deporteydesafio.com

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	- Requisitos generales: Desde el punto de vista del responsable. - Número de página modificado a la derecha. - Incluida información sobre expectativas del sistema. - Anexo añadido. - Enlaces externos en el pie de página. - Página web de la fundación añadida. - Índice actualizado. - Acta de reunión con cliente.	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	- Requisitos generales: generalización de los procesos de gestión de proyectos, emisión de mensajes y gestión de cuentas. - Requisitos de información: faltaban referencias a entidades de importancia. - Pruebas de aceptación: se añade identificadores unívocos para cada una. - Modelado conceptual: mejora de algunas asociaciones, inclusión de algunas restricciones, especificación del tipo de herencia.	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

Sobre el cliente

Deporte y Desafío[1] 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.

[1] Página web de la fundación: https://deporteydesafio.com

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**[2]: Ejercicio o juego que desempeñan los participantes (en eventos o programas deportivos) para disfrutar o entretenerse con otras personas.
- **Boletín**[3]: 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**[4]: 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**[5]: Documento donde quedan recogidos los resultados y el desarrollo de los proyectos, junto con pequeños informes de cada actividad, efectuados durante el año.
- **Newsletter**[6]: 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.

[2] Esquí alpino: https://deporteydesafio.com/programas-deportivos/actividades-squi-alpino.html
[3] Sección de noticias: https://www.deporteydesafio.com/noticias/page/2/
[4] Sección de eventos: https://www.deporteydesafio.com/noticias/eventos/
[5] Memoria anual 2017: https://www.deporteydesafio.com/newsletter/newsletter-2017.html
[6] Newsletter: https://mailchi.mp/31acfe3133bc/jornadas-8-de-junio-senderismo-y-geocaching-1659937?e=41becb9e91

- **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**[7]: Entidad pública o privada, que subvenciona actividades con fines publicitarios. Según la aportación realizada, se clasifican en oro, plata, bronce o colaborador.
- **Programa deportivo**[8]: 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.

[7] Directorio de patrocinadores: https://deporteydesafio.com/patrocinadores.html
[8] Oferta de programas deportivos: https://deporteydesafio.com/programas-deportivos.html

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.

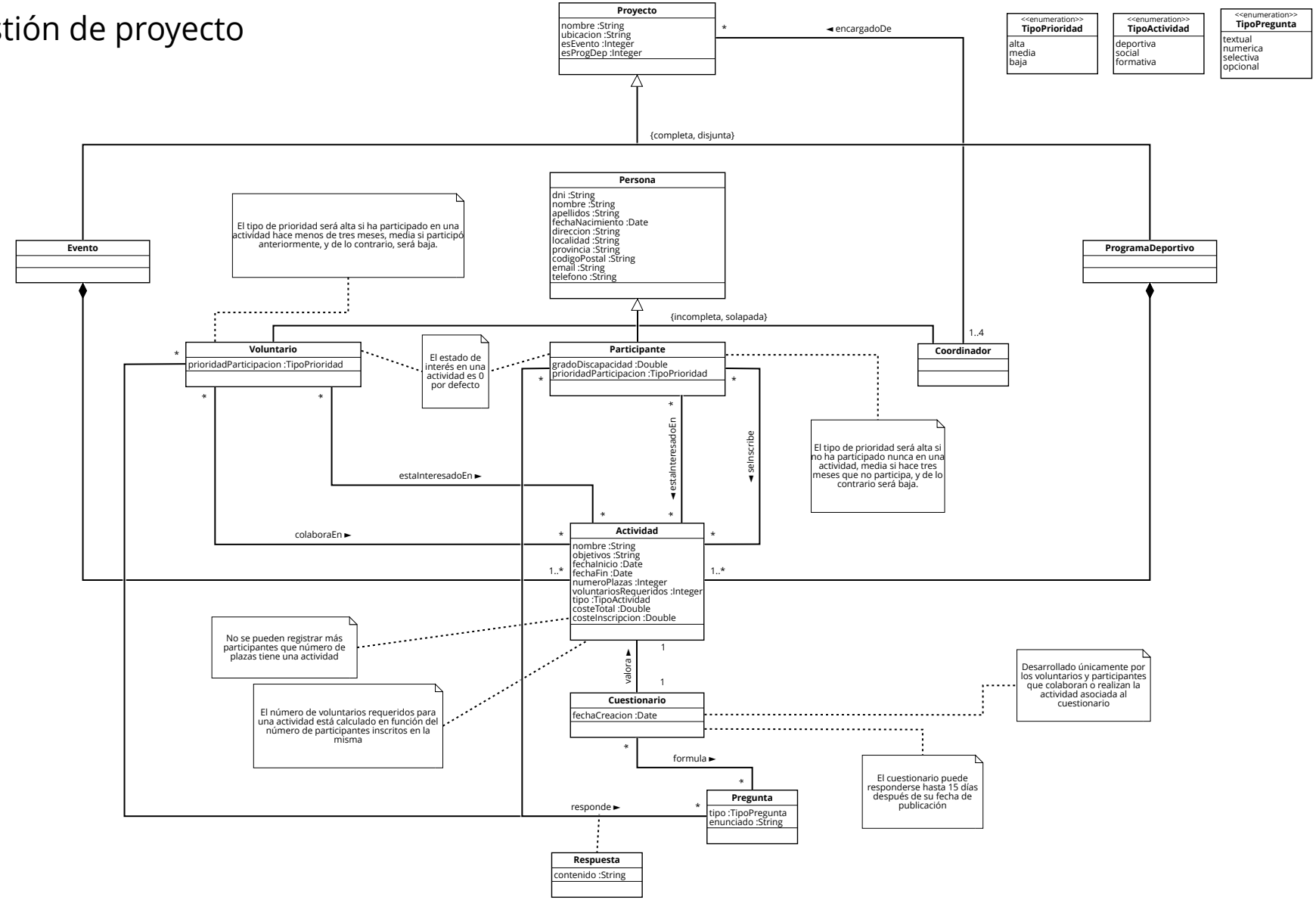
RNF – 5 Usabilidad I
El sistema debe ser accesible para los usuarios desde navegadores web tanto de dispositivos móviles como de ordenadores portátiles y de sobremesa.

