE inicializar AS
BEGIN
   DELETE FROM PROYECTOS:
END inicializar;
PROCEDURE insertar(nombre prueba VARCHAR2, w nombre VARCHAR2, w ubicacion CHAR, w esEvento SMALLINT, w esProgDep SMALLINT, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   provecto PROYECTOS%ROWTYPE;
   w OID Proj INTEGER;
BEGIN
   INSERT INTO PROYECTOS(nombre, ubicacion, esEvento, esProgDep) VALUES(w nombre, w ubicacion, w esEvento, w esProgDep);
   w OID Proj:=SEC PROJ.CURRVAL;
   SELECT * INTO proyecto FROM PROYECTOS WHERE OID Proj=w OID Proj;
   IF (proyecto.nombreND proyecto.ubicacionIF (proyecto.nombreND proyecto.esProgDepNw_esProgDepNw_esProgDepOr No.
       salida:=false:
   END IF;
   COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
           ROLLBACK;
END insertar;
PROCEDURE actualizar(nombre_prueba VARCHAR2,w_OID_Proj INTEGER, w_nombre VARCHAR2, w_ubicacion CHAR, w_esEvento SMALLINT,
   w esProgDep SMALLINT, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   proyecto PROYECTOS%ROWTYPE;
BEGIN
   UPDATE PROYECTOS SET nombre=w nombre, ubicacion=w ubicacion, esEvento=w esEvento, esProgDep=w esProgDep WHERE OID Proj=w OID Proj;
   SELECT * INTO proyecto FROM PROYECTOS WHERE OID Proj=w OID Proj;
   IF (proyecto.nombre<>w nombre AND proyecto.ubicacion<>w ubicacion AND proyecto.esEvento<>w esEvento AND proyecto.esProgDep<>w esProgDep<) THEN
       salida:=false;
   END IF:
   COMMIT WORK;
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
```

```
DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
      ROLLBACK;
END actualizar;
PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Proj INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true:
   n proyectos INTEGER;
BEGIN
   DELETE FROM PROYECTOS WHERE OID Proi=w OID Proi:
   SELECT COUNT (*) INTO n proyectos FROM PROYECTOS WHERE OID Proj=w OID Proj;
   IF(n proyectos<>0) THEN
       salida:=false;
   END IF:
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
       ROLLBACK:
END eliminar;
END PRUEBAS PROYECTOS;
-- PRUEBAS ACTIVIDADES
create or replace
PACKAGE BODY PRUEBAS ACTIVIDADES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM ACTIVIDADES;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w OID Proj INTEGER, w nombre VARCHAR2, w objetivo VARCHAR2, w fechalnicio DATE,
       w_fechaFin_DATE, w_numeroPlazas INTEGER, w_voluntariosRequeridos INTEGER, w_tipo VARCHAR2, w_costeTotal NUMBER,
       w costeInscripcion NUMBER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   actividad ACTIVIDADES%ROWTYPE;
   w OID Act INTEGER;
       INSERT INTO ACTIVIDADES(OID Proj, nombre, objetivo, fechaInicio, fechaFin, numeroPlazas, voluntariosRequeridos, tipo,
            costeTotal, costeInscripcion) VALUES(w OID Proj, w nombre, w objetivo, w fechaInicio, w fechaFin, w numeroPlazas,
            w_voluntariosRequeridos, w_tipo, w_costeTotal, w_costeInscripcion);
       w OID Act:= SEC ACT.CURRVAL:
       SELECT * INTO actividad FROM ACTIVIDADES WHERE OID Act=w OID Act;
       IF (actividad.OID Proj<>w OID Proj AND actividad.nombre<>w nombre AND actividad.objetivo<>w objetivo
            AND actividad.fechaInicio<>w_fechaInicio AND actividad.fechaFin<>w_fechaFin AND actividad.numeroPlazas<>w_numeroPlazas
            AND actividad.voluntariosRequeridos<br/>>w voluntariosRequeridos AND actividad.tipo<>w tipo AND actividad.costeTotal<>w costeTotal
            AND actividad.costeInscripcion<>w costeInscripcion) THEN
            salida:=false;
```

```
END IF:
    COMMIT WORK;
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
        DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Act INTEGER, w OID Proj INTEGER, w nombre VARCHAR2, w objetivo VARCHAR2,
    w fechaInicio DATE, w fechaFin DATE, w numeroPlazas INTEGER, w voluntariosRequeridos INTEGER, w tipo VARCHAR2, w costeTotal NUMBER,
    w costeInscripcion NUMBER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
actividad ACTIVIDADES%ROWTYPE:
BEGIN
UPDATE ACTIVIDADES SET OID Proj=w OID Proj, nombre=w nombre, objetivo=w objetivo, fechaInicio=w fechaInicio, fechaInicio, fechaFin=w fechaFin,
    numeroPlazas=w numeroPlazas, voluntariosRequeridos=w voluntariosRequeridos,tipo=w tipo, costeTotal=w costeTotal,
    costeInscripcion=w costeInscripcion WHERE OID Act=w OID Act;
SELECT * INTO actividad FROM ACTIVIDADES WHERE OID Act= w OID Act;
IF (actividad.OID_Proj<>w_OID_Proj AND actividad.nombre<>w_nombre AND actividad.objetivo<>w_objetivo AND actividad.fechaInicio<>w_fechaInicio
    AND actividad.fechaFinw fechaFin AND actividad.numeroPlazas>w numeroPlazas AND actividad.voluntariosRequeridos>w voluntariosRequeridos
    AND actividad.tipo<>w tipo AND actividad.costeTotal<>w costeTotal AND actividad.costeInscripcion<>w costeInscripcion) THEN
salida:=false;
END IF:
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EOUALS(false, salidaEsperada));
   ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Act INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n actividades INTEGER;
DELETE FROM ACTIVIDADES WHERE OID Act= w OID Act;
SELECT COUNT (*) INTO n actividades FROM ACTIVIDADES WHERE OID Act= w OID Act;
IF(n_actividades<>0) THEN
    salida:=false:
END IF;
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
```

**EXCEPTION** 

```
WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS ACTIVIDADES;
-- PRUEBAS ENCARGADOS
create or replace
PACKAGE BODY PRUEBAS ENCARGADOS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM ENCARGADOS:
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2,w OID Proj INTEGER, w OID Coord INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   encargado ENCARGADOS%ROWTYPE;
   w OID RP INTEGER;
   BEGIN
       INSERT INTO ENCARGADOS(OID Proj, OID Coord) VALUES(w OID Proj, w OID Coord);
       w_OID_RP:= SEC_RP.CURRVAL;
       SELECT * INTO encargado FROM ENCARGADOS WHERE OID RP=w OID RP:
       IF (encargado.OID Proj<>w OID Proj AND encargado.OID Coord<>w OID Coord) THEN
            salida:=false;
       END IF:
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
           ROLLBACK:
   END insertar;
   PROCEDURE actualizar(nombre_prueba VARCHAR2,w_OID_RP_INTEGER, w_OID_Proj_INTEGER, w_OID_Coord_INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   encargado ENCARGADOS%ROWTYPE;
   BEGIN
   UPDATE ENCARGADOS SET OID_Proj=w_OID_Proj, OID_Coord=w_OID_Coord WHERE OID_RP= w_OID_RP;
   SELECT * INTO encargado FROM ENCARGADOS WHERE OID RP= w OID RP;
   IF (encargado.OID_Proj<>w_OID_Proj AND encargado.OID_Coord<>w_OID_Coord) THEN
   salida:=false:
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
```

**EXCEPTION** 

```
WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(false, salidaEsperada));
      ROLLBACK:
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID RP INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n encargados INTEGER;
   BEGIN
   DELETE FROM ENCARGADOS WHERE OID RP= w OID RP;
   SELECT COUNT (*) INTO n_encargados FROM ENCARGADOS WHERE OID_RP= w_OID_RP;
   IF(n encargados<>0) THEN
      salida:=false:
   END IF;
   COMMIT WORK;
   EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
      ROLLBACK;
   END eliminar:
END PRUEBAS ENCARGADOS;
-- PRUEBAS COLABORACIONES
create or replace
PACKAGE BODY PRUEBAS COLABORACIONES AS
   PROCEDURE inicializar AS
   BEGIN
      DELETE FROM COLABORACIONES;
   END inicializar;
   PROCEDURE insertar(nombre_prueba VARCHAR2, w_OID_Vol INTEGER, w_OID_Act INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   colaboracion COLABORACIONES%ROWTYPE;
   w OID Colab INTEGER;
   BEGIN
      INSERT INTO COLABORACIONES(OID Vol, OID Act) VALUES(w OID Vol, w OID Act);
      w OID Colab:= SEC COLAB.CURRVAL;
      SELECT * INTO colaboracion FROM COLABORACIONES WHERE OID Colab=w OID Colab;
      IF (colaboracion.OID_Vol<>w_OID_Vol AND colaboracion.OID_Act<>w_OID_Act) THEN
          salida:=false:
      END IF;
      COMMIT WORK;
      EXCEPTION
```

```
WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
           ROLLBACK:
   END insertar;
   PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Colab INTEGER, w OID Vol INTEGER, w OID Act INTEGER, salidaEsperada BOOLEAN) AS
  salida BOOLEAN:=true;
   colaboracion COLABORACIONES%ROWTYPE;
   BEGIN
   UPDATE COLABORACIONES SET OID Vol=w OID Vol, OID Act=w OID Act WHERE OID Colab= w OID Colab;
   SELECT * INTO colaboracion FROM COLABORACIONES WHERE OID Colab= w OID Colab;
   IF (colaboracion.OID Vol<>w OID Vol AND colaboracion.OID Act<>w OID Act) THEN
   salida:=false:
   END IF;
  COMMIT WORK;
   EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
      ROLLBACK;
   END actualizar:
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Colab INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true:
   n colaboraciones INTEGER;
   BEGIN
   DELETE FROM COLABORACIONES WHERE OID Colab: w OID Colab;
   SELECT COUNT (*) INTO n colaboraciones FROM COLABORACIONES WHERE OID Colab= w OID Colab;
   IF(n colaboraciones<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK:
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
       ROLLBACK;
   END eliminar;
END PRUEBAS COLABORACIONES:
--PRUEBAS RECIBOS
create or replace
PACKAGE BODY PRUEBAS RECIBOS AS
   PROCEDURE inicializar AS
```

BEGIN

```
DELETE FROM RECIBOS:
END inicializar;
PROCEDURE insertar(nombre prueba VARCHAR2, w_OID_Act INTEGER, w_OID_Part INTEGER, w_fechaEmision DATE, w_fechaVencimiento DATE,
       w importe NUMBER, w estado VARCHAR2, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
recibo RECIBOS%ROWTYPE;
w OID Rec INTEGER;
BEGIN
       INSERT INTO RECIBOS(OID Act, OID Part, fechaEmision, fechaVencimiento, importe, estado) VALUES(w OID Act, w OID Part, w fechaEmision,
                w fechaVencimiento, w importe, w estado);
       w OID Rec:= SEC REC.CURRVAL;
       SELECT * INTO recibo FROM RECIBOS WHERE OID Rec=w OID Rec:
       IF (recibo.OID Act<>w OID Act AND recibo.OID Part<>w OID Part AND recibo.fechaEmision<>w fechaEmision
               AND recibo.fechaVencimiento
AND recibo.importew importe AND recibo.estado
AND recibo.estado
per and per a
               salida:=false:
       END IF;
       COMMIT WORK;
       DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
               DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
               ROLLBACK;
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Rec INTEGER, w OID Act INTEGER, w OID Part INTEGER, w fechaEmision DATE,
       w fechaVencimiento DATE, w importe NUMBER, w estado VARCHAR2, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true:
recibo RECIBOS%ROWTYPE;
BEGIN
UPDATE RECIBOS SET OID Act=w OID Act, OID Part=w OID Part, fechaEmision=w fechaEmision, fechaVencimiento=w fechaVencimiento,
       importe=w importe, estado=w estado WHERE OID Rec= w OID Rec:
SELECT * INTO recibo FROM RECIBOS WHERE OID_Rec= w_OID_Rec;
IF (recibo.OID Act<>w OID Act AND recibo.OID Part<>w OID Part AND recibo.fechaEmision<>w fechaEmision
       AND recibo.fechaVencimiento<>w fechaVencimiento
               AND recibo.importe<>w importe AND recibo.estado<>w estado) THEN
salida:=false;
END IF;
COMMIT WORK;
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
     ROLLBACK:
END actualizar;
```

