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

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

1.1. 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.

1.2. 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.

1.3. 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.

.

1.4. 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.

1.5. 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

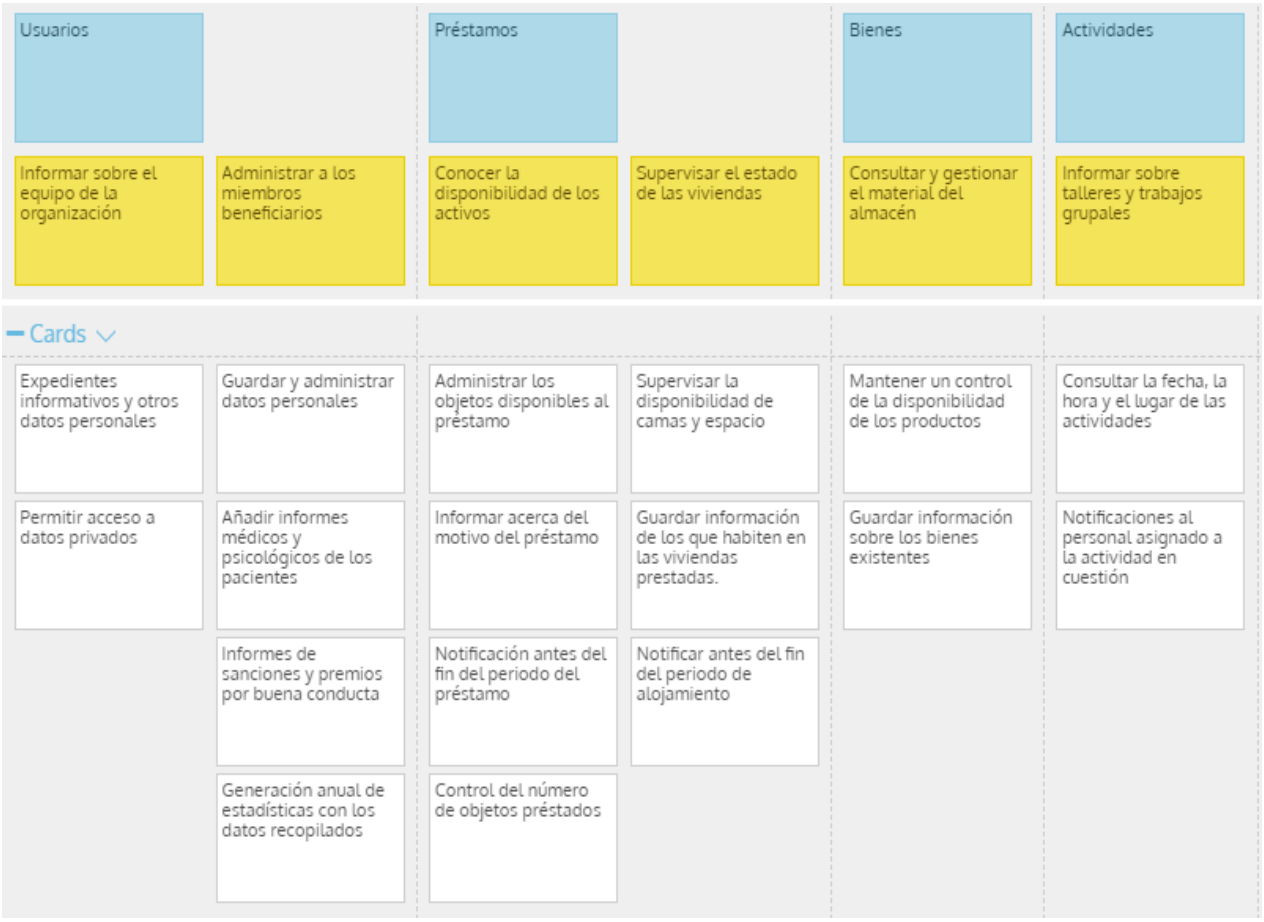
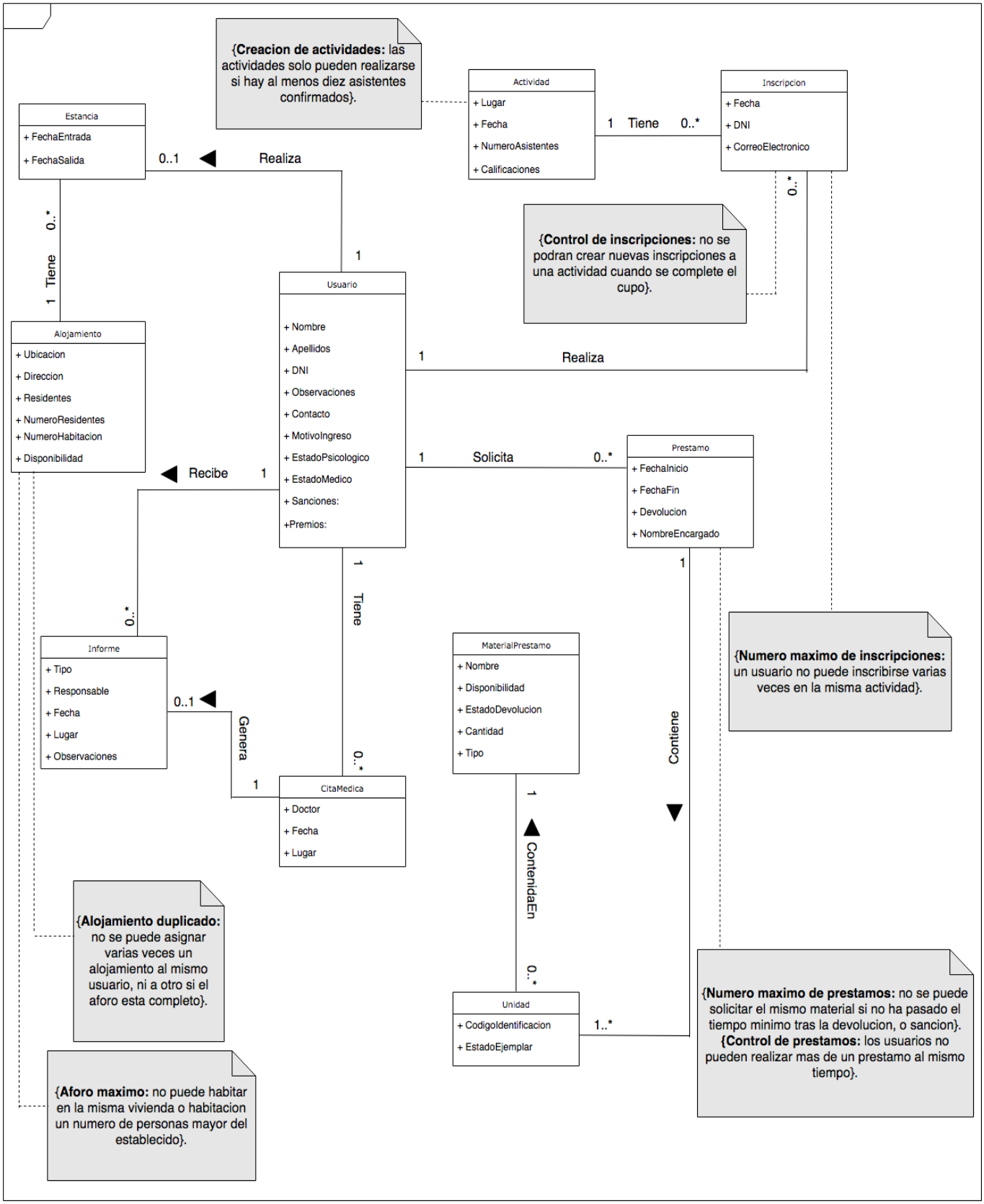


