21-10-2018

Proyecto AFAR

Programas que ayudan a la gente

Cáceres Romero, David

Crespo Bravo, Natalia

González Martín, Alejandro

Pradas Fernández, Vanessa

Enlace a Projetsii:

<https://projetsii.informatica.us.es/projects/mfhxda39he4hzl26l3c>

Proyecto iissi 1

|  |  |  |
| --- | --- | --- |
| Versión | Descripción | Participantes |
| Primer entregable | Informe recopilatorio del problema y del funcionamiento del sistema. | Cáceres Romero, David  Crespo Bravo, Natalia  González Martín, Alejandro  Pradas Fernández, Vanessa |
| SEGUNDO  ENTREGABLE | Modificación de la visión general del sistema, añadido requisitos (generales, de información, funcionales y no funcionales), pruebas de aceptación, reglas de negocio, diagrama conceptual, mapa de historias de usuario, casos de prueba, acta de reunión (anexo 2), referencias externas y matrices de trazabilidad. | Cáceres Romero, David  Crespo Bravo, Natalia  González Martín, Alejandro  Pradas Fernández, Vanessa |
| TERCER ENTREGABLE | Correspondientes al segundo entregable: añadido el objetivo préstamo; modificación de todos los requisitos, así como las reglas de negocio y sus correspondientes pruebas de aceptación. Modificación completa de la matriz de trazabilidad y del diagrama conceptual.  Correspondientes al tercer entregable: realización del modelo relacional en 3FN, elaboración de tablas, secuencias, funciones , procedimientos, triggers y pruebas de aceptación en sql. | Cáceres Romero, David  Crespo Bravo, Natalia  González Martín, Alejandro  Pradas Fernández, Vanessa |

Índice

[Introducción al problema 3](#_Toc535186837)

[Glosario de términos 5](#_Toc535186838)

[Visión general del sistema 6](#_Toc535186839)

[Requisitos del Sistema y pruebas de aceptación 7](#_Toc535186840)

[1.1. Requisitos Generales (Objetivos) 7](#_Toc535186841)

[1.2. Requisitos de Información 8](#_Toc535186842)

[1.3. Reglas de negocio 10](#_Toc535186843)

[1.4. Requisitos funcionales 12](#_Toc535186844)

[1.5. Requisitos no funcionales. 14](#_Toc535186845)

[Pruebas de aceptación 16](#_Toc535186846)

[Mapas de historias de usuario 18](#_Toc535186847)

[Diagrama conceptual 19](#_Toc535186848)

[Matriz de trazabilidad 22](#_Toc535186849)

[Casos de Prueba 23](#_Toc535186850)

[Modelo relacional 3FN 28](#_Toc535186851)

[Referencias externas 29](#_Toc535186852)

[Anexo A. Acta de la reunión con los responsables de AFAR 30](#_Toc535186853)

[Anexo B: Acta de la segunda reunión con AFAR 32](#_Toc535186854)

[Anexo C: Código SQL 33](#_Toc535186855)

# Introducción al problema

La entidad a la que queremos ofrecer nuestro servicio presenta como principales problemas una base de datos desfasada e incompleta, y una página web carente de actualizaciones; la base de datos fue implementada por un informático contratado por la entidad y la página web fue proporcionada por una subvención de la Junta de Andalucía. Dicha entidad se llama AFAR y es una organización con treinta años de experiencia, que actualmente consta de doce profesionales como equipo técnico que gestionan centros en los que trabajan con personas sin hogar o en riesgo de exclusión social.

Algunos de los beneficiados por esta asociación viven en el recinto de esta, situado en Alcalá de Guadaíra (Sevilla). Y cuentan con centros médicos, donde se realizan pruebas paulatinamente, tales como exámenes psicológicos o tratamiento de enfermedades. Además, se les proporciona la supervisión constante de monitores que les apoyan en el ámbito cotidiano y organizan una serie de actividades diarias, desde asambleas en grupo hasta lecturas.

Las unidades administrativas de las que se dispone son, entre otros, centros CAPI que proporcionan acceso a internet a personas que están tecnológicamente atrasadas tanto en conocimiento como monetariamente; centros de acogida a personas sin hogar, donde se les pone al día con su información personal (DNI, salud, situación y demás documentación); y centros de formación profesional, que les ayuda a integrarse en el mundo laboral.

La base de datos actual se realizó en 2007 y genera una copia de seguridad mensual. Al final del año, los trabajadores tienen que extraer manualmente datos estadísticos de la escasa información almacenada en los expedientes (citas médicas, estado físico y mental, sanciones, número de residentes, sexo, estado civil y familiar, edad, nacionalidad…), esto supone un problema de tiempo, comodidad y aumenta la posibilidad de cometer errores a la hora de hacer los cálculos y pasar los datos a la plataforma virtual.

La página del centro está desfasada e incompleta, ya que existen apartados sin elaborar como el de formación. Por otro lado, se limita a presentar la organización, por lo que es meramente informativa.



Figura 1. Entrada al recinto de la asociación.

# Glosario de términos

* **Actividad.** Trabajo o acción organizada por los monitores y trabajadores de los centros para contribuir al desarrollo personal y social de los usuarios.
* **Alojamiento.** Lugar de acogida a los usuarios.
* **Asamblea.** Reunión para la puesta en común de ideas y pensamientos.
* **Centros CAPI.** Centros de Acceso Público a Internet.
* **Director.** Persona que está a cargo de una actividad u organización.
* **Enlace Web.** Expresión que conecta una cierta información con otra en la web.
* **Entidad.** Colectivo considerado como unidad, y, en especial, cualquier corporación, compañía o institución tomada como persona jurídica.
* **Estadísticas.** Resumen con porcentajes y/o gráficos extraído de los datos registrados a lo largo de un año.
* **Exclusión social.** Falta de participación de segmentos de la población en la vida social, económica y cultural de sus respectivas sociedades debido a la carencia de derechos, recursos y capacidades básicas.
* **Expediente.** Conjunto de todos los archivos correspondientes a un asunto, negocio o formación laboral de la población.
* **Foro.** Mecanismo o medio mediante el cual podrían realizarse consultas directamente a los responsables de la organización o de la información de esta.
* **Monitor.** Persona que guía el aprendizaje deportivo o cultural.
* **Requisito.** Circunstancia o condición necesaria para realizar cualquier tipo de actividad (desde gestión de información hasta asambleas).
* **Recinto.** Espacio comprendido dentro de determinados límites, usualmente ocupado por alguna construcción.
* **Servidor.** Ordenador especialmente preparado para almacenar datos.
* **Unidad administrativa.** Órgano que tiene acceso a funciones propias que lo distinguen de los demás en la institución.
* **Usuario.** Es «aquel que usa un servicio» o «que usa ordinariamente un servicio».

# Visión general del sistema

Como solución al problema del cliente, proponemos una aplicación software que permita satisfacer las necesidades de esta empresa. Para llevarlo a cabo, nuestro objetivo principal sería la ampliación de la taxonomía y mejora del sistema de clasificación de información, que permita gestionar mejor la recogida de datos de los usuarios por parte del equipo de coordinadores.

Por otra parte, los usuarios tendrán acceso a una mayor cantidad de información y se les facilitará el seguimiento de las actividades que realiza el programa de la entidad. Otro objetivo a tener en cuenta sería el alojamiento en los centros de recogida, que puede solucionarse proporcionando datos sobre la ubicación de dichos lugares asociados a un contacto telefónico. Por último, sería conveniente la existencia de un foro como mecanismo de consulta.

Para terminar, de esta información podemos extraer a modo de resumen el identificador único de cada objetivo:

* **Objetivo1 (obj-1).** Como responsable de AFAR, quiero gestionar los usuarios y

toda la información relacionada con estos, como el nombre, los apellidos, la edad, la nacionalidad, las fechas de nacimiento y de alta, el estado de salud y la asistencia a eventos, entre otras cosas.

* **Objetivo 2 (obj-2).** Como responsable de AFAR, quiero gestionar las

actividades y toda la información relacionada con estas, como fecha de realización, hora de realización, duración y tipos de actividades, como comidas benéficas, lecturas en grupo, etcétera.

* **Objetivo 3 (obj-3).** Como responsable de AFAR, quiero gestionar los

alojamientos y la información relacionada con estos, como los residentes, fecha de entrada y de salida, precio y lugar, entre otros.

* **Objetivo 4 (obj-4).** Como responsable de AFAR, quiero gestionar los

préstamos y toda la información relacionada con estos, como la fecha de realización y de devolución, los atrasos y el estado físico del objeto.

# Requisitos del Sistema y pruebas de aceptación

# Requisitos Generales (Objetivos)

* **RG-1:** Control administrativo.

Como monitora,

quiero tener control sobre la gente que ha pasado por el centro,

para organizar mejor la empresa.

* **RG-2:** Reparto de personal.

Como directora del centro,

quiero controlar el desarrollo de las actividades,

para mejorar la distribución de personal.

* **RG-3:** Control de alojamiento.

Como monitora del centro,

quiero saber la situación de disponibilidad de las habitaciones (ocupado o no) y el estado físico en el que se encuentran,

para ofrecérselas nuevos miembros o antiguos que regresan, con la mayor calidad posible.

* **RG-4:** Revisión de bienes.

Como encargado de préstamos,

quiero controlar el mantenimiento del inventario,

para asegurar las necesidades básicas, evitando así posibles faltas de material y manteniendo la calidad del centro.

# Requisitos de Información

RELACIONADOS CON EL USUARIO

* **RI-1:** Estado médico.

Como monitora autorizada,

quiero poder introducir datos en nuestro sistema de las citas médicas de nuestros internos e informes de salud,

para facilitar el seguimiento médico.

* **RI-2:** Registro de usuarios.

Como monitora autorizada,

quiero poder introducir datos en nuestro sistema de las personas como DNI, estado civil, campo de observaciones, causas de ingreso, entre otros aspectos;

para tener un registro.

* **RI-3:** Estado psicológico.

Como psicóloga del centro,

quiero guardar los resultados de cada sesión,

para llevar un control del estado mental de los pacientes.

* **RI-4:** Evitar faltas de material.

Como gestor de alojamientos,

quiero poder introducir información sobre el material para el uso exclusivo de los residentes,

para así evitar faltas de fármacos, alimentos u otras carencias de primera clase y tener una lista de la compra.

RELACIONADOS CON LAS ACTIVIDADES

* **RI-5:** Actividades.

Como monitora,

quiero que se cuente el número de asistentes, saber la fecha, la hora y lugar de las actividades,

para controlar la motivación de los residentes.

* **RI-6:** Calificaciones.

Como monitora,

quiero que almacenar información relacionada con las actividades y los usuarios, como actitud, habilidad y trabajo en equipo,

poder realizar estadísticas de forma anual.

RELACIONADOS CON EL ALOJAMIENTO

* **RI-7:** Alojamientos.

Como gestor de alojamientos,

quiero poder introducir información sobre las habitaciones asignadas a los beneficiarios: apariencia, ocupada o libre.

RELACIONADOS CON LOS PRÉSTAMOS

* **RI-8:** Mantenimiento del almacén.

Como encargado de mantenimiento,

quiero guardar todos los datos acerca de los artículos disponibles (nombre del objeto, cantidad, si es prestable),

para mantener un control de la disponibilidad a la hora del préstamo.

* **RI-9:** Préstamos.

Como encargado de préstamos,

quiero disponer de información relacionada con quién y por qué motivo solicita el préstamo, fecha de entrada y salida del material en cuestión, así como el estado físico del objeto devuelto.

# Reglas de negocio

RELACIONADAS CON EL USUARIO

* **RN-1:** No es posible tener una sexta sanción.

Como monitora,

quiero que el sistema no sea capaz de almacenar más de cinco sanciones, ya que no sería necesario, porque a la quinta se le expulsa del centro.

* **RN-2:** Seguridad de datos.

Como monitora encargada de la ofimática,

quiero que el sistema sea capaz de diferenciar entre una persona no autorizada y una que sí, mediante el sistema de usuario y contraseña,

para evitar filtrado de información y cumplir con la Ley de protección de datos.

* **RN-3:** Imposibilidad de eliminación de datos.

Como monitora,

quiero que no se pueda eliminar información del sistema de ninguna persona que haya pasado por el centro,

para tener en cuenta el regreso de antiguos miembros.

RELACIONADAS CON LAS ACTIVIDADES

* **RN-4:** Creación de actividades.

Como monitora,

quiero que las actividades solo puedan realizarse si hay un número mínimo de personas confirmadas establecido. Normalmente diez.

.

* **RN-5:** Restricciones para apuntarse a una actividad.

Como monitora, quiero que el sistema no permita apuntarse a una actividad a

alguien con más de tres sanciones por mala conducta.

RELACIONADAS CON EL ALOJAMIENTO

* **RN-6:** Seguimiento de habitaciones.

Como monitora,

quiero que cuando busque las habitaciones, el sistema diferencie las que no están llenas de las que sí,

para facilitarme su seguimiento.

RELACIONADAS CON LOS PRÉSTAMOS

* **RN-7:** Control de préstamos.

Como encargado de préstamos,

quiero que nadie pueda tener dos préstamos al mismo tiempo,

para controlar mejor el inventario.

* **RN-8:** Disponibilidad de préstamos.

Como encargado de préstamos,

quiero que no puedan realizarse préstamos de materiales que no estén disponibles.

.

# Requisitos funcionales

RELACIONADOS CON EL USUARIO

* **RF-1:** Información sobre miembros.

Como monitora,

quiero tener una lista de los miembros,

para registrar información sobre ellos.

* **RF-2:** Estadísticas.

Como monitora encargada de la ofimática,

quiero que el sistema cree estadísticas de forma anual sobre la actitud, las habilidades y el trabajo en equipo de los miembros para las distintas actividades que se realicen, para conocer el avance evolutivo de estos.

* **RF-3:** Actas de conducta.

Como monitora autorizada,

quiero poder introducir informes relacionados con las sanciones o premios por conducta de los usuarios,

para conocer la motivación de los participantes.

* **RF-4:** Buena estancia.

Como residentes de la asociación,

nos gustaría poder informar de nuestras necesidades básicas,

para facilitar nuestra vida en las instalaciones.

* **RF-5:** Alerta por sanciones o por buena actitud.

Como monitora,

quiero que el sistema realice un aviso cuando algún usuario reciba un número máximo de cinco sanciones o más de diez reconocimientos por buen trabajo,

para mediar en consecuencia.

* **RF-6:** Alerta por seguridad de datos.

Como monitora encargada de la ofimática,

quiero que el sistema mande un aviso cuando una persona no autorizada intente acceder a una cuenta de usuario,

para evitar filtrado de información y cumplir con la Ley de protección de datos.

RELACIONADOS CON LAS ACTIVIDADES

* **RF-7:** Aviso de actividades.

Como monitora,

quiero que se realicen avisos para el personal asignado a la nueva actividad,

para evitar faltas de personal u olvidos.

* **RF-8:** Información sobre talleres.

Como monitora,

quiero informar a los miembros sobre los talleres que tenemos en las instalaciones,

para fomentar así la asistencia.

RELACIONADOS CON EL ALOJAMIENTO

* **RF-9:** Expiración del periodo de alojamiento.

Como monitora,

quiero que el sistema me avise si ha expirado el periodo de alojamiento de un usuario,

para confirmar que la habitación está disponible para la llegada de un nuevo miembro.

RELACIONADOS CON LOS PRÉSTAMOS

* **RF-10:** Artículos no devueltos en el periodo.

Como encargado de préstamos,

quiero que el sistema me avise cuando alguien no haya devuelto el artículo prestado dentro del periodo establecido,

para evitar así pérdidas del material.

* **RF-11:** Aviso de devolución.

Como encargado de los préstamos,

quiero que se notifique al beneficiario dos días antes del plazo de devolución, para evitar posibles atrasos.

* **RF-12:** Normativa de préstamos.