```
PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Rec INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n recibos INTEGER;
   BEGIN
   DELETE FROM RECIBOS WHERE OID Rec= w OID Rec:
   SELECT COUNT (*) INTO n recibos FROM RECIBOS WHERE OID Rec= w OID Rec;
   IF(n recibos<>0) THEN
       salida:=false;
   END IF:
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS RECIBOS;
--PRUEBAS_INSCRIPCIONES
create or replace
PACKAGE BODY PRUEBAS INSCRIPCIONES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM INSCRIPCIONES;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w OID Part INTEGER, w OID Act INTEGER, w OID Rec INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   inscripcion INSCRIPCIONES%ROWTYPE;
   w OID Ins INTEGER;
   BEGIN
       INSERT INTO INSCRIPCIONES(OID Part, OID Act, OID Rec) VALUES(w_OID Part, w_OID Act, w_OID Rec);
       w OID Ins:= SEC INS.CURRVAL;
       SELECT * INTO inscripcion FROM INSCRIPCIONES WHERE OID Ins=w OID Ins;
       IF (inscripcion.OID Part<>w OID Part AND inscripcion.OID Act<>w OID Act AND inscripcion.OID Rec<>w OID Rec) THEN
            salida:=false;
       END IF;
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
   END insertar;
```

```
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Ins INTEGER, w OID Part INTEGER, w OID Act INTEGER, w OID Rec INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   inscripcion INSCRIPCIONES%ROWTYPE;
   BEGIN
   UPDATE INSCRIPCIONES SET OID Part=w OID Part, OID Act=w OID Act, OID Rec=w OID Rec WHERE OID Ins= w OID Ins;
   SELECT * INTO inscripcion FROM INSCRIPCIONES WHERE OID Ins= w OID Ins;
   IF (inscripcion.OID Part<>w OID Part AND inscripcion.OID Act<>w OID Act AND inscripcion.OID Rec<>w OID Rec) THEN
   salida:=false;
   END IF:
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
      ROLLBACK:
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Ins INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true:
   n_inscripciones INTEGER;
   BEGIN
   DELETE FROM INSCRIPCIONES WHERE OID Ins= w OID Ins;
   SELECT COUNT (*) INTO n inscripciones FROM INSCRIPCIONES WHERE OID Ins= w OID Ins;
   IF(n inscripciones<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK;
   END eliminar;
END PRUEBAS INSCRIPCIONES;
-- PRUEBAS PATROCIONIOS
create or replace
PACKAGE BODY PRUEBAS_PATROCINIOS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM PATROCINIOS;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w cif CHAR, w cantidad NUMBER, w OID Act INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
```

```
patrocinio PATROCINIOS%ROWTYPE:
w OID Fin INTEGER;
BEGIN
    INSERT INTO PATROCINIOS(cif, cantidad, OID Act) VALUES(w cif, w cantidad, w OID Act);
   w OID Fin:= SEC Fin.CURRVAL:
   SELECT * INTO patrocinio FROM PATROCINIOS WHERE OID Fin=w OID Fin;
    IF (patrocinio.cif<>w cif AND patrocinio.cantidad<>w cantidad AND patrocinio.OID Act<>w OID Act) THEN
        salida:=false;
    END IF:
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
        ROLLBACK:
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Fin INTEGER, w cif CHAR, w cantidad NUMBER, w OID Act INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true:
patrocinio PATROCINIOS%ROWTYPE;
UPDATE PATROCINIOS SET cif=w cif, cantidad=w cantidad, OID Act=w OID Act WHERE OID Fin= w OID Fin;
SELECT * INTO patrocinio FROM PATROCINIOS WHERE OID Fin= w OID Fin;
IF (patrocinio.cif<>w cif AND patrocinio.cantidad<>w cantidad AND patrocinio.OID Act<>w OID Act) THEN
salida:=false;
END IF;
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
   ROLLBACK;
END actualizar;
PROCEDURE eliminar(nombre_prueba VARCHAR2, w_OID_Fin INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n patrocinios INTEGER;
BEGIN
DELETE FROM PATROCINIOS WHERE OID_Fin= w_OID_Fin;
SELECT COUNT (*) INTO n patrocinios FROM PATROCINIOS WHERE OID Fin: w OID Fin:
IF(n patrocinios<>0) THEN
    salida:=false;
END IF;
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
```

```
EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS PATROCINIOS;
-- PRUEBAS TIPOSDONACIONES
create or replace
PACKAGE BODY PRUEBAS TIPODONACIONES AS
PROCEDURE inicializar AS
BEGIN
   DELETE FROM TIPODONACIONES;
END inicializar;
PROCEDURE insertar(nombre prueba VARCHAR2, w nombre VARCHAR2, w tipoUnidad VARCHAR2, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   tipoDonacion TIPODONACIONES%ROWTYPE:
   w_OID_TDon INTEGER;
BEGIN
   INSERT INTO TIPODONACIONES(nombre, tipoUnidad) VALUES(w nombre, w tipoUnidad);
   w OID TDon:=SEC TDon.CURRVAL;
   SELECT * INTO tipoDonacion FROM TIPODONACIONES WHERE OID TDon=w OID TDon;
   IF (tipoDonacion.nombre<>w nombre or tipoDonacion.tipoUnidad<>w tipoUnidad) THEN
        salida:=false;
   END IF;
   COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
           ROLLBACK;
END insertar;
PROCEDURE actualizar (nombre prueba VARCHAR2, w OID TDon INTEGER, w nombre VARCHAR2, w tipoUnidad VARCHAR2, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   tipoDonacion TIPODONACIONES%ROWTYPE;
BEGIN
   UPDATE TIPODONACIONES SET nombre=w_nombre, tipoUnidad=w_tipoUnidad WHERE OID_TDon=w_OID_TDon;
   SELECT * INTO tipoDonacion FROM TIPODONACIONES WHERE OID TDon=w OID TDon;
   IF (tipoDonacion.nombre<>w nombre or tipoDonacion.tipoUnidad<>w tipoUnidad) THEN
        salida:=false;
   END IF;
   COMMIT WORK;
```

```
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
      ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre prueba VARCHAR2, w OID TDon INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n tipoDonaciones INTEGER;
BEGIN
   DELETE FROM TIPODONACIONES WHERE OID TDon=w OID TDon:
   SELECT COUNT (*) INTO n_tipoDonaciones FROM TIPODONACIONES WHERE OID TDon=w OID TDon;
   IF(n tipoDonaciones<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK;
END eliminar;
END PRUEBAS TIPODONACIONES;
--PRUEBAS DONACIONES
create or replace
PACKAGE BODY PRUEBAS DONACIONES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM DONACIONES;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w cif CHAR, w dni CHAR, w OID TDon INTEGER, w cantidad NUMBER, w valorUnitario NUMBER,
       w fecha DATE, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   donacion DONACIONES%ROWTYPE;
   w OID Don INTEGER;
   BEGIN
       INSERT INTO DONACIONES(cif, dni, OID TDon, cantidad, valorUnitario, fecha) VALUES(w cif, w dni, w OID TDon, w cantidad, w valorUnitario, w fecha);
       w OID Don:= SEC Don.CURRVAL;
       SELECT * INTO donacion FROM DONACIONES WHERE OID Don=w OID Don;
       IF (donacion.cif<>w_cif AND donacion.dni<>w_dni AND donacion.OID_TDon<>w_OID_TDon AND donacion.cantidad<>w_cantidad
           AND donacion.valorUnitario<>w valorUnitario AND donacion.fecha<>w fecha) THEN
            salida:=false;
       END IF;
```

```
COMMIT WORK:
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
       ROLLBACK;
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Don INTEGER, w cif CHAR, w dni CHAR, w OID TDon INTEGER, w cantidad NUMBER,
   w valorUnitario NUMBER, w fecha DATE, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true:
donacion DONACIONES%ROWTYPE;
BEGIN
UPDATE DONACIONES SET cif=w cif, dni=w dni, OID TDon=w OID TDon, cantidad=w cantidad, valorUnitario=w valorUnitario,
   fecha=w fecha WHERE OID Don= w OID Don;
SELECT * INTO donacion FROM DONACIONES WHERE OID Don= w OID Don;
IF (donacion.cif<>w cif AND donacion.dni<>w dni AND donacion.OID TDon<>w OID TDon AND donacion.cantidad<>w cantidad
   AND donacion.valorUnitario<>w valorUnitario AND donacion.fecha<>w fecha) THEN
salida:=false;
END IF;
COMMIT WORK:
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
  ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre_prueba VARCHAR2, w_OID_Don INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true:
n_donaciones INTEGER;
BEGIN
DELETE FROM DONACIONES WHERE OID Don= w OID Don;
SELECT COUNT (*) INTO n donaciones FROM DONACIONES WHERE OID Don= w OID Don;
IF(n donaciones<>0) THEN
   salida:=false;
END IF;
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
   ROLLBACK;
END eliminar;
```

```
END PRUEBAS DONACIONES;
--PRUEBAS_MENSAJES
create or replace
PACKAGE BODY PRUEBAS MENSAJES AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM MENSAJES:
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w OID Coord INTEGER, w tipo VARCHAR2, w fechaEnvio DATE, w asunto VARCHAR2,
       w contenido LONG, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   mensaje MENSAJES%ROWTYPE;
   w OID M INTEGER;
   BEGIN
       INSERT INTO MENSAJES(OID Coord, tipo, fechaEnvio, asunto, contenido) VALUES(w OID Coord, w tipo, w fechaEnvio, w asunto, w contenido);
       w OID M:= SEC M.CURRVAL:
       SELECT * INTO mensaje FROM MENSAJES WHERE OID M=w OID M;
       IF (mensaje.OID Coord<>w OID Coord AND mensaje.tipo<>w tipo AND mensaje.fechaEnvio<>w fechaEnvio AND mensaje.asunto<>w asunto
            AND mensaje.contenido<>w_contenido) THEN
            salida:=false:
        END IF;
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
            ROLLBACK;
   END insertar;
   PROCEDURE actualizar(nombre_prueba VARCHAR2, w_OID_M INTEGER, w_OID_Coord INTEGER, w_tipo VARCHAR2, w_fechaEnvio DATE,
       w asunto VARCHAR2, w contenido LONG, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   mensaje MENSAJES%ROWTYPE;
   BEGIN
   UPDATE MENSAJES SET OID M=w OID M, OID Coord=w OID Coord, tipo=w tipo, fechaEnvio=w fechaEnvio, asunto=w asunto,
        contenido=w contenido WHERE OID M=w OID M;
   SELECT * INTO mensaje FROM MENSAJES WHERE OID M= w OID M;
   IF (mensaje.OID_Coord<>w_OID_Coord AND mensaje.tipo<>w_tipo AND mensaje.fechaEnvio<>w_fechaEnvio AND mensaje.asunto<>w_asunto
       AND mensaie.contenido<>w contenido) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
```