Diagrama conceptual



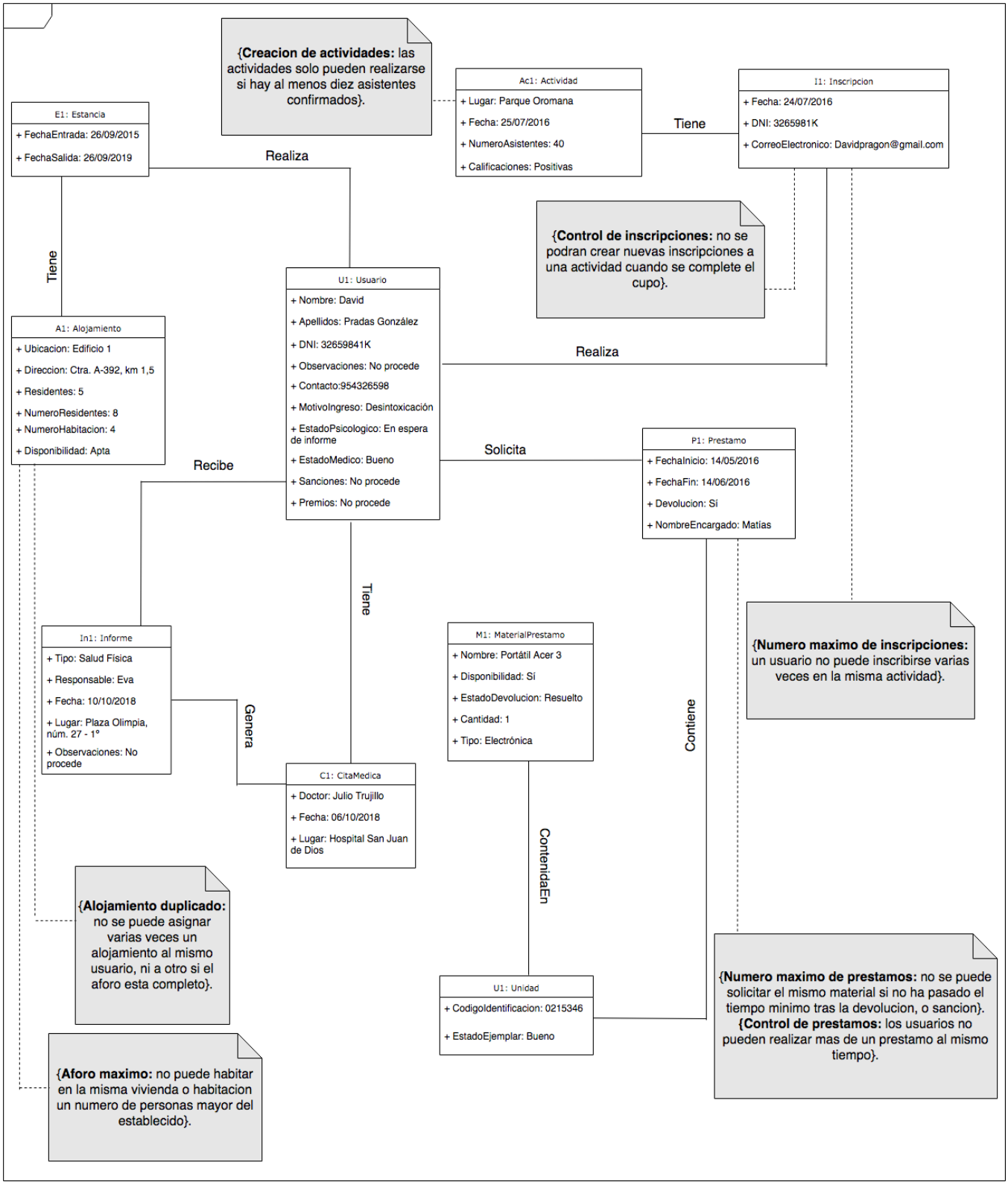
C/A	RI-1	RI-2	RI-3	RI-4	RI-5	RI-6	RI-7	RI-8	RI-9	RNF-1	RNF-2	RNF-3	RNF-4	RNF-5	RNF-6	RNF-7	RNF-8
Usuario	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓			✓	✓	✓
Estancia							✓							✓			✓
Alojamiento							✓							✓			✓
Inscripcion					✓	✓									✓		
Actividad					✓	✓											
Prestamo								✓	✓			✓	✓			✓	
Unidad				✓	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