Como encargado de préstamos,

quiero que el sistema me avise si voy a realizar un préstamo a un usuario antes del final de su periodo de sanción por no haber devuelto un artículo en el periodo establecido,

para cumplir con la normativa.

* **RF-13:** Ampliación del inventario según la necesidad del público.

Como miembros de la asociación,

nos gustaría poder informar de los artículos que necesitamos tomar prestados,

para que la asociación nos los pueda facilitar.

# Requisitos no funcionales.

* **RNF-1:** Ley de protección de datos.

Como encargada de los registros de los usuarios,

quiero que el acceso a esta información este restringido únicamente a mí y al personal autorizado, para cumplir con la Ley de Protección de Datos.

* **RNF-2:** Sistema con usuario y contraseña.

Como directora de las instalaciones,

quiero un sistema que cuente con usuario y contraseña para todo el equipo de AFAR,

para permitirles acceso a los datos de todos los miembros.

* **RNF-3:** Organización del sistema de préstamos.

Como encargado de los préstamos,

quiero que el sistema organice préstamos de tal modo que los miembros cuenten con un usuario y una contraseña,

para facilitarme el seguimiento de los artículos, así como a ellos la solicitud del objeto en cuestión.

* **RNF-4:** Horario de préstamos.

Como encargado de los préstamos,

quiero que la disponibilidad del sistema de ventas sea de lunes a viernes, a excepción de festivos, con el horario de 9:00 a 14:00 y de 15:00 a 19:00,

para organizar mejor los préstamos.

* **RNF-5:** Alojamiento duplicado.

Como gestor de alojamientos,

quiero que no se asigne el mismo alojamiento a diferentes usuarios si el aforo máximo está completo, o que no se asigne el mismo alojamiento más de una vez al mismo usuario.

* **RNF-6:** Número máximo inscripciones.

Como monitora,

quiero que una vez que se haya completado el cupo de inscripciones a una actividad,

se impida a los usuarios acceder al trámite de inscripción, aunque algunos puedan asistir a los eventos.

* **RNF-7:** Número máximo de préstamos.

Como gestor de préstamos,

quiero que los usuarios no puedan realizar varias veces el mismo pedido si no ha pasado un periodo mínimo de tiempo desde la última vez que lo hicieron, para que el material pueda ser solicitado por los demás usuarios.

* **RNF-8:** Aforo máximo permitido.

Como gestor de alojamientos,

quiero que exista un número máximo de personas que puedan vivir en la vivienda o habitación asignada, para que no se descompense el reparto ni empeore la calidad de vida.

# Pruebas de aceptación

* **PA-1**

Cuando intento poner una sanción a alguien que ya posee seis, el sistema no

me lo permite (RN-1).

* **PA-2**

Cuando un usuario intente acceder el sistema de datos y no posea una

contraseña, se le denegará el acceso (RN-2).

* **PA-3**

Si intento eliminar datos de algún antiguo residente, no podré (RN-3).

* **PA-4**

Cuando intente crear una actividad, si especifico una fecha más cercana a dos

días, dicha actividad no llegará a crearse nunca (RN-4).

* **PA-5**

Si intento apuntarme a una actividad y tengo más de tres sanciones, no podré

(RN-5).

* **PA-6**

Si intento apuntarme a una actividad y queda menos de una hora, no podré

(RN-5).

* **PA-7**

Al consultar la lista de actividades se me presentarán ordenadas por fecha

(RN-6).

* **PA-8**

Cuando busque las habitaciones, se me presentarán diferenciadas las

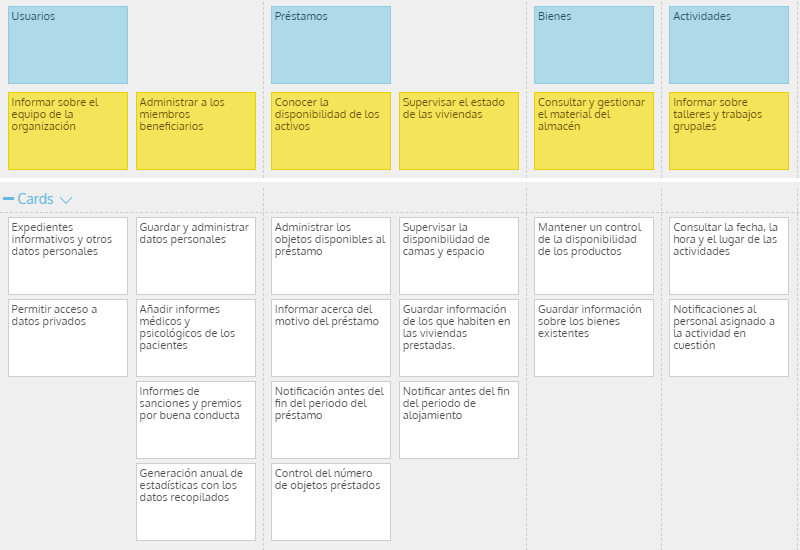
disponibles de las no disponibles (RN-7).

* **PA-9**

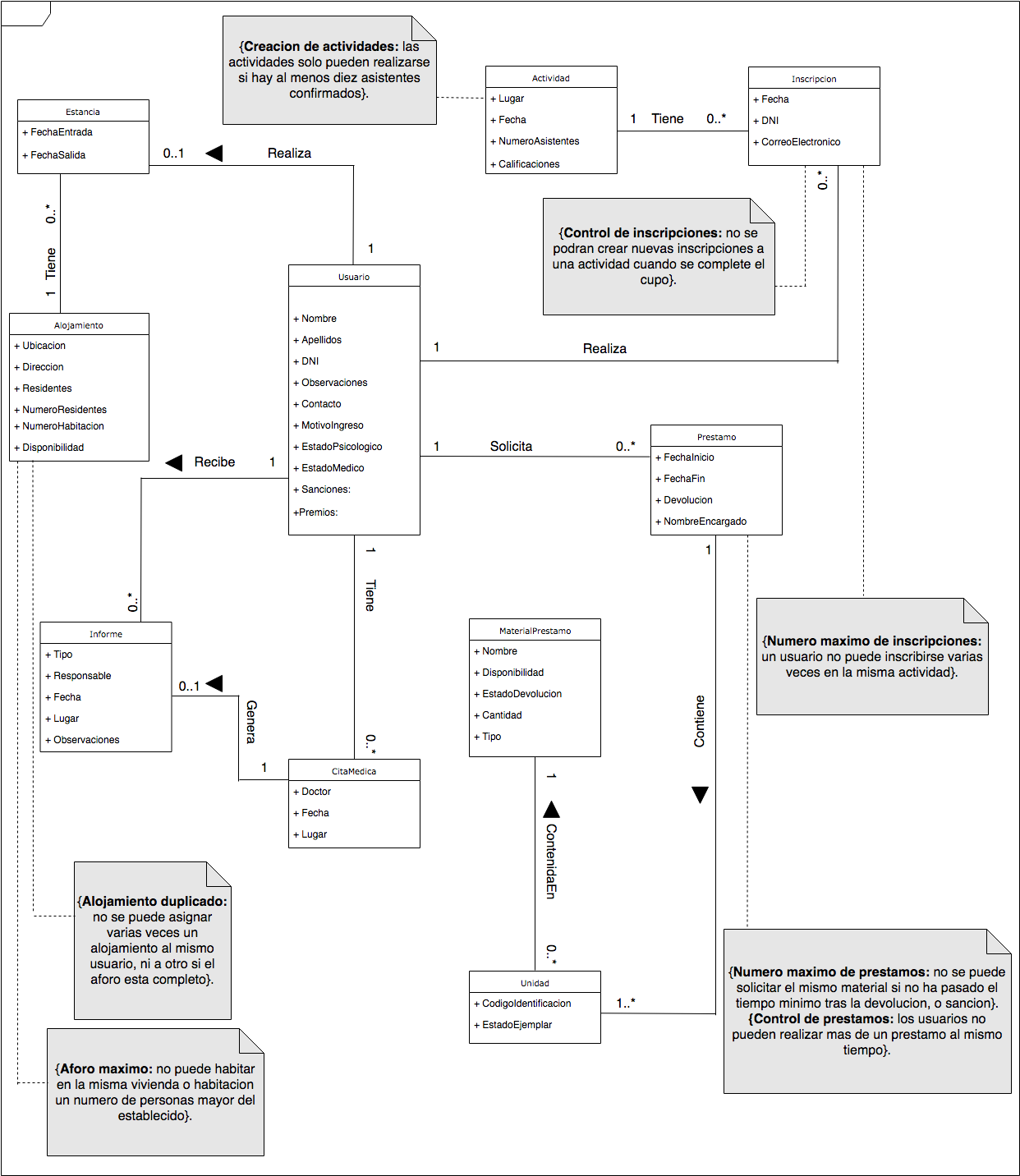
Tras añadir un préstamo, recibiré un error al intentar añadir un segundo (RN

8).

# Mapas de historias de usuario



# Diagrama conceptual



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C/A | RI-1 | RI-2 | RI-3 | RI-4 | RI-5 | RI-6 | RI-7 | RI-8 | RI-9 | RNF-1 | RNF-2 | RNF-3 | RNF-4 | RNF-5 | RNF-6 | RNF-7 | RNF-8 |
| Usuario | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ |  |  | ✓ | ✓ | ✓ |
| Estancia |  |  |  |  |  |  | ✓ |  |  |  |  |  |  | ✓ |  |  | ✓ |
| Alojamiento |  |  |  |  |  |  | ✓ |  |  |  |  |  |  | ✓ |  |  | ✓ |
| Inscripcion |  |  |  |  | ✓ | ✓ |  |  |  |  |  |  |  |  | ✓ |  |  |
| Actividad |  |  |  |  | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |
| Prestamo |  |  |  |  |  |  |  | ✓ | ✓ |  |  | ✓ | ✓ |  |  | ✓ |  |
| Unidad |  |  |  | ✓ |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |
| MaterialPrestamo |  |  |  | ✓ |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |
| CitaMedica | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Informe | ✓ |  | ✓ |  |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |
| Realiza |  |  |  |  | ✓ | ✓ | ✓ |  |  |  |  |  |  |  | ✓ |  | ✓ |
| Recibe | ✓ |  | ✓ |  |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |
| Tiene | ✓ |  | ✓ |  | ✓ | ✓ | ✓ |  |  |  |  |  |  | ✓ |  |  | ✓ |
| Genera | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Solicita |  |  |  |  |  |  |  |  | ✓ |  |  | ✓ |  |  |  | ✓ |  |
| Contiene |  |  |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |
| ContenidaEn |  |  |  | ✓ |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |

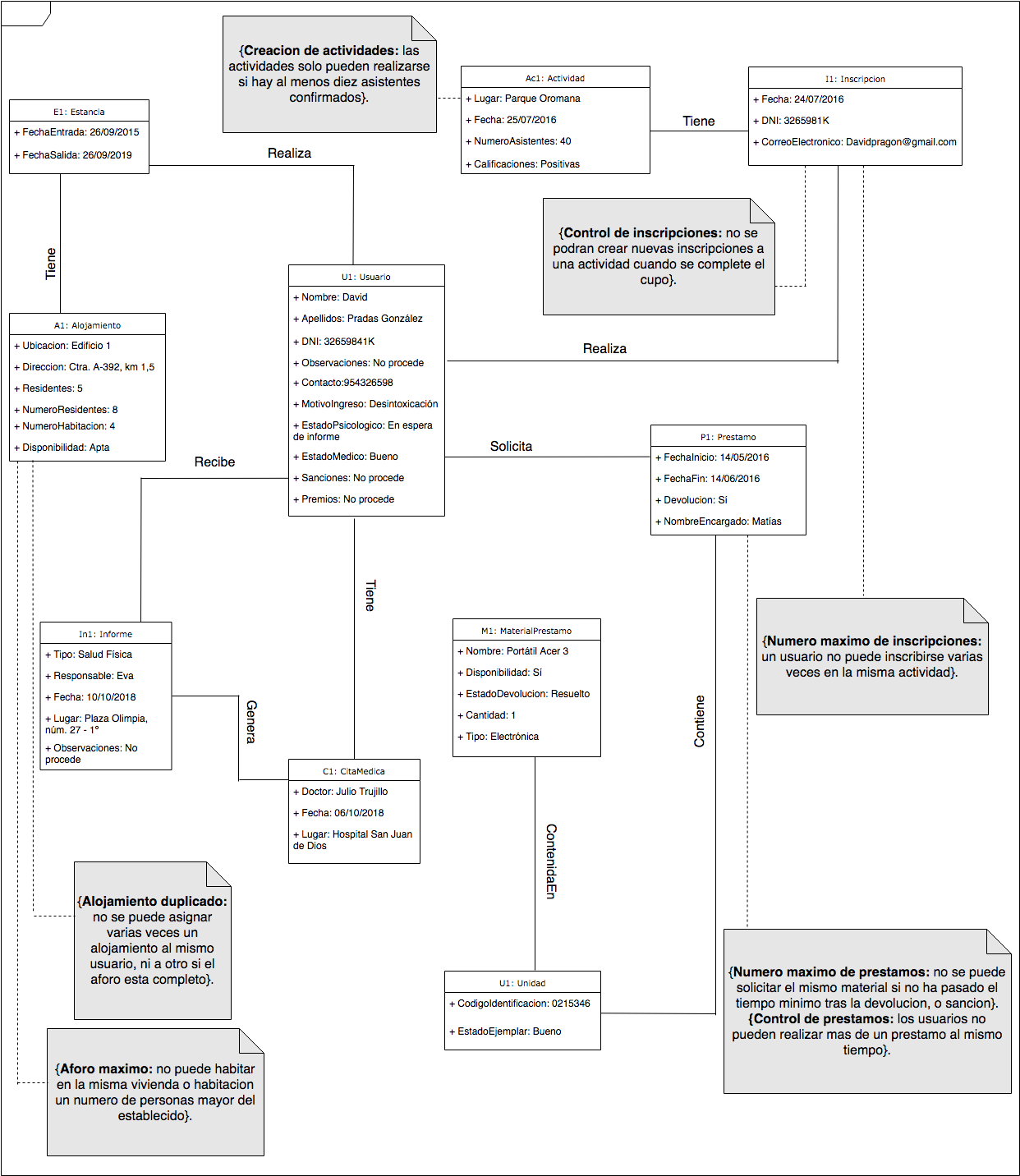
# Matriz de trazabilidad

R

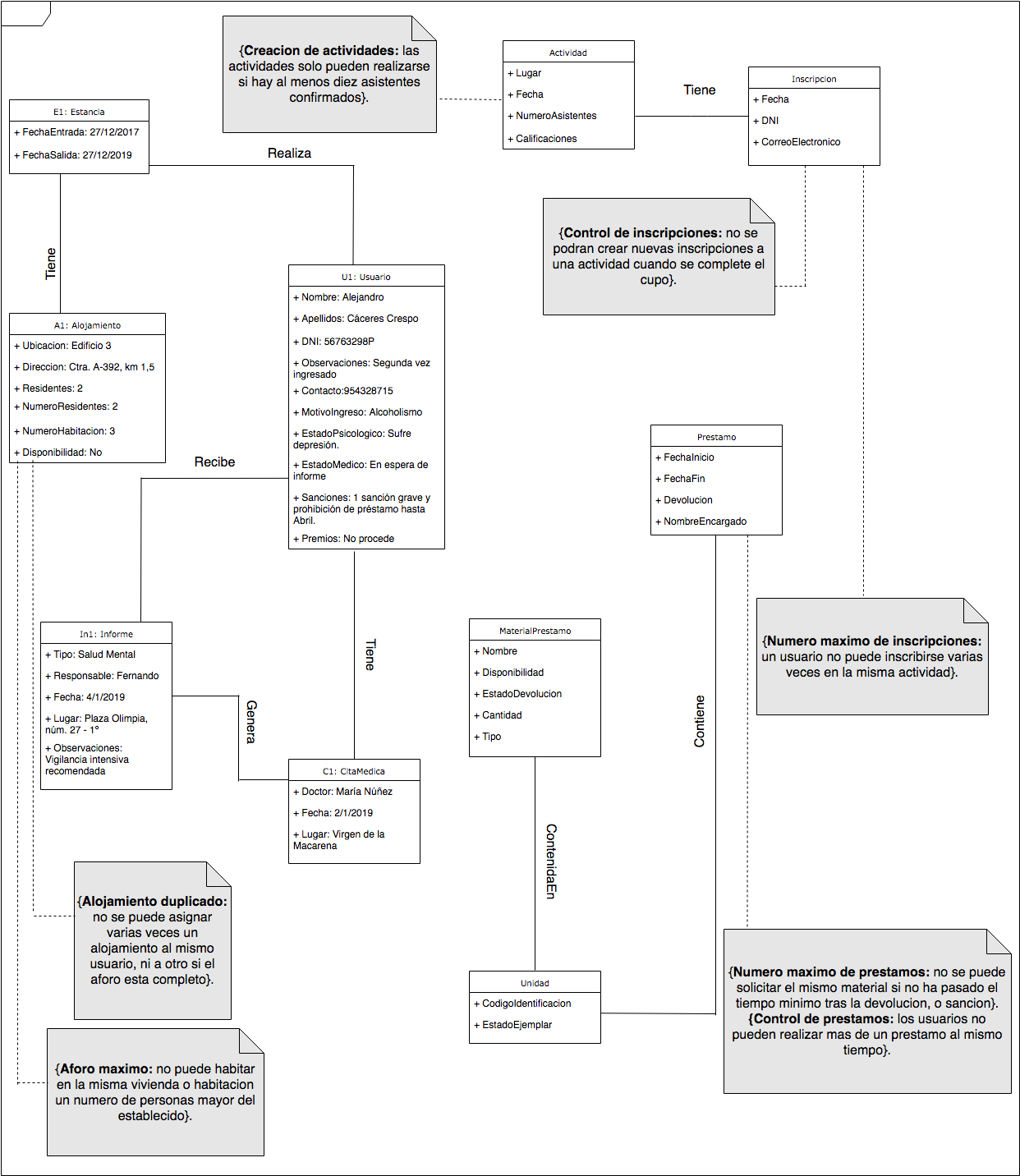
# Casos de Prueba

* **Caso de Prueba 1**: Usuario corriente con alojamiento en la asociación, informe

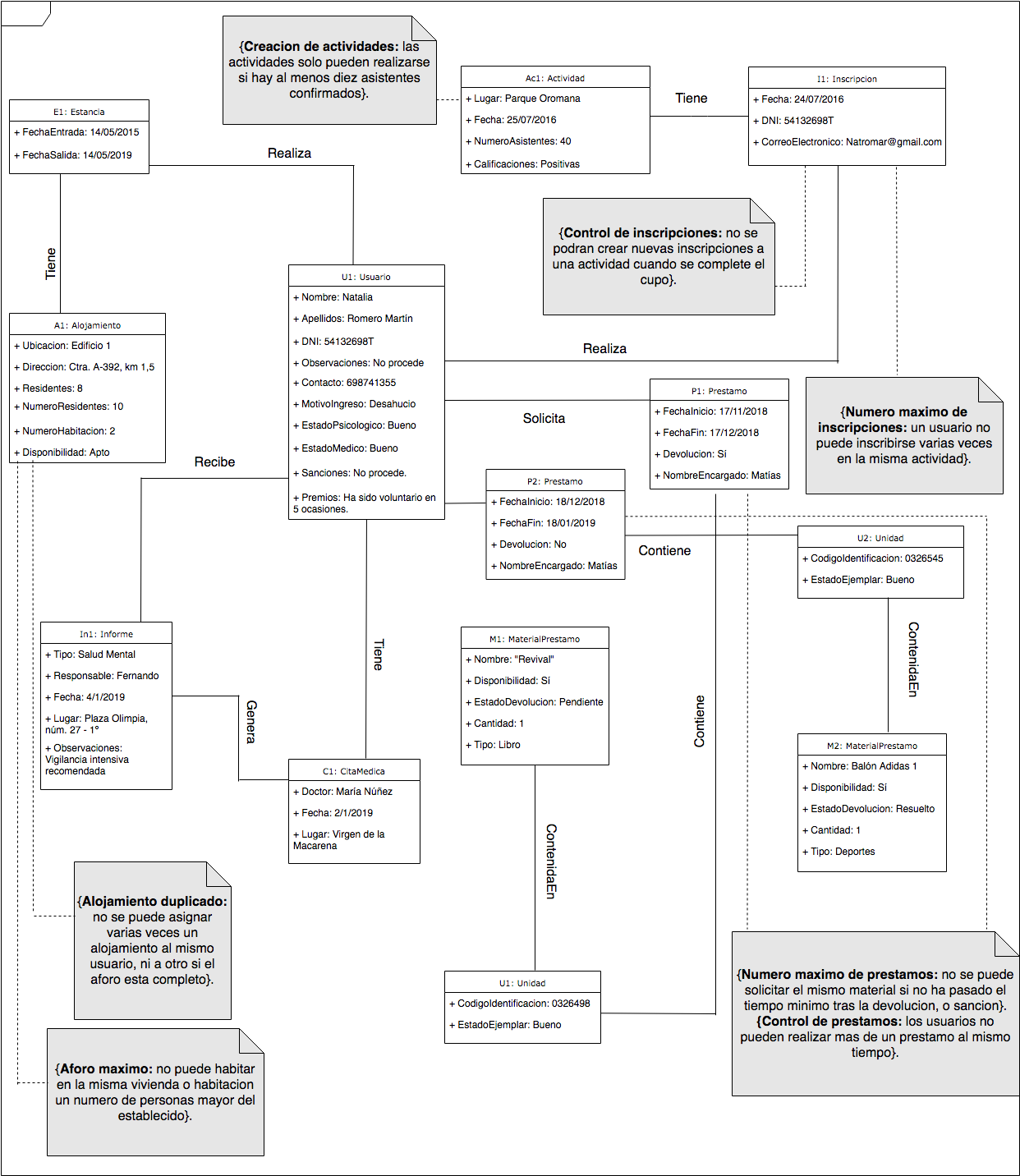
médico físico y no psicológico, un préstamo y participa en actividades.



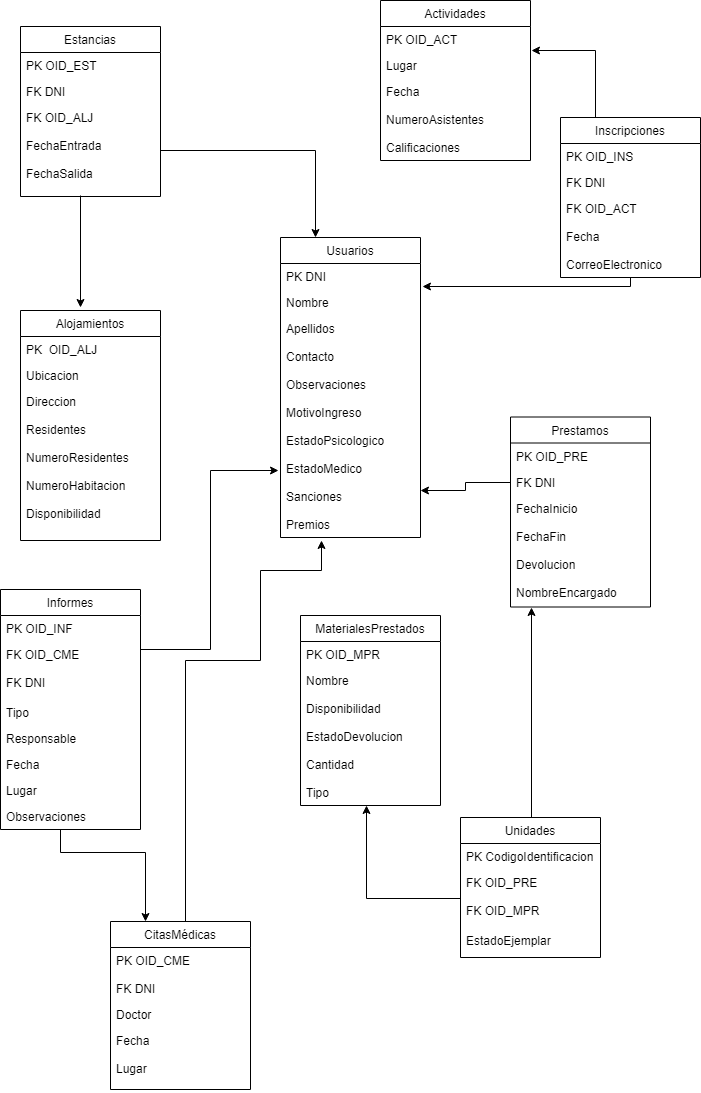
* **Caso de Prueba 2:** Usuario alojado en residencias de AFAR, informe médico psicológico, sin préstamos y no participa en actividades.



* **Caso de Prueba 3:** Usuario con residencia en AFAR, informe médico psicológico e informe médico físico, dos préstamos atribuidos y participó en una actividad.



# Modelo relacional 3FN



# Referencias externas

* [Página web de AFAR](http://www.afar.es/)

Entidades que colaboran:

* [Junta de Andalucía](https://www.juntadeandalucia.es/index.html)
* [Fundación Cajasol](https://fundacioncajasol.com/)
* [Obra Social la Caixa](https://obrasociallacaixa.org/es/)
* [Unión Europea](https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=LEGISSUM:l60015)
* [Ayuntamiento de Sevilla](https://www.sevilla.org/)
* [Ministerio de Bienestar, Consumo y Bienestar Social](http://www.mscbs.gob.es/)
* [Grupo Cementos Portlan Valderrivas](http://www.valderrivas.es/en/)
* [Ayuntamiento de Alcalá](http://www.ciudadalcala.org/)
* [Cáritas](https://www.caritas-sevilla.org/)
* [Hermandad del Águila](http://www.virgendelaguila.com/?ver=Hermandad&pag=Saludo)
* [Protelsur](http://www.protelsur.com/)
* [Dacolim S.L.](http://www.dacolim.com/)
* [Machenieto](https://tiendamanchenieto.com/)
* [Martín Casillas](http://www.martincasillas.com/)
* [Ingtar](http://www.ingtar.es/)

# Anexo A. Acta de la reunión con los responsables de AFAR

Se concertó una reunión con los directivos de AFAR el miércoles 10 de octubre de 2018 en la sede de la organización para proceder con una entrevista en la que poder aclarar los objetivos y necesidades del software que requiere la empresa. Para esta, nos preparamos una serie de preguntas, tanto para informarnos de cómo debe ser el programa o programas que precisan como para conocer mejor la organización, su forma de trabajo, el tipo de personas con las que trabajan, etc.

Al llegar al lugar (Véase figura 1), fuimos recibidos por Rosario Rioja, miembro de la Junta Directiva, quien nos dio una pequeña introducción durante el tiempo en el que llegaban el resto de los directivos. Ante nuestra petición de grabar la conversación para el proyecto, ella nos pidió que si no era obligatorio no lo hiciéramos, ya que le producía cierta incomodidad y distracción, motivo por el cual no tenemos ninguna prueba audiovisual de la entrevista. Viendo el retraso de los compañeros de Rosario, debido a otra reunión que mantenían en ese momento, comenzamos la entrevista sin ellos.



Figura 2: Patio de actividades de AFAR.

Para empezar, preguntamos acerca de la empresa y sus objetivos para conseguir algo de información que podríamos usar más adelante sobre el programa que tendríamos que hacer. Cuando Rosario terminó de explicarnos este apartado, sus compañeros se unieron a la entrevista. Ellos son Miguel Ángel Caro, presidente de AFAR, y María Palacios, otro miembro de la Junta Directiva. Entonces, continuamos la entrevista centrándonos en el sistema de información que desearían y nos comentaron que, a pesar de ya tener uno, este está desfasado y sólo cumple funciones básicas como registrar los nombres de los usuarios junto con un número de expediente y pocos datos más. Por ello, les ofrecimos un sistema que pueda cumplir las mismas funciones que el anterior y además añadirle nuevas opciones y funcionalidades, como la posibilidad de ordenar la información siguiendo unos criterios y de recopilar datos estadísticos. Llegamos a un acuerdo y con ello finalizamos la entrevista.

Antes de irnos, nuestra guía nos enseñó las instalaciones (véase figuras 2 y 3), donde pudimos asistir a una de las actividades que hacen con los participantes de la asociación y, tras eso, una de las trabajadoras nos mostró el programa que usan actualmente para poder hacernos una mejor idea de qué es lo que buscan. Finalmente recogimos nuestras pertenencias y nos marchamos.



Figura 3: Patio secundario de AFAR.

# Anexo B: Acta de la segunda reunión con AFAR

Acordamos una pequeña reunión con Rosario para que nos aclarase algunos detalles relacionados principalmente con las reglas de negocio. Nos especifica que cuando llega un nuevo usuario se registran sus datos personales y además se toman otros datos relacionados con el alojamiento, los informes médicos (que sólo tratan la parte del estado de salud físico del usuario), los diferentes reconocimientos (tratan una parte más psicológica), las actividades a las que asisten y los préstamos que solicitan, que van abarcan una cantidad muy amplia de posibilidades (libros, balones, películas, etc.).

Si un usuario comete una falta muy grave, queda invalidado su alojamiento indefinidamente. Otra de las reglas es que hay un máximo de préstamos (no más de uno por persona), y si se pasa la fecha límite de devolución de este, el usuario en cuestión recibe una sanción equivalente al número de días que pasa de la fecha. Aclarados estos detalles, nos despedimos de Rosario y nos vamos de la asociación.

