Proyecto

Espacio, Vida & Música

Introducción a la Ingeniería del Software

y los Sistemas de Información

https://projetsii.informatica.us.es/projects/trfqdtp3ph2qxehazh2

Iglesias Pérez, Daniel

Muñoz de Souza, Carlos

Outin, Louis

Rodríguez Martín, Javier

# Historial de versiones

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Descripción |
| 1 | 18/10/2015 | Versión inicial |
| 2 | 22/11/2015 | * Mejora del primer entregable * Catálogo de requisitos * Pruebas de aceptación * Modelo conceptual * Matrices de trazabilidad |
| 3 | 10/01/2016 | * Correcciones en todos los BPMN * Nueva realización de los RI. * Se han añadido nuevos RF y RNF. * Pruebas de aceptación de los requisitos añadidos. * Cambios en el UML y escenarios de prueba * Nueva realización de las matrices. * Contenido de la nueva entrega. |
|  |  |  |

Índice

[Historial de versiones 2](#_Toc440110143)

[1 - Introducción 4](#_Toc440110144)

[2 – Glosario 5](#_Toc440110145)

[3 – Modelo de negocio 6](#_Toc440110146)

[4 – Visión general del sistema 10](#_Toc440110147)

[5 – Catálogo de requisitos 11](#_Toc440110148)

[Requisitos de información 11](#_Toc440110149)

[Reglas de negocio 12](#_Toc440110150)

[Requisitos funcionales 13](#_Toc440110151)

[Requisitos no funcionales 15](#_Toc440110152)

[6 – Pruebas de aceptación 16](#_Toc440110153)

[Pruebas de aceptación de los requisitos de información 16](#_Toc440110154)

[Pruebas de aceptación de las reglas de negocio 17](#_Toc440110155)

[Pruebas de aceptación de los requisitos funcionales 18](#_Toc440110156)

[Pruebas de aceptación de los requisitos no funcionales 20](#_Toc440110157)

[7 – Modelo conceptual 21](#_Toc440110158)

[8 – Matrices de trazabilidad 25](#_Toc440110159)

[Pruebas de aceptación/Requisitos 25](#_Toc440110160)

[Requisitos/UML 27](#_Toc440110161)

[Pruebas de aceptación/Escenarios de prueba 28](#_Toc440110162)

[9- Modelo Relacional 30](#_Toc440110163)

[10 – Modelo Tecnológico 31](#_Toc440110164)

[Script de creación de tablas, restricciones, secuencias, triggers asociados a la gestión de secuencias e índices. 31](#_Toc440110165)

[Script de creación de funciones y procedimientos 40](#_Toc440110166)

[Script de creación de triggers no asociados a secuencias 51](#_Toc440110167)

[Script de pruebas 54](#_Toc440110168)

[Anexo 112](#_Toc440110169)

# 1 - Introducción

Nuestro sistema de información va destinado a dar solución a las necesidades del proyecto “Espacio, Vida & Música” (EVM en adelante). Este es un proyecto que está bajo la cobertura de la “Fundación Pasión y Compromiso” y su objetivo final es trabajar aspectos tan importantes para el desarrollo de un niño como son la comunicación, el respeto, la tolerancia o la generosidad, entre otros. Además promueve una mejor conexión entre padres, hijos y profesores.

EVM es en definitiva una escuela de música, situada en la Iglesia Evangelista de la calle Casiodoro de Reina, donde los niños tienen la oportunidad de conocer el mundo de los instrumentos mientras desarrollan sus capacidades y aptitudes sociales, también buscan fomentar el compañerismo, el trabajo en equipo y reforzar los vínculos entre padres e hijos. El enfoque es cooperativo, y nunca competitivo, de forma que todos los alumnos puedan avanzar a su propio ritmo con la ayuda de los demás.

Para tener éxito en el camino, EVM recalca la importancia de que la familia se involucre en el aprendizaje del alumno. Entre todas las responsabilidades que se le pide a la familia, podemos destacar la práctica diaria con el niño y la presencia obligatoria del padre en algunas de las materias que el alumno aprende en la sede, para alumnos de 3 a 11 años.

Por otra parte, EVM está subvencionado por la fundación, de forma que el alumno solo tiene que pagar aproximadamente un 20% del coste total y en caso de no poder adquirir el instrumento que necesite, se le prestará uno. Además, cada caso es especial, se estudia personalmente, pudiendo reducir aún más las tasas.

El equipo que hace posible esto consta de un director académico, dos secretarios y once profesores que trabajan para más de 90 alumnos actualmente. El principal problema que tienen ahora mismo es que la administración puede llegar a ser un poco caótica.

Aunque tienen [página web](http://www.espaciovidaymusica.es), es solo a nivel informativo y carecen de más medios que hojas de papel, es decir, las hojas de registro que son archivadas en carpetas, y un archivo de hoja de cálculos tipo Excel, en el que hay una lista de alumnos y los pagos correspondientes a cada mes, para llevar el registro de los alumnos y la contabilidad.

En cuanto nos pusimos en contacto con ellos se mostraron muy agradecidos y sus expectativas son agilizar el sistema administrativo con nuestra ayuda, tener una mejor organización tanto para la administración de los alumnos, pagos, instrumentos prestados y profesores, dedicándonos todo el tiempo que creamos necesario en reuniones para aclarar los requisitos del sistema.

# 2 – Glosario

**Calendario “EVM”:** El calendario de la escuela tiene muchos eventos. Por ejemplo, acción de Gracias en el mes de septiembre y conciertos todos los meses.

**Clases:**

* **Expresión corporal y danza:** Para los niños de 3 a 6 años. En esta asignatura, el alumno tiene una introducción a los diferentes tipos de danzas y de coreografías
* **Instrumento:** La asignatura de instrumento se desarrolla en pequeños grupos con 2 o 3 estudiantes y el padre o madre de cada niño. En esta asignatura, el alumno puede aprender la práctica del instrumento que ha elegido en su matrícula.
* **Lenguaje musical:** Esta asignatura trata el fundamento de la música: como aprender a leer las partituras musicales, cómo funcionan los instrumentos, etc.

**Encuentros semanales con las familias:** Todas las semanas, los padres de los niños y los profesores tienen encuentros lo que permite conocer más a las otras personas del proyecto.

**Inversión económica:** La tasa de matrícula es de 25€ y las cuotas mensuales serán de 35€ por un hijo, 40€ por dos y 45€ por tres hijos.

**Matrícula**: Para matricularse, el niño (con sus padres) debe rellenar un formulario con los datos del alumno, los datos de los padres, aceptar la filosofía del proyecto “EVM” y elegir el instrumento que quiere aprender.

**Proyecto “EVM”:** Para que un niño pueda participar en el proyecto “EVM”, tiene que tener al menos 3 años. Además, el padre o madre debería estar regularmente presentes en las clases hasta que el niño cumpla los 13 años. El objetivo de este proyecto no es solamente enseñar música al niño sino ayudarle a desarrollar sus capacidades cognitivas, sociales y enseñarle también sobre la cultura. Todo esto es posible gracias a la cooperación entre el niño, los padres y los profesores.

**Responsable:** El responsable del alumno, o responsable, es el padre, madre o tutor que se responsabiliza de que el niño que participa en este proyecto tenga una continuidad en sus estudios. Además deberá acompañarle a algunas clases si el alumno está en el rango de edad comentado anteriormente y también deberá asistir a conciertos y reuniones del proyecto. Puede darse el caso excepcionalmente de que el alumno sea su propio responsable si es mayor.

**Tiempo del estudio:** Se trata de un periodo de 4 años consecutivos, dejando preparado estudiante para el acceso al Conservatorio u otro curso de música de carácter oficial.

**Usuario:** Un usuario, o alumno, es en nuestro contexto tanto un niño que quiere apuntarse en la escuela de música, como aquellos que ya están apuntados a ella.