```
EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
       ROLLBACK;
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID M INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n mensajes INTEGER;
   BEGIN
   DELETE FROM MENSAJES WHERE OID M= w OID M;
   SELECT COUNT (*) INTO n mensajes FROM MENSAJES WHERE OID M= w OID M;
   IF(n mensaies<>0) THEN
        salida:=false;
   END IF;
   COMMIT WORK;
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
        ROLLBACK:
   END eliminar;
END PRUEBAS_MENSAJES;
--PRUEBAS ENVIOS
create or replace
PACKAGE BODY PRUEBAS ENVIOS AS
   PROCEDURE inicializar AS
   BEGIN
        DELETE FROM ENVIOS;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w OID M INTEGER, w dni CHAR, w cif CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   envio ENVIOS%ROWTYPE;
   w OID Env INTEGER;
   BEGIN
        INSERT INTO ENVIOS(OID M, dni, cif) VALUES(w OID M, w dni, w cif);
       w OID Env:= SEC ENV.CURRVAL;
       SELECT * INTO envio FROM ENVIOS WHERE OID_Env=w_OID_Env;
        IF (envio.OID M<>w OID M AND envio.dni<>w dni AND envio.cif<>w cif) THEN
            salida:=false;
        END IF;
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
```

```
EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(false, salidaEsperada));
           ROLLBACK;
   END insertar:
   PROCEDURE actualizar(nombre prueba VARCHAR2, w OID Env INTEGER, w OID M INTEGER, w dni CHAR, w cif CHAR, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   envio ENVIOS%ROWTYPE;
   BEGIN
   UPDATE ENVIOS SET OID M=w OID M, dni=w dni, cif=w cif WHERE OID Env=w OID Env;
   SELECT * INTO envio FROM ENVIOS WHERE OID Env= w OID Env;
   IF (envio.OID M<>w OID M AND envio.dni<>w dni AND envio.cif<>w cif) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
      ROLLBACK:
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Env INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n envios INTEGER;
   BEGIN
   DELETE FROM ENVIOS WHERE OID Env= w OID Env;
   SELECT COUNT (*) INTO n envios FROM ENVIOS WHERE OID Env= w OID Env;
   IF(n envios<>0) THEN
       salida:=false;
   END IF:
   COMMIT WORK;
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EOUALS(false, salida Esperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS ENVIOS;
--PRUEBAS_PREGUNTAS
create or replace
PACKAGE BODY PRUEBAS PREGUNTAS AS
   PROCEDURE inicializar AS
```

```
BEGIN
DELETE FROM PREGUNTAS;
END inicializar;
PROCEDURE insertar(nombre prueba VARCHAR2, w tipo VARCHAR2, w enunciado VARCHAR, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
pregunta PREGUNTAS%ROWTYPE;
w_OID_Q INTEGER;
BEGIN
INSERT INTO PREGUNTAS(tipo, enunciado) VALUES(w tipo, w enunciado);
w OID Q:=SEC Q.CURRVAL;
SELECT * INTO pregunta FROM PREGUNTAS WHERE OID O=w OID O;
IF (pregunta.tipo<>w tipo AND pregunta.enunciado<>w enunciado) THEN
    salida:=false;
END IF;
COMMIT WORK;
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
       ROLLBACK:
END insertar;
PROCEDURE actualizar(nombre_prueba VARCHAR2,w_OID_Q INTEGER, w_tipo VARCHAR2, w_enunciado VARCHAR, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
pregunta PREGUNTAS%ROWTYPE;
BEGIN
UPDATE PREGUNTAS SET tipo=w tipo, enunciado=w enunciado WHERE OID Q=w OID Q;
SELECT * INTO pregunta FROM PREGUNTAS WHERE OID O=w OID O;
IF (pregunta.tipo<>w tipo AND pregunta.enunciado<>w enunciado) THEN
    salida:=false;
END IF:
COMMIT WORK;
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
   ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre_prueba VARCHAR2, w_OID_Q INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n_preguntas INTEGER;
BEGIN
DELETE FROM PREGUNTAS WHERE OID 0=w OID 0;
SELECT COUNT (*) INTO n preguntas FROM PREGUNTAS WHERE OID Q=w OID Q;
```

```
IF(n preguntas<>0) THEN
       salida:=false;
   END IF:
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EOUALS(false, salida Esperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS PREGUNTAS;
-- PRUEBAS CUESTIONARIOS
create or replace
PACKAGE BODY PRUEBAS CUESTIONARIOS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM CUESTIONARIOS;
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2, w OID Act INTEGER, w fechaCreacion DATE, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   cuestionario CUESTIONARIOS%ROWTYPE;
   w_OID_Cues INTEGER;
   BEGIN
       INSERT INTO CUESTIONARIOS(OID Act, fechaCreacion) VALUES(w OID Act, w fechaCreacion);
       w OID Cues:= SEC CUES.CURRVAL;
       SELECT * INTO cuestionario FROM CUESTIONARIOS WHERE OID Cues=w OID Cues;
       IF ( cuestionario.OID Act<>w OID Act AND cuestionario.fechaCreacion<>w fechaCreacion) THEN
            salida:=false;
       END IF:
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false, salidaEsperada));
           ROLLBACK:
   END insertar;
   PROCEDURE actualizar(nombre prueba VARCHAR2, w_OID_Cues INTEGER, w_OID_Act INTEGER, w_fechaCreacion DATE, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   cuestionario CUESTIONARIOS%ROWTYPE;
   BEGIN
   UPDATE CUESTIONARIOS SET OID Act=w OID Act, fechaCreacion= w fechaCreacion WHERE OID Cues= w OID Cues;
   SELECT * INTO cuestionario FROM CUESTIONARIOS WHERE OID Cues= w OID Cues;
```

```
IF (cuestionario.OID Act<>w OID Act AND cuestionario.fechaCreacion<>w fechaCreacion) THEN
   salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(false, salidaEsperada));
      ROLLBACK:
   END actualizar;
   PROCEDURE eliminar(nombre_prueba VARCHAR2, w_OID_Cues INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n cuestionarios INTEGER;
   BEGIN
   DELETE FROM CUESTIONARIOS WHERE OID Cues= w OID Cues;
   SELECT COUNT (*) INTO n cuestionarios FROM CUESTIONARIOS WHERE OID Cues= w OID Cues;
   IF(n_cuestionarios<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK:
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK:
   END eliminar;
END PRUEBAS CUESTIONARIOS;
--PRUEBAS_FORMULARIOS
create or replace
PACKAGE BODY PRUEBAS FORMULARIOS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM FORMULARIOS;
   END inicializar;
   PROCEDURE insertar(nombre_prueba VARCHAR2, w_OID_Cues INTEGER, w_OID_Q INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true:
   formulario FORMULARIOS%ROWTYPE;
   w OID Form INTEGER;
   BEGIN
       INSERT INTO FORMULARIOS(OID Cues, OID Q) VALUES(w OID Cues, w OID Q);
       w OID Form:= SEC FORM.CURRVAL;
       SELECT * INTO formulario FROM FORMULARIOS WHERE OID_Form=w_OID_Form;
```

```
IF (formulario.OID Cues<>w OID Cues AND formulario.OID Q<>w OID Q) THEN
        salida:=false;
    END IF:
    COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(salida, salidaEsperada));
    EXCEPTION
    WHEN OTHERS THEN
        DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
        ROLLBACK:
END insertar;
PROCEDURE actualizar(nombre prueba VARCHAR2, w OID_Form INTEGER, w OID_Cues INTEGER, w OID_Q INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
formulario FORMULARIOS%ROWTYPE;
BEGIN
UPDATE FORMULARIOS SET OID Cues=w OID Cues, OID O= w OID O WHERE OID Form= w OID Form;
SELECT * INTO formulario FROM FORMULARIOS WHERE OID Form= w OID Form;
    IF (formulario.OID Cues<>w OID Cues AND formulario.OID O<>w OID O) THEN
salida:=false;
END IF;
COMMIT WORK:
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
  DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EOUALS(false, salidaEsperada));
   ROLLBACK:
END actualizar;
PROCEDURE eliminar(nombre prueba VARCHAR2, w OID Form INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true:
n_formularios INTEGER;
BEGIN
DELETE FROM FORMULARIOS WHERE OID Form= w OID Form;
SELECT COUNT (*) INTO n formularios FROM FORMULARIOS WHERE OID Form= w OID Form;
IF(n formularios<>0) THEN
    salida:=false;
END IF;
COMMIT WORK;
DBMS OUTPUT.put line(nombre prueba | | ':' | | ASSERT EQUALS(salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
   DBMS OUTPUT.put line(nombre prueba | | ':' | ASSERT EQUALS(false, salida Esperada));
    ROLLBACK;
END eliminar;
```

```
END PRUEBAS FORMULARIOS;
--PRUEBAS_RESPUESTAS
create or replace
PACKAGE BODY PRUEBAS RESPUESTAS AS
   PROCEDURE inicializar AS
   BEGIN
       DELETE FROM RESPUESTAS:
   END inicializar;
   PROCEDURE insertar(nombre prueba VARCHAR2,w OID Form INTEGER, w OID Part INTEGER, w OID Vol INTEGER, w contenido VARCHAR2,
        salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   respuesta RESPUESTAS%ROWTYPE;
   w OID Ans INTEGER;
   BEGIN
       INSERT INTO RESPUESTAS(OID Form, OID Part, OID Vol, contenido) VALUES(w OID Form, w OID Part, w OID Vol, w contenido);
       w OID Ans:= SEC ANS.CURRVAL;
       SELECT * INTO respuesta FROM RESPUESTAS WHERE OID Ans=w OID Ans;
       IF (respuesta.OID Form<>w OID Form AND respuesta.OID Part<>w OID Part AND respuesta.OID Vol<>w OID Vol
            AND respuesta.contenido<>w contenido) THEN
            salida:=false:
        END IF;
       COMMIT WORK;
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida, salidaEsperada));
       EXCEPTION
       WHEN OTHERS THEN
           DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
            ROLLBACK:
   END insertar;
   PROCEDURE actualizar(nombre_prueba VARCHAR2,w OID Ans INTEGER, w OID Form INTEGER, w OID Part INTEGER, w OID Vol INTEGER,
       w contenido VARCHAR2, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   respuesta RESPUESTAS%ROWTYPE;
   BEGIN
   UPDATE RESPUESTAS SET OID Form=w OID Form, OID Part= w OID Part, OID Vol=w OID Vol, contenido=w contenido WHERE OID Ans= w OID Ans;
   SELECT * INTO respuesta FROM RESPUESTAS WHERE OID Ans= w OID Ans;
       IF (respuesta.OID Form<>>w OID Form AND respuesta.OID Part<>>w OID Part AND respuesta.OID Vol<>>w OID Vol
            AND respuesta.contenido<>w_contenido) THEN
   salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
```

```
WHEN OTHERS THEN
      DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EOUALS(false, salidaEsperada));
      ROLLBACK;
   END actualizar;
   PROCEDURE eliminar(nombre prueba VARCHAR2, w_OID Ans INTEGER, salidaEsperada BOOLEAN) AS
   salida BOOLEAN:=true;
   n_respuestas INTEGER;
   BEGIN
   DELETE FROM RESPUESTAS WHERE OID_Ans= w_OID_Ans;
   SELECT COUNT (*) INTO n_respuestas FROM RESPUESTAS WHERE OID_Ans= w_OID_Ans;
   IF(n respuestas<>0) THEN
       salida:=false;
   END IF;
   COMMIT WORK;
   DBMS OUTPUT.put line(nombre prueba || ':' || ASSERT EQUALS(salida, salidaEsperada));
   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK;
   END eliminar;
END PRUEBAS RESPUESTAS;
```