# Anexo C: Código SQL

DROP TABLE INSCRIPCIONES;

DROP TABLE ACTIVIDADES;

DROP TABLE INFORMES;

DROP TABLE CITAS\_MEDICAS;

DROP TABLE UNIDADES;

DROP TABLE PRESTAMOS;

DROP TABLE MATERIALES\_PRESTADOS;

DROP TABLE ESTANCIAS;

DROP TABLE ALOJAMIENTOS;

DROP TABLE USUARIOS;

CREATE TABLE USUARIOS (

DNI VARCHAR2(50) NOT NULL CHECK((DNI='T')OR(DNI='F')), /\*DNI=T SIGNIFICA USUARIO AUTORIZADO.\*/

NOMBRE VARCHAR2(50) NOT NULL,

APELLIDOS VARCHAR2(50) NOT NULL,

OBSERVACIONES VARCHAR2(200),

CONTACTO VARCHAR2(50) NOT NULL UNIQUE,

MOTIVO\_INGRESO VARCHAR2(200) NOT NULL,

ESTADO\_PSICOLOGICO VARCHAR2(200) NOT NULL,

ESTADO\_MEDICO VARCHAR2(200) NOT NULL,

SANCIONES NUMBER(10),

PREMIOS NUMBER(10),

PRIMARY KEY(DNI)

);

/

CREATE TABLE ACTIVIDADES (

OID\_ACT NUMBER(10) NOT NULL,

LUGAR VARCHAR2(50) NOT NULL,

FECHA DATE NOT NULL,

NUMERO\_ASISTENTES NUMBER(10),

CALIFICACIONES VARCHAR2(200),

PRIMARY KEY(OID\_ACT)

);

/

CREATE TABLE INSCRIPCIONES (

OID\_INS NUMBER(10) NOT NULL,

DNI VARCHAR2(50) NOT NULL,

OID\_ACT NUMBER(10) NOT NULL,

FECHA DATE NOT NULL,

CORREO\_ELECTRONICO VARCHAR2(50) NOT NULL,

PRIMARY KEY(OID\_INS),

FOREIGN KEY(DNI) REFERENCES USUARIOS,

FOREIGN KEY(OID\_ACT) REFERENCES ACTIVIDADES

);

/

CREATE TABLE CITAS\_MEDICAS (

OID\_CM NUMBER(10) NOT NULL,

DNI VARCHAR2(50 BYTE) NOT NULL,

DOCTOR VARCHAR2(50 BYTE) NOT NULL,

FECHA DATE NOT NULL,

LUGAR VARCHAR2(50 BYTE) NOT NULL,

PRIMARY KEY(OID\_CM),

FOREIGN KEY(DNI) REFERENCES USUARIOS

);

/

CREATE TABLE INFORMES (

OID\_INF NUMBER(10) NOT NULL,

OID\_CM NUMBER(10) NOT NULL,

DNI VARCHAR2(50 BYTE) NOT NULL,

TIPO VARCHAR2(50 BYTE) NOT NULL,

RESPONSABLE VARCHAR2(50 BYTE) NOT NULL,

FECHA DATE NOT NULL,

LUGAR VARCHAR2(50 BYTE) NOT NULL,

OBSERVACIONES LONG,

PRIMARY KEY(OID\_INF),

FOREIGN KEY(OID\_CM) REFERENCES CITAS\_MEDICAS,

FOREIGN KEY(DNI) REFERENCES USUARIOS

);

/

CREATE TABLE PRESTAMOS(

OID\_PRE NUMBER(10) NOT NULL,

DNI VARCHAR2(50 BYTE) NOT NULL,

FECHA\_INICIO DATE NOT NULL,

FECHA\_FIN DATE NOT NULL,

DEVOLUCION VARCHAR2(1 BYTE) CHECK((DEVOLUCION='T')OR(DEVOLUCION='F')),

NOMBRE\_ENCARGADO VARCHAR2(50 BYTE),

CONSTRAINT FECHAS CHECK (FECHA\_INICIO < FECHA\_FIN),

PRIMARY KEY(OID\_PRE),

FOREIGN KEY(DNI) REFERENCES USUARIOS

);

/

CREATE TABLE MATERIALES\_PRESTADOS(

OID\_MPR NUMBER(10) NOT NULL,

NOMBRE VARCHAR2(50 BYTE),

DISPONIBILIDAD VARCHAR2(1 BYTE) CHECK((DISPONIBILIDAD= 'T')OR(DISPONIBILIDAD= 'F')),

ESTADO\_DEVOLUCION VARCHAR2(50 BYTE) CHECK((ESTADO\_DEVOLUCION IN('BIEN', 'MAL'))),

CANTIDAD VARCHAR2(50 BYTE),

TIPO VARCHAR2(50 BYTE),

PRIMARY KEY(OID\_MPR)

);

/

CREATE TABLE UNIDADES(

OID\_PRE NUMBER(10) NOT NULL,

OID\_MPR NUMBER(10) NOT NULL,

CODIGO\_IDENTIFICACION VARCHAR2(50) NOT NULL,

ESTADO\_EJEMPLAR VARCHAR2(50) CHECK((ESTADO\_EJEMPLAR='BUENO')OR(ESTADO\_EJEMPLAR='MALO')),

PRIMARY KEY(CODIGO\_IDENTIFICACION),

FOREIGN KEY(OID\_PRE) REFERENCES PRESTAMOS,

FOREIGN KEY(OID\_MPR) REFERENCES MATERIALES\_PRESTADOS

);

/

CREATE TABLE ALOJAMIENTOS(

OID\_ALJ NUMBER(10) NOT NULL,

UBICACION VARCHAR2(50 BYTE),

DIRECCION VARCHAR2(50 BYTE),

RESIDENTES VARCHAR2(200 BYTE),

NUMERO\_RESIDENTES NUMBER(10),

NUMERO\_HABITACION NUMBER(10),

DISPONIBILIDAD VARCHAR2(1 BYTE) CHECK((DISPONIBILIDAD='T')OR(DISPONIBILIDAD='F')),

PRIMARY KEY(OID\_ALJ)

);

/

CREATE TABLE ESTANCIAS(

OID\_EST NUMBER(10) NOT NULL,

DNI VARCHAR2(50 BYTE) NOT NULL,

OID\_ALJ NUMBER(10) NOT NULL,

FECHA\_ENTRADA DATE NOT NULL,

FECHA\_SALIDA DATE,

CONSTRAINT FECHAS\_ESTANCIA CHECK (FECHA\_ENTRADA < FECHA\_SALIDA),

PRIMARY KEY(OID\_EST),

FOREIGN KEY(DNI) REFERENCES USUARIOS,

FOREIGN KEY(OID\_ALJ) REFERENCES ALOJAMIENTOS

);

/

-----------------------------------------------/\* SECUENCIAS \*/------------------------------------------

DROP SEQUENCE SEC\_OID\_ACT;

CREATE SEQUENCE SEC\_OID\_ACT START WITH 1 INCREMENT BY 1;

/\* DUAL: tabla que tiene oracle para probar funciones on hacer calculos rápidos\*/

CREATE OR REPLACE TRIGGER NUEVO\_OID\_ACT

BEFORE INSERT ON ACTIVIDADES

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_ACT.NEXTVAL INTO :NEW.OID\_ACT FROM DUAL;

END NUEVO\_OID\_ACT;

/

DROP SEQUENCE SEC\_OID\_INS;

CREATE SEQUENCE SEC\_OID\_INS START WITH 1 INCREMENT BY 1;

/\* DUAL: tabla que tiene oracle para probar funciones on hacer calculos rápidos\*/

CREATE OR REPLACE TRIGGER NUEVO\_OID\_INS

BEFORE INSERT ON INSCRIPCIONES

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_INS.NEXTVAL INTO :NEW.OID\_INS FROM DUAL;

END NUEVO\_OID\_INS;

/

DROP SEQUENCE SEC\_OID\_CM;

CREATE SEQUENCE SEC\_OID\_CM START WITH 1 INCREMENT BY 1;

/\* DUAL: tabla que tiene oracle para probar funciones on hacer calculos rápidos\*/

CREATE OR REPLACE TRIGGER NUEVO\_OID\_CM

BEFORE INSERT ON CITAS\_MEDICAS

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_CM.NEXTVAL INTO :NEW.OID\_CM FROM DUAL;

END NUEVO\_OID\_CM;

/

DROP SEQUENCE SEC\_OID\_INF;

CREATE SEQUENCE SEC\_OID\_INF START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER NUEVO\_OID\_INF

BEFORE INSERT ON INFORMES

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_INF.NEXTVAL INTO :NEW.OID\_INF FROM DUAL;

END NUEVO\_OID\_INF;

/

DROP SEQUENCE SEC\_OID\_PRE;

CREATE SEQUENCE SEC\_OID\_PRE START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER NUEVO\_OID\_PRE

BEFORE INSERT ON PRESTAMOS

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_PRE.NEXTVAL INTO :NEW.OID\_PRE FROM DUAL;

END NUEVO\_OID\_PRE;

/

DROP SEQUENCE SEC\_OID\_MPR;

CREATE SEQUENCE SEC\_OID\_MPR START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER NUEVO\_OID\_MPR

BEFORE INSERT ON MATERIALES\_PRESTADOS

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_MPR.NEXTVAL INTO :NEW.OID\_MPR FROM DUAL;

END NUEVO\_OID\_MPR;

/

DROP SEQUENCE SEC\_OID\_ALJ;

CREATE SEQUENCE SEC\_OID\_ALJ START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER NUEVO\_OID\_ALJ

BEFORE INSERT ON ALOJAMIENTOS

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_ALJ.NEXTVAL INTO :NEW.OID\_ALJ FROM DUAL;

END NUEVO\_OID\_ALJ;

/

DROP SEQUENCE SEC\_OID\_EST;

CREATE SEQUENCE SEC\_OID\_EST START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER NUEVO\_OID\_EST

BEFORE INSERT ON ESTANCIAS

FOR EACH ROW

BEGIN

SELECT SEC\_OID\_EST.NEXTVAL INTO :NEW.OID\_EST FROM DUAL;

END NUEVO\_OID\_EST;

/

--------------------------------------------------------/\* FUNCIONES \*/-----------------------------------------------------------

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;

/

CREATE OR REPLACE FUNCTION obtener\_premios (w\_dni IN usuarios.dni%TYPE) RETURN NUMBER IS w\_premios usuarios.premios%TYPE;

BEGIN

SELECT premios INTO w\_premios FROM usuarios WHERE dni = w\_dni;

RETURN w\_premios;

END obtener\_premios;

/

CREATE OR REPLACE FUNCTION obtener\_sanciones (w\_dni IN usuarios.dni%TYPE) RETURN NUMBER IS w\_sanciones usuarios.sanciones%TYPE;

BEGIN

SELECT sanciones INTO w\_sanciones FROM usuarios WHERE dni = w\_dni;

RETURN w\_sanciones;

END obtener\_sanciones;

/

------------------------------------------------/\* PROCEDIMIENTOS \*/---------------------------------------------------------

CREATE OR REPLACE PROCEDURE NUEVO\_USUARIO(

w\_DNI IN USUARIOS.DNI%Type, w\_NOMBRE IN USUARIOS.NOMBRE%Type, w\_APELLIDOS IN USUARIOS.APELLIDOS%Type,

w\_OBSERVACIONES IN USUARIOS.OBSERVACIONES%type, w\_CONTACTO IN USUARIOS.CONTACTO%type, w\_MOTIVO\_INGRESO IN

USUARIOS.MOTIVO\_INGRESO%type, w\_ESTADO\_PSICOLOGICO IN USUARIOS.ESTADO\_PSICOLOGICO%type,

w\_ESTADO\_MEDICO IN USUARIOS.ESTADO\_MEDICO%type, w\_SANCIONES IN USUARIOS.SANCIONES%type, w\_PREMIOS IN USUARIOS.PREMIOS%type) IS

BEGIN

INSERT INTO USUARIOS(DNI,NOMBRE, APELLIDOS, OBSERVACIONES, CONTACTO, MOTIVO\_INGRESO, ESTADO\_PSICOLOGICO,

ESTADO\_MEDICO, SANCIONES, PREMIOS)

VALUES (w\_DNI,w\_NOMBRE,w\_APELLIDOS,w\_OBSERVACIONES,w\_CONTACTO,w\_MOTIVO\_INGRESO,w\_ESTADO\_PSICOLOGICO,

w\_ESTADO\_MEDICO,w\_SANCIONES, w\_PREMIOS);

COMMIT WORK;

END NUEVO\_USUARIO;

/

CREATE OR REPLACE PROCEDURE NUEVA\_UNIDAD(

w\_OID\_PRE IN UNIDADES.OID\_PRE%Type, w\_OID\_MPR IN UNIDADES.OID\_MPR%Type,

w\_CODIGO\_IDENTIFICACION IN UNIDADES.CODIGO\_IDENTIFICACION%Type,

w\_ESTADO\_EJEMPLAR IN UNIDADES.ESTADO\_EJEMPLAR%type) IS

BEGIN

INSERT INTO UNIDADES(OID\_PRE,OID\_MPR, CODIGO\_IDENTIFICACION, ESTADO\_EJEMPLAR)

VALUES (w\_OID\_PRE,w\_OID\_MPR, w\_CODIGO\_IDENTIFICACION, w\_ESTADO\_EJEMPLAR);

COMMIT WORK;

END NUEVA\_UNIDAD;

/

CREATE OR REPLACE PROCEDURE NUEVO\_PRESTAMO(

w\_DNI IN PRESTAMOS.DNI%Type, w\_FECHA\_INICIO IN PRESTAMOS.FECHA\_INICIO%Type,

w\_FECHA\_FIN IN PRESTAMOS.FECHA\_FIN%Type,

w\_DEVOLUCION IN PRESTAMOS.DEVOLUCION%type, w\_NOMBRE\_ENCARGADO IN PRESTAMOS.NOMBRE\_ENCARGADO%type) IS

BEGIN

INSERT INTO PRESTAMOS(OID\_PRE, DNI,FECHA\_INICIO,FECHA\_FIN,DEVOLUCION,NOMBRE\_ENCARGADO)

VALUES (SEC\_OID\_PRE.nextval,w\_DNI,w\_FECHA\_INICIO,w\_FECHA\_FIN,w\_DEVOLUCION,w\_NOMBRE\_ENCARGADO);

COMMIT WORK;

END NUEVO\_PRESTAMO;

/

CREATE OR REPLACE PROCEDURE NUEVO\_MATERIAL\_PRESTADO(

w\_NOMBRE IN MATERIALES\_PRESTADOS.NOMBRE%Type, w\_DISPONIBILIDAD IN MATERIALES\_PRESTADOS.DISPONIBILIDAD%Type,

w\_ESTADO\_DEVOLUCION IN MATERIALES\_PRESTADOS.ESTADO\_DEVOLUCION%Type,

w\_CANTIDAD IN MATERIALES\_PRESTADOS.CANTIDAD%type, w\_TIPO IN MATERIALES\_PRESTADOS.TIPO%type) IS

BEGIN

INSERT INTO MATERIALES\_PRESTADOS(OID\_MPR,NOMBRE,DISPONIBILIDAD,ESTADO\_DEVOLUCION,CANTIDAD,TIPO)

VALUES (SEC\_OID\_MPR.nextval,w\_NOMBRE,w\_DISPONIBILIDAD,w\_ESTADO\_DEVOLUCION,w\_CANTIDAD,w\_TIPO);

COMMIT WORK;

END NUEVO\_MATERIAL\_PRESTADO;

/

CREATE OR REPLACE PROCEDURE NUEVA\_INSCRIPCION(

w\_DNI IN INSCRIPCIONES.DNI%Type, w\_OID\_ACT IN INSCRIPCIONES.OID\_ACT%Type) IS

BEGIN

INSERT INTO INSCRIPCIONES(OID\_INS, DNI, OID\_ACT)

VALUES (SEC\_OID\_INS.nextval, w\_DNI, w\_OID\_ACT);

COMMIT WORK;

END NUEVA\_INSCRIPCION;

/

CREATE OR REPLACE PROCEDURE NUEVO\_INFORME(

w\_OID\_CM IN INFORMES.OID\_CM%Type, w\_DNI IN INFORMES.DNI%Type, w\_TIPO IN INFORMES.TIPO%Type,

w\_RESPONSABLE IN INFORMES.RESPONSABLE%Type, w\_FECHA IN INFORMES.FECHA%Type, w\_LUGAR IN INFORMES.LUGAR%Type,

w\_OBSERVACIONES IN INFORMES.OBSERVACIONES%Type) IS

BEGIN

INSERT INTO INFORMES(OID\_INF,OID\_CM,DNI,TIPO,RESPONSABLE,FECHA,LUGAR,OBSERVACIONES)

VALUES (SEC\_OID\_INF.nextval,w\_OID\_CM,w\_DNI,w\_TIPO,w\_RESPONSABLE,w\_FECHA,w\_LUGAR,w\_OBSERVACIONES);

COMMIT WORK;

END NUEVO\_INFORME;

/

CREATE OR REPLACE PROCEDURE NUEVA\_ESTANCIA(

w\_DNI IN ESTANCIAS.DNI%Type, w\_OID\_ALJ IN ESTANCIAS.OID\_ALJ%Type, w\_FECHA\_ENTRADA IN ESTANCIAS.FECHA\_ENTRADA%Type,

w\_FECHA\_SALIDA IN ESTANCIAS.FECHA\_SALIDA%Type) IS

BEGIN

INSERT INTO ESTANCIAS(OID\_EST,DNI,OID\_ALJ,FECHA\_ENTRADA,FECHA\_SALIDA)

VALUES (SEC\_OID\_EST.nextval,w\_DNI,w\_OID\_ALJ,w\_FECHA\_ENTRADA,w\_FECHA\_SALIDA);