# 3 – Modelo de negocio

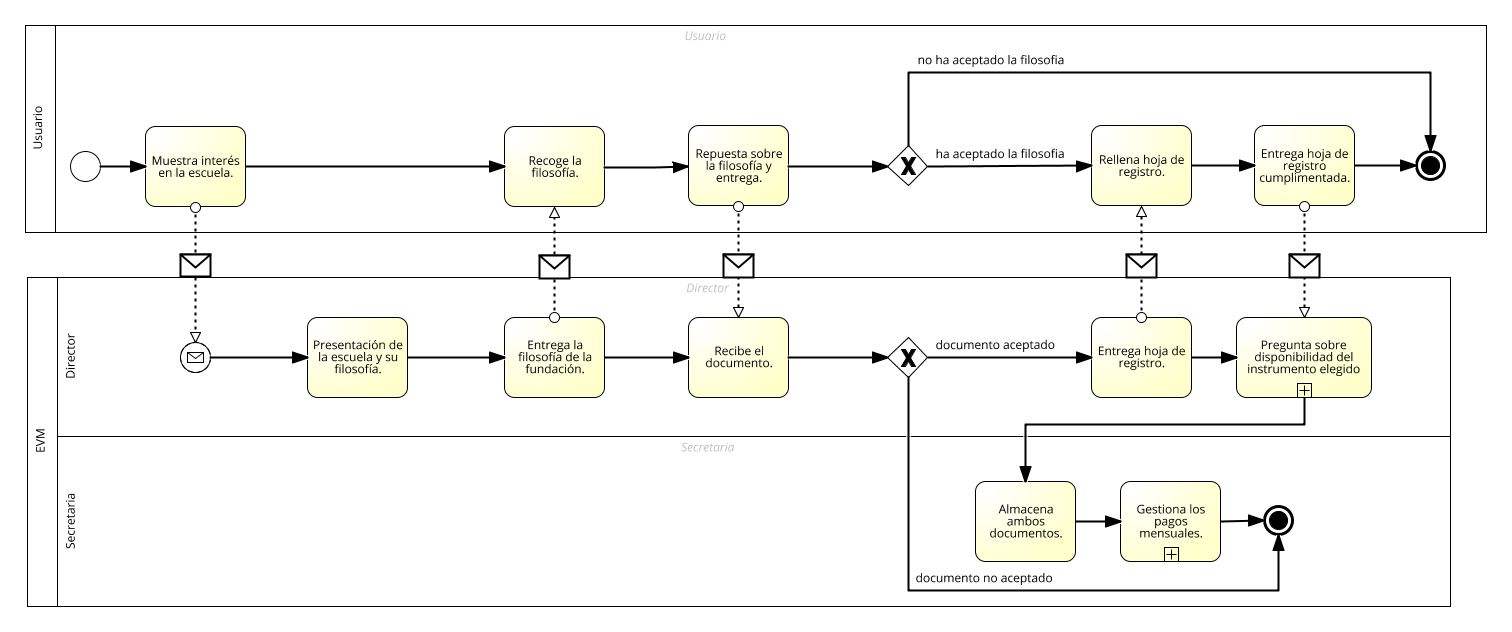
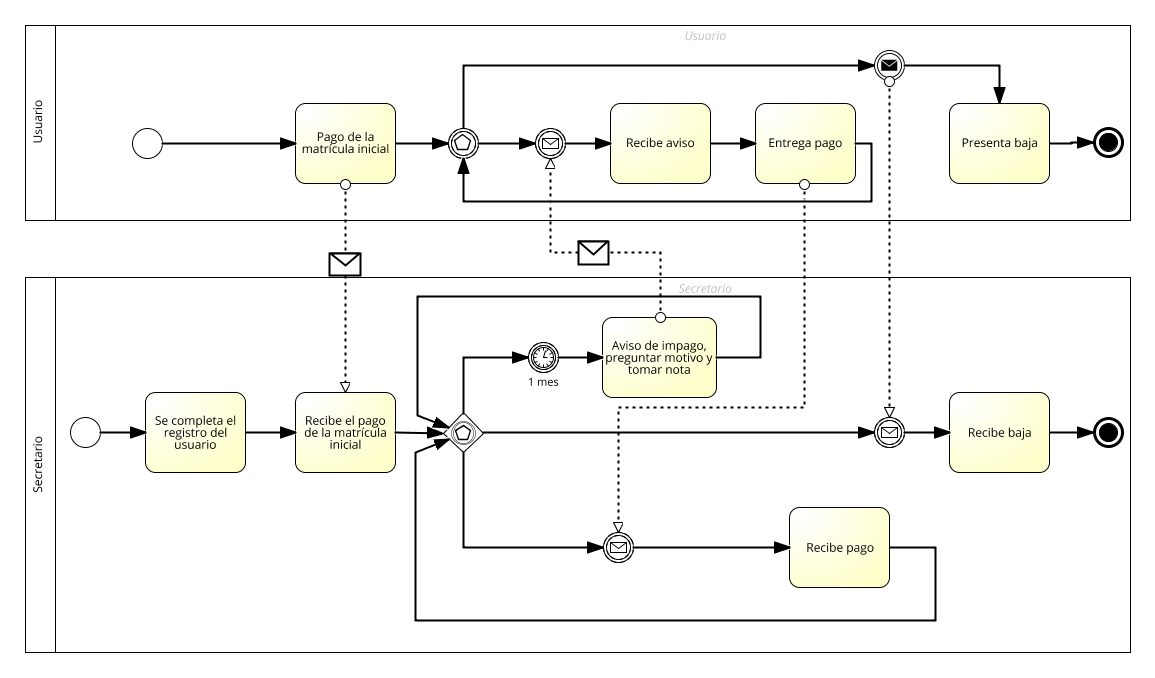


Imagen : BPMN registro

 Imagen : BPMN pagos

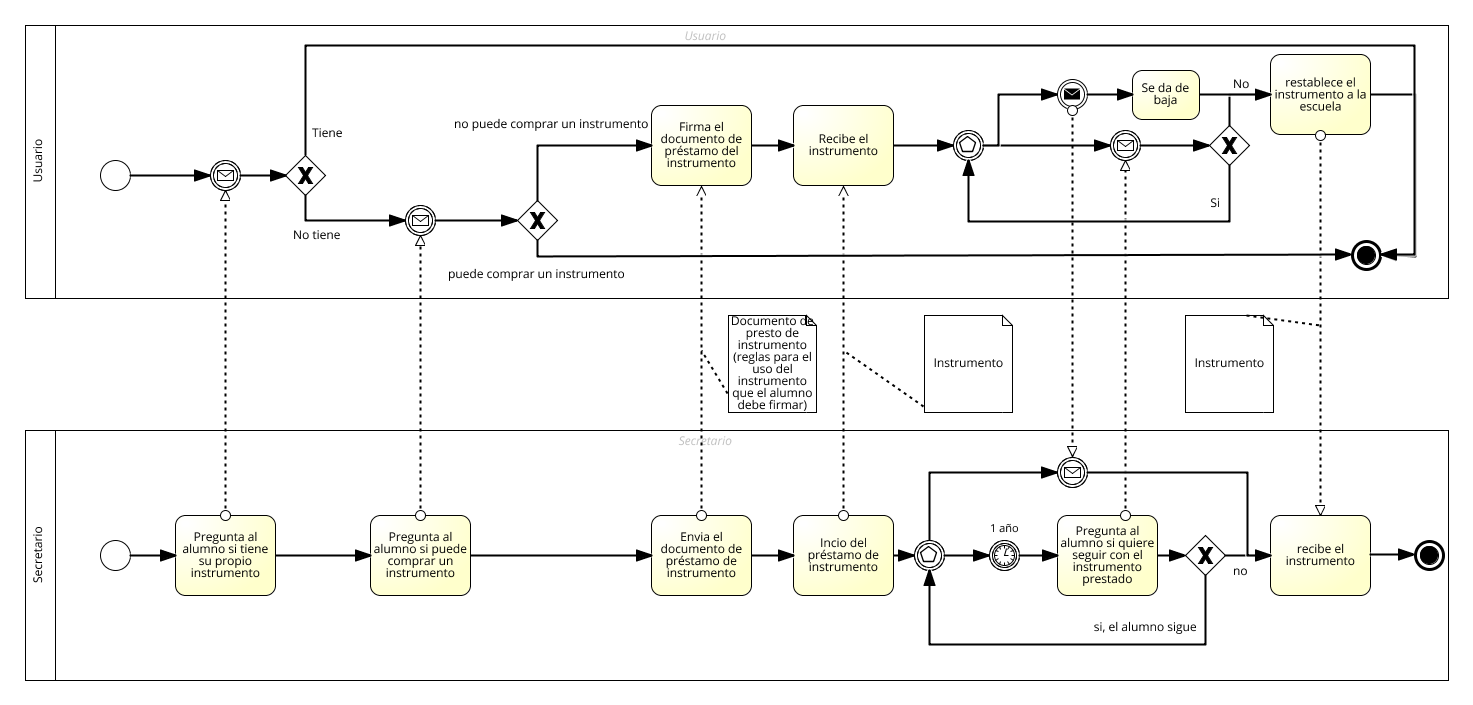


Imagen : BPMN préstamos

**Descripciones de los BPMN**

**- BPMN de registros:**

El BPMN de registros empieza cuando un usuario o su responsable muestran interés en la escuela y este va a tener una entrevista con el director de la misma. El director habla con él y le presenta la filosofía y una vista general del proyecto. Le entrega una copia de la filosofía, la cual tiene que aceptar y firmar el responsable antes de devolvérsela si el usuario quiere matricularse. En caso afirmativo se le entrega la hoja de registro, la cual rellena y firma el responsable con sus datos y los del usuario. Es entregada al secretario el cual se encarga de archivarla y, mientras el usuario siga en la escuela el secretario se encargará de gestionar sus cuotas (visto en el siguiente BPMN).

**- BPMN de pagos:**

En este BPMN podemos ver el proceso de la realización de pagos. El punto de comienzo se encuentra en el BPMN anterior en el que el usuario se registra. Una vez registrado, el secretario estará pendiente de recibir el pago de la matrícula inicial y después de recibirlo controlará los pagos mensuales por parte de este usuario. Si algún mes no recibe el pago se pondrá en contacto para saber el motivo, anotándolo en la tabla Excel y siendo este abonado por parte de la fundación si fuese necesario. Cuando recibe el pago del usuario también lo anota en dicha tabla e ingresa el pago. Por otro lado, si un usuario quiere darse de baja del proyecto, el secretario lo tiene en cuenta para dejar de controlar los pagos de dicho usuario, punto en el que acabaría nuestro BPMN.