```
-- PRUEBAS de PAOUETES
-- PRUEBAS PERSONAS
BEGIN
pruebas personas.inicializar;
pruebas personas.insertar('Prueba 1- insertar persona correctamente','47339192V', 'Mario', 'Ruano Fernández', '01/11/1989', 'C/ San Antonio, 17', 'Fuentes de Andalucía', 'Sevilla',
'41420', 'mrf1989@hotmail.com', '615337550', true);
pruebas personas insertar ('Prueba 2- insertar persona correctamente', '13227182M', 'Ignacio', 'Vázquez Rial', '11/12/1991', 'Paseo de los Marineros, 23', 'Lepe', 'Huelva', '40004',
'ignatius1991@gmail.com', '621499732', true);
pruebas personas.insertar('Prueba 3- insertar persona correctamente','98385816W', 'Cristina', 'Caro Caro', '15/08/1991', 'C/ Vallehermoso, 62, 1ºA', 'Madrid', 'Madrid', '28023',
'ccaroc@deporteydesafio.com', '629123456' ,true);
pruebas personas.insertar('Prueba 4- insertar persona correctamente', '95487762B', 'Ana', 'Cuevas Hernández', '04/04/1988', 'C/ Mayor, 12', 'Segovia', 'Segovia', '20456',
'anacuhe@gmail.com', '644181345',true);
pruebas personas.insertar('Prueba 5- insertar persona correctamente','55612222B', 'Estafanía', 'Gómez Gómez', '19/10/1990', 'C/ Periodistas, 44, 1ºB','Madrid', 'Madrid', '28006',
'estefa gg@gmail.com', '695777100',true);
pruebas personas.insertar('Prueba 6- insertar persona correctamente','38656621L', 'Antonio Luis', 'Orellana Smith', '01/09/1980', 'C/ Molineros, 7, 3ºB','Madrid', 'Madrid', '28017',
null. '643998765'.true):
pruebas personas.insertar('Prueba 7- insertar persona correctamente','57153559V', 'Mateo', 'Ruiz López', '06/02/1978', 'C/ Miguel Hernández, 27, 2ºB','Madrid', 'Madrid', '28017',
'mateo ruiz lopez@gmail.com', '614888909'.true):
pruebas personas.insertar('Prueba 8- insertar persona correctamente'.'761843590'. 'Ramón', 'Román Romo'. '25/11/1981'. 'C/ Bandoleros de la Sierra, 10, 2ºB'.'Cercedilla'. 'Madrid'.
'27004', 'ramonromo@gmail.com', '678411522',true);
pruebas personas.insertar('Prueba 9- insertar persona correctamente', '46687881E', 'María Dolores', 'Pilatos Suárez de la Hoz', '15/03/1991', 'C/ Aluminio, 23, 1ºA', 'Madrid', 
'28021', 'maripi suhoz@gmail.com', '617829955',true);
pruebas personas.insertar('Prueba 10- insertar persona correctamente','12345678A', 'María', 'Núñez Ortiz', '12/10/1990', 'Avd. Kansas City, 12, 4ºC','Sevilla', 'Sevilla', '41007',
'm nuor@gmail.com', '645234184',true);
pruebas personas.insertar('Prueba 11- insertar persona correctamente','57231289R', 'Marcos', 'Gómez Arce', '15/05/1993', 'C/ Tregua, 21, 5ºB','Alcorcón', 'Madrid', '23102',
'magoarce@gmail.com', '698543761',true);
pruebas personas.insertar('Prueba 12- insertar persona correctamente','23987795C', 'Pablo', 'Cabello Marín', '11/11/1982', 'C/ Narciso, 46, 8ºA', 'Madrid', 'Madrid', '28015',
'cabello_marin@gmail.com', '658679123',true);
pruebas personas.insertar('Prueba 13- insertar persona correctamente','66641234M', 'Ángela María', 'Castilla de la Huerta Domínguez', '21/02/1990', 'C/ Anís, 12','Collado Villalba',
'Madrid', '28510', 'angie castilladom@gmail.com', '610223734',true);
pruebas personas.insertar('Prueba 14- insertar persona correctamente','89125642G', 'Sara', 'Orellana Luna', '10/06/2009', 'C/ Molineros, 7, 3ºB', 'Madrid', 'Madrid', '28017', null,
'643998765',true);
pruebas personas.insertar('Prueba 15- insertar persona con dni menor a 9 caracteres', '89321345', 'Pedro', 'López Durán', '23/03/1986', 'C/ Laurel, 23, 1ºB', 'Sevilla', 'Sevilla',
'41005', 'pedro86 lodu@hotmail.com', '611765473', false);
pruebas personas.insertar('Prueba 16- insertar persona con dni null', NULL, 'Pedro', 'López Durán', '23/03/1986', 'C/ Laurel, 23, 1ºB', 'Sevilla', 'Sevilla', '41005',
'pedro86 lodu@hotmail.com', '611765473', false);
pruebas personas.insertar('Prueba 17- insertar persona con nombre null','43761927D', NULL, 'Ramirez Doblado', '08/01/1998', 'C/ Militante, 12', 'Alcorcón', 'Madrid', '23104',
'glorado1998@gmail.com', '642132534', false);
pruebas personas.insertar('Prueba 18- insertar persona con fecha de nacimiento null','43761927D', 'Gloria', 'Ramirez Doblado', NULL, 'C/ Militante, 12', 'Alcorcón', 'Madrid', '23104',
 'glorado1998@gmail.com', '642132534', false);
pruebas personas.insertar('Prueba 19- insertar persona con teléfono null','43761927D', 'Gloria', 'Ramirez Doblado', '08/01/1998', 'C/ Militante, 12', 'Alcorcón', 'Madrid', '23104',
'glorado1998@gmail.com', NULL, false);
pruebas personas.actualizar('Prueba 20- actualizar persona no insertada', '12345678S', 'Manuel', 'Núñez Ortiz', '12/10/1990', 'Avd. Kansas City, 12, 4ºC', 'Sevilla', 'Sevilla',
'41007', 'm nuor@gmail.com', '645234184', false);
pruebas personas actualizar ('Prueba 21- actualizar persona', '13227182M', 'Ignacio', 'Vázquez Rial', '11/12/1991', 'Paseo de los Marineros, 25', 'Ayamonte', 'Huelva', '21400',
'ignatius1991@gmail.com', '621499732', true);
pruebas personas.eliminar('Prueba 22- eliminar persona', '47339192V', true);
END:
```

```
--PRUEBAS COORDINADORES
DECLARE
OID Coord INTEGER;
OID Coord2 INTEGER:
BEGIN
pruebas coordinadores.inicializar;
pruebas coordinadores.insertar('Prueba 23- insertar coordinador correctamente', '98385816W', true);
OID Coord:=SEC COORD.CURRVAL;
pruebas coordinadores.insertar('Prueba 24- insertar un coordinador correctamente', '13227182M', true);
OID Coord2:=SEC COORD.CURRVAL:
pruebas coordinadores.insertar('Prueba 25- insertar un coordinador correctamente','55612222B'.true);
pruebas coordinadores.insertar('Prueba 26- insertar coordinador con dni menor a 9 caracteres', '23559156', false);
pruebas coordinadores.insertar('Prueba 27- insertar coordinador que no existe como persona', '10226644R', false);
pruebas coordinadores.insertar('Prueba 28- insertar coordinador con dni null', NULL, false);
pruebas coordinadores.actualizar('Prueba 29- actualizar coordinador con OID Coord null', NULL , '23559156L' , false);
pruebas coordinadores.actualizar('Prueba 30- actualizar coordinador', OID Coord2 , '95487762B' , true);
pruebas coordinadores.eliminar('Prueba 31- eliminar coordinador', OID Coord ,true);
-- PRUEBAS TUTORESLEGALES
DECLARE
OID Tut INTEGER:
OID Tut2 INTEGER;
BEGIN
pruebas tutoreslegales.inicializar;
pruebas tutoreslegales.insertar('Prueba 32- insertar tutor legal correctamente', '38656621L',true);
OID Tut:=SEC TUT.CURRVAL:
pruebas tutoreslegales.insertar('Prueba 33- insertar tutor legal correctamente', '13227182M',true);
OID Tut2:=SEC TUT.CURRVAL:
pruebas tutoreslegales.insertar('Prueba 34- insertar tutor legal correctamente', '761843590',true);
pruebas_tutoreslegales.insertar('Prueba 35- insertar tutor legal con dni menor a 9 caracteres', '38656621B',false);
pruebas tutoreslegales.insertar('Prueba 36- insertar tutor legal con dni null', NULL, false);
pruebas_tutoreslegales.insertar('Prueba 37- insertar tutor legal con dos letras en su dni', '89321A45B',false);
pruebas tutoreslegales.actualizar('Prueba 38- actualizar tutor legal con OID Tut null', NULL, '38656621L', false):
pruebas tutoreslegales.actualizar('Prueba 39- actualizar tutor legal', OID Tut2, '57153559V', true);
pruebas tutoreslegales.eliminar('Prueba 40- eliminar tutor legal', OID Tut, true);
--PRUEBAS VOLUNTARIOS
DECLARE
OID Vol INTEGER:
OID Vol2 INTEGER:
pruebas voluntarios.inicializar;
pruebas_voluntarios.insertar('Prueba 41- insertar voluntario correctamente','46687881E', 'baja',true);
OID Vol:=SEC VOL.CURRVAL;
pruebas voluntarios.insertar('Prueba 42- insertar voluntario correctamente','12345678A', 'media',true);
OID Vol2:=SEC VOL.CURRVAL;
pruebas voluntarios.insertar('Prueba 43- insertar voluntario correctamente', '57231289R', 'media', true);
pruebas voluntarios.insertar('Prueba 44- insertar voluntario con dni null', NULL, 'baja',false):
pruebas voluntarios.insertar('Prueba 45- insertar voluntario con dni con dni menor a 9 caracteres', '57231289', 'media', false);
pruebas voluntarios.insertar('Prueba 46- insertar voluntario con dos letras en su dni','123B5678A','media',false);
pruebas voluntarios.actualizar('Prueba 47- actualizar voluntario correctamente',oid vol,'46687881E', 'alta',true);
pruebas voluntarios.actualizar('Prueba 48- actualizar voluntario con dni incorrecto'.oid vol. NULL.'alta'.false);
pruebas voluntarios.eliminar('Prueba 49- eliminar voluntario',oid vol2,true);
END:
```