RNF – 6 Usabilidad II
El sistema debe dar acceso a cualquier punto de la aplicación en un máximo de tres clics.

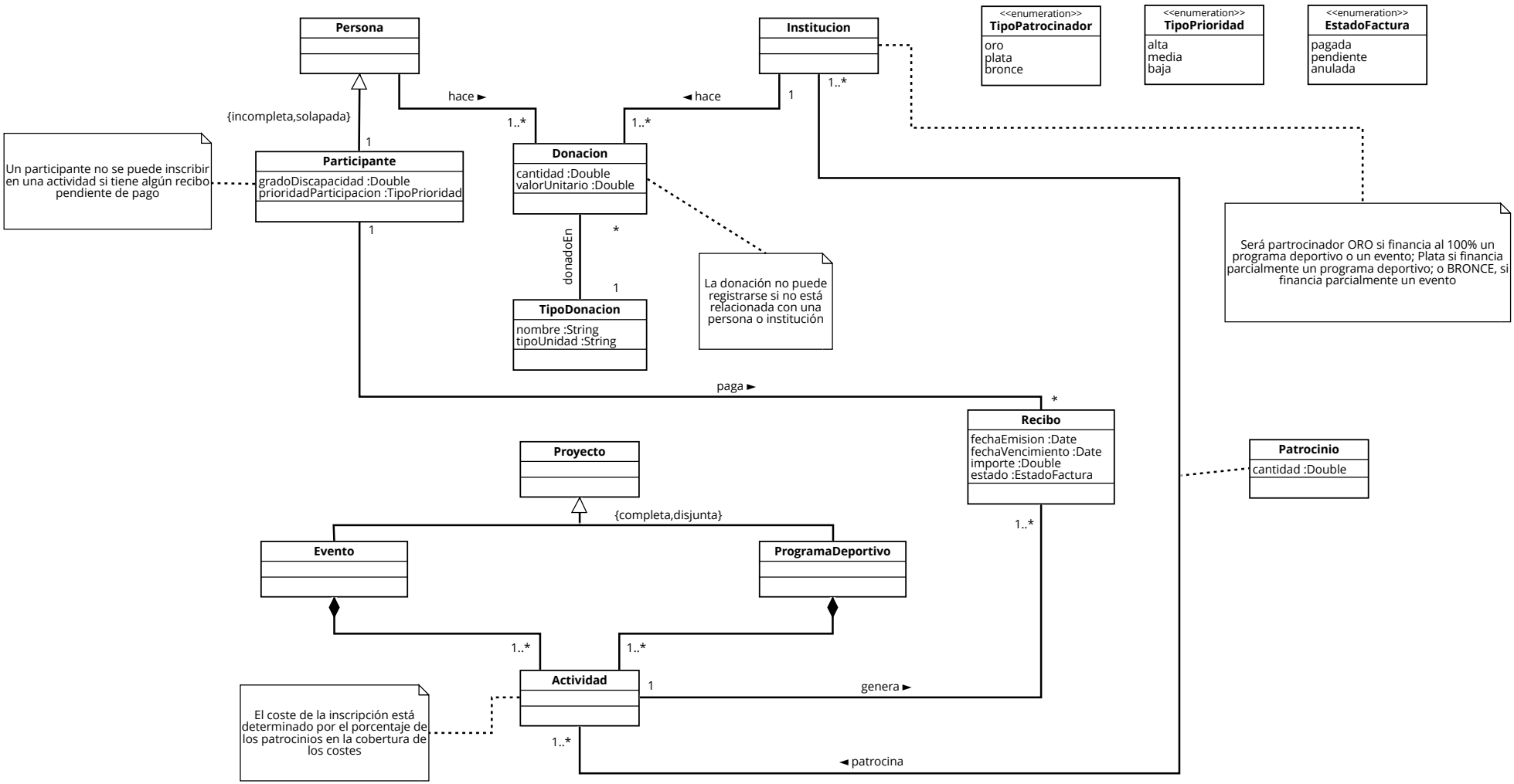
5. MODELO CONCEPTUAL

Modelado UML