**- BPMN de préstamos:**

Este BPMN muestra el proceso por el cual un usuario puede adquirir un instrumento prestado por parte del proyecto. El proceso comienza en el primer BPMN, una vez que el usuario se registra se le pregunta si tiene instrumento propio. En caso de que tuviera terminaría nuestro proceso. Si no es así, se le preguntará también si puede comprarlo, y si es así de nuevo termina nuestro proceso. En el caso de que no pueda comprar el instrumento en cuestión y de que haya un instrumento disponible, se le da un documento en el que están las condiciones de uso y que debe ser firmado por el usuario. Si acepta esas condiciones se le presta finalmente el instrumento, anotando la fecha de préstamo y que tiene validez por un año. Una vez finalizado el año deberá devolverlo o renovarlo.

# 4 – Visión general del sistema

**HU-1**. Como director de la escuela de música,

quiero tener un sistema que regule la administración de los alumnos

para agilizar la tramitación de estos.

**HU-2.** Como director de la escuela,

quiero un sistema que regule los pagos de las cuotas

para llevar un control mejor de las cuentas.

**HU-3.** Como director de la escuela,

quiero un sistema que regule el préstamo de instrumentos a los alumnos

para tener un control de estos instrumentos.

# 5 – Catálogo de requisitos

## Requisitos de información

**RI-01 Listado de usuarios**

Como director de la escuela

quiero disponer de esta información sobre los usuarios: nombre, apellidos, fecha de nacimiento, dirección, e-mail, teléfono de contacto y su conformidad a los derechos de imagen.

**RI-02 Responsables de menores**

Como director de la escuela

quiero disponer de esta información sobre los responsables de los usuarios: nombre, apellidos, teléfono de contacto y e-mail.

**RI-03 Instrumentos de préstamo**

Como director de la escuela

quiero disponer de esta información sobre los instrumentos: tipo de instrumento, si está disponible para préstamo, nombre y estado (nuevo, usado o deteriorado).

**RI-04 Pagos**

Como director de la escuela

quiero disponer de esta información sobre los pagos: fecha, cantidad a pagar, concepto y estado (pagado o pendiente).

**RI-05 Control de faltas**

Como director de la escuela

quiero disponer de esta información sobre las faltas: tipo de falta (retraso en el pago o falta de asistencia), fecha y saber si está justificada.

**RI-06 Matrículas**

Como director de la escuela

quiero disponer de esta información sobre las matrículas: fecha de matriculación, curso en el que se matricula y código de matrícula.

**RI-07 Listado de asignaturas**

Como director de la escuela

quiero disponer como información de las asignaturas su nombre correspondiente.

**RI-08 Préstamos**

Como director de la escuela

quiero disponer como información de los préstamos la fecha en la que se ha realizado.

**RI-09 Relaciones usuario-responsable**

Como director de la escuela

quiero disponer del tipo de relación (familiar o legal) que hay entre los usuarios y sus responsables.

## Reglas de negocio

**RN-01 Edad de integrantes**

Como director de la escuela

quiero que se cumpla la siguiente regla de negocio: para que alguien pueda registrarse al sistema debe de tener al menos 3 años.

**RN-02 Instrumento complementario**

Como director de la escuela

quiero que se cumpla la siguiente regla de negocio: a partir del tercer año de estudios será obligatorio el estudio de un instrumento complementario (guitarra o piano) para todos los usuarios.

**RN-03 Clases obligatorias para menores:**

Como director de la escuela

quiero que se cumpla la siguiente regla de negocio: los usuarios entre 3 y 6 años deberán asistir a clases de Expresión Corporal y Danza en horarios comunes.

## Requisitos funcionales

**RF-01 Consulta de instrumentos**

Como director

quiero tener un acceso a los instrumentos disponibles según su categoría

para saber cuáles están libres.

**RF-02 Relaciones entre director y responsable**

Como director

quiero poder obtener toda la información sobre el responsable de cualquier usuario

para poder hablar con este cuando sea necesario.

**RF-03 Faltas de asistencia**

Como director

quiero que un usuario reciba una falta cuando falte a alguna clase

para tener un control sobre la asistencia a clase.

**RF-04 Acumulación de faltas**

Como director

quiero recibir un aviso cuando un usuario tenga al menos 5 faltas sin justificar, y mandarle una notificación al responsable

para tener una reunión con el responsable.

**RF-05 Acompañantes para menores**

Como director de la escuela

quiero que si un usuario menor de 15 años va a clase sin estar acompañado por su responsable reciba una falta

para tener registradas las faltas.

**RF-06** **Faltas de pagos**

Como director

quiero que cuando un usuario se retrase con algún pago reciba una falta

para tener un control sobre los pagos.

**RF-07** **Lista de matrículas actuales**

Como director

quiero tener acceso a una lista de todas las matrículas en vigor

para tener un control de las matrículas de este curso.

**RF-08** **Lista de matrículas por curso**

Como director

quiero tener acceso a una lista de todas las matrículas agrupadas por el curso al que pertenezcan

para ver la evolución de las matrículas por curso.

**RF-09** **Pagos para un usuario**

Como director

quiero tener acceso a una lista de todos los pagos correspondientes a un usuario

para obtener información sobre las tasas abonadas por el usuario.

**RF-10** **Préstamos de un instrumento**

Como director

quiero obtener la lista de usuarios a los que se les ha prestado un instrumento, así como la fecha del préstamo

para llevar un control del uso del instrumento.

**RF-11** **Asignaturas de la matrícula actual**

Como director

quiero tener acceso a las asignaturas de la matrícula en vigor de un usuario

para consultar dichos datos del usuario.

**RF-12** **Faltas de un usuario**

Como director

quiero tener acceso a una lista de las faltas que ha cometido un usuario en el curso actual

para consultar las faltas de un usuario.

## Requisitos no funcionales

**RNF-01** **Copia de seguridad**

Como director

quiero que se realice una copia de seguridad de los datos cada mes

para respaldar los datos.

**RNF-02** **Sistema óptimo**

Como director

quiero que el sistema responda siempre en menos de 10 segundos

para agilizar consultas.

# 6 – Pruebas de aceptación

## Pruebas de aceptación de los requisitos de información

**PRI-01**

* **PRI-01.1** Se registra un usuario, se pide una lista de usuarios y aparece en ella.
* **PRI-01.2** Se da de baja un usuario, se pide la lista de usuarios matriculados actualmente, y no aparece en ella.

**PRI-02**

* **PRI-02.1** Se registra un usuario y su responsable, se pide la lista de responsables y aparece en ella.
* **PRI-02.2** Un usuario se hace mayor de edad y se borra su responsable, se pide la lista de responsables y no aparece en ella.

**PRI-03**

* **PRI-03.1** Se añade un instrumento, se pide la lista de instrumentos y aparece en ella.
* **PRI-03.2** Se borra un instrumento, se pide la lista de instrumentos y no aparece en ella.

**PRI-04**

* **PRI-04.1** Se registra una nueva matrícula, se pide la lista pagos asociados a la matrícula y aparecen todos los pagos sin realizar.
* **PRI-04.2** Se realiza un pago, se pide la lista de pagos y aparece dicho pago con el estado Pagado.

**PRI-05**

* **PRI-05.1** Un usuario realiza una falta, se pide la lista de faltas asociadas a la matrícula actual del usuario y aparece en ella.
* **PRI-05.2** Se justifica una falta, se pide la lista de faltas y aparece justificada.

**PRI-06**

* **PRI-06.1** Se matricula un usuario por primera vez, se pide la lista de matrículas y aparece dicha matrícula.
* **PRI-06.2** Se matricula un usuario por segundo año, se pide la lista de matrículas y aparecen todas las matrículas asociadas a ese usuario.

**PRI-07**

* **PRI-07.1** Se añade una asignatura, se pide la lista de asignaturas y aparece en ella.
* **PRI-07.2** Se elimina una asignatura, se pide la lista de asignaturas y no aparece en ella.

**PRI-08**

* **PRI-08.1** Se realiza un nuevo préstamo, se pide la lista de préstamos y aparece en ella.
* **PRI-08.2** Se renueva un préstamo, se pide la lista de préstamos actuales y aparece el que tiene la fecha actualizada.

**PRI-09**

* **PRI-09.1** Se pide una lista de las relaciones entre responsables y usuarios, aparecen todas.

## Pruebas de aceptación de las reglas de negocio

**PRN-01**

* **PRN-01.1** Alguien intenta registrarse con menos de 3 años, no se logra registrar.
* **PRN-01.2** Alguien intenta registrarse con 3 años o más, se registra exitosamente.

**PRN-02**

* **PRN-02.1** Un usuario que está cursando su tercer año elige un instrumento complementario con el que trabajar, queda registrado correctamente.
* **PRN-02.2** Un usuario que está cursando su tercer año no elige un instrumento complementario con el que trabajar, se le asigna uno aleatoriamente.

**PRN-03**

* **PRN-03.1** Un usuario entre 3 y 6 años está matriculado en Expresión Corporal y Danza, no hay problema.
* **PRN-03.2** Un usuario entre 3 y 6 años está matriculado en Expresión Corporal y Danza, debe matricularse en dicha asignatura.