```
-- PRUEBAS PARTICIPANTES
DECLARE
OID Part INTEGER;
OID Part2 INTEGER:
OID Part3 INTEGER:
REGIN
pruebas participantes.inicializar:
pruebas participantes.insertar('Prueba 50- insertar participante correctamente','23987795C', '0,4', 'alta', 2, 1,true);
OID Part:=SEC PART.CURRVAL:
pruebas participantes.insertar('Prueba 51- insertar participante correctamente', '66641234M', '0.6', 'media', 3, 3, true);
OID Part2:=SEC PART.CURRVAL:
pruebas participantes.insertar('Prueba 52- insertar participante correctamente', '89125642G', '0.6', 'alta', 3, NULL.true);
OID Part3:=SEC PART.CURRVAL:
pruebas participantes.insertar('Prueba 53- insertar participante con dni null', NULL, '0,6', 'media', 3, 1, false);
pruebas participantes.insertar('Prueba 54- insertar participante con grado de discapacidad null', '89125642G', NULL, 'baja', 3, 1 ,false);
pruebas participantes.insertar('Prueba 55- insertar participante con oid Tut null', '66641234M', '0,6', 'media', NULL , 3, false);
pruebas participantes.actualizar('Prueba 56- actualizar participante',OID Part2,'66641234M', '0,6', 'alta', 3, 3,true);
pruebas participantes.actualizar('Prueba 57- actualizar participante con dni null', OID Part3, NULL, '0,6', 'alta', 3, 1, false);
pruebas participantes.eliminar('Prueba 58- eliminar participante', OID Part.true):
--PRUEBAS INFORMESMEDICOS
DECLARE
OID Inf INTEGER:
OID Inf2 INTEGER;
BEGIN
pruebas informesmedicos.inicializar:
pruebas informesmedicos.insertar('Prueba 1- insertar informemedico correctamente', 2, 'Se trata de un paciente femenino que presenta claros sintomas...', '04/06/2017', true);
OID Inf:=SEC INF.CURRVAL:
pruebas informesmedicos.insertar('Prueba 1- insertar informemedico correctamente', 3, 'Se trata de un paciente femenino que presenta claros sintomas...', '05/08/2018', true);
OID Inf2:=SEC INF.CURRVAL;
pruebas informesmedicos.insertar('Prueba 1- insertar informemedico correctamente', 2,'La mejora notable de las sesiones de fisioterapia...', '23/06/2018', true);
pruebas informesmedicos.insertar('Prueba 2- insertar informemedico con OID Part null', NULL, 'Se trata de un paciente femenino que presenta claros sintomas...', '05/06/2018', false);
pruebas informesmedicos.insertar('Prueba 3- insertar informemedico con descripcion null'. 2.NULL. '24/01/2016'.false):
pruebas informesmedicos.insertar('Prueba 4- insertar informemedico con fecha null', 2, 'Se trata de un paciente femenino que presenta claros sintomas...', NULL, false);
pruebas informesmedicos.actualizar('Prueba 1- actualizar informemedico',oid Inf2, 3,'A día 01/06/2018, nos encontramos ante...', '04/06/2018', true);
pruebas informesmedicos.actualizar('Prueba 1- actualizar informemedico con descripción null',oid Inf2, 3,NULL, '04/06/2018', false);
pruebas informesmedicos.eliminar('Prueba 1- eliminar informemedico',oid Inf, true);
-- PRUEBAS PROYECTOS
DECLARE
OID Proi INTEGER:
OID Proj2 INTEGER;
BEGIN
pruebas proyectos.inicializar;
pruebas proyectos.insertar('Prueba 1- inserción proyecto correctamente', 'Curso de Padel', 'Urban Padel Madrid', 1, 0, true);
OID Proi:=SEC PROJ.CURRVAL:
pruebas proyectos.insertar('Prueba 2- inserción proyecto correctamente', 'Ruta Senderista Cascadas del Purgatorio', 'Valle Alto del Lozoya, Madrid', 0, 1, true);
OID Proi2:=SEC PROJ.CURRVAL:
pruebas provectos.insertar('Prueba 3- inserción provecto correctamente'. 'Esquí sobre hielo'.'Pabellón de Hielo de Leganés'.0.1. true):
pruebas proyectos.insertar('Prueba 4- inserción proyecto correctamente', 'Curso de Natación avanzado', 'Centro Deportivo Municipal Francos Rodrigez',0,1, true);
pruebas proyectos.insertar('Prueba 5- inserción proyecto nombre null', NULL, 'Pabellón de Hielo de Leganés',0,1,false);
pruebas provectos.insertar('Prueba 6- inserción provecto con ubicación null', 'Torneo de Fútbol', NULL, 1, 0, false);
pruebas proyectos.insertar('Prueba 7- inserción proyecto con esEvento null', 'Ruta Senderista Monte Abantos', 'San Lorenzo del Escorial, Madrid', NULL, 0, false);
pruebas proyectos.insertar('Prueba 8- inserción proyecto con esProgDep null', 'Curso de Padel', 'Urban Padel Madrid', 1, NULL, false);
pruebas proyectos.actualizar('Prueba 1- actualizar proyecto correctamente', OID Proj,'Curso de Padel','Padel Space Madrid',1,0,true);
```

```
pruebas proyectos.actualizar('Prueba 2- actualizar proyecto con nombre null',OID Proj2,NULL,'Valle Alto del Lozoya, Madrid',1,0, false );
pruebas proyectos.eliminar('Prueba 1- elimiar proyecto',OID Proj, true);
/
-- PRUEBAS ENCARGADOS
DECLARE
OID RP INTEGER;
OID RP2 INTEGER;
BEGIN
pruebas encargados.inicializar:
pruebas encargados.insertar('Prueba 1- insertar encargado', 2, 2, true):
OID RP:=SEC RP.CURRVAL:
pruebas encargados.insertar('Prueba 2-insertar encargado', 2, 2,true);
OID RP2:=SEC RP.CURRVAL:
pruebas encargados.insertar('Prueba 3- inertar encargado', 4, 3, true);
pruebas encargados.insertar('Prueba 4- insertar encargado con OID Proj null', NULL, 2, false);
pruebas encargados.insertar('Prueba 5- insertar encargado con OID Coord null', 2, NULL, false);
pruebas encargados.actualizar('Prueba 1- actualizar encargado con OID RP null', NULL, 2, 2, false);
pruebas encargados actualizar ('Prueba 2- actualizar encargado', OID RP2, 3,2,true):
pruebas encargados.eliminar('Prueba 1- eliminar encargado', OID RP, true);
END;
--PRUEBAS ACTIVIDADES
DECLARE
OID Act INTEGER;
OID Act2 INTEGER:
BEGIN
pruebas actividades.insertar('Prueba 1- insertar actividad', 3, 'Esquí sobre hielo', 'felicidad, diversión y compañía', '22/12/2018', '22/12/2018', '70', '70', 'deportiva', '700', '5',
true):
OID Act:=SEC ACT.CURRVAL;
pruebas actividades.insertar('Prueba 2- insertar actividad', 2, 'Ruta Senderista Cascadas del Purgatorio', 'Conocer historia y nuevos paisajes', '08/01/2017', '08/01/2017', '20', '15',
'deportiva', '360', '8', true);
OID Act2:=Sec ACT.CURRVAL;
pruebas actividades.insertar('Prueba 3- insertar actividad', 4.'Curso de Natación avanzado', 'Mejora en las hablilidades físicas', '01/02/2019', '01/06/2019', '10', '10', 'deportiva',
'500', '0', true);
pruebas actividades.insertar('Prueba 4- insertar actividad con nombre null', 3, NULL, 'Divertirse, aprender y hacer nuevos amigos','22/12/2018', '22/12/2018', '30', 'deportiva',
'200', '4', false );
pruebas actividades.insertar('Prueba 5- insertar actividad con objetivo null', 3, 'Esquí sobre hielo', NULL, '22/12/2018', '22/12/2018', '70', 'deportiva', '400', '5', false);
pruebas actividades.insertar ('Prueba 6- insertar actividad con fechalnicio null', 3, 'Esquí sobre hielo', 'felicidad, diversión y compañía', NULL, '22/12/2018', '70', '70',
'deportiva', '400', '5', false);
pruebas actividades.insertar('Prueba 7- insertar actividad con fechaFin null', 3, 'Esquí sobre hielo', 'felicidad, diversión v compañía', '22/12/2018', NULL, '70', '70', 'deportiva',
'400', '5', false):
pruebas actividades.insertar('Prueba 8- insertar actividad con OID Proj null', NULL, 'Ruta Senderista Cascadas del Purgatorio', 'Conocer historia y nuevos paisajes',
'08/01/2017','08/01/2017', '20', '15', 'deportiva', '160', '8', false);
pruebas_actividades.insertar('Prueba 9- insertar actividad con numeroPlazas null',3, 'Esquí sobre hielo', 'felicidad, diversión y compañía', '22/12/2018', '22/12/2018', NULL, '70',
'deportiva', '400', '5', false);
pruebas actividades.insertar('Prueba 10- insertar actividad con VoluntariosRegueridos null',3, 'Esquí sobre hielo', 'felicidad, diversión y compañía', '22/12/2018', '22/12/2018', '70',
NULL, 'deportiva', '400', '5', false);
pruebas actividades.insertar('Prueba 11- insertar actividad con tipo null',2, 'Ruta Senderista Cascadas del Purgatorio', 'Conocer historia y nuevos paisajes', '08/01/2017',
'08/01/2017', '20', '15', NULL, '160', '8', false);
pruebas actividades.insertar('Prueba 12- insertar actividad con costeTotal null'.2, 'Ruta Senderista Cascadas del Purgatorio', 'Conocer historia y nuevos paisajes', '08/01/2017',
'08/01/2017', '20', '15', 'deportiva', NULL, '8', false);
pruebas actividades.insertar('Prueba 13- insertar actividad con costeIscripcion null', 2, 'Ruta Senderista Cascadas del Purgatorio', 'Conocer historia y nuevos paisajes', '08/01/2017',
'08/01/2017', '20', '15', 'deportiva', '160', NULL, false);
pruebas actividades.actualizar ('Prueba 1- actualizar actividad', OID Act2, 2, 'Ruta Senderista Cascadas del Purgatorio', 'Conocer historia y nuevos paisajes', '05/05/2017',
'05/05/2017', '20', '15', 'deportiva', '360', '8', true);
```

```
pruebas actividades.actualizar ('Prueba 2- actualizar actividad con OID null', NULL, 3, 'Mercadillo de Navidad', 'Dar y recibir amor, felicidad y compañía', '22/12/2018', '22/12/2018',
'70', '70', 'social', '600', '5', false);
pruebas actividades.eliminar('Prueba 1- eliminar actividad', OID Act, true);
-- PRUEBAS ESTAINTERESADOEN
DECLARE
OID Int INTEGER;
OID Int2 INTEGER:
BEGIN
pruebas estaInteresadoEn.inicializar:
pruebas estaInteresadoEn.insertar('Prueba 1- insertar el interés de participante', 2, NULL, 2, 1,true);
OID Int:=SEC INT.CURRVAL:
pruebas esta Interesado En. insertar ('Prueba 2- insertar el interés de voluntario', NULL, 1, 2, 1, true);
OID Int2:=SEC INT.CURRVAL;
pruebas estaInteresadoEn.insertar('Prueba 3- insertar el interés de participante', 3, NULL, 3, 1, true );
pruebas estaInteresadoEn.insertar('Prueba 3- insertar el interés con OID Act null', 2,NULL, NULL, 1, false);
pruebas estaInteresadoEn.insertar('Prueba 4- insertar el interés con estado null',NULL, 1, 2, NULL,false);
pruebas estaInteresadoEn.actualizar('Prueba 1- actualizar el interés'.OID Int. 2.NULL. 2. 0.true):
pruebas estaInteresadoEn.actualizar('Prueba 2- actualizar el interés con OID null', NULL, 2, NULL, 2, 0, false);
pruebas estaInteresadoEn.eliminar('Prueba 1- eliminar interés', OID Int2, true);
END:
-- PRUEBAS COLABORACIONES
DECLARE
OID Colab INTEGER:
OID Colab2 INTEGER:
REGTN
pruebas colaboraciones.inicializar:
pruebas_colaboraciones.insertar('Prueba 1- insertar colaboración de voluntario', 1, 2, true);
OID Colab:=SEC COLAB.CURRVAL;
pruebas colaboraciones.insertar('Prueba 2- insertar colaboración de voluntario', 1, 2, true);
OID Colab2:=SEC COLAB.CURRVAL;
pruebas colaboraciones.insertar('Prueba 3- insertar colaboración de voluntario', 1, 3, true);
pruebas colaboraciones.insertar('Prueba 4- insertar colaboración de voluntario con OID Vol null', NULL, 4, false);
pruebas colaboraciones.insertar('Prueba 5- insertar colaboración de voluntario con OID Act null', 1, NULL, false);
pruebas colaboraciones.actualizar('Prueba 1- actualizar colaboración de voluntario con OID Colab null', NULL, 3, 2, false );
pruebas colaboraciones.actualizar('Prueba 2- actualizar colaboración de voluntario', OID Colab2, 3, 2, true );
pruebas colaboraciones.eliminar('Prueba 1- eliminar colaboración de voluntario', OID Colab, true):
END:
-- PRUEBAS RECIBOS
DECLARE
OID Rec INTEGER;
OID Rec2 INTEGER;
BEGIN
pruebas recibos.inicializar;
pruebas recibos.insertar('Prueba 1- insertar recibo de una actividad', 2, 2, '01/02/2017', '20/04/2017', 5, 'anulado', true);
OID Rec:=SEC REC.CURRVAL;
pruebas recibos.insertar('Prueba 2- insertar recibo de una actividad', 2, 3, '01/02/2017', '20/04/2017', '8', 'pagado', true);
OID Rec2:=SEC REC.CURRVAL:
pruebas recibos.insertar('Prueba 3- insertar recibo de una actividad', 3, 2, '01/01/2019', '31/01/2019', '0', 'pagado', true);
pruebas recibos.insertar('Prueba 4- insertar recibo de una actividad', 3, 3, '01/01/2019', '31/01/2019', '0', 'pagado', true);
pruebas_recibos.insertar('Prueba 5- insertar recibo de una actividad con fechaEmision null',2, 2, NULL,'20/04/2017', '8', 'pagado',false);
pruebas recibos.insertar('Prueba 6- insertar recibo de una actividad con fechaVencimiento null', 2, 2,'01/02/2017',NULL, '8', 'pagado',false);
pruebas recibos.insertar('Prueba 7- insertar recibo de una actividad con importe null', 2, 2, '01/02/2017', '01/02/2017', NULL, 'pagado', false);
pruebas_recibos.insertar('Prueba 8- insertar recibo de una actividad con estado null', 2, 2, '01/02/2017', '01/02/2017', '8', NULL, false);
```