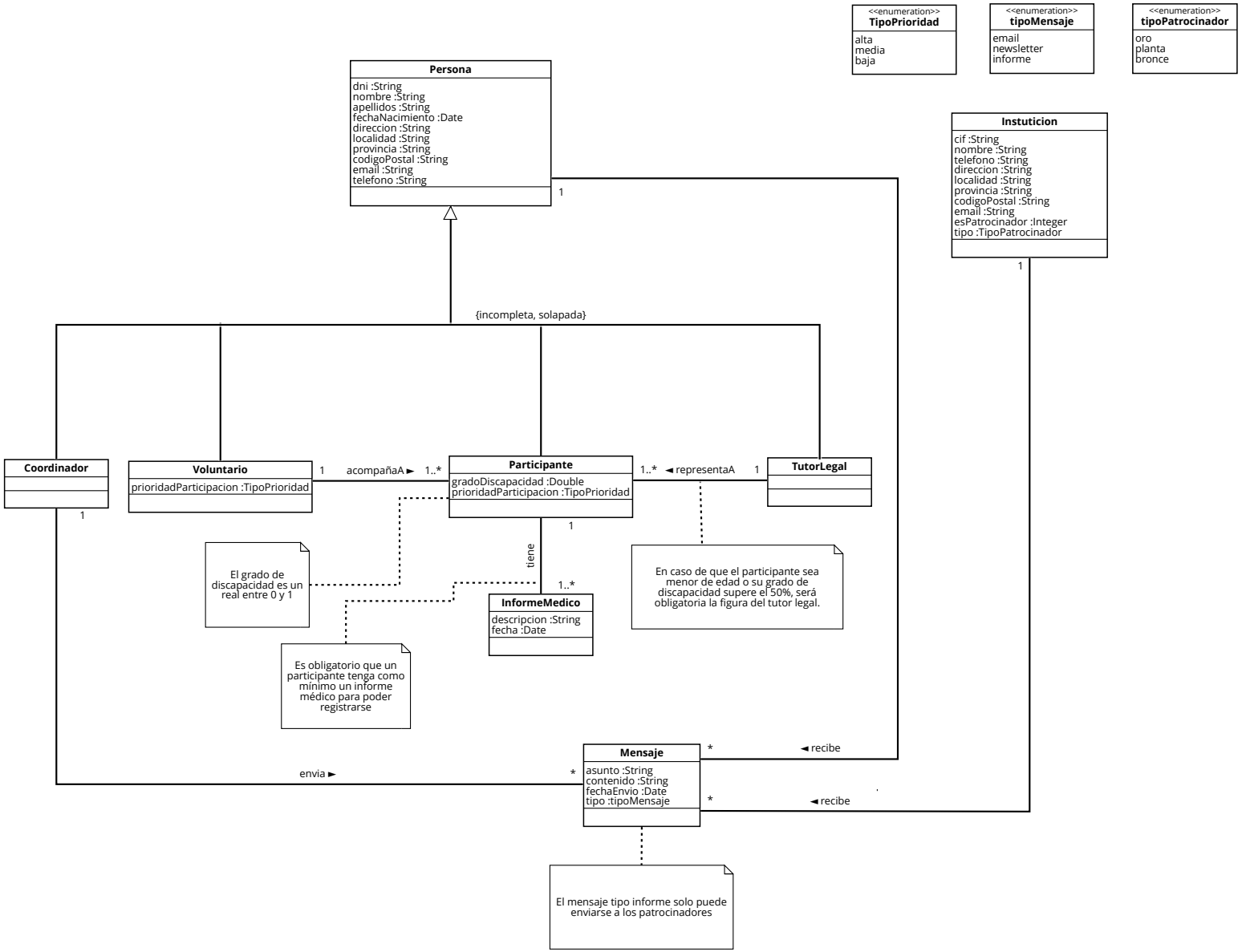
Gestión de proyecto



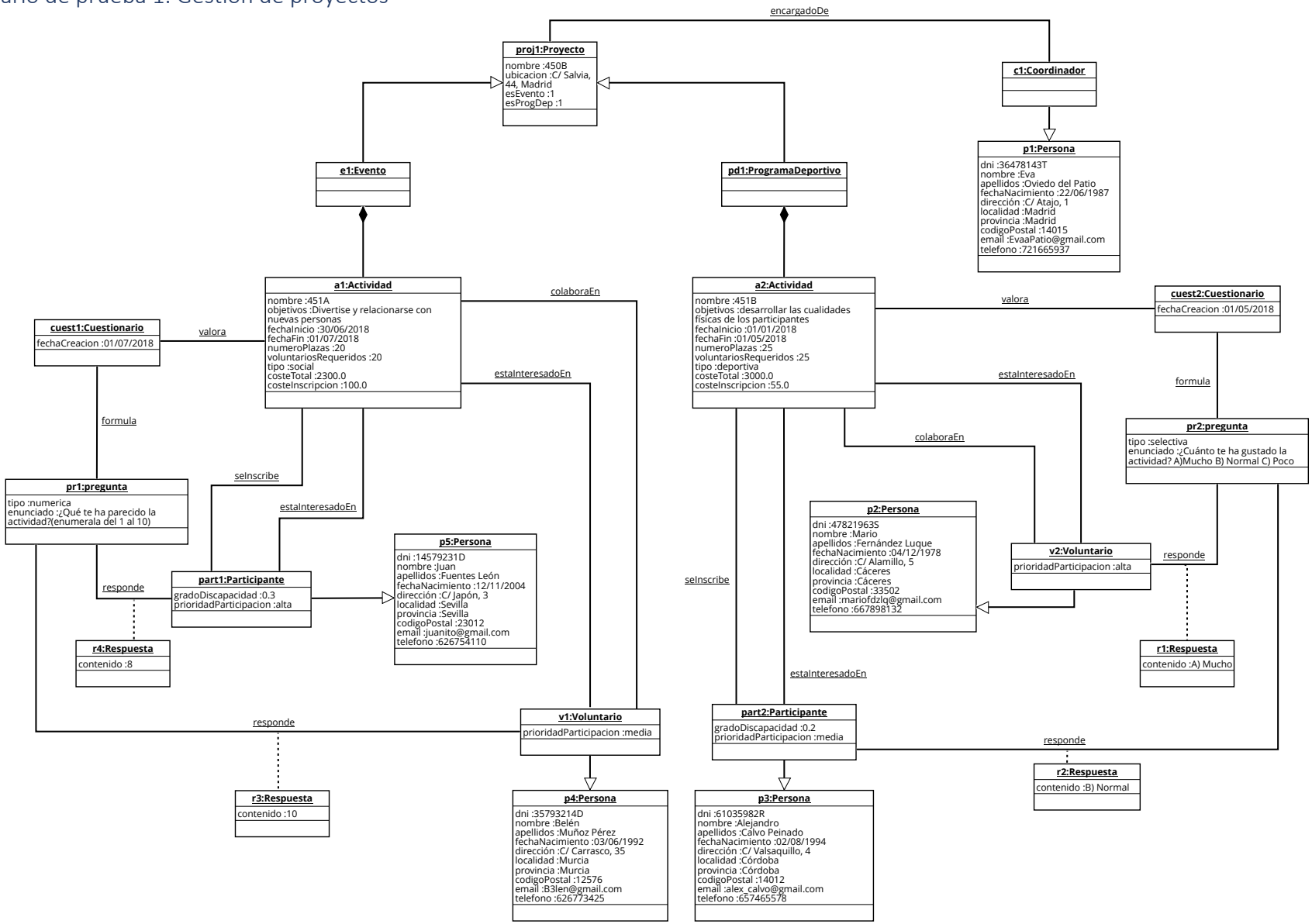
Gestión de cuentas



Gestión de comunicaciones



Escenario de prueba 1. Gestión de proyectos



Descripción textual