## **Pruebas de aceptación de los** requisitos funcionales

**PRF-01**

* **PRF-01.1** Se realiza la búsqueda de los instrumentos de cuerda libres, se obtiene la lista resultante.
* **PRF-01.2** Se realiza la búsqueda de los instrumentos de viento libres cuando no queda ninguno libre, se obtiene una lista vacía.

**PR****F-02**

* **PRF-02.1** El director quiere obtener la información del responsable de cierto usuario, realiza la búsqueda y la consigue.
* **PRF-02.2** El director quiere obtener la información del responsable de un usuario mayor de edad, no llega al resultado porque el usuario no tiene ningún resultado.

**PRF-03**

* **PRF-03.1** Un usuario no va a clases, recibe una falta.
* **PRF-03.2** Un usuario va a clases, el usuario no recibe faltas.

**PRF-04**

* **PRF-04.1** Un usuario tiene 5 faltas, el director llama a su responsable.
* **PRF-04.2** Un usuario tiene 3 faltas, el director no llama a su responsable.

**PRF-05**

* **PRF-05.1** Un usuario con menos de 15 años va a clase sin su responsable, se le registra una falta.
* **PRF-05.2** Un usuario con menos de 15 años va a clase con su responsable, no se registra falta.

**PRF-06**

* **PRF-06.1** Un usuario se retrasa con un pago, recibe una falta.
* **PRF-06.2** Un usuario no se retrasa con un pago, no recibe una falta.

**PRF-07**

* **PRF-07.1** El director pide una lista de las matrículas en vigor, la obtiene.
* **PRF-07.2** Una matrícula pasa a ser del curso pasado, cuando el director pide la lista, no aparece.

**PRF-08**

* **PRF-08.1** El director pide una lista de las matrículas por curso, la obtiene.
* **PRF-08.2** Se añade una nueva matrícula a un curso, y aparece cuando se pide la lista por cursos.

**PRF-09**

* **PRF-09.1** El director pide la lista de pagos de un usuario, la obtiene.
* **PRF-09.2** Un usuario realiza un pago, aparece en la lista actualizado.

**PRF-10**

* **PRF-10.1** El director pide la lista de los usuarios a los que s eles ha prestado un instrumento, la obtiene.
* **PRF-10.2** Un instrumento libre es prestado a un usuario, el préstamo correspondiente aparece en la lista de ese instrumento.

**PRF-11**

* **PRF-11.1** El director pide la lista de asignaturas y el grupo de un usuario, la obtiene.
* **PRF-11.2** Un usuario se cambia de grupo, aparece el cambio reflejado en la lista.

**PRF-12**

* **PRF-12.1** El director pide la lista de faltas de un usuario, la obtiene.
* **PRF-12.2** Un usuario comete una falta, aparece en la lista.

## Pruebas de aceptación de los requisitos no funcionales

**PRNF-01**

* **PRNF-01.1** Pasa un mes y se crea una copia de seguridad.
* **PRNF-01.1** Pasan dos meses y se actualiza la copia de seguridad.

**PRNF-02**

* **PRNF-02.1** Se realiza una consulta, el sistema responde en 5 segundos.