COMMIT WORK;

END NUEVA\_ESTANCIA;

/

CREATE OR REPLACE PROCEDURE NUEVA\_CITA\_MEDICA(

w\_DNI IN CITAS\_MEDICAS.DNI%Type, w\_DOCTOR IN CITAS\_MEDICAS.DOCTOR%Type, w\_FECHA IN CITAS\_MEDICAS.FECHA%Type,

w\_LUGAR IN CITAS\_MEDICAS.LUGAR%Type) IS

BEGIN

INSERT INTO CITAS\_MEDICAS(OID\_CM,DNI,DOCTOR,FECHA,LUGAR)

VALUES (SEC\_OID\_CM.nextval,w\_DNI,w\_DOCTOR,w\_FECHA,w\_LUGAR);

COMMIT WORK;

END NUEVA\_CITA\_MEDICA;

/

CREATE OR REPLACE PROCEDURE NUEVO\_ALOJAMIENTO(

w\_UBICACION IN ALOJAMIENTOS.UBICACION%Type, w\_DIRECCION IN ALOJAMIENTOS.DIRECCION%Type,

w\_RESIDENTES IN ALOJAMIENTOS.RESIDENTES%Type,

w\_NUMERO\_RESIDENTES IN ALOJAMIENTOS.NUMERO\_RESIDENTES%Type, w\_NUMERO\_HABITACION IN ALOJAMIENTOS.NUMERO\_HABITACION%Type,

w\_DISPONIBILIDAD IN ALOJAMIENTOS.DISPONIBILIDAD%Type) IS

BEGIN

INSERT INTO ALOJAMIENTOS(OID\_ALJ,UBICACION,DIRECCION,RESIDENTES,NUMERO\_RESIDENTES,NUMERO\_HABITACION,DISPONIBILIDAD)

VALUES (SEC\_OID\_ALJ.nextval,w\_UBICACION,w\_DIRECCION,w\_RESIDENTES,w\_NUMERO\_RESIDENTES,w\_NUMERO\_HABITACION,w\_DISPONIBILIDAD);

COMMIT WORK;

END NUEVO\_ALOJAMIENTO;

/

CREATE OR REPLACE PROCEDURE NUEVO\_ACTIVIDAD(

w\_LUGAR IN ACTIVIDADES.LUGAR%Type, w\_FECHA IN ACTIVIDADES.FECHA%Type,

w\_NUMERO\_ASISTENTES IN ACTIVIDADES.NUMERO\_ASISTENTES%Type,

w\_CALIFICACIONES IN ACTIVIDADES.CALIFICACIONES%Type) IS

BEGIN

INSERT INTO ACTIVIDADES(OID\_ACT,LUGAR,FECHA,NUMERO\_ASISTENTES,CALIFICACIONES)

VALUES (SEC\_OID\_ACT.nextval,w\_LUGAR,w\_FECHA,w\_NUMERO\_ASISTENTES,w\_CALIFICACIONES);

COMMIT WORK;

END NUEVO\_ACTIVIDAD;

/

------------------------------------------------------------/\* CURSORES \*/---------------------------------------------------------------

DECLARE

CURSOR cursor\_usuario IS

SELECT dni, nombre, apellidos, contacto, observaciones, motivo\_ingreso, estado\_psicologico, estado\_medico, sanciones, premios

FROM usuarios ORDER BY dni;

w\_dni usuarios.dni%TYPE;

w\_nombre usuarios.nombre%TYPE;

w\_apellidos usuarios.apellidos%TYPE;

w\_contacto usuarios.contacto%TYPE;

w\_observaciones usuarios.observaciones%TYPE;

w\_motivo\_ingreso usuarios.motivo\_ingreso%TYPE;

w\_estado\_psicologico usuarios.estado\_psicologico%TYPE;

w\_estado\_medico usuarios.estado\_medico%TYPE;

w\_sanciones usuarios.sanciones%TYPE;

w\_premios usuarios.premios%TYPE;

BEGIN

IF NOT (cursor\_usuario%ISOPEN) THEN OPEN cursor\_usuario;

END IF;

LOOP

FETCH cursor\_usuario INTO w\_dni, w\_nombre, w\_apellidos, w\_contacto, w\_observaciones, w\_motivo\_ingreso, w\_estado\_psicologico, w\_estado\_medico, w\_sanciones, w\_premios;

EXIT WHEN cursor\_usuario%NOTFOUND;

UPDATE usuarios SET PREMIOS = w\_premios + 1 WHERE DNI = w\_dni;

END LOOP;

CLOSE cursor\_usuario;

END cursor\_usuario;

/

DECLARE

CURSOR cursor\_actividad IS

SELECT oid\_act, lugar, fecha, numero\_asistentes, calificaciones

FROM actividades ORDER BY lugar;

w\_oid\_act actividades.oid\_act%TYPE;

w\_lugar actividades.lugar%TYPE;

w\_fecha actividades.fecha%TYPE;

w\_numero\_asistentes actividades.numero\_asistentes%TYPE;

w\_calificaciones actividades.calificaciones%TYPE;

BEGIN

IF NOT(cursor\_actividad%ISOPEN) THEN OPEN cursor\_actividad;

END IF;

LOOP

FETCH cursor\_actividad INTO w\_oid\_act, w\_lugar, w\_fecha, w\_numero\_asistentes, w\_calificaciones;

EXIT WHEN cursor\_actividad%NOTFOUND;

UPDATE actividades SET FECHA= w\_fecha +1 WHERE LUGAR= w\_lugar;

END LOOP;

CLOSE cursor\_actividad;

END cursor\_actividad;

/

DECLARE

CURSOR cursor\_usuario2 IS

SELECT dni, nombre, apellidos, contacto, observaciones, motivo\_ingreso, estado\_psicologico, estado\_medico, sanciones, premios

FROM usuarios ORDER BY dni;

w\_dni usuarios.dni%TYPE;

w\_nombre usuarios.nombre%TYPE;

w\_apellidos usuarios.apellidos%TYPE;

w\_contacto usuarios.contacto%TYPE;

w\_observaciones usuarios.observaciones%TYPE;

w\_motivo\_ingreso usuarios.motivo\_ingreso%TYPE;

w\_estado\_psicologico usuarios.estado\_psicologico%TYPE;

w\_estado\_medico usuarios.estado\_medico%TYPE;

w\_sanciones usuarios.sanciones%TYPE;

w\_premios usuarios.premios%TYPE;

BEGIN

IF NOT (cursor\_usuario2%ISOPEN) THEN OPEN cursor\_usuario2;

END IF;

LOOP

FETCH cursor\_usuario2 INTO w\_dni, w\_nombre, w\_apellidos, w\_contacto, w\_observaciones, w\_motivo\_ingreso, w\_estado\_psicologico, w\_estado\_medico, w\_sanciones, w\_premios;

EXIT WHEN cursor\_usuario2%NOTFOUND;

UPDATE usuarios SET SANCIONES = w\_sanciones + 1 WHERE DNI = w\_dni;

END LOOP;

CLOSE cursor\_usuario2;

END cursor\_usuario2;

/

DECLARE

CURSOR cursor\_informes IS

SELECT dni, tipo, responsable, fecha, lugar, observaciones

FROM informes ORDER BY dni;

w\_dni informes.dni%TYPE;

w\_tipo informes.tipo%TYPE;

w\_responsable informes.responsable%TYPE;

w\_fecha informes.fecha%TYPE;

w\_lugar informes.lugar%TYPE;

w\_observaciones usuarios.observaciones%TYPE;

BEGIN

IF NOT (cursor\_informes%ISOPEN) THEN OPEN cursor\_informes;

END IF;

LOOP

FETCH cursor\_informes INTO w\_dni, w\_tipo, w\_responsable, w\_fecha,w\_lugar, w\_observaciones;

EXIT WHEN cursor\_informes%NOTFOUND;

UPDATE informes SET TIPO = w\_tipo + 1 WHERE DNI = w\_dni;

END LOOP;

CLOSE cursor\_informes;

END cursor\_informes;

/

-------------------------------------------------/\*Triggers\*/---------------------------------------------------------

/\*? RN-1: No es posible tener una sexta sanción.

Como monitora,

quiero que el sistema no sea capaz de almacenar más de cinco sanciones, ya que no sería necesario, porque a la quinta se le expulsa del centro.

\*/

CREATE OR REPLACE TRIGGER RN1\_SANCION

BEFORE UPDATE OF SANCIONES ON USUARIOS

FOR EACH ROW

BEGIN

IF :NEW.SANCIONES =6

THEN RAISE\_APPLICATION\_ERROR

(-20600, ' NO SE PUEDEN TENER MAS DE 5 SANCIONES, LA CANTIDAD A ACTUALIZAR ERA: ' || :NEW.SANCIONES);

END IF;

END RN1\_SANCION;

/

/\*? RN-3: Imposibilidad de eliminación de datos.

Como monitora,

quiero que no se pueda eliminar información del sistema de ninguna persona que haya pasado por el centro,

para tener en cuenta el regreso de antiguos miembros.

\*/

CREATE OR REPLACE TRIGGER RN3\_ELIMINACION\_DATOS

BEFORE DELETE ON USUARIOS

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR (-20601,'NO SE PUEDEN ELIMINAR DATOS DE LOS USUARIOS');

END RN3\_ELIMINACION\_DATOS;

/

/\*? RN-4: Creación de actividades.

Como monitora,

quiero que si hay menos de 10 asistentes inscritos no se pueda realizar la actividad.

\*/

CREATE OR REPLACE TRIGGER RN4\_CREACION\_ACTIVIDADES

BEFORE UPDATE OF NUMERO\_ASISTENTES ON ACTIVIDADES

FOR EACH ROW

DECLARE

BEGIN

IF :NEW.NUMERO\_ASISTENTES < 10

THEN RAISE\_APPLICATION\_ERROR(-20603, 'SOLO PUEDEN REALIZARSE ACTIVIDADES CON AL MENOS 10 ASISTENTES. EL NÚMERO DE ASISTENTES A ACTUALIZAR ERA: ' || :NEW.NUMERO\_ASISTENTES);

END IF;

END RN4\_CREACION\_ACTIVIDADES;

/

/\*? RN-5: Restricciones para apuntarse a una actividad.

Como monitora, quiero que el sistema no permita apuntarse a una actividad a

alguien después con más de tres sanciones por mala conducta.

\*/

CREATE OR REPLACE TRIGGER RN5\_INSCRIPCION\_ACTIVIDAD

BEFORE INSERT ON INSCRIPCIONES

FOR EACH ROW

DECLARE NUMERO\_SANCIONES NUMBER(10);

BEGIN

SELECT SANCIONES INTO NUMERO\_SANCIONES FROM USUARIOS WHERE DNI = :NEW.DNI;

IF (NUMERO\_SANCIONES > 3)

THEN RAISE\_APPLICATION\_ERROR (-20602, 'NO SE PUEDEN INSCRIBIR USUARIOS CON MÁS DE 3 SANCIONES');

END IF;

END RN5\_INSCRIPCION\_ACTIVIDAD;

/

/\*? RN-8: Disponibilidad de los prestamos.

Como encargado de prestamos,

quiero que no puedan realizarse prestamos de materiales que no esten disponibles.

\*/

CREATE OR REPLACE TRIGGER RN8\_DISPONIBILIDAD\_PRESTAMOS

BEFORE INSERT ON UNIDADES

FOR EACH ROW

DECLARE DISPONIBILIDAD\_MATERIAL VARCHAR2(50);

BEGIN

SELECT DISPONIBILIDAD INTO DISPONIBILIDAD\_MATERIAL FROM MATERIALES\_PRESTADOS WHERE OID\_MPR = :NEW.OID\_MPR;

IF (DISPONIBILIDAD\_MATERIAL = 'F')

THEN RAISE\_APPLICATION\_ERROR(-20603, 'NO PUEDEN REALIZARSE PRESTAMOS DE MATERIALES NO DISPONIBLES');

END IF;

END RN8\_DISPONIBILIDAD\_PRESTAMOS;

/

/\*? RN-8: Control de préstamos.

Como encargado de préstamos,

quiero que nadie pueda tener dos préstamos al mismo tiempo,

para controlar mejor el inventario.

\*/

/\*

CREATE OR REPLACE TRIGGER RN7\_CONTROL\_PRESTAMOS

BEFORE INSERT ON PRESTAMOS

FOR EACH ROW

BEGIN

FOR LISTA\_PRESTAMOS IN (SELECT DEVOLUCION FROM PRESTAMOS WHERE DNI = :NEW.DNI)

LOOP

IF LISTA\_PRESTAMOS.DEVOLUCION IS NOT NULL AND LISTA\_PRESTAMOS.DEVOLUCION = 'F'

THEN RAISE\_APPLICATION\_ERROR

(-20604, 'NO SE PUEDEN TENER DOS PRESTAMOS SIMULTANEAMENTE');

END IF;

END LOOP;

END RN7\_CONTROL\_PRESTAMOS;

/

\*/

/\*Un usuario no puede cambiar de alojamiento si no ha pasado al menos un mes en el\*/

/\*CREATE OR REPLACE TRIGGER T\_ALOJAMIENTOS

BEFORE UPDATE OF OID\_EST ON ESTANCIAS

FOR EACH ROW

DECLARE

DURACION\_ESTANCIA NUMBER;

BEGIN

SELECT DATEDIFF(DAY, FECHA\_ENTRADA, FECHA\_SALIDA) INTO DURACION\_ESTANCIA FROM ESTANCIAS;

IF ((DURACION\_ESTANCIA != 30)OR(DURACION\_ESTANCIA != 31))

THEN RAISE\_APPLICATION\_ERROR(-20605, :NEW.OID\_EST

|| 'NO SE PUEDE CAMBIAR DE ALOJAMIENTO SI LA DURACION DE LA ULTIMA ESTANCIA NO ES DE AL MENOS UN MES');

END IF;

END T\_ALOJAMIENTOS;

/

\*/

/\*No podemos quedarnos sin materiales para los prestamos\*/

CREATE OR REPLACE TRIGGER T\_MATERIALES

BEFORE UPDATE OF OID\_PRE ON PRESTAMOS

FOR EACH ROW

DECLARE

CANTIDAD\_DISPONIBLE VARCHAR2(50);

BEGIN

SELECT CANTIDAD INTO CANTIDAD\_DISPONIBLE FROM MATERIALES\_PRESTADOS;

IF((CANTIDAD\_DISPONIBLE = 'UNO')OR(CANTIDAD\_DISPONIBLE='1'))

THEN RAISE\_APPLICATION\_ERROR(-20606, :NEW.OID\_PRE

||'NO PODEMOS QUEDARNOS SIN MATERIAL PARA LOS PRESTAMOS');

END IF;

END T\_MATERIALES;

/

/\*No se puede solicitar otro prestamo si no se ha devuelto el anterior\*/

CREATE OR REPLACE TRIGGER T\_DEVOLUCION

BEFORE UPDATE OF OID\_PRE ON PRESTAMOS

FOR EACH ROW

DECLARE

ESTADO\_DEVOLUCION VARCHAR2(50);

BEGIN

SELECT DEVOLUCION INTO ESTADO\_DEVOLUCION FROM PRESTAMOS;

IF(ESTADO\_DEVOLUCION = 'F')

THEN RAISE\_APPLICATION\_ERROR(-20607, :NEW.OID\_PRE

||'DEBE DEVOLVERSE EL MATERIAL SOLICITADO ANTERIORMENTE');

END IF;

END T\_DEVOLUCION;

/

/\*Un usuario no puede inscribirse varias veces en la misma actividad\*/

CREATE OR REPLACE TRIGGER T\_INSCRIPCIONES

BEFORE UPDATE OF OID\_INS ON INSCRIPCIONES

FOR EACH ROW

DECLARE

ACTIVIDAD\_NEW NUMBER;

ACTIVIDAD\_OLD NUMBER;