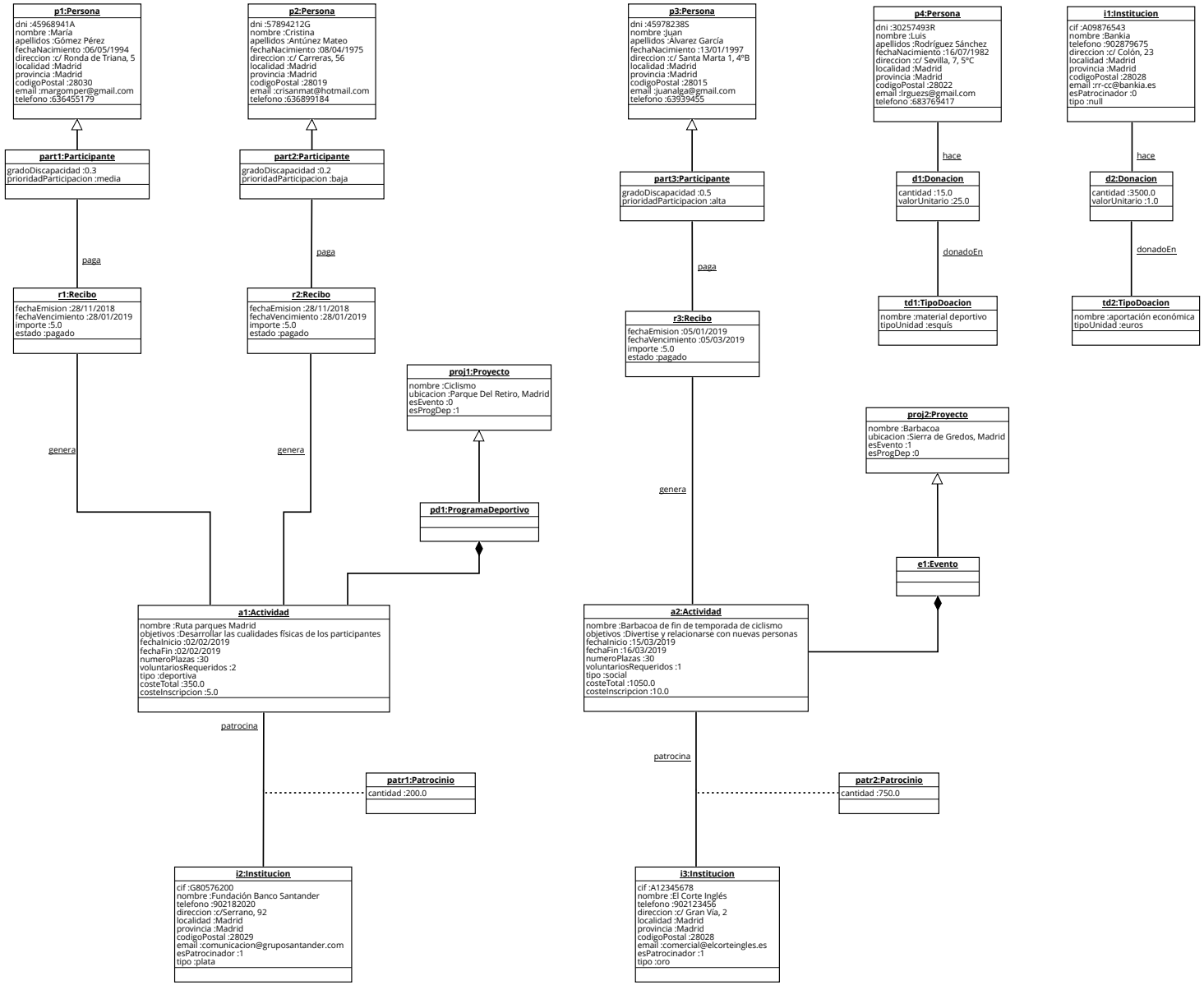
En este escenario de prueba definimos 5 personas (p1, p2, p3, p4, p5), siendo p1, coordinador (c1), p4 y p2, voluntarios (v1, v2) y p5 y p3, participantes (part1, part2).

