



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 5](#_Toc534422995)

[Sobre el cliente 5](#_Toc534422996)

[Sobre los usuarios 5](#_Toc534422997)

[Estructura organizacional 6](#_Toc534422998)

[Gestión de proyectos 6](#_Toc534422999)

[Medición de resultados y evaluación de proyectos 6](#_Toc534423000)

[Dominio del problema 7](#_Toc534423001)

[2. GLOSARIO DE TÉRMINOS 8](#_Toc534423002)

[3. VISIÓN GENERAL DEL SISTEMA 10](#_Toc534423003)

[Expectativas del sistema 10](#_Toc534423004)

[Tipos de usuarios 10](#_Toc534423005)

[4. CATÁLOGO DE REQUISITOS 11](#_Toc534423006)

[Requisitos generales 11](#_Toc534423007)

[Requisitos de información 12](#_Toc534423008)

[Reglas de negocio y pruebas de aceptación 15](#_Toc534423009)

[Requisitos funcionales 20](#_Toc534423010)

[Requisitos no funcionales 23](#_Toc534423011)

[5. MODELO CONCEPTUAL 24](#_Toc534423012)

[Modelado UML 24](#_Toc534423013)

[Escenarios de prueba 27](#_Toc534423014)

[Escenario de prueba 1. Gestión de proyectos 27](#_Toc534423015)

[Escenario de prueba 2. Gestión de cuentas 29](#_Toc534423016)

[Escenario de prueba 3. Gestión de comunicaciones 31](#_Toc534423017)

[6. MATRICES DE TRAZABILIDAD 33](#_Toc534423018)

[Matriz de clases de entidad y requisitos de información 33](#_Toc534423019)

[Matriz de clases de entidad y reglas de negocio 34](#_Toc534423020)

[Matriz de asociaciones y requisitos de información 35](#_Toc534423021)

[Matriz de asociaciones y reglas de negocios 36](#_Toc534423022)

[Matriz de restricciones y requisitos de información 37](#_Toc534423023)

[Matriz de restricciones y reglas de negocio 38](#_Toc534423024)

[7. MODELO RELACIONAL 39](#_Toc534423025)

[Justificación de estrategias de transformación de jerarquías 40](#_Toc534423026)

[8. MODELO TECNOLÓGICO 42](#_Toc534423027)

[Tablas, secuencias y triggers de secuencias 42](#_Toc534423028)

[Funciones y procedimientos 53](#_Toc534423029)

[Cursores y listados de consultas 63](#_Toc534423030)

[Triggers 73](#_Toc534423031)

[Pruebas 79](#_Toc534423032)

[9. ANEXOS 130](#_Toc534423033)

[Anexo I: Prototipo de newsletter 130](#_Toc534423034)

[Anexo II: Acta de reunión 131](#_Toc534423035)

# INTRODUCCIÓN AL PROBLEMA

## Sobre el cliente

Deporte y Desafío[[1]](#footnote-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.

## 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).

# GLOSARIO DE TÉRMINOS

* **Actividad**[[2]](#footnote-2): Ejercicio o juego que desempeñan los participantes (en eventos o programas deportivos) para disfrutar o entretenerse con otras personas.
* **Boletín**[[3]](#footnote-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]](#footnote-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]](#footnote-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]](#footnote-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.
* **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]](#footnote-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]](#footnote-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.

# 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.

# 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. |

# MODELO CONCEPTUAL

## Modelado UML

## Escenarios de prueba

### 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).

### Escenario de prueba 3. Gestión de comunicaciones

#### Descripción textual

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

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

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

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

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

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

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

# MATRICES DE TRAZABILIDAD

## Matriz de clases de entidad y requisitos de información



## Matriz de clases de entidad y reglas de negocio



## Matriz de asociaciones y requisitos de información



## Matriz de asociaciones y reglas de negocios



## Matriz de restricciones y requisitos de información



## Matriz de restricciones y reglas de negocio



# MODELO RELACIONAL



## 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.

# 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');

# 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.

1. 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.

1. 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.

1. 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.

1. Página web de la fundación: <https://deporteydesafio.com> [↑](#footnote-ref-1)
2. Esquí alpino: <https://deporteydesafio.com/programas-deportivos/actividades-squi-alpino.html> [↑](#footnote-ref-2)
3. Sección de noticias: <https://www.deporteydesafio.com/noticias/page/2/> [↑](#footnote-ref-3)
4. Sección de eventos: <https://www.deporteydesafio.com/noticias/eventos/> [↑](#footnote-ref-4)
5. Memoria anual 2017: <https://www.deporteydesafio.com/newsletter/newsletter-2017.html> [↑](#footnote-ref-5)
6. Newsletter: <https://mailchi.mp/31acfe3133bc/jornadas-8-de-junio-senderismo-y-geocaching-1659937?e=41becb9e91> [↑](#footnote-ref-6)
7. Directorio de patrocinadores: <https://deporteydesafio.com/patrocinadores.html> [↑](#footnote-ref-7)
8. Oferta de programas deportivos: <https://deporteydesafio.com/programas-deportivos.html> [↑](#footnote-ref-8)