BEGIN

SELECT :NEW.OID\_ACT INTO ACTIVIDAD\_NEW FROM INSCRIPCIONES;

SELECT :OLD.OID\_ACT INTO ACTIVIDAD\_OLD FROM INSCRIPCIONES;

IF(ACTIVIDAD\_NEW = ACTIVIDAD\_OLD )

THEN RAISE\_APPLICATION\_ERROR(-20608, :NEW.OID\_ACT ||'YA SE HA INSCRITO EN ESTA ACTIVIDAD');

END IF;

END T\_INSCRIPCIONES;

/

-----------------------------------------------------------/\* PAQUETES \*/----------------------------------------------------------------

create or replace package PRUEBAS\_USUARIOS as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_DNI VARCHAR2, w\_nombre VARCHAR2, w\_apellidos VARCHAR2, w\_observaciones VARCHAR2, w\_contacto VARCHAR2,

w\_motivo\_ingreso VARCHAR2, w\_estado\_psicologico VARCHAR2, w\_estado\_medico VARCHAR2, w\_sanciones NUMBER, w\_premios NUMBER,

salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, W\_DNI VARCHAR2, w\_nombre VARCHAR2, w\_apellidos VARCHAR2, w\_observaciones VARCHAR2, w\_contacto VARCHAR2,

w\_motivo\_ingreso VARCHAR2, w\_estado\_psicologico VARCHAR2, w\_estado\_medico VARCHAR2, w\_sanciones NUMBER, w\_premios NUMBER,

salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_DNI VARCHAR2, salidaEsperada BOOLEAN);

end PRUEBAS\_USUARIOS;

/

create or replace package PRUEBAS\_UNIDADES as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_OID\_PRE NUMBER, w\_OID\_MPR NUMBER, w\_ESTADO\_EJEMPLAR VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_PRE NUMBER, w\_OID\_MPR NUMBER, w\_CODIGO\_IDENTIFICACION VARCHAR2,

w\_ESTADO\_EJEMPLAR VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_CODIGO\_IDENTIFICACION VARCHAR2, salidaEsperada BOOLEAN);

end PRUEBAS\_UNIDADES;

/

create or replace package PRUEBAS\_PRESTAMOS as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_DNI VARCHAR2, w\_FECHA\_INICIO DATE, w\_FECHA\_FIN DATE, w\_DEVOLUCION VARCHAR2, w\_NOMBRE\_ENCARGADO VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_PRE NUMBER, w\_DNI VARCHAR2, w\_FECHA\_INICIO DATE, w\_FECHA\_FIN DATE, w\_DEVOLUCION VARCHAR2,

w\_NOMBRE\_ENCARGADO VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_PRE VARCHAR2, salidaEsperada BOOLEAN);

end PRUEBAS\_PRESTAMOS;

/

create or replace package PRUEBAS\_MATERIALES\_PRESTADOS as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_NOMBRE VARCHAR2, w\_DISPONIBILIDAD VARCHAR2, w\_ESTADO\_DEVOLUCION VARCHAR2,

w\_CANTIDAD VARCHAR2, w\_TIPO VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_MPR NUMBER, w\_NOMBRE VARCHAR2, w\_DISPONIBILIDAD VARCHAR2, w\_ESTADO\_DEVOLUCION VARCHAR2,

w\_CANTIDAD VARCHAR2, w\_TIPO VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_MPR VARCHAR2, salidaEsperada BOOLEAN);

end PRUEBAS\_MATERIALES\_PRESTADOS;

/

create or replace package PRUEBAS\_INFORMES as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_OID\_CM NUMBER, w\_DNI VARCHAR2, w\_TIPO VARCHAR2, w\_RESPONSABLE VARCHAR2, w\_FECHA DATE,

w\_LUGAR VARCHAR2, w\_OBSERVACIONES LONG, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2,

w\_OID\_INF NUMBER, w\_OID\_CM NUMBER, w\_DNI VARCHAR2, w\_TIPO VARCHAR2, w\_RESPONSABLE VARCHAR2, w\_FECHA DATE, w\_LUGAR VARCHAR2, w\_OBSERVACIONES LONG, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_INF VARCHAR2, salidaEsperada BOOLEAN);

end PRUEBAS\_INFORMES;

/

create or replace package PRUEBAS\_CITAS\_MEDICAS as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_DNI VARCHAR2, w\_DOCTOR VARCHAR2, w\_FECHA DATE, w\_LUGAR VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_CM NUMBER, w\_DNI VARCHAR2, w\_DOCTOR VARCHAR2, w\_FECHA DATE, w\_LUGAR VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_CM NUMBER, salidaEsperada BOOLEAN);

end PRUEBAS\_CITAS\_MEDICAS;

/

create or replace package PRUEBAS\_ALOJAMIENTOS as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_UBICACION VARCHAR2, w\_DIRECCION VARCHAR2, w\_RESIDENTES VARCHAR2, w\_NUMERO\_RESIDENTES NUMBER,

w\_NUMERO\_HABITACION NUMBER, w\_DISPONIBILIDAD VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_ALJ NUMBER, w\_UBICACION VARCHAR2, w\_DIRECCION VARCHAR2, w\_RESIDENTES VARCHAR2, w\_NUMERO\_RESIDENTES NUMBER,

w\_NUMERO\_HABITACION NUMBER, w\_DISPONIBILIDAD VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_ALJ NUMBER, salidaEsperada BOOLEAN);

end PRUEBAS\_ALOJAMIENTOS;

/

create or replace package PRUEBAS\_ACTIVIDADES as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_LUGAR VARCHAR2, w\_FECHA DATE, w\_NUMERO\_ASISTENTES NUMBER, w\_CALIFICACIONES VARCHAR2,

salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_ACT VARCHAR2, w\_LUGAR VARCHAR2, w\_FECHA DATE, w\_NUMERO\_ASISTENTES NUMBER, w\_CALIFICACIONES VARCHAR2,

salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_ACT NUMBER, salidaEsperada BOOLEAN);

END PRUEBAS\_ACTIVIDADES;

/

create or replace package PRUEBAS\_ESTANCIAS as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_DNI VARCHAR2, w\_OID\_ALJ NUMBER, w\_FECHA\_ENTRADA DATE, w\_FECHA\_SALIDA DATE, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_est NUMBER, w\_DNI VARCHAR2, w\_OID\_ALJ NUMBER, w\_FECHA\_ENTRADA DATE, w\_FECHA\_SALIDA DATE,

salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_EST VARCHAR2, salidaEsperada BOOLEAN);

end PRUEBAS\_ESTANCIAS;

/

create or replace package PRUEBAS\_INSCRIPCIONES as

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_DNI VARCHAR2, w\_OID\_ACT NUMBER, w\_FECHA DATE, w\_correo\_electronico VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_OID\_INS NUMBER, w\_DNI VARCHAR2, w\_OID\_ACT NUMBER, w\_FECHA DATE, w\_correo\_electronico VARCHAR2, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2, w\_OID\_INS NUMBER, salidaEsperada BOOLEAN);

END PRUEBAS\_INSCRIPCIONES;

/

--------------------------------------------------------------------------/\*Pruebas\*/-----------------------------------------------------------------------------

SET SERVEROUTPUT ON;

DECLARE

cod\_act NUMBER;

cod\_alj NUMBER;

cod\_ins NUMBER;

cod\_cm NUMBER;

cod\_inf NUMBER;

BEGIN

pruebas\_usuarios.inicializar;

pruebas\_usuarios.insertar('Prueba usuario 1 - Insercion usuario', '30476281Z', 'Antonio', 'Palomares', 'Payaso', 'Telefono', 'Desintoxicacion', 'Depresion', 'Bueno', 0, 3, true);

pruebas\_usuarios.insertar('Prueba usuario 2 - Insercion usuario con nombre null', '30476181Z', null, 'Palomares', 'Payaso', 'Telefono', 'Desintoxicacion', 'Depresion', 'Bueno', 0, 3, false);

pruebas\_usuarios.actualizar('Prueba usuario 3 - Actualizacion apellido usuario', '30472381Z', 'Antonio', 'Villuela', 'Payaso', 'Telefono', 'Desintoxicacion', 'Depresion', 'Bueno', 0, 3, true);

pruebas\_usuarios.actualizar('Prueba usuario 4 - Actualizacion estado medico usuario a null', '32476281Z', 'Antonio', 'Palomares', 'Payaso', 'Telefono', 'Desintoxicacion', 'Depresion', null, 0, 3, false);

pruebas\_usuarios.eliminar('Prueba usuario 5 - Eliminar usuario', '30476265Z', true);

pruebas\_actividades.inicializar;

pruebas\_actividades.insertar('Prueba actividad 1 - Insercion actividad', 'Parque', TO\_DATE('01-03-2011','DD-MM-YYYY'), 40, 'Positivas', true);

cod\_act := SEC\_OID\_ACT.currval;

pruebas\_actividades.insertar('Prueba actividad 2 - Insercion actividad con lugar null', null, TO\_DATE('01-03-2011','DD-MM-YYYY'), 40, 'Positivas', false);

pruebas\_actividades.insertar('Prueba actividad 3 - Insercion actividad con fecha null', 'Parque', null, 40, 'Positivas', false);

pruebas\_actividades.actualizar('Prueba actividad 4 - Actualizacion lugar actividad', cod\_act, 'Bolera', TO\_DATE('01-03-2011','DD-MM-YYYY'), 40, 'Positivas', true);

pruebas\_actividades.actualizar('Prueba actividad 5 - Actualizacion fecha actividad a null', cod\_act, 'Parque', null, 40, 'Positivas', false);

pruebas\_actividades.eliminar('Prueba actividad 6 - Eliminar actividad', cod\_act, true);

pruebas\_alojamientos.inicializar;

pruebas\_alojamientos.insertar('Prueba alojamiento 1 - Insercion alojamiento', 'Edificio 1', 'Ctr. A-392, km 1, 5', '2', 8, 4, 'T', true);

cod\_alj := SEC\_OID\_ALJ.currval;

pruebas\_alojamientos.actualizar('Prueba alojamiento 2 - Actualizacion residentes alojamiento', cod\_alj, 'Edificio 1', 'Ctr. A-392, km 1, 5', '3', 8, 4, 'T', true);

pruebas\_alojamientos.actualizar('Prueba alojamiento 3 - Actualizacion codigo alojamiento a null', null, 'Edificio 1', 'Ctr. A-392, km 1, 5', '5', 8, 4, 'T', false);

pruebas\_alojamientos.eliminar('Prueba alojamiento 4 - Eliminar alojamiento', cod\_alj, true);

pruebas\_inscripciones.inicializar;

cod\_act := SEC\_OID\_ACT.currval;

pruebas\_inscripciones.insertar('Prueba Inscripcion 1 - Insertar inscripcion', '23653221H', cod\_act, TO\_DATE('01-03-2011','DD-MM-YYYY'), 'manueboliva@gmail.com', true);

cod\_ins := SEC\_OID\_INS.currval;

cod\_act := SEC\_OID\_ACT.currval;

pruebas\_inscripciones.insertar('Prueba Inscripcion 2 - Insertar fecha y correo inscripcion a null', '23653221H', cod\_act, null, null, false);

pruebas\_inscripciones.actualizar('Prueba Inscripcion 3 - Actualizacion inscripcion', cod\_ins, '38548694H', cod\_act, TO\_DATE('01-03-2011','DD-MM-YYYY'), 'manueboliva@gmail.com', true);

pruebas\_inscripciones.actualizar('Prueba Inscripcion 4 - Actualizacion correo inscripcion a null', cod\_ins, '23653221H', cod\_act, TO\_DATE('01-03-2011','DD-MM-YYYY'), null, false);

pruebas\_inscripciones.eliminar('Pruebas Inscripcion 5 - Eliminar inscripcion', cod\_ins, true);

pruebas\_citas\_medicas.inicializar;

pruebas\_citas\_medicas.insertar('Prueba Cita Medica 1 - Insertar cita', '13264987L', 'Jesús Postigo', TO\_DATE('01-03-2011','DD-MM-YYYY'), 'Hospital Virgen de la Macarena', true);

cod\_cm := SEC\_OID\_CM.currval;

pruebas\_citas\_medicas.insertar('Prueba Cita Medica 2 - Insertar cita con lugar null', '13264987L', 'Jesús Postigo', TO\_DATE('01-03-2011','DD-MM-YYYY'), null, false);

pruebas\_citas\_medicas.actualizar('Prueba Cita Medica 3 - Actualizar lugar cita', cod\_cm, '13264987L', 'Jesús Postigo', TO\_DATE('01-03-2011','DD-MM-YYYY'), 'Hospital San Juan de Dios', true);

pruebas\_citas\_medicas.actualizar('Prueba Cita Medica 4 - Actualizar DNI cita a null', cod\_cm, null, 'Jesús Postigo', TO\_DATE('01-03-2011','DD-MM-YYYY'), 'Hospital Virgen de la Macarena', false);

pruebas\_citas\_medicas.eliminar('Pruebas Cita Medica 5 - Eliminar cita', cod\_cm, true);

/\*pruebas\_informes.inicializar;

pruebas\_informes.insertar('Prueba Informe 1', cod\_cm, '12358964N', 'Psicológico', 'Francisco Alejo', TO\_DATE('01-03-2011','DD-MM-YYYY'),

'Plaza Olimpia, num 27 - 1ş', null, true);

cod\_inf := SEC\_OID\_INF.currval;

cod\_cm := SEC\_OID\_CM.currval;

pruebas\_informes.insertar('Prueba Informe 2', cod\_cm, '12358964N', 'Psicológico', null, null, null, false);

pruebas\_informes.actualizar('Prueba Informe 3', cod\_inf, cod\_cm, '12358964N', 'Psicológico', 'Francisco Alejo', TO\_DATE('01-03-2011','DD-MM-YYYY'),

'Plaza Olimpia, num 27 - 1ş', 'No procede', true);

pruebas\_informes.actualizar('Prueba Informe 4', cod\_inf, cod\_cm, '12358964N', 'Psicológico', 'Francisco Alejo', TO\_DATE('01-03-2011','DD-MM-YYYY'),

null, null, false);

pruebas\_informes.eliminar('Pruebas Informe 5', cod\_inf, true);

\*/

END;

/

--------USUARIOS------------------------------------------------------------------------------

EXECUTE NUEVO\_USUARIO('DNI1', 'NOMBRE1', 'AP1', 'OB1', 'CN1','MI1','EP1','EM1',1,1 );

EXECUTE NUEVO\_USUARIO('DNI2', 'NOMBRE2', 'AP2', 'OB2', 'CN2','MI2','EP2','EM2',2,2 );

EXECUTE NUEVO\_USUARIO('DNI3', 'NOMBRE3', 'AP3', 'OB3', 'CN3','MI3','EP3','EM3',3,3 );

EXECUTE NUEVO\_USUARIO('DNI4', 'NOMBRE4', 'AP4', 'OB4', 'CN4','MI4','EP4','EM4',4,4 );

INSERT INTO USUARIOS(DNI,NOMBRE, APELLIDOS, OBSERVACIONES, CONTACTO, MOTIVO\_INGRESO, ESTADO\_PSICOLOGICO,

ESTADO\_MEDICO, SANCIONES, PREMIOS)

VALUES ('DNI1', 'NOMBRE1', 'AP1', 'OB1', 'CN1','MI1','EP1','EM1',1,1);

INSERT INTO USUARIOS(DNI,NOMBRE, APELLIDOS, OBSERVACIONES, CONTACTO, MOTIVO\_INGRESO, ESTADO\_PSICOLOGICO,

ESTADO\_MEDICO, SANCIONES, PREMIOS)

VALUES ('DNI2', 'NOMBRE2', 'AP2', 'OB2', 'CN2','MI2','EP2','EM2',2,2);