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

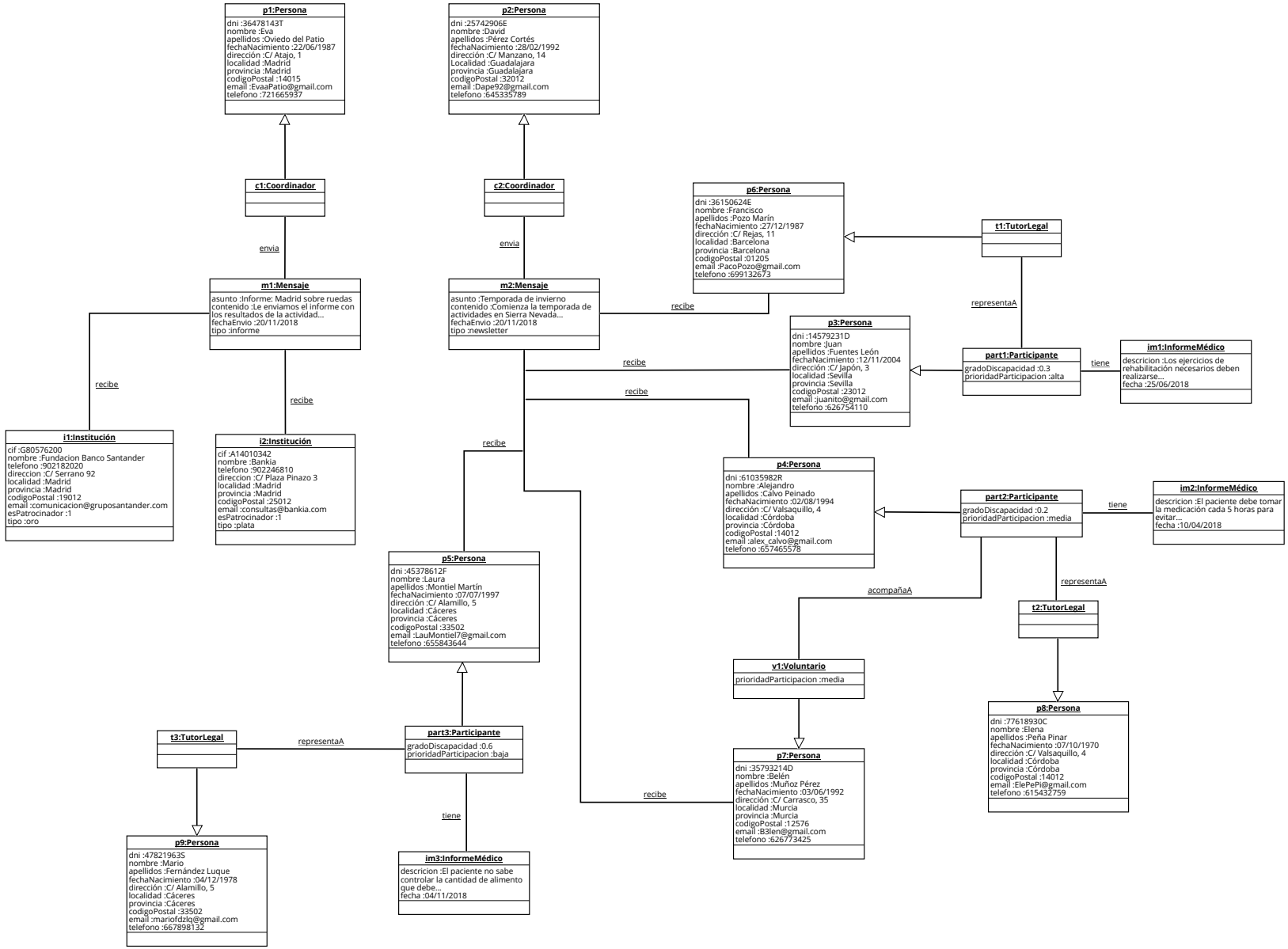
En este escenario de prueba definimos 4 personas (p1, p2, p3, p4), siendo 3 de ellas participantes (part1, part2, part3).

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).