```
pruebas recibos.actualizar('Prueba 1- actualizar recibo de una actividad',OID Rec,2, 3,'01/02/2017','20/04/2017', '8', 'anulado', true);
pruebas recibos.actualizar('Prueba 2- actualizar recibo de una actividad con OID null', NULL, 2, 2, '01/02/2017', '20/04/2017', '8', 'pagado', false);
pruebas recibos.eliminar('Prueba 1- eliminar recibo de una actividad', OID Rec, true);
--PRUEBAS INSCRIPCIONES
DECLARE
OID Ins INTEGER;
OID Ins2 INTEGER:
BEGIN
pruebas inscripciones.inicializar:
pruebas inscripciones.insertar('Prueba 1- insertar inscripción de participante', 2, 2, 3, true):
OID Ins:=SEC INS.CURRVAL:
pruebas inscripciones.insertar('Prueba 2- insertar inscripción de participante', 2, 3, 3, true);
OID Ins2:=SEC INS.CURRVAL;
pruebas inscripciones.insertar('Prueba 3- insertar inscripción de participante', 3, 2, 2, true);
pruebas inscripciones.insertar('Prueba 4- insertar inscripción de participante con OID Part null', NULL, 2, 2, false);
pruebas inscripciones.insertar('Prueba 5- insertar inscripción de participante con OID Act null', 3, NULL, 2, false);
pruebas inscripciones.actualizar('Prueba 1- actualizar inscripción de participante con OID Ins null'.NULL. 2. 3. 3. false):
pruebas inscripciones.actualizar ('Prueba 2- actualizar inscripción de participante', OID Ins2, 3, 3, 4, true);
pruebas inscripciones.eliminar('Prueba 1-eliminar inscripción de participante', OID Ins, true);
END:
-- PRUEBAS INSTITUCIONES
BEGIN
pruebas instituciones.inicializar:
pruebas instituciones.insertar('Prueba 1- insertar institución correctamente','A58223209', 'Banco Santander', '678943212', 'C/ Gran Vía, 22','Madrid', 'Madrid', '28001',
'contacto@santander.com', 1, 'oro', true);
pruebas instituciones.insertar('Prueba 2- insertar institución correctamente', 'A87674532', 'El Corte Inglés, S.A.', '654123987', 'C/ Preciados, 1', 'Madrid', 'Madrid', '28001',
'contacto@elcorteingles.com', 0, NULL, true);
pruebas instituciones.insertar('Prueba 3- insertar institución correctamente', 'A33219876', 'BBVA', '672129086', 'C/ Colón, 2', 'Madrid', 'Madrid', '28001', 'contacto@bbva.com', 1,
'plata', true);
pruebas instituciones.insertar('Prueba 4- insertar institución correctamente','H45681524', 'Deportiteka S.L.', '651675440', 'C/ Milagros, 74', 'Madrid', '28005',
'contacto@deportiteka.com', 0, NULL, true):
pruebas instituciones.insertar('Prueba 5- insertar institución con cif menor a 9 caracteres ', 'H4568152', 'Deportiteka S.L.', '651675440', 'C/ Milagros, 74', 'Madrid', 'Madrid
'28005', 'contacto@deportiteka.com', 1, 'plata', false);
pruebas instituciones.insertar('Prueba 6- insertar institución con cif null',null, 'Materiales Domínguez S.L.', '639553337', 'C/ Danubio, 12, 4ºB', 'Madrid', 'Madrid', '28002',
'm dominguez@gmail.com',1,'bronce', false);
pruebas instituciones.insertar('Prueba 7- insertar institución con nombre null', 'B78998456', NULL, '677213157', 'C/ Vuelos Altos, 14', 'Madrid', 'Madrid', '28004', 'mail@amisa.es'
,0,NULL, false);
pruebas instituciones.insertar('Prueba 8- insertar institución con telefono null','A33219876', 'BBVA', NULL, 'C/ Colón, 2', 'Madrid', 'Madrid', '28001', 'contacto@bbva.com', 1,
'plata'.false):
pruebas instituciones.insertar('Prueba 9- insertar institución con esPatrocinador null', 'B78998456', 'Fundación AMISA', '677213157', 'C/ Vuelos Altos, 14', 'Madrid', '28001',
 'mail@amisa.es', NULL, NULL, false);
pruebas_instituciones.actualizar('Prueba 1- actualizar institución con cif menor a 9 caracteres', 'A5822320', 'Banco Santander', '678943212', 'C/ Gran Vía, 22', 'Madrid', 'Madrid', 'Madrid', 'A5822320', 'Banco Santander', '678943212', 'C/ Gran Vía, 22', 'Madrid', 'M
'28001', 'contacto@santander.com', 1, 'oro', false);
pruebas instituciones.actualizar('Prueba 2- actualizar institución con nueva dirección', 'A87674532', 'El Corte Inglés, S.A.', '654123987', 'C/ Dr. Gómez Ulla, 2', 'Madrid', 'M
'28001', 'contacto@elcorteingles.com', 1,'oro', true);
pruebas instituciones.eliminar('Prueba 1- eliminar institución', 'A58223209', true);
END:
--PRUEBAS PATROCIONIOS
DECLARE
OID Fin INTEGER:
OID Fin2 INTEGER;
BEGIN
pruebas patrocinios.inicializar;
```

```
pruebas patrocinios.insertar('Prueba 1- insertar patrocinio','A87674532','500', 3, true);
OID Fin:=SEC FIN.CURRVAL:
pruebas patrocinios.insertar('Prueba 2- insertar patrocinio', 'A33219876','100', 2, true);
OID Fin2:=SEC FIN.Currval:
pruebas patrocinios.insertar('Prueba 3- insertar patrocinio con cif null', NULL,'200', 2, false);
pruebas patrocinios.insertar('Prueba 4- insertar patrocinio con OID Act null', 'A33219876','200',NULL, false);
pruebas_patrocinios.insertar('Prueba 5- insertar patrocinio con cantidad null', 'A33219876',NULL,2, false);
pruebas patrocinios.actualizar ('Prueba 1- actualizar patrocinio', OID Fin2, 'A33219876', '200', 2, true);
pruebas patrocinios.actualizar('Prueba 2- actualizar patrocinio con OID null', NULL, 'A87674532', '400', 3, false);
pruebas patrocinios.eliminar('Prueba 1- eliminar patrocinio', OID Fin. true):
--PRUEBAS TIPOSDONACIONES
DECLARE
OID TDon INTEGER;
OID TDon2 INTEGER;
BEGIN
pruebas tipodonaciones.inicializar;
pruebas tipodonaciones.insertar('Prueba 1- insertar tipodonaciones correctamente', 'Equipaciones de Esquí', 'material',true):
OID TDon:=SEC TDON.CURRVAL;
pruebas_tipodonaciones.insertar('Prueba 2- insertar tipodonaciones correctamente', 'Sillas de ruedas Baloncesto', 'material', true);
pruebas tipodonaciones insertar ('Prueba 3- insertar tipodonaciones correctamente', 'Equipaciones de Golf', 'material', true);
OID TDon2:=SEC TDON.CURRVAL;
pruebas tipodonaciones.insertar('Prueba 4- insertar tipodonaciones con numbre null', NULL, 'Euro', false);
pruebas tipodonaciones.insertar('Prueba 5- insertar tipodonaciones con tipoUnidad null', 'Bicicletas adaptadas', NULL, false);
pruebas tipodonaciones.actualizar('Prueba 1- actualizar tipodonacion correctamente'. OID TDon2. 'Conjunto de senderismo invernal'. 'equipaciones', true ):
pruebas tipodonaciones.actualizar('Prueba 2- actualizar tipodonacion con oid null', NULL, 'Equipaciones de Golf', 'material', false);
pruebas tipodonaciones.actualizar('Prueba 3- actualizar tipodonacion con nombre null', OID TDon, NULL, 'material', false);
pruebas tipodonaciones.actualizar('Prueba 4- actualizar tipodonacion con tipoUnidad null', OID TDon, 'Equipaje deportivo adaptado', NULL, false);
pruebas tipodonaciones.eliminar('Prueba 1- eliminar tipodonacion ', OID TDon, true);
END:
-- PRUEBAS DONACIONES
DECLARE
OID Don INTEGER;
OID Don2 INTEGER:
BEGIN
pruebas donaciones.inicializar;
pruebas donaciones.insertar('Prueba 1- insertar donación de persona', NULL, '13227182M', 2, '12','1200','10/12/2018', true);
OID Don:=SEC DON.CURRVAL:
pruebas donaciones.insertar('Prueba 2- insertar donación de institución', 'H45681524', NULL, 3, '4', '4500','15/11/2018', true);
OID Don2:=SEC DON.CURRVAL:
pruebas_donaciones.insertar('Prueba 3- insertar donación con cantidad null', NULL, '13227182M', 2, NULL,'1200','07/10/2018', false);
pruebas donaciones.insertar('Prueba 4- insertar donación con valorUnitario null', 'H45681524', NULL, 3, '4', NULL,'02/05/2017', false);
pruebas_donaciones.insertar('Prueba 5- insertar donación con OID_TDon null', 'H45681524', NULL, '4', '4500','08/04/2017', false);
pruebas_donaciones.actualizar('Prueba 1- actualizar donación de persona', OID_Don,NULL, '13227182M', 2, '10','900','26/05/2017', true);
pruebas donaciones.actualizar('Prueba 2- actualizar donación con OID null', NULL, NULL, '13227182M', 2, '18','2500','21/12/2018', false);
pruebas donaciones.eliminar('Prueba 1- eliminar donación', OID Don2, true);
--PRUEBAS MENSAJES
DECLARE
OID M INTEGER:
OID M2 INTEGER:
BEGIN
pruebas mensaies.inicializar:
pruebas_mensajes.insertar('Prueba 1- insertar mensaje', 3, 'newsletter', '26/12/2018', '¡Llega el Mercadillo de Navidad!', 'Código HTML de la newsletter', true);
```