INSERT INTO USUARIOS(DNI,NOMBRE, APELLIDOS, OBSERVACIONES, CONTACTO, MOTIVO\_INGRESO, ESTADO\_PSICOLOGICO,

ESTADO\_MEDICO, SANCIONES, PREMIOS)

VALUES ('DNI3', 'NOMBRE3', 'AP3', 'OB3', 'CN3','MI3','EP3','EM3',3,3);

INSERT INTO USUARIOS(DNI,NOMBRE, APELLIDOS, OBSERVACIONES, CONTACTO, MOTIVO\_INGRESO, ESTADO\_PSICOLOGICO,

ESTADO\_MEDICO, SANCIONES, PREMIOS)

VALUES ('DNI4', 'NOMBRE4', 'AP4', 'OB4', 'CN4','MI4','EP4','EM4',4,4);

-------------------------------------------UNIDADES--------------------------------------------------

EXECUTE NUEVA\_UNIDAD('COD1', 'E1' );

EXECUTE NUEVA\_UNIDAD('COD2', 'E2' );

EXECUTE NUEVA\_UNIDAD('COD3', 'E3' );

EXECUTE NUEVA\_UNIDAD('COD4', 'E4' );

INSERT INTO UNIDADES(OID\_PRE,OID\_MPR, CODIGO\_IDENTIFICACION, ESTADO\_EJEMPLAR)

VALUES('COD1', 'E1');

INSERT INTO UNIDADES(OID\_PRE,OID\_MPR, CODIGO\_IDENTIFICACION, ESTADO\_EJEMPLAR)

VALUES('COD2', 'E2');

INSERT INTO UNIDADES(OID\_PRE,OID\_MPR, CODIGO\_IDENTIFICACION, ESTADO\_EJEMPLAR)

VALUES('COD3', 'E3');

INSERT INTO UNIDADES(OID\_PRE,OID\_MPR, CODIGO\_IDENTIFICACION, ESTADO\_EJEMPLAR)

VALUES('COD4', 'E4');

-----------------------PRESTAMOS------------------------------------------------------------------

EXECUTE NUEVO\_PERSTAMO('DNI1', 'FI1', 'FF1','T', 'NE1' );

EXECUTE NUEVO\_PRESTAMO('DNI2', 'FI2', 'FF2','F', 'NE2' );

EXECUTE NUEVO\_PRESTAMO('DNI3', 'FI3', 'FF3','T', 'NE3' );

EXECUTE NUEVO\_PRESTAMO('DNI4', 'FI4', 'FF4','F', 'NE4' );

INSERT INTO PRESTAMOS(OID\_PRE, DNI,FECHA\_INICIO,FECHA\_FIN,DEVOLUCION,NOMBRE\_ENCARGADO)

VALUES ('DNI1', 'FI1', 'FF1','T', 'NE1');

INSERT INTO PRESTAMOS(OID\_PRE, DNI,FECHA\_INICIO,FECHA\_FIN,DEVOLUCION,NOMBRE\_ENCARGADO)

VALUES ('DNI2', 'FI2', 'FF2','F', 'NE2' );

INSERT INTO PRESTAMOS(OID\_PRE, DNI,FECHA\_INICIO,FECHA\_FIN,DEVOLUCION,NOMBRE\_ENCARGADO)

VALUES ('DNI3', 'FI3', 'FF3','T', 'NE3' );

INSERT INTO PRESTAMOS(OID\_PRE, DNI,FECHA\_INICIO,FECHA\_FIN,DEVOLUCION,NOMBRE\_ENCARGADO)

VALUES ('DNI4', 'FI4', 'FF4','F', 'NE4' );

----------------------------------------MATERIAL\_PRESTADO-----------------------------------------------

EXECUTE NUEVO\_MATERIAL\_PRESTADO('N1', 'D1', 'T',1, 'T1' );

EXECUTE NUEVO\_MATERIAL\_PRESTADO('N2', 'D2', 'F',2, 'T2' );

EXECUTE NUEVO\_MATERIAL\_PRESTADO('N3', 'D3', 'T',3, 'T3' );

EXECUTE NUEVO\_MATERIAL\_PRESTADO('N4', 'D4', 'F',4, 'T4' );

INSERT INTO MATERIALES\_PRESTADOS(OID\_MPR,NOMBRE,DISPONIBILIDAD,ESTADO\_DEVOLUCION,CANTIDAD,TIPO)

VALUES('N1', 'D1', 'T',1, 'T1');

INSERT INTO MATERIALES\_PRESTADOS(OID\_MPR,NOMBRE,DISPONIBILIDAD,ESTADO\_DEVOLUCION,CANTIDAD,TIPO)

VALUES('N2', 'D2', 'F',2, 'T2');

INSERT INTO MATERIALES\_PRESTADOS(OID\_MPR,NOMBRE,DISPONIBILIDAD,ESTADO\_DEVOLUCION,CANTIDAD,TIPO)

VALUES('N3', 'D3', 'T',3, 'T3');

INSERT INTO MATERIALES\_PRESTADOS(OID\_MPR,NOMBRE,DISPONIBILIDAD,ESTADO\_DEVOLUCION,CANTIDAD,TIPO)

VALUES('N4', 'D4', 'F',4, 'T4');

---------------------------INSCRIPCIONES--------------------------------------------------------------

EXECUTE NUEVA\_INSCRIPCION('DNI1', 'F1', 'C1');

EXECUTE NUEVA\_INSCRIPCION('DNI2', 'F2', 'C2' );

EXECUTE NUEVA\_INSCRIPCION('DNI3', 'F3', 'C3' );

EXECUTE NUEVA\_INSCRIPCION('DNI4', 'F4', 'C4' );

INSERT INTO INSCRIPCIONES(OID\_INS, DNI, OID\_ACT, FECHA, CORREO\_ELECTRONICO)

VALUES('DNI1', 'F1', 'C1');

INSERT INTO INSCRIPCIONES(OID\_INS, DNI, OID\_ACT, FECHA, CORREO\_ELECTRONICO)

VALUES('DNI2', 'F2', 'C2');

INSERT INTO INSCRIPCIONES(OID\_INS, DNI, OID\_ACT, FECHA, CORREO\_ELECTRONICO)

VALUES('DNI3', 'F3', 'C3');

INSERT INTO INSCRIPCIONES(OID\_INS, DNI, OID\_ACT, FECHA, CORREO\_ELECTRONICO)

VALUES('DNI4', 'F4', 'C4');

-------------------------------------------INFORMES-------------------------------------------------------

EXECUTE NUEVO\_INFORME('DN1', 'T1', 'R1', 'F1', 'L1', 'OB1');

EXECUTE NUEVO\_INFORME('DN2', 'T2', 'R2', 'F2', 'L2', 'OB2' );

EXECUTE NUEVO\_INFORME('DN3', 'T3', 'R3', 'F3', 'L3', 'OB3' );

EXECUTE NUEVO\_INFORME('DN4', 'T4', 'R4', 'F4', 'L4', 'OB4');

INSERT INTO INFORMES(OID\_INF,OID\_CM,DNI,TIPO,RESPONSABLE,FECHA,LUGAR,OBSERVACIONES)

VALUES('DN1', 'T1', 'R1', 'F1', 'L1', 'OB1');

INSERT INTO INFORMES(OID\_INF,OID\_CM,DNI,TIPO,RESPONSABLE,FECHA,LUGAR,OBSERVACIONES)

VALUES('DN2', 'T2', 'R2', 'F2', 'L2', 'OB2');

INSERT INTO INFORMES(OID\_INF,OID\_CM,DNI,TIPO,RESPONSABLE,FECHA,LUGAR,OBSERVACIONES)

VALUES('DN3', 'T3', 'R3', 'F3', 'L3', 'OB3' );

INSERT INTO INFORMES(OID\_INF,OID\_CM,DNI,TIPO,RESPONSABLE,FECHA,LUGAR,OBSERVACIONES)

VALUES('DN4', 'T4', 'R4', 'F4', 'L4', 'OB4' );

----------------------------------------ESTANCIAS-------------------------------------------------------

EXECUTE NUEVA\_ESTANCIA('DN1', 'FE1', 'FS1');

EXECUTE NUEVA\_ESTANCIA('DN2', 'FE2', 'FS2' );

EXECUTE NUEVA\_ESTANCIA('DN3', 'FE3', 'FS3' );

EXECUTE NUEVA\_ESTANCIA('DN4', 'FE4', 'FS4');

INSERT INTO ESTANCIAS (OID\_EST,DNI,OID\_ALJ,FECHA\_ENTRADA,FECHA\_SALIDA)

VALUES('DN1', 'FE1', 'FS1');

INSERT INTO ESTANCIAS (OID\_EST,DNI,OID\_ALJ,FECHA\_ENTRADA,FECHA\_SALIDA)

VALUES('DN2', 'FE2', 'FS2');

INSERT INTO ESTANCIAS (OID\_EST,DNI,OID\_ALJ,FECHA\_ENTRADA,FECHA\_SALIDA)

VALUES('DN3', 'FE3', 'FS3');

INSERT INTO ESTANCIAS (OID\_EST,DNI,OID\_ALJ,FECHA\_ENTRADA,FECHA\_SALIDA)

VALUES('DN4', 'FE4', 'FS4');

-----------------------------------CITAS\_MEDICAS---------------------------------------------------

EXECUTE NUEVA\_CITA\_MEDICA('DN1', 'DOC1', 'F1','L1');

EXECUTE NUEVA\_CITA\_MEDICA('DN2', 'DOC2', 'F2','L2');

EXECUTE NUEVA\_CITA\_MEDICA('DN3', 'DOC3', 'F3','L3' );

EXECUTE NUEVA\_CITA\_MEDICA('DN4', 'DOC4', 'F4','L4');

INSERT INTO CITAS\_MEDICAS(OID\_CM,DNI,DOCTOR,FECHA,LUGAR)

VALUES('DN1', 'DOC1', 'F1','L1');

INSERT INTO CITAS\_MEDICAS(OID\_CM,DNI,DOCTOR,FECHA,LUGAR)

VALUES('DN2', 'DOC2', 'F2','L2');

INSERT INTO CITAS\_MEDICAS(OID\_CM,DNI,DOCTOR,FECHA,LUGAR)

VALUES('DN3', 'DOC3', 'F3','L3');

INSERT INTO CITAS\_MEDICAS(OID\_CM,DNI,DOCTOR,FECHA,LUGAR)

VALUES('DN4', 'DOC4', 'F4','L4');

-----------------------------------ALOJAMIENTOS-----------------------------------------------

EXECUTE NUEVO\_ALOJAMIENTO('U1', 'D1', 'R1',1, 1, 'T');

EXECUTE NUEVO\_ALOJAMIENTO('U2', 'D2', 'R2',2, 2, 'F');

EXECUTE NUEVO\_ALOJAMIENTO('U3', 'D3', 'R3',3, 3, 'T');

EXECUTE NUEVO\_ALOJAMIENTO('U4', 'D4', 'R4',4, 4, 'F');

INSERT INTO ALOJAMIENTOS(OID\_ALJ,UBICACION,DIRECCION,RESIDENTES,NUMERO\_RESIDENTES,NUMERO\_HABITACION,DISPONIBILIDAD)

VALUES('U1', 'D1', 'R1',1, 1, 'T');

INSERT INTO ALOJAMIENTOS(OID\_ALJ,UBICACION,DIRECCION,RESIDENTES,NUMERO\_RESIDENTES,NUMERO\_HABITACION,DISPONIBILIDAD)

VALUES('U2', 'D2', 'R2',2, 2, 'F');

INSERT INTO ALOJAMIENTOS(OID\_ALJ,UBICACION,DIRECCION,RESIDENTES,NUMERO\_RESIDENTES,NUMERO\_HABITACION,DISPONIBILIDAD)

VALUES('U3', 'D3', 'R3',3, 3, 'T');

INSERT INTO ALOJAMIENTOS(OID\_ALJ,UBICACION,DIRECCION,RESIDENTES,NUMERO\_RESIDENTES,NUMERO\_HABITACION,DISPONIBILIDAD)

VALUES('U4', 'D4', 'R4',4, 4, 'F');

----------------------------------------------ACTIVIDADES------------------------------------------------

EXECUTE NUEVA\_ACTIVIDAD('L1', 'F1', 1,'C1');

EXECUTE NUEVA\_ACTIVIDAD('L2', 'F2', 2,'C2');

EXECUTE NUEVA\_ACTIVIDAD('L3', 'F3', 3,'C3');

EXECUTE NUEVA\_ACTIVIDAD('L4', 'F4', 4,'C4');

INSERT INTO ACTIVIDADES(OID\_ACT,LUGAR,FECHA,NUMERO\_ASISTENTES,CALIFICACIONES)

VALUES('L1', 'F1', 1,'C1');

INSERT INTO ACTIVIDADES(OID\_ACT,LUGAR,FECHA,NUMERO\_ASISTENTES,CALIFICACIONES)

VALUES('L2', 'F2', 2,'C2');

INSERT INTO ACTIVIDADES(OID\_ACT,LUGAR,FECHA,NUMERO\_ASISTENTES,CALIFICACIONES)

VALUES('L3', 'F3', 3,'C3');

INSERT INTO ACTIVIDADES(OID\_ACT,LUGAR,FECHA,NUMERO\_ASISTENTES,CALIFICACIONES)

VALUES('L4', 'F4', 4,'C4');

----------------------------------------------PRUEBAS DE TABLAS------------------------------------------------

create or replace PACKAGE BODY PRUEBAS\_USUARIOS AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM usuarios;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, NOMBRE VARCHAR2, APELLIDOS VARCHAR2, OBSERVACIONES VARCHAR2, CONTACTO VARCHAR2,

MOTIVO\_INGRESO VARCHAR2, ESTADO\_PSICOLOGICO VARCHAR2, ESTADO\_MEDICO VARCHAR2, SANCIONES NUMBER, PREMIOS NUMBER,

salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

usuario usuarios%ROWTYPE;

w\_DNI VARCHAR2;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

INSERT INTO usuarios VALUES (DNI, w\_nombre, w\_apellidos, w\_observaciones, w\_contacto, w\_motivo\_ingreso, w\_estado\_psicologico,

w\_estado\_medico, w\_sanciones, w\_premios);

SELECT \* INTO usuario FROM usuarios WHERE DNI = w\_DNI;

IF (usuario.NOMBRE<>w\_NOMBRE OR usuario.APELLIDOS<>w\_APELLIDOS OR usuario.OBSERVACIONES<>w\_OBSERVACIONES

OR usuario.CONTACTO<>w\_CONTACTO OR usuario.MOTIVO\_INGRESO<>w\_MOTIVO\_INGRESO OR usuario.ESTADO\_PSICOLOGICO<>w\_ESTADO\_PSICOLOGICO

OR usuario.ESTADO\_MEDICO<>w\_ESTADO\_MEDICO OR usuario.SANCIONES<>w\_SANCIONES OR usuario.PREMIOS<>w\_PREMIOS) 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, DNI VARCHAR2, NOMBRE VARCHAR2, APELLIDOS VARCHAR2, OBSERVACIONES VARCHAR2, CONTACTO VARCHAR2,

MOTIVO\_INGRESO VARCHAR2, ESTADO\_PSICOLOGICO VARCHAR2, ESTADO\_MEDICO VARCHAR2, SANCIONES NUMBER, PREMIOS NUMBER,

salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

usuario usuarios%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE usuarios SET NOMBRE=w\_NOMBRE, APELLIDOS=w\_APELLIDOS, OBSERVACIONES=w\_OBSERVACIONES,

CONTACTO=w\_CONTACTO, MOTIVO\_INGRESO=w\_MOTIVO\_INGRESO, ESTADO\_PSICOLOGICO=w\_ESTADO\_PSICOLOGICO,

ESTADO\_MEDICO=w\_ESTADO\_MEDICO, SANCIONES=w\_SANCIONES, PREMIOS=w\_PREMIOS WHERE DNI = w\_DNI;