p1:Persona
dni :36478143T
nombre :Eva
apellidos :Oviedo del Patio
fechaNacimiento :22/06/1987
dirección :C/ Atajo, 1
localidad :Madrid
provincia :Madrid
codigoPostal :14015
email :EvaaPatio@gmail.com
telefono :721665937
p2:Persona
dni :25742906E
nombre :David
apellidos :Pérez Cortés
fechaNacimiento :28/02/1992
dirección :C/ Manzano, 14
Localidad :Guadalajara
provincia :Guadalajara
codigoPostal :32012
email :Dape92@gmail.com
telefono :645335789
c1:Coordinador
c2:Coordinador
envia
envia
m1:Mensaje
asunto :Informe: Madrid sobre ruedas
contenido :Le enviamos el informe con los resultados de la actividad...
fechaEnvio :20/11/2018
tipo :informe
m2:Mensaje
asunto :Temporada de invierno
contenido :Comienza la temporada de actividades en Sierra Nevada...
fechaEnvio :20/11/2018
tipo :newsletter
p6:Persona
dni :36150624E
nombre :Francisco
apellidos :Pozo Marín
fechaNacimiento :27/12/1987
dirección :C/ Rejas, 11
localidad :Barcelona
provincia :Barcelona
codigoPostal :01205
email :PacoPozo@gmail.com
telefono :699132673
t1:TutorLegal
representaA
recibe
recibe
recibe
i1:Institución
cif :G80576200
nombre :Fundacion Banco Santander
telefono :902182020
direccion :C/ Serrano 92
localidad :Madrid
provincia :Madrid
codigoPostal :19012
email :comunicacion@gruposantander.com
esPatrocinador :1
tipo :oro
i2:Institución
cif :A14010342
nombre :Bankia
telefono :902246810
direccion :C/ Plaza Pinazo 3
localidad :Madrid
provincia :Madrid
codigoPostal :25012
email :consultas@bankia.com
esPatrocinador :1
tipo :plata
p3:Persona
dni :14579231D
nombre :Juan
apellidos :Fuentes León
fechaNacimiento :12/11/2004
dirección :C/ Japón, 3
localidad :Sevilla
provincia :Sevilla
codigoPostal :23012
email :juanito@gmail.com
telefono :626754110
part1:Participante
gradoDiscapacidad :0.3
prioridadParticipacion :alta
tiene
im1:InformeMédico
descricion :Los ejercicios de rehabilitación necesarios deben realizarse...
fecha :25/06/2018
recibe
p4:Persona
dni :61035982R
nombre :Alejandro
apellidos :Calvo Peinado
fechaNacimiento :02/08/1994
dirección :C/ Valsaquillo, 4
localidad :Córdoba
provincia :Córdoba
codigoPostal :14012
email :alex_calvo@gmail.com
telefono :657465578
part2:Participante
gradoDiscapacidad :0.2
prioridadParticipacion :media
tiene
im2:InformeMédico
descricion :El paciente debe tomar la medicación cada 5 horas para evitar...
fecha :10/04/2018
p5:Persona
dni :45378612F
nombre :Laura
apellidos :Montiel Martín
fechaNacimiento :07/07/1997
dirección :C/ Alamillo, 5
localidad :Cáceres
provincia :Cáceres
codigoPostal :33502
email :LauMontiel7@gmail.com
telefono :655843644
acompañaA
representaA
t2:TutorLegal
v1:Voluntario
prioridadParticipacion :media
p8:Persona
dni :77618930C
nombre :Elena
apellidos :Peña Pinar
fechaNacimiento :07/10/1970
dirección :C/ Valsaquillo, 4
localidad :Córdoba
provincia :Córdoba
codigoPostal :14012
email :ElePePi@gmail.com
telefono :615432759
t3:TutorLegal
representaA
part3:Participante
gradoDiscapacidad :0.6
prioridadParticipacion :baja
recibe
p7:Persona
dni :35793214D
nombre :Belén
apellidos :Muñoz Pérez
fechaNacimiento :03/06/1992
dirección :C/ Carrasco, 35
localidad :Murcia
provincia :Murcia
codigoPostal :12576
email :B3len@gmail.com
telefono :626773425
tiene
p9:Persona
dni :47821963S
nombre :Mario
apellidos :Fernández Luque
fechaNacimiento :04/12/1978
dirección :C/ Alamillo, 5
localidad :Cáceres
provincia :Cáceres
codigoPostal :33502
email :mariofdzlq@gmail.com
telefono :667898132
im3:InformeMédico
descricion :El paciente no sabe controlar la cantidad de alimento que debe...
fecha :04/11/2018

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					✔							
Coordinador	✔											
Cuestionario												✔
Donación							✔					
Evento				✔								
Informe Médico		✔										
Institución								✔				
Mensaje											✔	
Participante		✔										
Patrocinador								✔				
Patrocinio									✔			
Persona	✔	✔	✔			✔	✔					
Pregunta												✔
Programa Deportivo				✔								
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	✔	✔					✔		✔		✔	✔			✔
Coordinador								✔							
Cuestionario													✔		
Donación										✔					
Evento	✔	✔					✔								
Informe Médico					✔										
Institución								✔		✔					
Mensaje								✔							
Participante	✔	✔	✔	✔	✔		✔							✔	
Patrocinador									✔						
Patrocinio									✔						
Persona								✔		✔					
Pregunta													✔		
Programa Deportivo	✔	✔					✔								
Proyecto									✔						
Recibo			✔								✔				
Respuesta													✔		
Tutor Legal				✔											
Voluntario						✔	✔								✔

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		✔	✔									
colaboraEn			✔		✔							
donadoEn							✔					
encargadoDe	✔			✔								
envía	✔										✔	
estaInteresadoEn		✔	✔		✔							
formula												✔
genera					✔					✔		
hace							✔					
paga		✔								✔		
patrocina					✔			✔	✔			
recibe		✔	✔			✔	✔	✔			✔	
representaA		✔				✔						
responde		✔	✔									✔
seInscribe		✔			✔							
tiene		✔										
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						✔									
colaboraEn						✔									✔
donadoEn										✔					
encargadoDe								✔							
envía								✔							
estaInteresadoEn							✔								
formula													✔		
genera			✔								✔				
hace										✔					
paga			✔												
patrocina									✔						
recibe								✔							
representaA				✔											
responde													✔		
seInscribe	✔	✔										✔			
tiene					✔									✔	
valora													✔		