# 

# 7 – Modelo conceptual

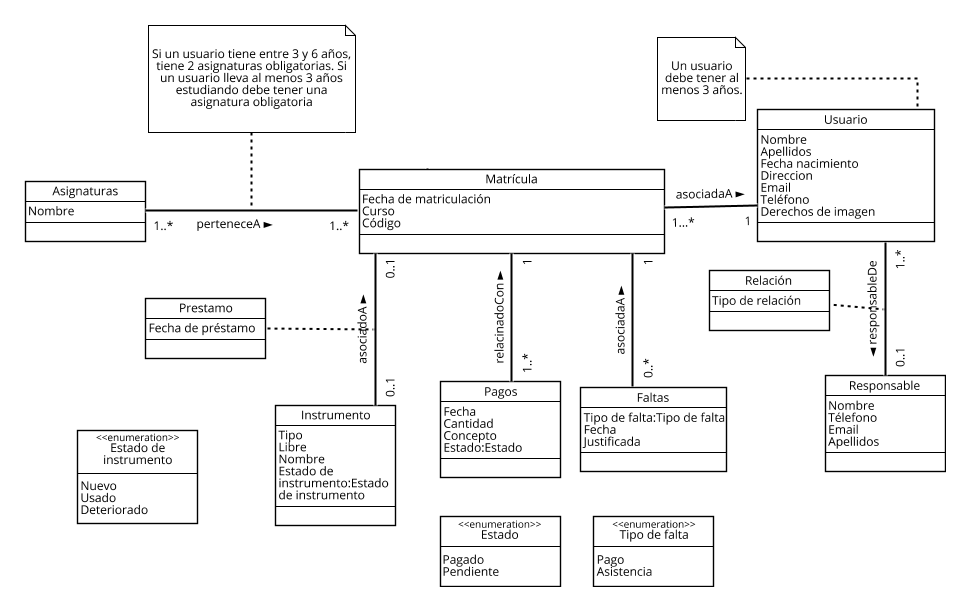


Imagen : UML

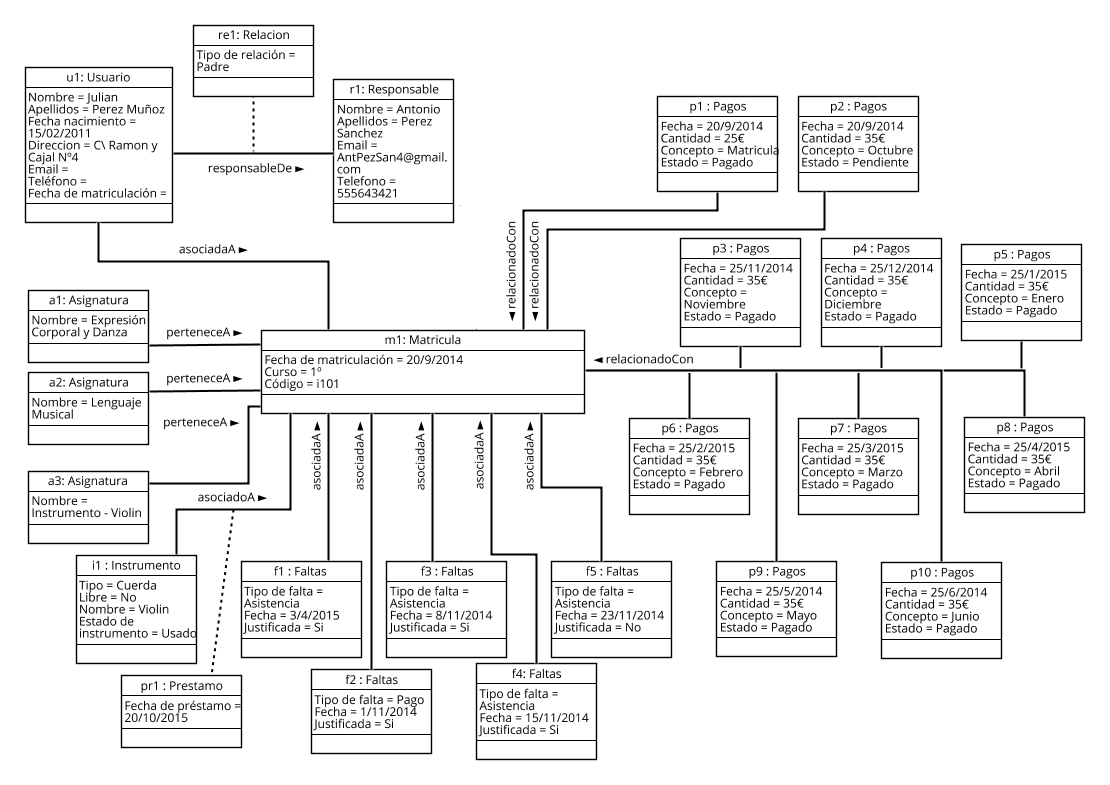


Imagen : Escenario de pruebas 1

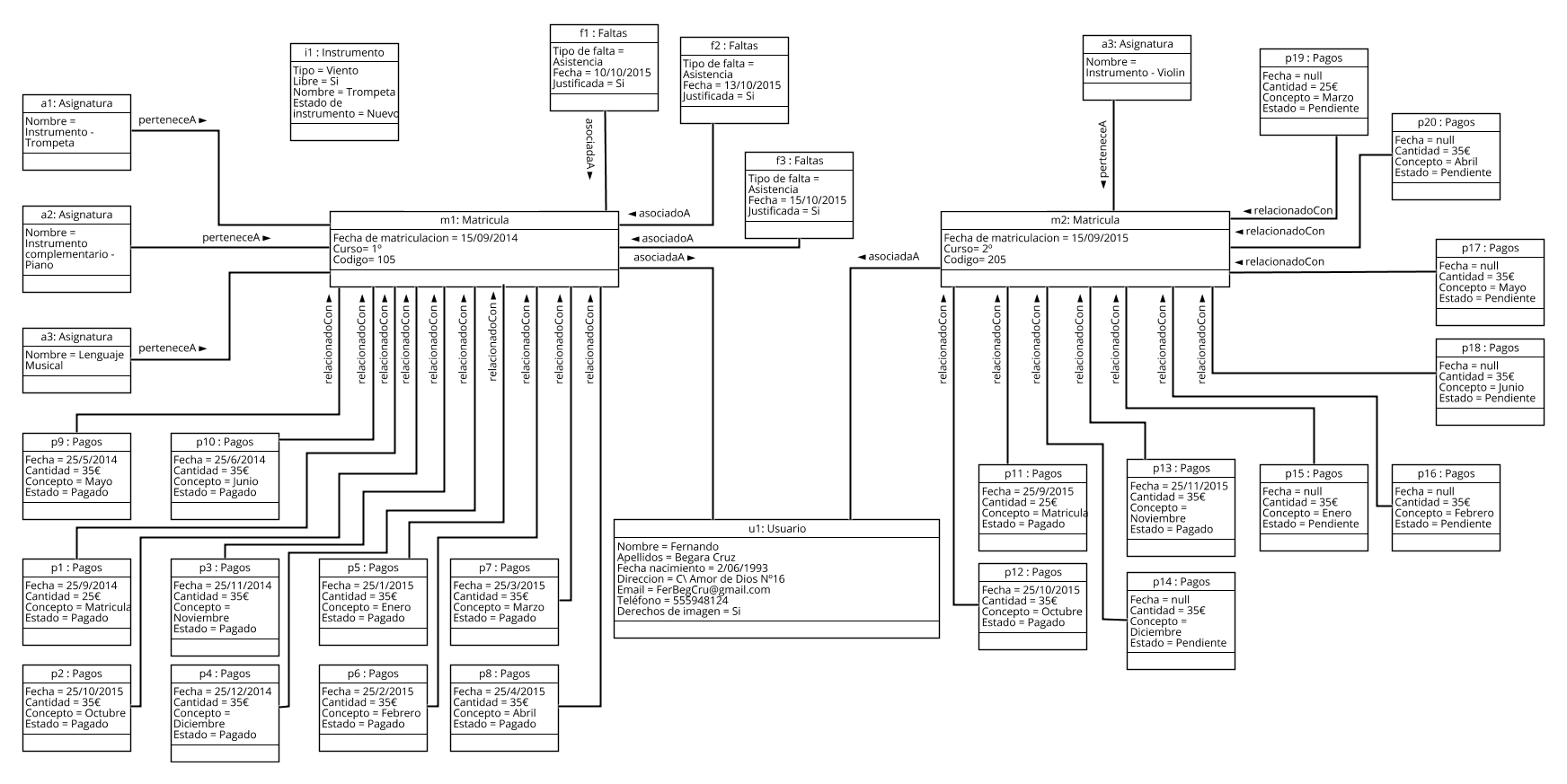


Imagen : Escenario de pruebas 2

# 8 – Matrices de trazabilidad

## Pruebas de aceptación/Requisitos

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Requisitos de información** | | | | | | | | |
|  | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** |
| **PRI-01.1** | 1 |  |  |  |  |  |  |  |  |
| **PRI-01.2** | 1 |  |  |  |  |  |  |  |  |
| **PRI-02.1** |  | 1 |  |  |  |  |  |  |  |
| **PRI-02.2** |  | 1 |  |  |  |  |  |  |  |
| **PRI-03.1** |  |  | 1 |  |  |  |  |  |  |
| **PRI-03.2** |  |  | 1 |  |  |  |  |  |  |
| **PRI-04.1** |  |  |  | 1 |  |  |  |  |  |
| **PRI-04.2** |  |  |  | 1 |  |  |  |  |  |
| **PRI-05.1** |  |  |  |  | 1 |  |  |  |  |
| **PRI-05.2** |  |  |  |  | 1 |  |  |  |  |
| **PRI-06.1** |  |  |  |  |  | 1 |  |  |  |
| **PRI-06.2** |  |  |  |  |  | 1 |  |  |  |
| **PRI-07.1** |  |  |  |  |  |  | 1 |  |  |
| **PRI-07.2** |  |  |  |  |  |  | 1 |  |  |
| **PRI-08.1** |  |  |  |  |  |  |  | 1 |  |
| **PRI-08.2** |  |  |  |  |  |  |  | 1 |  |
| **PRI-09.1** |  |  |  |  |  |  |  |  | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Reglas de Negocio** | | | **Requisitos no funcionales** | |
|  | **01** | **02** | **03** | **01** | **02** |
| **PRN-01.1** | 1 |  |  |  |  |
| **PRN-01.2** | 1 |  |  |  |  |
| **PRN-02.1** |  | 1 |  |  |  |
| **PRN-02.2** |  | 1 |  |  |  |
| **PRN-03.1** |  |  | 1 |  |  |
| **PRN-03.2** |  |  | 1 |  |  |
| **PRNF-01.1** |  |  |  | 1 |  |
| **PRNF-01.2** |  |  |  | 1 |  |
| **PRNF-02.1** |  |  |  |  | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Requisitos funcionales** | | | | | | | | | | | |
|  | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | **10** | **11** | **12** |
| **PRF-01.1** | 1 |  |  |  |  |  |  |  |  |  |  |  |
| **PRF-01.2** | 1 |  |  |  |  |  |  |  |  |  |  |  |
| **PRF-02.1** |  | 1 |  |  |  |  |  |  |  |  |  |  |
| **PRF-02.2** |  | 1 |  |  |  |  |  |  |  |  |  |  |
| **PRF-03.1** |  |  | 1 |  |  |  |  |  |  |  |  |  |
| **PRF-03.2** |  |  | 1 |  |  |  |  |  |  |  |  |  |
| **PRF-04.1** |  |  |  | 1 |  |  |  |  |  |  |  |  |
| **PRF-04.2** |  |  |  | 1 |  |  |  |  |  |  |  |  |
| **PRF-05.1** |  |  |  |  | 1 |  |  |  |  |  |  |  |
| **PRF-05.2** |  |  |  |  | 1 |  |  |  |  |  |  |  |
| **PRF-06.1** |  |  |  |  |  | 1 |  |  |  |  |  |  |
| **PRF-06.2** |  |  |  |  |  | 1 |  |  |  |  |  |  |
| **PRF-07.1** |  |  |  |  |  |  | 1 |  |  |  |  |  |
| **PRF-07.2** |  |  |  |  |  |  | 1 |  |  |  |  |  |
| **PRF-08.1** |  |  |  |  |  |  |  | 1 |  |  |  |  |
| **PRF-08.2** |  |  |  |  |  |  |  | 1 |  |  |  |  |
| **PRF-09.1** |  |  |  |  |  |  |  |  | 1 |  |  |  |
| **PRF-09.2** |  |  |  |  |  |  |  |  | 1 |  |  |  |
| **PRF-10.1** |  |  |  |  |  |  |  |  |  | 1 |  |  |
| **PRF-10.2** |  |  |  |  |  |  |  |  |  | 1 |  |  |
| **PRF-11.1** |  |  |  |  |  |  |  |  |  |  | 1 |  |
| **PRF-11.2** |  |  |  |  |  |  |  |  |  |  | 1 |  |
| **PRF-12.1** |  |  |  |  |  |  |  |  |  |  |  | 1 |
| **PRF-12.2** |  |  |  |  |  |  |  |  |  |  |  | 1 |

## 

## Requisitos/UML

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Usuario** | **Responsable** | **Instrumentos** | **Pagos** | **Faltas** | **Matrículas** | **Asignaturas** | **Préstamos** | **Relaciones** |
| **RI-01** | 1 |  |  |  |  |  |  |  |  |
| **RI-02** |  | 1 |  |  |  |  |  |  |  |
| **RI-03** |  |  | 1 |  |  |  |  |  |  |
| **RI-04** |  |  |  | 1 |  |  |  |  |  |
| **RI-05** |  |  |  |  | 1 |  |  |  |  |
| **RI-06** |  |  |  |  |  | 1 |  |  |  |
| **RI-07** |  |  |  |  |  |  | 1 |  |  |
| **RI-08** |  |  | 1 |  |  | 1 |  | 1 |  |
| **RI-09** | 1 | 1 |  |  |  |  |  |  | 1 |
| **RN-01** | 1 |  |  |  |  |  |  |  |  |
| **RN-02** |  |  |  |  |  | 1 | 1 |  |  |
| **RN-03** | 1 |  |  |  |  |  | 1 |  |  |
| **RF-01** |  |  | 1 |  |  |  |  |  |  |
| **RF-02** | 1 | 1 |  |  |  |  |  |  | 1 |
| **RF-03** | 1 |  |  |  | 1 | 1 |  |  |  |
| **RF-04** | 1 | 1 |  |  | 1 | 1 |  |  |  |
| **RF-05** | 1 |  |  |  | 1 | 1 |  |  |  |
| **RF-06** | 1 |  |  | 1 | 1 | 1 |  |  |  |
| **RF-07** |  |  |  |  |  | 1 |  |  |  |
| **RF-08** |  |  |  |  |  | 1 |  |  |  |
| **RF-09** | 1 |  |  | 1 |  | 1 |  |  |  |
| **RF-10** | 1 |  | 1 |  |  | 1 |  | 1 |  |
| **RF-11** | 1 |  |  |  |  | 1 | 1 |  |  |
| **RF-12** | 1 |  |  |  | 1 | 1 |  |  |  |
| **RNF-01** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **RNF-02** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## Pruebas de aceptación/Escenarios de prueba