Solicita									✓			✓				✓	
Contiene R								✓									
ContenidaEn				✓				✓									

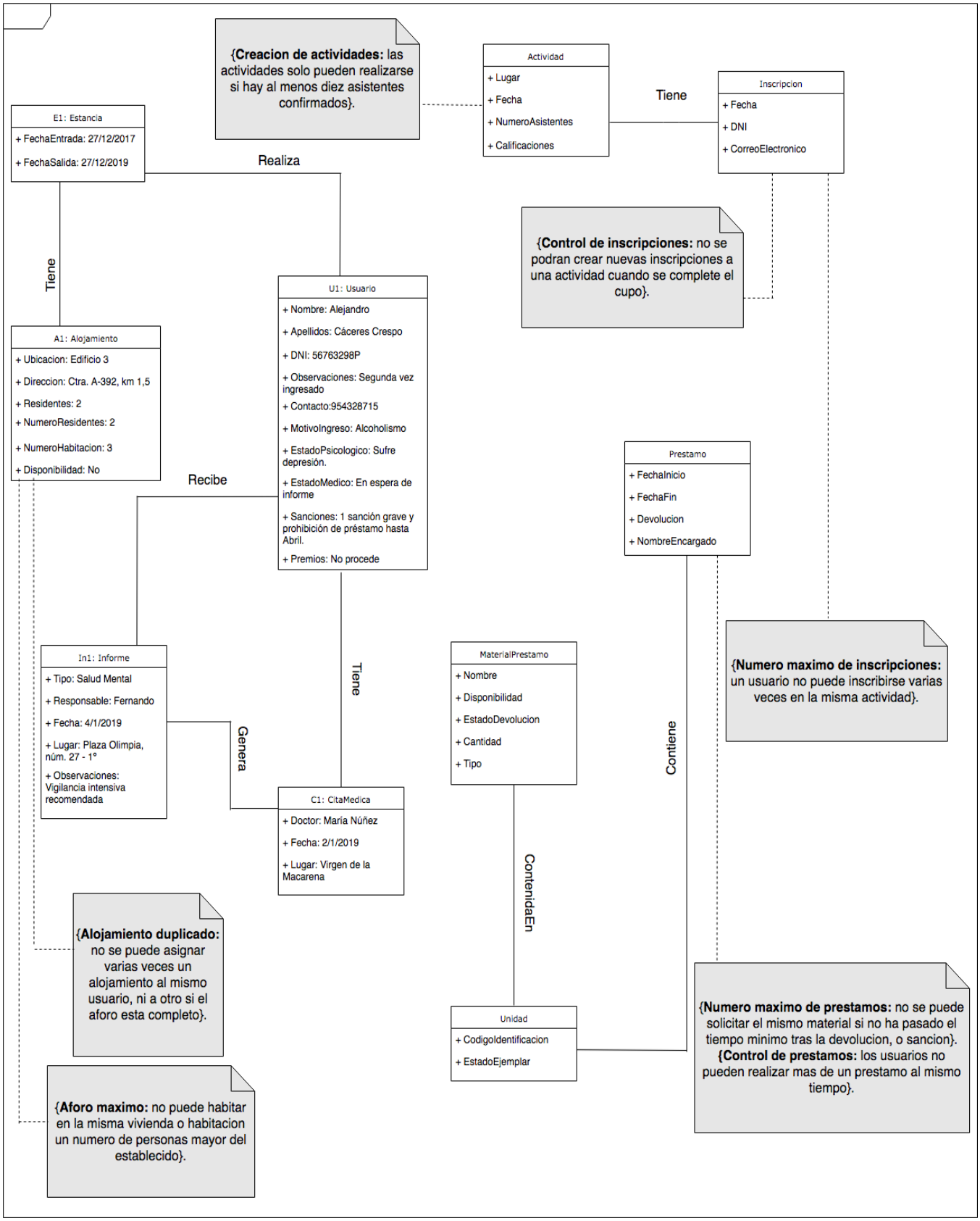
Matriz de trazabilidad

Casos de Prueba