```
OID M:=SEC M.CURRVAL:
pruebas mensajes.insertar ('Prueba 2- insertar mensaje', 2, 'email', '12/05/2018', 'Recordatorio de las Actividades próximas', '¿Te gustaría participar en alguna de estás
actividades?...', true);
OID M2:=SEC M.CURRVAL:
pruebas mensajes.insertar('Prueba 4- insertar mensaje', 2, 'informe', '13/01/2017', 'Recuerdos del 2016', 'Comenzamos el año recordando las actividades que se realizaron en 2016...',
pruebas mensajes.insertar('Prueba 5- insertar mensaje con OID Coord null', NULL, 'informe', '13/01/2017', 'Recuerdos del 2016', 'Comenzamos el año recordando las actividades que se
realizaron en 2016...', false);
pruebas mensajes.insertar('Prueba 6- insertar mensaje con tipo null', 2, NULL, '13/01/2018', 'Recuerdos del 2017', 'Comenzamos el año recordando las actividades que se realizaron en
2017...', false):
pruebas mensaies insertar ('Prueba 7- insertar mensaie con fechaEnvio null', 3, 'informe', NULL, 'Recuerdos del 2016', 'Comenzamos el año recordando las actividades que se realizaron en
2016...', false):
pruebas mensajes.insertar('Prueba 8- insertar mensaje con asunto null', 2, 'informe', '13/01/2017', NULL, 'Comenzamos el año recordando las actividades que se realizaron en 2016...',
pruebas mensajes.insertar('Prueba 9- insertar mensaje con contenido null', 3, 'informe', '13/01/2017', 'Recuerdos del 2016', NULL, false);
pruebas mensajes.actualizar('Prueba 1- actualizar mensaje con OID M null', NULL, 3, 'newsletter', '26/12/2018', '¡Llega el Mercadillo de Navidad!', 'Código HTML de la newsletter',
pruebas mensajes.actualizar('Prueba 2- actualizar mensaje', OID M, 3, 'newsletter', '26/12/2017', '¡Llega el Mercadillo de Navidad!', 'Código HTML de la newsletter', true);
pruebas mensaies.eliminar('Prueba 1- eliminar mensaie', OID M2, true):
--PRUEBAS ENVIOS
DECLARE
OID Env INTEGER:
OID Env2 INTEGER;
BEGIN
pruebas envios.inicializar:
pruebas envios.insertar('Prueba 1- insertar un envío', 1, '66641234M', NULL, true);
OID Env:=SEC ENV.CURRVAL:
pruebas envios.insertar('Prueba 2- insertar un envío', 1, NULL, 'A33219876', true);
OID Env2:=SEC ENV.CURRVAL;
pruebas envios.insertar('Prueba 3- insertar un envío con OID M null', NULL, 'A33219876', false);
pruebas envios.actualizar('Prueba 1- actualizar un envío con OID Env null', NULL, 3, NULL, 'A33219876', false);
pruebas envios.actualizar('Prueba 2- actualizar un envío'. OID Env. 3. '66641234M'. NULL. true):
pruebas envios.eliminar('Prueba 1- eliminar un envío', OID Env2, true);
END:
--PRUEBAS PREGUNTAS
DECLARE
OID Q INTEGER;
OID 02 INTEGER:
pruebas preguntas.inicializar;
pruebas preguntas.insertar('Prueba 1- insertar pregunta', 'numerica', 'Valore el progama deportivo del 1 al 10', true);
OID_Q:=SEC_Q.CURRVAL;
pruebas preguntas.insertar('Prueba 2- insertar pregunta', 'textual', '¿Qué parte de la actividad le ha gustado más?', true);
OID 02:=SEC O.CURRVAL;
pruebas preguntas.insertar('Prueba 2- insertar pregunta', 'textual', 'Si tuviera la posibilidad, ¿Realizaría de nuevo la actividad?', true);
pruebas preguntas.insertar('Prueba 3- insertar pregunta con tipo null', NULL, '¿Qué has aprendido en el evento?', false );
pruebas preguntas.insertar('Prueba 4- insertar pregunta con enunciado null', 'opcional', NULL, false):
pruebas preguntas.actualizar('Prueba 1- actualizar pregunta con OID O null'.NULL.'textual', '¿Qué parte de la actividad le ha gustado más?', false);
pruebas preguntas.actualizar('Prueba 2- actualizar pregunta',OID Q2, 'textual', '¿Qué mejoraría de la actividad?', true);
pruebas preguntas.eliminar('Prueba 1- eliminar pregunta', OID Q, true);
END:
```

```
-- PRUEBAS CUESTIONARIOS
DECLARE
OID Cues INTEGER;
OID Cues2 INTEGER:
BEGIN
pruebas cuestionarios.inicializar;
pruebas cuestionarios.insertar('Prueba 1- insertar cuestionarios', 2,'06/05/2017', true);
OID Cues:=SEC CUES.CURRVAL;
pruebas cuestionarios.insertar('Prueba 2- insertar cuestionarios', 3, '02/06/2019', true);
OID Cues2:=SEC Cues.CURRVAL:
pruebas cuestionarios.insertar('Prueba 3- insertar cuestionarios con fecha null', 3, NULL, false):
pruebas cuestionarios.insertar('Prueba 4- insertar cuestionarios con OID Act null', NULL, '02/06/2019', false);
pruebas cuestionarios.actualizar('Prueba 1- actualizar cuestionario con OID Cues null', 3, NULL,'02/06/2019', false);
pruebas cuestionarios.actualizar('Prueba 2- actializar cuestionario', OID_Cues2, 3,'05/01/2019', true);
pruebas cuestionarios.eliminar('Prueba 1- eliminar cuestionario', OID Cues, true);
--PRUEBAS FORMULARIOS
DECLARE
OID Form INTEGER:
OID Form2 INTEGER;
BEGIN
pruebas formularios.inicializar;
pruebas formularios.insertar('Prueba 1- insertar formulario'.2 ,2, true):
OID Form:=SEC FORM.CURRVAL;
pruebas formularios.insertar('Prueba 2- insertar formulario'.2. 3.true):
OID Form2:=SEC FORM.CURRVAL:
pruebas formularios.insertar('Prueba 3- insertar formulario con OID Cues null', NULL, 2, false);
pruebas formularios.insertar('Prueba 4- insertar formulario con OID Q null', 2, NULL, false);
pruebas formularios.actualizar('Prueba 1- actualizar formulario con OID Form null', NULL,2,2, false);
pruebas formularios.actualizar('Prueba 2- acualizar formulario', OID Form, 2,3,true);
pruebas formularios.eliminar('Prueba 1- eliminar formulario', OID Form2,true);
--PRUEBAS RESPUESTAS
DECLARE
OID Ans INTEGER;
OID Ans2 INTEGER;
BEGIN
pruebas respuestas.inicializar;
pruebas respuestas.insertar('Prueba 1- insertar respuesta', 1, 2, NULL, 'Sin ninguna duda volvería a realizarla', true):
OID Ans:=SEC ANS.CURRVAL:
pruebas respuestas.insertar('Prueba 2- insertar respuesta', 1, NULL, 1, 'No me ha gustado, me esperaba mucho más, no volvería a hacerla', true);
OID Ans2:=SEC ANS.CURRVAL;
pruebas_respuestas.insertar('Prueba 3- insertar respuesta con OID_Form null', NULL, NULL, 3, 'Me ha parecido muy interesante y me encantaría repetirla con más amigos',false);
pruebas_respuestas.insertar('Prueba 3- insertar respuesta con contenido null', 1, 3, NULL, NULL ,false);
pruebas respuestas.actualizar('Prueba 1- actualizar respuesta con OID Ans null', NULL, 1, 2, NULL, 'Sin ninguna duda volvería a realizarla', false);
pruebas_respuestas.actualizar('Prueba 1- actualizar respuesta',OID_Ans2,1, NULL, 3, 'No me ha gustado, me esperaba mucho más, no volvería a hacerla',true);
pruebas respuestas.eliminar('Prueba 1- eliminar respuesta', OID Ans, true);
END:
```

```
-- PRUEBAS de FUNCIONES
-- Calcular coste de inscripción a actividad
BEGIN
DBMS OUTPUT.PUT LINE(calcularCosteInscripcion(1650, 45) | ' €');
-- Calcular fecha de vencimiento de pago de recibo
DBMS OUTPUT.PUT LINE(calcularFechaVencimiento(SYSDATE));
END:
-- PRUEBAS de PROCEDIMIENTOS
EXEC Registrar_Persona('87554158T', 'María', 'López López', '08/10/1990', 'C/ Almirante Alcatraz, 45, 5ºC'. 'Madrid'. 'Madrid'. '28023'. 'malolopez@email'. '695487265'):
EXEC Registrar Persona ('52109812T', 'Juan', 'Calero Galera', '23/03/1986', 'C/ Colibrí, 12, 1ºA', 'Madrid', 'Madrid', '28023', 'juan_calega@hotmail.com', '615423515');
EXEC Registrar Persona('12312542V', 'Pedro', 'Hidalgo Pérez', '07/08/2010', 'C/ Torreones, 16, 36B', 'Madrid', 'Madrid', '28023', null, '674444897');
EXEC Registrar Persona('84478354]', 'Amaia', 'Domínguez Cordón', '29/03/1999', 'C/ Batalla de Bailén, 12, 4ºB', 'Madrid', 'Madrid', '28003', 'amaia docor@hotmail.com', '666555848');
EXEC Registrar Persona('45873564M', 'María Dolores', 'Cordón Durán', '23/03/1974', 'C/ Batalla de Bailén, 12, 4ºB', 'Madrid', 'Madrid', '28003', 'dolo cduran@hotmail.com',
'631554915');
EXEC Act Persona('52109812T', 'Juan Antonio', 'Calero Galera', '23/03/1986', 'C/ Colibrí, 12, 1ºA', 'Madrid', 'Madrid', '28023', 'juan calega@hotmail.com', '615423515');
EXEC Eliminar Persona('87554158T'):
EXEC Registrar Coordinador('43623893B', 'Mari Carmen', 'Muñoz Jiménez', '15/08/1989', 'C/ Cerros, 62, 1ºA', 'Madrid', 'Madrid', '28023', 'mari-mimenez@deporteydesafio.com',
'629123456'):
EXEC Registrar_Voluntario('65971564N', 'Pablo', 'Cabello Marín', '11/11/1982', 'C/ Narciso, 46, 8ºA', 'Madrid', 'Madrid', '28015', 'cabello_marin@gmail.com', '658679123');
EXEC Registran_Voluntario('54786321L', 'Pedro', 'Castaño Moreno', '23/11/1984', 'C/ Lorenzo, 57, 1ºA', 'Madrid', 'Madrid', '28015', 'petercamo@gmail.com', '688132231');
EXEC Registrar TutorLegal('11123453V', 'Mateo', 'Ruiz López', '06/02/1978', 'C/ Miguel Hernández, 27, 2ºB', 'Madrid', 'Madrid', '28017', 'mateo ruiz lopez@gmail.com', '614888909');
EXEC Registrar_Participante('15484468E', 'Alicia', 'Torcal Molar', '17/09/2006', 'C/ Cerezo, 2, 4ºC', 'Madrid', 'Madrid', '28086', null, '623168465', '0,45', '54786321L', '11123453V');
EXEC Registrar Institucion('A15359575', 'Animaciones Ilusiones Mágicas, S.A.', '677599884', 'C/ Malagueños, 1', 'Madrid', 'Madrid', '28001', 'info@ilusionesmagicas.com');
EXEC Act Institucion('A87614532', 'Agencia de Seguros Eliseo, S.A.', '654123989', 'C/ Preciados, 45', 'Madrid', 'Madrid', '18001', 'info@seguros-eliseo.com');
EXEC Eliminar Institucion('A15359575'):
EXEC Registrar Proyecto ('43623893B', 'Sierra Nevada Adaptada. Temporada 2019', 'Granada', 0, 1);
EXEC Registrar Proyecto('43623893B', 'IV Jornadas sobre la Fibrosis Quística', 'Centro Deportivo Pío Baroja, Madrid', 1, 0);
EXEC Add Actividad('Esquí adaptado', 'Mejorar la condición física de los participantes e introducirles en el deporte adaptado de invierno', '10/01/2019', '16/01/2019', '30, 'deportiva',
2300, 9);
EXEC Add Actividad('Jornada sobre el reto solidario 2plega2', 'Informar a los interesados sobre la iniciativa solidaria del joven jerezano.', '08/01/2019', '08/012019', 45, 'social',
100, 10);
EXEC Registrar Patrocinador('A87984532', 'Inditex, S.A.', '654123989', 'C/ Gran Vía, 12', 'Madrid', 'Madrid', '18001', 'info@inditex.com');
EXEC Add Patrocinio('A87984532', 14, 1800);
EXEC Registrar_Donacion('45873564M', null, 'Materiales lúdicos', 'juegos', 25, '30,90');
EXEC Add InformeMedico(5, 'Problemas respiratorios derivados de la fibrosis quística. Debe tomar la medicación prescrita rigurosamente.');
EXEC Inscribir Participante('15484468E', 14);
EXEC Act EstadoRecibo(9, 'pagado');
EXEC Inscribir Participante('15484468E', 3);
EXEC Inscribir Voluntario('65971564N', 2):
EXEC Registrar Mensaje('newsletter', '26/12/2018','¡Llega el Mercadillo de Navidad!', 'Código HTML de la newsletter', 2):
EXEC Registrar Envio('84478354J', null, 1);
EXEC Registrar Envio('65971564N', null, 1);
EXEC Registrar Envio('11123453V', null, 1):
EXEC Registrar Envio(null, 'A87984532', 1);
EXEC Act Recibo(3, '31/12/2018', '10,00', 'pagado');
EXEC Registrar_Pregunta('textual', '¿Qué sugerencias haría para mejorar la actividad de cara a futuras ediciones?');
```