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Escenario de pruebas 1** | **Escenario de pruebas 2** |
| **PRI-01.1** | | 1 | 1 |
| **PRI-01.2** | | 1 | 1 |
| **PRI-02.1** | | 1 |  |
| **PRI-02.2** | |  | 1 |
| **PRI-03.1** | | 1 | 1 |
| **PRI-03.2** | | 1 | 1 |
| **PRI-04.1** | | 1 | 1 |
| **PRI-04.2** | | 1 | 1 |
| **PRI-05.1** | | 1 | 1 |
| **PRI-05.2** | | 1 | 1 |
| **PRI-06.1** | | 1 | 1 |
| **PRI-06.2** | |  | 1 |
| **PRI-07.1** | | 1 | 1 |
| **PRI-07.2** | | 1 | 1 |
| **PRI-08.1** | | 1 |  |
| **PRI-08.2** | | 1 |  |
| **PRI.09.1** | | 1 |  |
| **PRF-01.1** | |  | 1 |
| **PRF-01.2** | | 1 | 1 |
| **PRF-02.1** | | 1 |  |
| **PRF-02.2** | |  | 1 |
| **PRF-03.1** | | 1 | 1 |
| **PRF-03.2** | | 1 | 1 |
| **PRF-04.1** | | 1 |  |
| **PRF-04.2** | |  | 1 |
| **PRF-05.1** | | 1 |  |
| **PRF-05.2** | | 1 |  |
| **PRF-06.1** | | 1 |  |
| **PRF-06.2** | |  | 1 |
| **PRF-07.1** | | 1 | 1 |
| **PRF-07.2** | |  | 1 |
| **PRF-08.1** | | 1 | 1 |
| **PRF-08.2** | | 1 | 1 |
| **PRF-09.1** | | 1 | 1 |
| **PRF-09.2** | | 1 | 1 |
| **PRF-10.1** | | 1 |  |
| **PRF-10.2** | | 1 |  |
| **PRF-11.1** | | 1 | 1 |
| **PRF-11.2** | | 1 | 1 |
| **PRF-12.1** | | 1 | 1 |
| **PRF-12.2** | | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Escenario de pruebas 1** | **Escenario de pruebas 2** |
| **PRN-01.1** | | 1 | 1 |
| **PRN-01.2** | | 1 | 1 |
| **PRN-02.1** | | 1 | 1 |
| **PRN-02.2** | | 1 | 1 |
| **PRN-03.1** | | 1 | 1 |
| **PRN-03.2** | | 1 | 1 |
| **PRNF-01.1** | | 1 | 1 |
| **PRNF-01.2** | | 1 | 1 |
| **PRNF-02.1** | | 1 | 1 |

# 9- Modelo Relacional

|  |  |  |
| --- | --- | --- |
| Table | PK | FK |
| **USUARIOS**  (OID\_U, NOMBRE, APELLIDOS, FECHA\_NACIMIENTO, DIRECCION, EMAIL, TELEFONO, DERECHO\_IMAGEN) | OID\_U |  |
| **RESPONSABLES**  (OID\_R, NOMBRE, APELLIDOS, TELEFONO, EMAIL) | OID\_R |  |
| **RELACIONES**  (OID\_REL, OID\_U, OID\_R, TIPO\_RELACION) | OID\_REL | OID\_U,  OID\_R |
| **MATRICULAS**  (OID\_M, FECHA\_MATRICULACION, CURSO, CODIGO, OID\_U) | OID\_M | OID\_U |
| **FALTAS**  (OID\_F, TIPO\_FALTA, FECHA\_FALTA, JUSTIFICADA, OID\_M) | OID\_F | OID\_M |
| **PAGOS**  (OID\_PA, FECHA\_PAGO, CANTIDAD, CONCEPTO, ESTADO, OID\_M) | OID\_PA | OID\_M |
| **INSTRUMENTOS**  (OID\_I, TIPO, LIBRE, NOMBRE, ESTADO\_INSTRUMENTO) | OID\_I | OID\_I |
| **PRESTAMOS**  (OID\_P, FECHA\_PRESTAMO, OID\_M, OID\_I) | OID\_P | OID\_M,  OID\_I |
| **ASIGNATURAS**  (OID\_A, NOMBRE) | OID\_A |  |
| **PERTENECE\_A**  (OID\_PE, OID\_M, OID\_A) | OID\_PE | OID\_M,  OID\_A |

# 10 – Modelo Tecnológico

## Script de creación de tablas, restricciones, secuencias, triggers asociados a la gestión de secuencias e índices.

DROP TABLE PERTENECE\_A;

DROP TABLE ASIGNATURAS;

DROP TABLE PRESTAMOS;

DROP TABLE INSTRUMENTOS;

DROP TABLE PAGOS;

DROP TABLE FALTAS;

DROP TABLE MATRICULAS;

DROP TABLE RELACIONES;

DROP TABLE RESPONSABLES;

DROP TABLE USUARIOS;

CREATE TABLE USUARIOS

(

OID\_U NUMBER(3),

NOMBRE VARCHAR2(50) NOT NULL,

APELLIDOS VARCHAR2(50) NOT NULL,

FECHA\_NACIMIENTO DATE NOT NULL,

DIRECCION VARCHAR2(60) NOT NULL,

EMAIL VARCHAR2(60),

TELEFONO NUMBER(9),

DERECHOS\_IMAGEN NUMBER(1) CHECK(DERECHOS\_IMAGEN=1 OR DERECHOS\_IMAGEN=0) NOT NULL,

PRIMARY KEY (OID\_U)

);

CREATE TABLE RESPONSABLES

(

OID\_R NUMBER(3),

NOMBRE VARCHAR2(50) NOT NULL,

APELLIDOS VARCHAR2(50) NOT NULL,

EMAIL VARCHAR2(60),

TELEFONO NUMBER(9) NOT NULL,

PRIMARY KEY (OID\_R)

);

CREATE TABLE RELACIONES

(

OID\_REL NUMBER(3),

OID\_U NUMBER(3),

OID\_R NUMBER(3),

TIPO\_RELACION VARCHAR2(60) NOT NULL,

PRIMARY KEY (OID\_REL),

FOREIGN KEY (OID\_U) REFERENCES USUARIOS,

FOREIGN KEY (OID\_R) REFERENCES RESPONSABLES

);

CREATE TABLE MATRICULAS

(

OID\_M NUMBER(3),

FECHA\_MATRICULACION DATE,

CURSO INT NOT NULL,

CODIGO VARCHAR2(5) NOT NULL,

OID\_U NUMBER(3),

PRIMARY KEY (OID\_M),

FOREIGN KEY (OID\_U) REFERENCES USUARIOS

);

CREATE TABLE FALTAS

(

OID\_F NUMBER(3),

TIPO\_FALTA VARCHAR2(10) CHECK(TIPO\_FALTA='Pago' OR TIPO\_FALTA='Asistencia'),

FECHA\_FALTA DATE,

JUSTIFICADA NUMBER(1) CHECK(JUSTIFICADA=0 OR JUSTIFICADA=1),

OID\_M NUMBER(3),

PRIMARY KEY (OID\_F),

FOREIGN KEY (OID\_M) REFERENCES MATRICULAS

);

CREATE TABLE PAGOS

(

OID\_PA NUMBER(3),

FECHA\_PAGO DATE,

CANTIDAD NUMBER(10) NOT NULL,

CONCEPTO CHAR(10) NOT NULL,

ESTADO VARCHAR2(9) CHECK(ESTADO='Pagado' OR Estado='Pendiente'),

OID\_M NUMBER(3),

PRIMARY KEY (OID\_PA),

FOREIGN KEY (OID\_M) REFERENCES MATRICULAS

);

CREATE TABLE INSTRUMENTOS

(

OID\_I NUMBER(3),

TIPO VARCHAR2(50),

LIBRE NUMBER(1) CHECK(LIBRE=1 OR LIBRE=0),

NOMBRE VARCHAR2(50),

ESTADO\_INSTRUMENTO VARCHAR2(11) CHECK(ESTADO\_INSTRUMENTO='Nuevo' OR ESTADO\_INSTRUMENTO='Usado' OR ESTADO\_INSTRUMENTO='Deteriorado'),

PRIMARY KEY (OID\_I)

);

CREATE TABLE PRESTAMOS

(

OID\_P NUMBER(3),

FECHA\_PRESTAMO DATE,

OID\_M NUMBER(3),

OID\_I NUMBER(3),

PRIMARY KEY (OID\_P),

FOREIGN KEY (OID\_M) REFERENCES MATRICULAS,

FOREIGN KEY (OID\_I) REFERENCES INSTRUMENTOS

);

CREATE TABLE ASIGNATURAS

(

OID\_A NUMBER(3),

NOMBRE VARCHAR2(60) NOT NULL,

PRIMARY KEY (OID\_A)

);

CREATE TABLE PERTENECE\_A

(

OID\_PE NUMBER(5),

OID\_M NUMBER(3),

OID\_A NUMBER(3),

PRIMARY KEY (OID\_PE),

FOREIGN KEY (OID\_M) REFERENCES MATRICULAS,

FOREIGN KEY (OID\_A) REFERENCES ASIGNATURAS

);