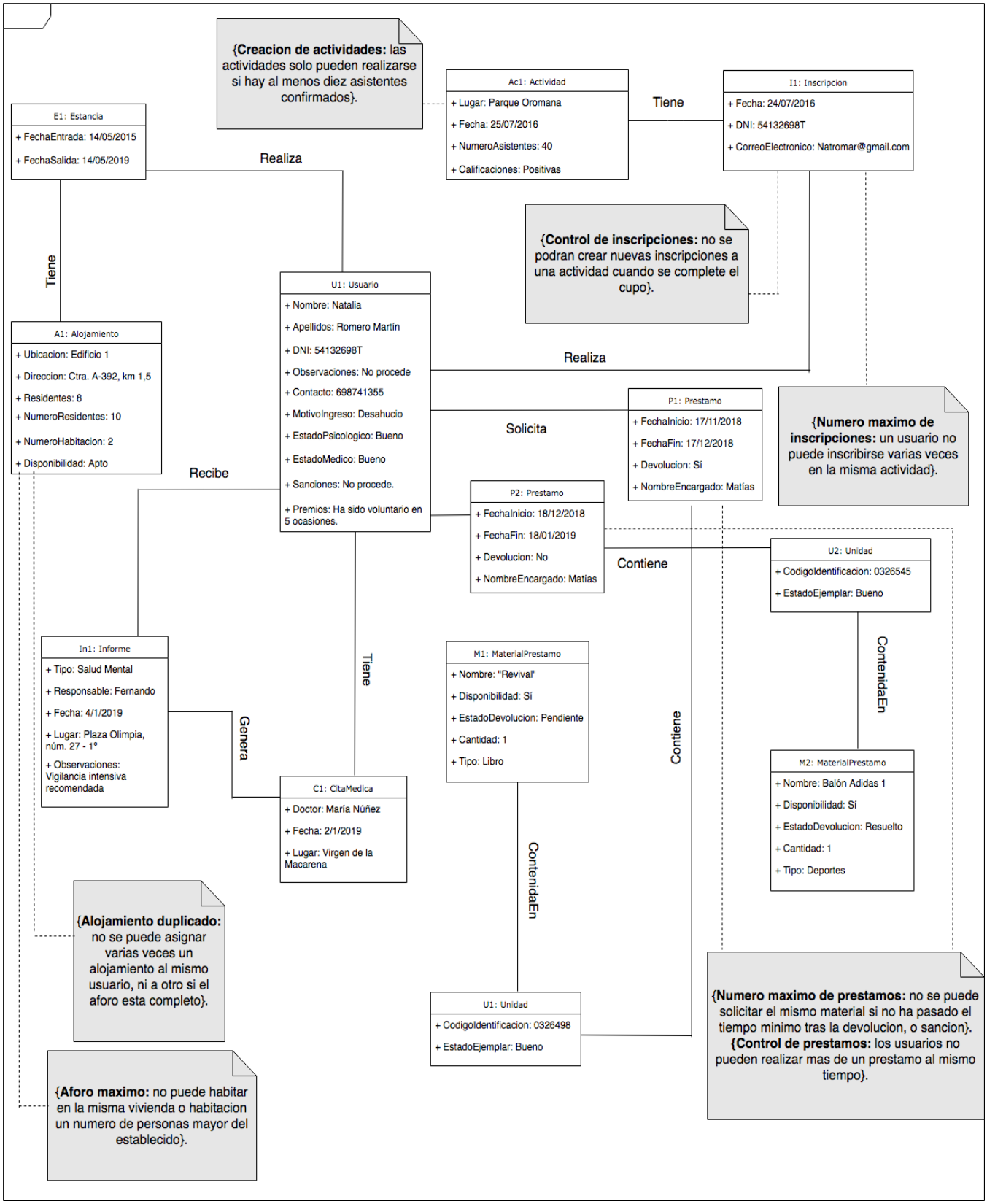
- **Caso de Prueba 1**: Usuario corriente con alojamiento en la asociación, informe médico físico y no psicológico, un préstamo y participa en actividades.