```
EXEC Registrar_Pregunta('numerica', 'Valore de 0 a 10 el evento');
EXEC Registrar Cuestionario(14);
EXEC Registrar Cuestionario(15);
EXEC Add Formulario(6, 5);
EXEC Add Formulario(7, 5);
EXEC Add Formulario(7, 6);
EXEC Registrar_Respuesta(null, 5, 5, 'Aumentar el número de días de actividades en la nieve.');
EXEC Registrar Respuesta(7, null, 6, '8');
EXEC Act Pregunta(2, 3, 'opcional', 'Eliga los tenderetes que más le gustaron');
_____
-- PRUEBAS de CURSORES
EXEC FichaParticipante(2);
EXEC FichaParticipante(5);
EXEC FichaVoluntario(1);
EXEC FichaVoluntario(7);
EXEC FichaPatrocinador('A87674532');
EXEC FichaPatrocinador('A87984532');
EXEC GET CuestionariosPart(14);
EXEC GET_CuestionariosVol(3);
EXEC ListaDonantes;
EXEC Lista Email('voluntarios');
EXEC Lista Email('participantes');
EXEC Lista VolAct(2);
EXEC Lista VolAct(3);
EXEC Lista HistVol(3);
EXEC Lista HistVol(7);
EXEC Lista_PartAct(2);
EXEC Lista_PartAct(3);
EXEC Lista_HistPart(2);
EXEC Lista HistPart(3);
EXEC Lista_PatrociniosAct(3);
EXEC Lista PatrociniosAct(14);
EXEC Lista_DonTemp('01/01/2017','31/12/2019');
EXEC Lista_ActTemp('01/01/2019','01/03/2019');
```

# 9. ANEXOS

# Anexo I: Prototipo de newsletter



Detalles jornada sabado 20 de ocutbre

# Anexo II: Acta de reunión

### Información de control

Proyecto	Deporte y Desafío
Cliente	Fundación Deporte y Desafío
Lugar de la reunión	Videoconferencia vía Skype (Sevilla – Madrid)
Fecha de realización	08/10/2018

### Lista de convocados

Nombre y Apellidos	Organismo	Asiste (S/N)
Mario Ruano Fernández	IS-G1-SSR Fundación	S
	Deporte y Desafío	
Cristina Caro Caro	Fundación Deporte y Desafío	S

### Orden del día

- 1. Estructura organizativa de la Fundación Deporte y Desafío.
- 2. Modelo de negocio y flujos de trabajo.
- 3. Principales problemáticas en la gestión de proyectos.
- 4. Expectativas del sistema.

# Aprobación de acta

Representante IS-G1-SSR Deporte y Desafío	Representante Fundación Deporte y Desafío	
	Ctra. El Pardo K. 3,400 - 28035 Mark	
Fdo.: Mario Ruano Fernández	Fdo.: Cristina Caro Caro	
Fecha: 10/10/2018	Fecha: 10/10/2018	

#### Desarrollo de la reunión

Cristina Caro Caro, en representación de la Fundación Deporte y Desafío, es coordinadora de programas deportivos en dicha institución. Junto con otras tres coordinadoras, forma un equipo encargado de la gestión íntegra de los diferentes programas deportivos y actividades que ofrece la fundación a sus participantes.

#### 1. Estructura organizativa de la Fundación Deporte y Desafío.

Deporte y Desafío es una fundación española sin ánimo de lucro que lleva trabajando veinte años por la inclusión social de las personas con discapacidad a través del deporte adaptado.

Con sede principal en Madrid, la fundación, de carácter privado, depende del Ministerio de Educación y Ciencia.

Fue fundada en 1998 por Jorge Pérez de Leza, a raíz de sufrir éste un accidente y quedar en silla de ruedas. Él fue una de las principales personas en España en apoyar y promover la práctica del deporte adaptado.

La fundación está gestionada a través de un patronato que se encarga de aprobar las cuentas anuales, las partidas de gastos, así como la representación institucional de la fundación.

La dirección de la fundación recae en manos de Carmen Pardo Martín, directora general de Deporte y Desafío, la cual se apoya en un equipo formado por cuatro coordinadoras de programas deportivos, encargadas de realizar todas las tareas de planificación y gestión de cada proyecto que se lleva a cabo en el seno de la fundación.

Deporte y Desafío también cuenta con un director técnico para el programa de esquí alpino adaptado que desarrollan anualmente en Sierra Nevada. Al ser una de las principales actividades de cada temporada y al contarse con amplio material adaptado para la nieve, desde el organismo consideran importante que exista un enlace permanente entre las oficinas de Madrid y Granada.

La fundación cuenta también con personal contable y una responsable de integración laboral.

### 2. Modelo de negocio y flujos de trabajo

En la actualidad, Deporte y Desafío cuenta con veintitrés programas deportivos activos diferentes, los cuales se reparten entre cursos trimestrales, jornadas puntuales y actividades de larga duración.

Los cursos trimestrales se reparten a lo largo del año a través de una sesión semanal; las jornadas suelen ser días concretos o eventos de un par de días de duración, normalmente los fines de semana; mientras que las actividades de larga duración comprenden actividades de mayor despliegue, como campamentos semanales, viajes, actividades en Sierra Nevada durante ocho semanas, o el Camino de Santiago.

Deporte y Desafío vive de subvenciones tanto públicas como privadas. Cuentan con un amplio número de patrocinadores que son los encargados de financiar los diferentes proyectos que

realizan cada año, existiendo una distinción entre patrocinadores según su nivel de importancia en cuanto a la cuantía subvencionada. Quedan clasificados como patrocinadores Oro, Plata, Bronce y Colaboradores.

Algunas instituciones patrocinadoras son La Caixa, Banco Santander, El Corte Inglés, Fundación Telefónica, Comunidad de Madrid, la Federación Madrileña de Deportes de Discapacitados Físicos, el Real Patronato de Discapacidad, Adidas, Coca-Cola etc.

Gracias a todas estas subvenciones es posible llevar a cabo todos y cada uno de los programas deportivos que se realizan en la fundación.

La labor de planificación y gestión recae en manos del equipo de cuatro coordinadoras, con la directora general al frente. Son ellas las encargadas de llevar a cabo todos los proyectos, desde la búsqueda de subvenciones, presentando el proyecto a los diferentes patrocinadores, pasando por la planificación de personal, gestión de inscripciones de participantes, formalización de equipos de voluntariado, alquiler de equipamiento, alojamientos y espacios, así como la ejecución y supervisión de las actividades en sí, terminando con la evaluación de estas y la elaboración de informes y memorias.

### 3. Principales problemáticas en la gestión de proyectos.

La principal problemática que resulta de la gestión de los programas deportivos que lleva a cabo la Fundación Deporte y Desafío es la mala economización de los tiempos durante el periodo de planificación y organización previo a la ejecución de las actividades.

Una de las causas de mayor peso en este problema es que no cuentan con una base de datos real en la que guardan de forma ordenada y clasificada la información de los participantes, voluntarios y demás agentes que han participado con anterioridad en actividades de la fundación. Esto es un problema a la hora de realizar una búsqueda o un proceso selectivo, de cara a organizar al personal, u obtener la lista de participantes seleccionados para una nueva edición de un programa deportivo.

Dado que los datos personales de contacto de estas personas están alojados en un documento de Excel, deben realizar búsquedas manuales y rastreos con cruce de otros documentos (como inscripciones pasadas) para, por ejemplo, ver qué participantes son los que hace más tiempo que no participan en una actividad. Además de que esta tarea les requiere bastante tiempo, este documento es bastante vulnerable, ya que también comentan que no existe un respaldo de este. No cuentan con copia de seguridad.

Anteriormente contaban con un programa informático que les registraba los datos personales de estas personas, pero dejaron de trabajar con él, ya que lo consideraron obsoleto y nunca llegaron a tenerlo actualizado.

Por ello, cada vez que se comienza a planificar una nueva actividad, en lo que a personal y a participantes se refiere, se repite prácticamente el mismo proceso de búsqueda manual, llamadas o cruce de correos electrónicos con cada persona que es posible candidata para participar.

Otra cuestión problemática es la relacionada con la recopilación de datos que derivan de la realización de una actividad, principalmente datos de cuestionarios de evaluación y satisfacción que se recogen tras la finalización de un programa deportivo. Estos cuestionarios

son respondidos tanto por participantes como por voluntarios, así como por tutores legales en algunas ocasiones. Dichos datos son escaneados y guardados en una carpeta, no existiendo posibilidad alguna de contabilizar de manera automática datos cuantitativos.

Toda esta rigidez de los datos impide realizar informes de calidad que posteriormente se envía a los patrocinadores de las propias actividades. En muchas ocasiones ni tan siquiera se envían, no siendo un gran problema, pero desde Deporte y Desafío consideran que es una necesidad para mantener buenas relaciones con los patrocinadores y asegurarse la continuidad de actividades en el futuro.

### 4. Expectativas del sistema.

Desde la Fundación Deporte y Desafío esperan que un sistema de información bien estructurado pueda dar solución a su problemática de tiempos, permitiéndole gestionar los proyectos con mayor brevedad y eficiencia durante la fase de planificación, pudiendo dedicar más tiempo a otros aspectos de la organización.

Del mismo modo, contar con un mejor almacenamiento de los datos y una mayor facilidad de tratar con estos, teniendo además una mayor seguridad de los mismos.

Por último, poder generar mejores informes, con mejores datos para el envío de las memorias de los programas a los patrocinadores correspondientes o para crear la memoria anual de manera más completa.