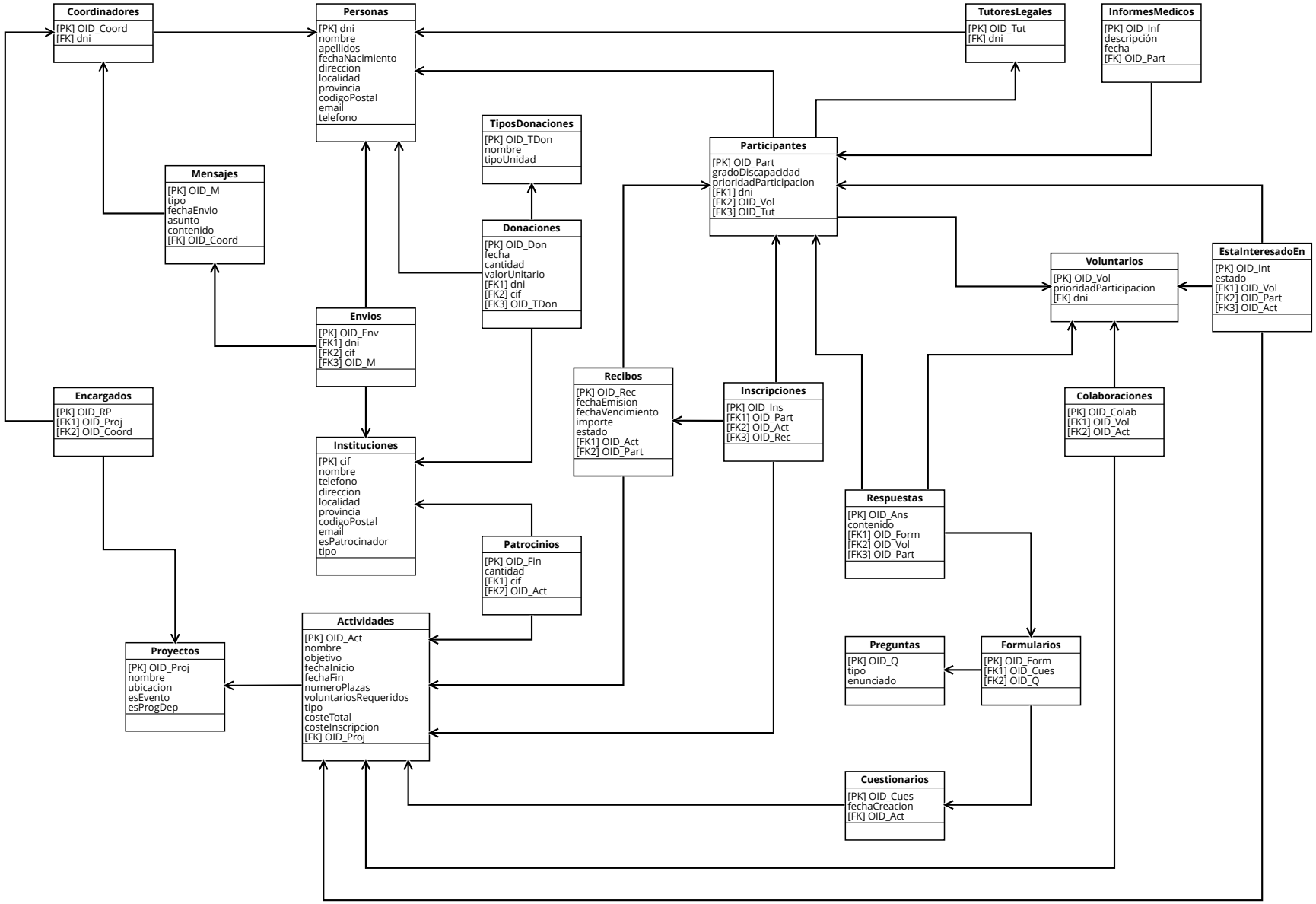
Matriz de restricciones y requisitos de información