{Creacion de actividades: las actividades solo pueden realizarse si hay al menos diez asistentes confirmados}.
Ac1: Actividad
+ Lugar: Parque Oromana
+ Fecha: 25/07/2016
+ NumeroAsistentes: 40
+ Calificaciones: Positivas
Tiene
I1: Inscripcion
+ Fecha: 24/07/2016
+ DNI: 3265981K
+ CorreoElectronico: Davidpragon@gmail.com
E1: Estancia
+ FechaEntrada: 26/09/2015
+ FechaSalida: 26/09/2019
Realiza
Tiene
{Control de inscripciones: no se podran crear nuevas inscripciones a una actividad cuando se complete el cupo}.
U1: Usuario
+ Nombre: David
+ Apellidos: Pradas González
+ DNI: 32659841K
+ Observaciones: No procede
+ Contacto:954326598
+ MotivoIngreso: Desintoxicación
+ EstadoPsicologico: En espera de informe
+ EstadoMedico: Bueno
+ Sanciones: No procede
+ Premios: No procede
A1: Alojamiento
+ Ubicacion: Edificio 1
+ Direccion: Ctra. A-392, km 1,5
+ Residentes: 5
+ NumeroResidentes: 8
+ NumeroHabitacion: 4
+ Disponibilidad: Apta
Realiza
Solicita
P1: Prestamo
+ FechaInicio: 14/05/2016
+ FechaFin: 14/06/2016
+ Devolucion: Sí
+ NombreEncargado: Matías
Recibe
Tiene
In1: Informe
+ Tipo: Salud Física
+ Responsable: Eva
+ Fecha: 10/10/2018
+ Lugar: Plaza Olimpia, núm. 27 - 1°
+ Observaciones: No procede
Genera
M1: MaterialPrestamo
+ Nombre: Portátil Acer 3
+ Disponibilidad: Sí
+ EstadoDevolucion: Resuelto
+ Cantidad: 1
+ Tipo: Electrónica
{Numero maximo de inscripciones: un usuario no puede inscribirse varias veces en la misma actividad}.
Contiene
C1: CitaMedica
+ Doctor: Julio Trujillo
+ Fecha: 06/10/2018
+ Lugar: Hospital San Juan de Dios
ContenidaEn
{Alojamiento duplicado: no se puede asignar varias veces un alojamiento al mismo usuario, ni a otro si el aforo esta completo}.
U1: Unidad
+ CodigoIdentificacion: 0215346
+ EstadoEjemplar: Bueno
{Numero maximo de prestamos: no se puede solicitar el mismo material si no ha pasado el tiempo minimo tras la devolucion, o sancion}.
{Control de prestamos: los usuarios no pueden realizar mas de un prestamo al mismo tiempo}.
{Aforo maximo: no puede habitar en la misma vivienda o habitacion un numero de personas mayor del establecido}.