DROP SEQUENCE SEC\_U;

CREATE SEQUENCE SEC\_U INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_U

BEFORE INSERT ON USUARIOS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_R;

CREATE SEQUENCE SEC\_R INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_R

BEFORE INSERT ON RESPONSABLES

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_REL;

CREATE SEQUENCE SEC\_REL INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_REL

BEFORE INSERT ON RELACIONES

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_M;

CREATE SEQUENCE SEC\_M INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_M

BEFORE INSERT ON MATRICULAS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_F;

CREATE SEQUENCE SEC\_F INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_F

BEFORE INSERT ON FALTAS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_PA;

CREATE SEQUENCE SEC\_PA INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_PA

BEFORE INSERT ON PAGOS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_I;

CREATE SEQUENCE SEC\_I INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_I

BEFORE INSERT ON INSTRUMENTOS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_P;

CREATE SEQUENCE SEC\_P INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_P

BEFORE INSERT ON PRESTAMOS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_A;

CREATE SEQUENCE SEC\_A INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_A

BEFORE INSERT ON ASIGNATURAS

FOR EACH ROW

BEGIN

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

END;

/

DROP SEQUENCE SEC\_PE;

CREATE SEQUENCE SEC\_PE INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER TRI\_PE

BEFORE INSERT ON PERTENECE\_A

FOR EACH ROW

BEGIN

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

END;

/

## Script de creación de funciones y procedimientos

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PROCEDIMIENTOS DE INSERCION

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE OR REPLACE PROCEDURE crear\_usuario

(nombre2 IN usuarios.nombre%TYPE,

apellidos2 IN usuarios.apellidos%TYPE,

fecha\_nacimiento2 IN usuarios.fecha\_nacimiento%TYPE,

direccion2 IN usuarios.direccion%TYPE,

email2 IN usuarios.email%TYPE,

telefono2 IN usuarios.telefono%TYPE,

derechos\_imagen2 IN usuarios.derechos\_imagen%TYPE) IS

BEGIN

INSERT INTO usuarios (nombre,apellidos,fecha\_nacimiento,direccion,email,telefono,derechos\_imagen)

VALUES (nombre2,apellidos2,fecha\_nacimiento2,direccion2,email2,telefono2,derechos\_imagen2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_responsable

(nombre2 IN responsables.nombre%TYPE,

apellidos2 IN responsables.apellidos%TYPE,

email2 IN responsables.email%TYPE,

telefono2 IN responsables.telefono%TYPE) IS

BEGIN

INSERT INTO responsables (nombre,apellidos,email,telefono)

VALUES (nombre2,apellidos2,email2,telefono2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_relacion

(oid\_u2 IN relaciones.oid\_u%TYPE,

oid\_r2 IN relaciones.oid\_r%TYPE,

tipo\_relacion2 IN relaciones.tipo\_relacion%TYPE) IS

BEGIN

INSERT INTO relaciones (oid\_u,oid\_r,tipo\_relacion)

VALUES (oid\_u2,oid\_r2,tipo\_relacion2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_matricula

(fecha\_matriculacion2 IN matriculas.fecha\_matriculacion%TYPE,

curso2 IN matriculas.curso%TYPE,

codigo2 IN matriculas.codigo%TYPE,

oid\_u2 IN matriculas.oid\_u%TYPE) IS

BEGIN

INSERT INTO matriculas (fecha\_matriculacion,curso,codigo,oid\_u)

VALUES (fecha\_matriculacion2,curso2,codigo2,oid\_u2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_falta

(tipo\_falta2 IN faltas.tipo\_falta%TYPE,

fecha\_falta2 IN faltas.fecha\_falta%TYPE,

justificada2 IN faltas.justificada%TYPE,

oid\_m2 IN faltas.oid\_m%TYPE) IS

BEGIN

INSERT INTO faltas (tipo\_falta,fecha\_falta,justificada,oid\_m)

VALUES (tipo\_falta2,fecha\_falta2,justificada2,oid\_m2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_pago

(fecha\_pago2 IN pagos.fecha\_pago%TYPE,

cantidad2 IN pagos.cantidad%TYPE,

concepto2 IN pagos.concepto%TYPE,

estado2 IN pagos.estado%TYPE,

oid\_m2 IN pagos.oid\_m%TYPE) IS

BEGIN

INSERT INTO pagos (fecha\_pago,cantidad,concepto,estado,oid\_m)

VALUES (fecha\_pago2,cantidad2,concepto2,estado2,oid\_m2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_instrumento

(tipo2 IN instrumentos.tipo%TYPE,

libre2 IN instrumentos.libre%TYPE,

nombre2 IN instrumentos.nombre%TYPE,

estado\_instrumento2 IN instrumentos.estado\_instrumento%TYPE) IS

BEGIN

INSERT INTO instrumentos (tipo,libre,nombre,estado\_instrumento)

VALUES (tipo2,libre2,nombre2,estado\_instrumento2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_prestamo

(fecha\_prestamo2 IN prestamos.fecha\_prestamo%TYPE,

oid\_m2 IN prestamos.oid\_m%TYPE,

oid\_i2 IN prestamos.oid\_i%TYPE) IS

BEGIN

INSERT INTO prestamos (fecha\_prestamo,oid\_m,oid\_i)

VALUES (fecha\_prestamo2,oid\_m2,oid\_i2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_asignatura

(nombre2 IN asignaturas.nombre%TYPE) IS

BEGIN

INSERT INTO asignaturas (nombre)

VALUES (nombre2);

END;

/

CREATE OR REPLACE PROCEDURE crear\_pertenece\_a

(oid\_m2 IN pertenece\_a.oid\_m%TYPE,

oid\_a2 IN pertenece\_a.oid\_a%TYPE) IS

BEGIN

INSERT INTO pertenece\_a (oid\_m,oid\_a)

VALUES (oid\_m2,oid\_a2);

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PROCEDIMIENTOS Y FUNCIONES DE APOYO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- Devuelve el oid de una asignatura a partir de su nombre.

CREATE OR REPLACE FUNCTION buscar\_oid\_a

(nombre2 asignaturas.nombre%TYPE) RETURN asignaturas.oid\_a%TYPE AS

oid\_a2 asignaturas.oid\_a%TYPE;

BEGIN

SELECT oid\_a into oid\_a2 from asignaturas where nombre=nombre2;

return oid\_a2;

END;

/

-- Funcion para la comparación de las pruebas.

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;

/

-- Borra tabla matriculas y sus asociadas.

CREATE OR REPLACE PROCEDURE BORRAR\_MATRICULA\_CASCADA AS

BEGIN

DELETE FROM PERTENECE\_A;

DELETE FROM PRESTAMOS;

DELETE FROM PAGOS;

DELETE FROM FALTAS;

DELETE FROM MATRICULAS;

END BORRAR\_MATRICULA\_CASCADA;

/

-- Devuelve nombre y apellidos de usuario a partir del oid.

CREATE OR REPLACE FUNCTION nombre\_usuario(usuario usuarios.oid\_u%type)

RETURN VARCHAR2 AS

nombre2 usuarios.nombre%type;

apellidos2 usuarios.apellidos%type;

BEGIN

SELECT nombre into nombre2 from usuarios where oid\_u=usuario;

SELECT apellidos into apellidos2 from usuarios where oid\_u=usuario;

return nombre2 || ' ' || apellidos2;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PROCEDIMIENTOS Y FUNCIONES SOBRE REQUISITOS FUNCIONALES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- REQUISITO FUNCIONAL 01

CREATE OR REPLACE PROCEDURE INSTRUMENTOS\_LIBRES AS

CURSOR c IS

SELECT oid\_i, nombre, estado\_instrumento FROM instrumentos where libre=1;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Instrumentos libres');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE('ID ' || fila.oid\_i ||' - '|| fila.nombre ||' - '|| fila.estado\_instrumento);

END LOOP;

END INSTRUMENTOS\_LIBRES;

/

--REQUISITO FUNCIONAL 02

CREATE OR REPLACE PROCEDURE RESPONSABLE\_DEL\_USUARIO (usuario usuarios.oid\_u%type) AS

CURSOR c IS

SELECT OID\_R, NOMBRE, APELLIDOS, EMAIL, TELEFONO FROM RESPONSABLES NATURAL JOIN RELACIONES WHERE OID\_U=usuario;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Los datos del responsable del usuario ' || nombre\_usuario(usuario) || ' son:');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(fila.oid\_r ||' - '|| fila.nombre ||' - '|| fila.APELLIDOS || ' - '|| fila.EMAIL ||' - '|| fila.TELEFONO);

END LOOP;

END RESPONSABLE\_DEL\_USUARIO;

/

--REQUISITO FUNCIONAL 07

CREATE OR REPLACE PROCEDURE MATRICULAS\_EN\_VIGOR AS

CURSOR c IS

SELECT curso, codigo, oid\_u FROM MATRICULAS WHERE FECHA\_MATRICULACION>(SYSDATE - 365);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Matriculas en vigor:');

DBMS\_OUTPUT.PUT\_LINE('CODIGO - CURSO - ALUMNO');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(fila.codigo ||' - '|| fila.curso ||' - '|| NOMBRE\_USUARIO(fila.oid\_u));

END LOOP;