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.							✔					
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.								✔				
Un patrocinador PLATA es aquel que financia parcialmente un programa deportivo.								✔				
Un patrocinador BRONCE es aquel que financia parcialmente un evento.								✔				
No se pueden registrar más participantes que número de plazas tiene una actividad.				✔	✔							
Los cuestionarios solo los pueden responder participantes y voluntarios.												✔
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.		✔										
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.				✔	✔							
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.									✔						
Un patrocinador PLATA es aquel que financia parcialmente un programa deportivo.									✔						
Un patrocinador BRONCE es aquel que financia parcialmente un evento.									✔						
No se pueden registrar más participantes que número de plazas tiene una actividad.												✔			
Los cuestionarios solo los pueden responder participantes y voluntarios.													✔		
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.	✔														
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.															✔
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, fechaInicio, fechaFin, numeroPlazas, Voluntarios requeridos, tipo, costeTotal, costeInscripcion, OID_Proj	OID_Act	OID_Proj / Proyectos
Colaboraciones	OID_Colab, OID_Vol, 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, fechaInicio, 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
        CONSTRAINT ck_dni CHECK (REGEXP_LIKE(dni, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]')),
    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
        CONSTRAINT ck_cif CHECK (REGEXP_LIKE(cif, '[A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')),
    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 SEQUENCE SEC_Int;
CREATE SEQUENCE SEC_Int INCREMENT BY 1 START WITH 1;
DROP SEQUENCE SEC_RP;
CREATE SEQUENCE SEC_RP INCREMENT BY 1 START WITH 1;
DROP SEQUENCE 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_Q;
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_EQUALS (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;
BEGIN
    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 mensajes y 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;
    ELSE
        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);
    END IF;
    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;
    ELSE
        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;
BEGIN
    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.direccion || ', ' || REG_Part.localidad || ', ' ||
            REG_Part.codigoPostal || ', ' || REG_Part.provincia);
        DBMS_OUTPUT.PUT_LINE('- Prioridad de participación: ' || REG_Part.prioridadParticipacion);
        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);
    END LOOP;
    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);
    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 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.PUT_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.direccion || ', ' || 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;
BEGIN
    FOR REG_Ins IN C_Patrocinador LOOP
        DBMS_OUTPUT.PUT_LINE('================================================================================');
        DBMS_OUTPUT.PUT_LINE('  FICHA DE PATROCINADOR');
        DBMS_OUTPUT.PUT_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.direccion || ', ' || REG_Ins.localidad || ', ' ||
            REG_Ins.codigoPostal || ', ' || REG_Ins.provincia);
    END 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);
    END LOOP;
    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;
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 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_Q=P.OID_Q
            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);
    END LOOP;
    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 LOOP;
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('--------------------------------------------------------------------------------');
    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.direccion || ', ' || 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));
        END 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('=============================================================================================================================');
    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));
    END 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 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,40) || 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('  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));
    END 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;
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('  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.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 || ' Y EL ' || 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 IF;
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 REQU.',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 IF;
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;
BEGIN
    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 IF;
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 PAQUETE
---------------------------------
--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<>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 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,salidaEsperada));
        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
        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_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
    BEGIN
        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;
    BEGIN
        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_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_Part INTEGER, salidaEsperada BOOLEAN) AS
    salida BOOLEAN:=true;
    n_participantes INTEGER;
    BEGIN
        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_EQUALS(false,salidaEsperada));
        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_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_Inf INTEGER, w_OID_Part INTEGER, w_descripcion LONG, w_fecha DATE, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
informeMedico INFORMESMEDICOS%ROWTYPE;
BEGIN
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_EQUALS(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,salidaEsperada));
```

```
        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_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_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_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_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,salidaEsperada));
    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;
    proyecto 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.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 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_Proj=w_OID_Proj;
    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,salidaEsperada));
        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_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;
    w_OID_Act INTEGER;
    BEGIN
        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<>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_EQUALS(false, salidaEsperada));
        ROLLBACK;
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, 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.fechaFin<>w_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_EQUALS(false, salidaEsperada));
   ROLLBACK;
END actualizar;

PROCEDURE eliminar(nombre_prueba VARCHAR2, w_OID_Act INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
n_actividades INTEGER;
BEGIN
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,salidaEsperada));
        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_EQUALS(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;

    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_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;

        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_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;

    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_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,salidaEsperada));
        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<>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 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));
            ROLLBACK;
    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_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_Fin INTEGER, w_cif CHAR , w_cantidad NUMBER, w_OID_Act INTEGER, salidaEsperada BOOLEAN) AS
salida BOOLEAN:=true;
patrocinio PATROCINIOS%ROWTYPE;
BEGIN
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;

    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_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,salidaEsperada));
    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_EQUALS(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 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_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_mensajes<>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_EQUALS(false,salidaEsperada));
        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_Q=w_OID_Q;
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_Q=w_OID_Q;
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 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_Q=w_OID_Q;
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_EQUALS(salida, salidaEsperada));

    EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
        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_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_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_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_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_Q= w_OID_Q 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_Q<>w_OID_Q) 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_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,salidaEsperada));
    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_EQUALS(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_EQUALS(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 PAQUETES
--------------------------------------------------------------------------------

--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','76184359Q', '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', '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);
END;
/
--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', '76184359Q',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);
END;
/
--PRUEBAS_VOLUNTARIOS
DECLARE
OID_Vol INTEGER;
OID_Vol2 INTEGER;
BEGIN
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;
BEGIN
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);
END;
/
--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);
END;
/
--PRUEBAS_PROYECTOS
DECLARE
OID_Proj 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_Proj:=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_Proj2:=SEC_PROJ.CURRVAL;
pruebas_proyectos.insertar('Prueba 3- inserción proyecto 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_proyectos.insertar('Prueba 6- inserción proyecto 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);
END;
/
--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', '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', '70', 'deportiva', '400', '5', false);
pruebas_actividades.insertar('Prueba 6- insertar actividad con fechaInicio 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 y 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 VoluntariosRequeridos 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);
END;
/
--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_estaInteresadoEn.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;
BEGIN
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);
END;
/
--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', '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', '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',
'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', 'Madrid',
'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);
END;
/
--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, 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);
END;
/
--PRUEBAS_MENSAJES
DECLARE
OID_M INTEGER;
OID_M2 INTEGER;
BEGIN
pruebas_mensajes.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...',
true);
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_mensajes.insertar('Prueba 7- insertar mensaje 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...',
false);
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',
false);
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_mensajes.eliminar('Prueba 1- eliminar mensaje', OID_M2, true);
END;
/
--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, 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_Q2 INTEGER;
BEGIN
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_Q2:=SEC_Q.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_Q 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);
END;
/
--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);
END;
/
--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) || ' €');
END;
/

-- Calcular fecha de vencimiento de pago de recibo
BEGIN
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@gmail', '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, 3ºB', 'Madrid', 'Madrid', '28023', null, '674444897');
EXEC Registrar_Persona('84478354J', '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 Registrar_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', '28001', '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', '28001', '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 Deporte y Desafío	S
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
 Fdo.: Mario Ruano Fernández Fecha: 10/10/2018	 Fdo.: Cristina Caro Caro 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.