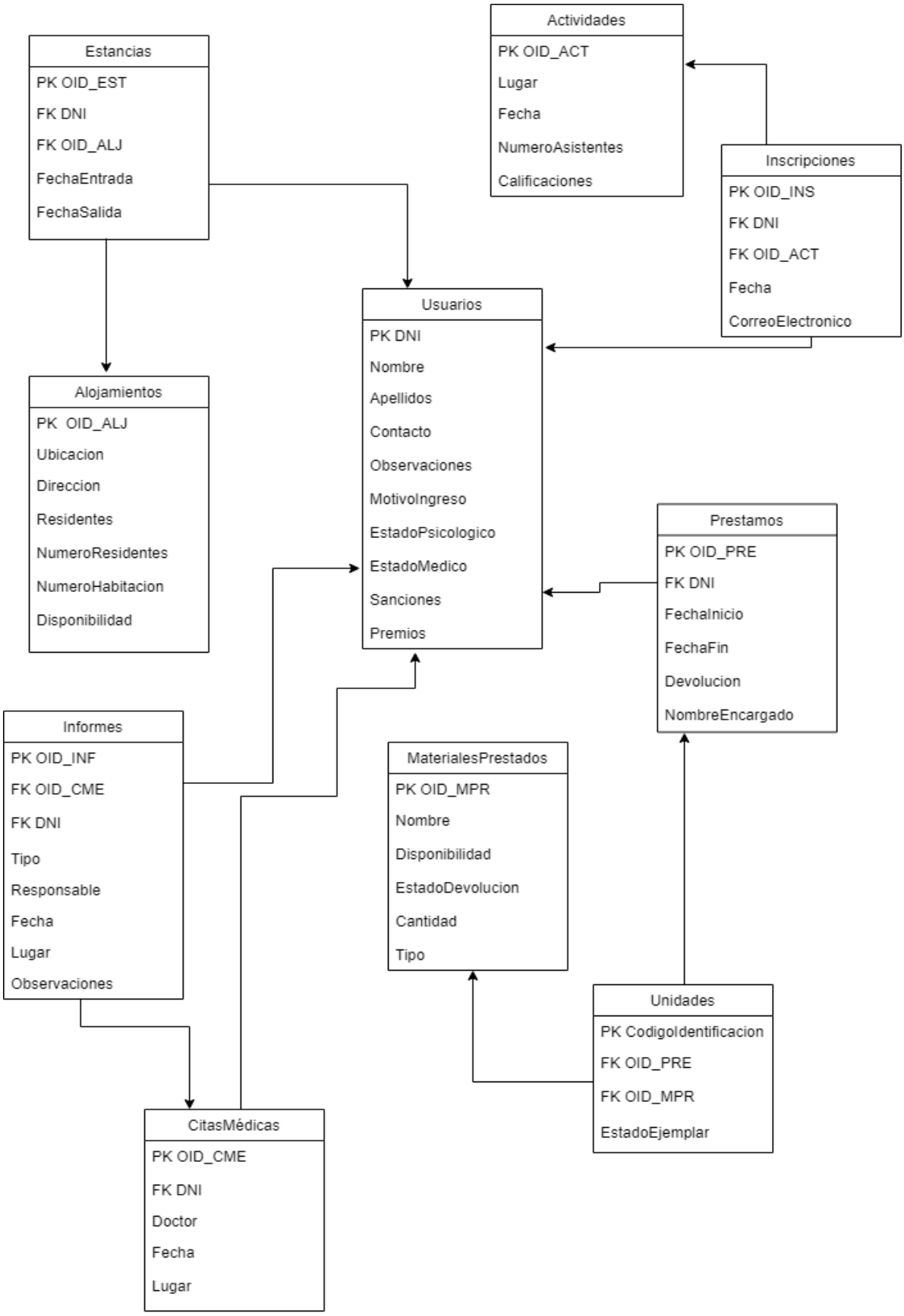
- **Caso de Prueba 2:** Usuario alojado en residencias de AFAR, informe médico psicológico, sin préstamos y no participa en actividades.



- **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

Entidades que colaboran:

- Junta de Andalucía
- Fundación Cajasol
- Obra Social la Caixa
- Unión Europea
- Ayuntamiento de Sevilla
- Ministerio de Bienestar, Consumo y Bienestar Social
- Grupo Cementos Portlan Valderrivas
- Ayuntamiento de Alcalá
- Cáritas
- Hermandad del Águila
- Protelsur
- Dacolim S.L.
- Machenieto
- Martín Casillas
- Ingtar

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_E
NCARGADO);

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,CANTID
AD,TIPO)

VALUES
(SEC_OID_MPR.nextval,w_NOMBRE,w_DISPONIBILIDAD,w_ESTADO_DEVOLUCION,w_CANTIDA
D,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_OBS
ERVACIONES);

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 - 1ș', 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 - 1ș', '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,CANTID
AD,TIPO)

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

INSERT INTO
MATERIALES_PRESTADOS(OID_MPR,NOMBRE,DISPONIBILIDAD,ESTADO_DEVOLUCION,CANTID
AD,TIPO)

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

INSERT INTO
MATERIALES_PRESTADOS(OID_MPR,NOMBRE,DISPONIBILIDAD,ESTADO_DEVOLUCION,CANTID
AD,TIPO)

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

INSERT INTO
MATERIALES_PRESTADOS(OID_MPR,NOMBRE,DISPONIBILIDAD,ESTADO_DEVOLUCION,CANTID
AD,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;

/
```