SELECT \* INTO usuario FROM usuarios WHERE DNI = w\_DNI;

IF (usuario.NOMBRE<>w\_NOMBRE OR usuario.APELLIDOS<>w\_APELLIDOS OR usuario.OBSERVACIONES<>w\_OBSERVACIONES

OR usuario.CONTACTO<>w\_CONTACTO OR usuario.MOTIVO\_INGRESO<>w\_MOTIVO\_INGRESO OR usuario.ESTADO\_PSICOLOGICO<>w\_ESTADO\_PSICOLOGICO

OR usuario.ESTADO\_MEDICO<>w\_ESTADO\_MEDICO OR usuario.SANCIONES<>w\_SANCIONES OR usuario.PREMIOS<>w\_PREMIOS) 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, DNI VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_usuarios INTEGER;

BEGIN

DELETE FROM usuarios WHERE DNI = w\_DNI;

SELECT COUNT(\*) INTO n\_usuarios FROM usuarios WHERE DNI = w\_DNI;

IF (n\_usuarios<>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\_USUARIOS;

/

create or replace PACKAGE BODY PRUEBAS\_ACTIVIDADES AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM actividades;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, LUGAR VARCHAR2, FECHA DATE, NUMERO\_ASISTENTES NUMBER, CALIFICACIONES VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

actividad actividades%ROWTYPE;

w\_oid\_act NUMBER;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nuevo\_actividad(w\_lugar, w\_fecha, w\_numero\_asistente, w\_calificaciones);

w\_oid\_act := oid\_act.currval;

SELECT \* INTO actividad FROM actividades WHERE OID\_ACT = w\_oid\_act;

IF (actividad.LUGAR<>w\_LUGAR OR actividad.FECHA<>w\_FECHA OR actividad.NUMERO\_ASISTENTES<>w\_NUMERO\_ASISTENTES OR actividad.CALIFICACIONES<>w\_CALIFICACIONES) 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, OID\_ACT NUMBER, LUGAR VARCHAR2, FECHA DATE, NUMERO\_ASISTENTES NUMBER, CALIFICACIONES VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

actividad actividades%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE actividades SET LUGAR=w\_LUGAR, FECHA=w\_FECHA, NUMERO\_ASISTENTES=w\_NUMERO\_ASISTENTES,

CALIFICACIONES=w\_CALIFICACIONES WHERE OID\_ACT = w\_OID\_ACT;

SELECT \* INTO actividad FROM actividades WHERE OID\_ACT = w\_OID\_ACT;

IF (actividad.LUGAR<>w\_LUGAR OR actividad.FECHA<>w\_FECHA OR actividad.NUMERO\_ASISTENTES<>w\_NUMERO\_ASISTENTES OR actividad.CALIFICACIONES<>w\_CALIFICACIONES) 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, OID\_ACT VARCHAR2, 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;

/

create or replace PACKAGE BODY PRUEBAS\_INSCRIPCIONES AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM inscripciones;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, DNI VARCHAR2, OID\_ACT NUMBER, FECHA DATE, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

inscripcion inscripciones%ROWTYPE;

w\_oid\_ins NUMBER;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nueva\_inscripción(w\_dni, w\_oid\_act, w\_fecha);

w\_oid\_ins := oid\_ins.currval;

SELECT \* INTO inscripcion FROM inscripciones WHERE OID\_INS = w\_oid\_ins;

IF (inscripcion.DNI<>w\_DNI OR inscripcion.OID\_ACT<>w\_OID\_ACT OR inscripcion.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, OID\_INS NUMBER, DNI VARCHAR2, OID\_ACT NUMBER, FECHA DATE, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

inscripcion inscripciones%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE inscripciones SET DNI=w\_DNI, OID\_ACT=w\_OID\_ACT, FECHA= W\_FECHA WHERE OID\_INS = w\_OID\_INS;

SELECT \* INTO inscripcion FROM inscripciones WHERE OID\_INS = w\_OID\_INS;

IF (inscripcion.DNI<>w\_DNI OR inscripcion.OID\_ACT<>w\_OID\_ACT OR inscripcion.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, OID\_INS NUMBER, 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;

/

create or replace PACKAGE BODY PRUEBAS\_CITAS\_MEDICAS AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM citas\_medicas;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, DNI VARCHAR2, DOCTOR VARCHAR2, FECHA DATE, LUGAR VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

cita\_medica citas\_medicas%ROWTYPE;

w\_oid\_cm NUMBER;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nueva\_cita\_medica(w\_dni, w\_doctor, w\_fecha, w\_lugar);

w\_oid\_cm := oid\_cm.currval;

SELECT \* INTO cita\_medica FROM citas\_medicas WHERE OID\_CM = w\_oid\_cm;

IF (cita\_medica.DNI<>w\_DNI OR cita\_medica.DOCTOR<>w\_DOCTOR OR cita\_medica.FECHA<>w\_FECHA OR cita\_medica.LUGAR<>w\_LUGAR) 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, OID\_CM NUMBER, DNI VARCHAR2, DOCTOR VARCHAR2, FECHA DATE, LUGAR VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

cita\_medica citas\_medicas%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE citas\_medicas SET DNI=w\_DNI, DOCTOR=w\_DOCTOR, FECHA= W\_FECHA, LUGAR=W\_LUGAR WHERE OID\_CM = w\_OID\_CM;

SELECT \* INTO cita\_medica FROM citas\_medicas WHERE OID\_CM = w\_OID\_CM;

IF (cita\_medica.DNI<>w\_DNI OR cita\_medica.DOCTOR<>w\_DOCTOR OR cita\_medica.FECHA<>w\_FECHA OR cita\_medica.LUGAR<>w\_LUGAR) 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, OID\_CM NUMBER, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_citas\_medicas INTEGER;

BEGIN

DELETE FROM citas\_medicas WHERE OID\_CM = w\_OID\_CM;

SELECT COUNT(\*) INTO n\_citas\_medicas FROM citas\_medicas WHERE OID\_CM = w\_OID\_CM;

IF (n\_citas\_medicas<>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\_CITAS\_MEDICAS;

/

create or replace PACKAGE BODY PRUEBAS\_INFORMES AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM informes;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, OID\_CM NUMBER, DNI VARCHAR2, TIPO VARCHAR2, RESPONSABLE VARCHAR2, FECHA DATE, LUGAR VARCHAR2, OBSERVACIONES LONG, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

informe informes%ROWTYPE;

w\_oid\_inf NUMBER;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nuevo\_informe(w\_oid\_cm, w\_dni, w\_tipo, w\_responsable, w\_fecha, w\_lugar, w\_observaciones);

w\_oid\_inf := oid\_inf.currval;

SELECT \* INTO informe FROM informes WHERE OID\_INF = w\_oid\_inf;

IF (informes.OID\_CM<>w\_OID\_CM OR informes.DNI<>w\_DNI OR informes.TIPO<>w\_TIPO OR informes.RESPONSABLES<>w\_RESPONSABLES OR informes.FECHA<>w\_FECHA,

informes.LUGAR<>w\_LUGAR, informes.OBSERVACIONES<>w\_OBSERVACIONES) 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,

OID\_INF NUMBER, OID\_CM NUMBER, DNI VARCHAR2, TIPO VARCHAR2, RESPONSABLE VARCHAR2, FECHA DATE, LUGAR VARCHAR2, OBSERVACIONES LONG, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

informe informess%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE informes SET OID\_CM=w\_OID\_CM, DNI=w\_DNI, TIPO=w\_TIPO,

RESPONSABLE=w\_RESPONSABLE, FECHA=w\_FECHA, LUGAR=w\_LUGAR,

OBSERVACIONES=w\_OBSERVACIONES WHERE OID\_INF = w\_OID\_INF;

SELECT \* INTO informes FROM informes WHERE OID\_INF = w\_OID\_INF;

IF (informes.OID\_CM<>w\_OID\_CM OR informes.DNI<>w\_DNI OR informes.TIPO<>w\_TIPO OR informes.RESPONSABLES<>w\_RESPONSABLES OR informes.FECHA<>w\_FECHA,

informes.LUGAR<>w\_LUGAR, informes.OBSERVACIONES<>w\_OBSERVACIONES) 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, OID\_INF VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_informes INTEGER;

BEGIN

DELETE FROM informes WHERE OID\_INF = w\_OID\_INF;

SELECT COUNT(\*) INTO n\_informes FROM informes WHERE OID\_INF = w\_OID\_INF;

IF (n\_informes<>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\_INFORMES;

/

create or replace PACKAGE BODY PRUEBAS\_PRESTAMOS AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM prestamos;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, DNI VARCHAR2, FECHA\_INICIO DATE, FECHA\_FIN DATE, DEVOLUCION VARCHAR2, NOMBRE\_ENCARGADO VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

prestamo prestamos%ROWTYPE;

w\_oid\_pre NUMBER;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nuevo\_prestamo(w\_dni, w\_fecha\_inicio, w\_fecha\_fin, w\_devolucion, w\_nombre\_encargado);

w\_oid\_pre := oid\_pre.currval;

SELECT \* INTO prestamo FROM prestamos WHERE OID\_PRE = w\_oid\_pre;

IF (prestamo.DNI<>w\_DNI OR prestamo.FECHA\_INICIO<>w\_FECHA\_INICIO OR prestamo.FECHA\_FIN<>w\_FECHA\_FIN OR prestamo.DEVOLUCION<>w\_DEVOLUCION, prestamo.NOMBRE\_ENCARGADO<>w\_NOMBRE\_ENCARGADO) 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, OID\_PRE NUMBER, DNI VARCHAR2, FECHA\_INICIO DATE, FECHA\_FIN DATE, DEVOLUCION VARCHAR2, NOMBRE\_ENCARGADO VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

prestamo prestamos%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE prestamos SET DNI=w\_DNI, FECHA\_INICIO=w\_FECHA\_INICIO, FECHA\_FIN=w\_FECHA\_FIN, DEVOLUCION=w\_DEVOLUCION, NOMBRE\_ENCARGADO=w\_NOMBRE\_ENCARGADO

WHERE OID\_PRE = w\_OID\_PRE;

SELECT \* INTO prestamo FROM prestamos WHERE OID\_PRE = w\_OID\_PRE;

IF (prestamo.DNI<>w\_DNI OR prestamo.FECHA\_INICIO<>w\_FECHA\_INICIO OR prestamo.FECHA\_FIN<>w\_FECHA\_FIN OR prestamo.DEVOLUCION<>w\_DEVOLUCION, prestamo.NOMBRE\_ENCARGADO<>w\_NOMBRE\_ENCARGADO) 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, OID\_PRE VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_prestamos INTEGER;

BEGIN

DELETE FROM prestamos WHERE OID\_PRE = w\_OID\_PRE;

SELECT COUNT(\*) INTO n\_prestamos FROM prestamos WHERE OID\_PRE = w\_OID\_PRE;

IF (n\_prestamos<>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\_PRESTAMOS;

/

create or replace PACKAGE BODY PRUEBAS\_MATERIALES\_PRESTADOS AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM materiales\_prestados;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, NOMBRE VARCHAR2, DISPONIBILIDAD VARCHAR2, ESTADO\_DEVOLUCION VARCHAR2, CANTIDAD VARCHAR2, TIPO VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

material\_prestado materiales\_prestados%ROWTYPE;

w\_oid\_mpr NUMBER;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nuevo\_material\_prestado(w\_nombre, w\_disponibilidad, w\_estado\_devolucion, w\_cantidad, w\_tipo);

w\_oid\_mpr := oid\_mpr.currval;

SELECT \* INTO material\_prestado FROM materiales\_prestados WHERE OID\_MPR = w\_oid\_mpr;

IF (material\_prestado.NOMBRE<>w\_NOMBRE OR material\_prestado.DISPONIBILIDAD<>w\_DISPONIBILIDAD OR material\_prestado.ESTADO\_DEVOLUCION<>w\_ESTADO\_DEVOLUCION OR material\_prestado.CANTIDAD<>w\_CANTIDAD,

material\_prestado.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, OID\_MPR NUMBER, NOMBRE VARCHAR2, DISPONIBILIDAD VARCHAR2, ESTADO\_DEVOLUCION VARCHAR2, CANTIDAD VARCHAR2, TIPO VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

material\_prestado materiales\_prestados%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE material\_prestado SET NOMBRE=w\_NOMBRE, DISPONIBILIDAD=w\_DISPONIBILIDAD, ESTADO\_DEVOLUCION=w\_ESTADO\_DEVOLUCION, CANTIDAD=w\_CANTIDAD, TIPO=w\_TIPO WHERE OID\_MPR = w\_OID\_MPR;

SELECT \* INTO material\_prestado FROM materiales\_prestados WHERE OID\_MPR = w\_OID\_MPR;

IF (material\_prestado.NOMBRE<>w\_NOMBRE OR material\_prestado.DISPONIBILIDAD<>w\_DISPONIBILIDAD OR material\_prestado.ESTADO\_DEVOLUCION<>w\_ESTADO\_DEVOLUCION OR material\_prestado.CANTIDAD<>w\_CANTIDAD,

material\_prestado.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, OID\_MPR VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_materiales\_prestados INTEGER;

BEGIN

DELETE FROM materiales\_prestados WHERE OID\_MPR = w\_OID\_MPR;

SELECT COUNT(\*) INTO n\_materiales\_prestados FROM materiales\_prestados WHERE OID\_MPR = w\_OID\_MPR;

IF (n\_materiales\_prestados<>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\_MATERIALES\_PRESTADOS;

/

create or replace PACKAGE BODY PRUEBAS\_UNIDADES AS

PROCEDURE inicializar AS

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_USUARIOS.inicializar

DELETE FROM unidades;

END inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, OID\_PRE NUMBER, OID\_MPR NUMBER, ESTADO\_EJEMPLAR VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

unidad unidades%ROWTYPE;

w\_codigo\_identificacion VARCHAR2;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

nueva\_unidad(w\_oid\_pre, w\_oid\_mpr, w\_codigo\_identificacion, w\_estado\_ejemplar);

SELECT \* INTO unidad FROM unidades WHERE codigo\_identificacion = w\_codigo\_identificacion;

IF (unidad.OID\_PRE<>w\_OID\_PRE OR unidad.OID\_MPR<>w\_OID\_MPR OR unidad.ESTADO\_EJEMPLAR<>w\_ESTADO\_EJEMPLAR) 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, OID\_PRE NUMBER, OID\_MPR NUMBER, CODIGO\_IDENTIFICACION VARCHAR2, ESTADO\_EJEMPLAR VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN:=true;

unidad unidades%ROWTYPE;

BEGIN

-- TAREA: Se necesita implantación para PROCEDURE PRUEBAS\_DEPARTAMENTOS.insertar

UPDATE unidad SET OID\_PRE=w\_OID\_PRE, OID\_MPR=w\_OID\_MPR, ESTADO\_EJEMPLAR=w\_ESTADO\_EJEMPLAR WHERE CODIGO\_IDENTIFICACION = w\_CODIGO\_IDENTIFICACION;

SELECT \* INTO unidad FROM unidades WHERE CODIGO\_IDENTIFICACION = w\_CODIGO\_IDENTIFICACION;

IF (unidad.OID\_PRE<>w\_OID\_PRE OR unidad.OID\_MPR<>w\_OID\_MPR OR unidad.ESTADO\_EJEMPLAR<>w\_ESTADO\_EJEMPLAR) 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, CODIGO\_IDENTIFICACION VARCHAR2, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_unidades INTEGER;

BEGIN

DELETE FROM unidades WHERE CODIGO\_IDENTIFICACION = w\_CODIGO\_IDENTIFICACION;

SELECT COUNT(\*) INTO n\_unidades FROM unidades WHERE CODIGO\_IDENTIFICACION = w\_CODIGO\_IDENTIFICACION;

IF (n\_unidades<>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\_UNIDADES;

/