END MATRICULAS\_EN\_VIGOR;

/

--REQUISITO FUNCIONAL 08

CREATE OR REPLACE PROCEDURE MATRICULAS\_POR\_CURSO AS

CURSOR c IS

SELECT fecha\_matriculacion, codigo, curso, oid\_u FROM MATRICULAS ORDER BY CURSO;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Matriculas por curso:');

DBMS\_OUTPUT.PUT\_LINE('CODIGO - FECHA - CURSO - ALUMNO');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(fila.codigo ||' - '|| fila.fecha\_matriculacion ||' - '|| fila.curso ||' - '|| NOMBRE\_USUARIO(fila.oid\_u));

END LOOP;

END MATRICULAS\_POR\_CURSO;

/

--REQUISITO FUNCIONAL 09

CREATE OR REPLACE PROCEDURE PAGOS\_DEL\_USUARIO (usuario usuarios.oid\_u%type) AS

CURSOR c IS

SELECT FECHA\_PAGO, CANTIDAD, CONCEPTO, ESTADO FROM PAGOS NATURAL JOIN MATRICULAS where oid\_u=usuario;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pagos del usuario '|| nombre\_usuario(usuario) || ' (ID ' || usuario || ')');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(fila.FECHA\_PAGO ||' - '|| fila.CANTIDAD ||' - '|| fila.CONCEPTO ||' - '|| fila.ESTADO);

END LOOP;

END PAGOS\_DEL\_USUARIO;

/

--REQUISITO FUNCIONAL 10

CREATE OR REPLACE PROCEDURE USUARIOS\_CON\_PRESTAMOS AS

CURSOR c IS

SELECT OID\_U, FECHA\_PRESTAMO FROM PRESTAMOS NATURAL JOIN MATRICULAS NATURAL JOIN USUARIOS;

BEGIN

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE('Al usuario '|| nombre\_usuario(fila.oid\_u) ||

' con ID ' || fila.oid\_u || ' se le presto un instrumento el ' || fila.fecha\_prestamo);

END LOOP;

END USUARIOS\_CON\_PRESTAMOS;

/

--REQUISITO FUNCIONAL 11

CREATE OR REPLACE PROCEDURE ASIGNATURAS\_DEL\_USUARIO(usuario usuarios.oid\_u%type) AS

CURSOR c IS

SELECT NOMBRE FROM ASIGNATURAS NATURAL JOIN PERTENECE\_A NATURAL JOIN MATRICULAS WHERE OID\_U=usuario AND

FECHA\_MATRICULACION>(SYSDATE - 365);

begin

DBMS\_OUTPUT.PUT\_LINE('Asignaturas del usuario ' || nombre\_usuario(usuario) || ':');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(fila.nombre);

END LOOP;

END ASIGNATURAS\_DEL\_USUARIO;

/

--REQUISITO FUNCIONAL 12

CREATE OR REPLACE PROCEDURE FALTAS\_DEL\_USUARIO(usuario usuarios.oid\_u%type) AS

CURSOR c IS

SELECT TIPO\_FALTA, FECHA\_FALTA, JUSTIFICADA FROM FALTAS NATURAL JOIN MATRICULAS WHERE OID\_U=usuario AND

FECHA\_MATRICULACION>(SYSDATE - 365);

begin

DBMS\_OUTPUT.PUT\_LINE('Faltas del usuario ' || nombre\_usuario(usuario) || ':');

FOR fila IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(fila.TIPO\_FALTA || ' - ' || fila.FECHA\_FALTA || ' - ' || fila.JUSTIFICADA);

END LOOP;

END FALTAS\_DEL\_USUARIO;

/

## Script de creación de triggers no asociados a secuencias

CREATE OR REPLACE TRIGGER EDAD\_USUARIOS

BEFORE INSERT OR UPDATE ON USUARIOS

FOR EACH ROW

BEGIN

IF (SYSDATE-:NEW.FECHA\_NACIMIENTO<365\*3)

THEN raise\_application\_error(-20501,'Un usuario debe tener al menos 3 anos');

END IF;

END;

/

CREATE OR REPLACE TRIGGER ASIGNATURAS\_MENORES

AFTER INSERT ON MATRICULAS

FOR EACH ROW

DECLARE

FN USUARIOS.FECHA\_NACIMIENTO%TYPE;

BEGIN

SELECT FECHA\_NACIMIENTO INTO FN FROM USUARIOS where oid\_u=:new.oid\_u;

IF(SYSDATE-FN>365\*3 AND SYSDATE-FN<365\*6)

THEN CREAR\_PERTENECE\_A(:NEW.oid\_m,buscar\_oid\_a('Expresion Corporal y Danza'));

END IF;

END;

/

CREATE OR REPLACE TRIGGER FACTURACION

AFTER INSERT ON MATRICULAS

FOR EACH ROW

BEGIN

CREAR\_PAGO(null, 25, 'Matricula', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO( null, 35, 'Octubre', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Noviembre', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Diciembre', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Enero', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Febrero', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Marzo', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Abril', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Mayo', 'Pendiente', :NEW.oid\_m);

CREAR\_PAGO(null, 35, 'Junio', 'Pendiente', :NEW.oid\_m);

END;

/

CREATE OR REPLACE TRIGGER ERROR\_INSTRUMENTO\_PRESTADO

BEFORE INSERT OR UPDATE ON PRESTAMOS

FOR EACH ROW

DECLARE

inst\_libre instrumentos.libre%type;

BEGIN

SELECT libre INTO inst\_libre FROM instrumentos WHERE oid\_i=:NEW.oid\_i;

IF (inst\_libre<>1)

THEN raise\_application\_error(-20502,'El instrumento ya esta prestado');

END IF;

END;

/

CREATE OR REPLACE TRIGGER INSTRUMENTO\_PRESTADO

AFTER INSERT OR UPDATE ON PRESTAMOS

FOR EACH ROW

BEGIN

UPDATE instrumentos SET libre = 0 WHERE oid\_i=:NEW.oid\_i;

END;

/

## Script de pruebas

/\* PAQUETES \*/

CREATE OR REPLACE PACKAGE PRUEBAS\_ASIGNATURAS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_nombre asignaturas.nombre%TYPE,salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2,w\_oid\_a asignaturas.oid\_a%TYPE, w\_nombre asignaturas.nombre%TYPE, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_a asignaturas.oid\_a%TYPE, salidaEsperada BOOLEAN);

END PRUEBAS\_ASIGNATURAS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_ASIGNATURAS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

DELETE FROM pertenece\_a;

DELETE FROM asignaturas;

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE ASIGNATURAS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_nombre asignaturas.nombre%TYPE,salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

asignatura asignaturas%ROWTYPE;

w\_oid\_a asignaturas.oid\_a%TYPE;

BEGIN

/\* Insertar asignatura \*/

CREAR\_ASIGNATURA(w\_nombre);

/\* Seleccionar asignatura y comprobar que los datos se insertaron correctamente \*/

w\_oid\_a := sec\_a.currval;

SELECT \* INTO asignatura FROM asignaturas WHERE oid\_a=w\_oid\_a;

IF (asignatura.nombre<>w\_nombre) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE ASIGNATURAS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2,w\_oid\_a asignaturas.oid\_a%TYPE, w\_nombre asignaturas.nombre%TYPE, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

asignatura asignaturas%ROWTYPE;

BEGIN

/\* Actualizar nombre \*/

UPDATE asignaturas SET nombre=w\_nombre WHERE oid\_a=w\_oid\_a;

/\* Seleccionar asignatura y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO asignatura FROM asignaturas WHERE oid\_a=w\_oid\_a;

IF (asignatura.nombre<>w\_nombre) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE ASIGNATURAS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_a asignaturas.oid\_a%TYPE, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_asignaturas INTEGER;

BEGIN

/\* Eliminar asignatura \*/

DELETE FROM asignaturas WHERE oid\_a=w\_oid\_a;

/\* Verificar que el asignatura no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_asignaturas FROM asignaturas WHERE oid\_a=w\_oid\_a;

IF (n\_asignaturas <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_ASIGNATURAS;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_USUARIOS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_nombre usuarios.nombre%type, w\_apellidos usuarios.apellidos%type,

w\_fecha\_nacimiento usuarios.fecha\_nacimiento%type, w\_direccion usuarios.direccion%type, w\_email usuarios.email%type,

w\_telefono usuarios.telefono%type, w\_derechos\_imagen usuarios.derechos\_imagen%type,salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_u usuarios.oid\_u%type, w\_nombre usuarios.nombre%type,

w\_apellidos usuarios.apellidos%type, w\_fecha\_nacimiento usuarios.fecha\_nacimiento%type, w\_direccion usuarios.direccion%type,

w\_email usuarios.email%type, w\_telefono usuarios.telefono%type, w\_derechos\_imagen usuarios.derechos\_imagen%type,

salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_u usuarios.oid\_u%type, salidaEsperada BOOLEAN);

END PRUEBAS\_USUARIOS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_USUARIOS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

DELETE FROM relaciones;

borrar\_matricula\_cascada;

DELETE FROM usuarios;

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE USUARIOS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_nombre usuarios.nombre%type, w\_apellidos usuarios.apellidos%type,

w\_fecha\_nacimiento usuarios.fecha\_nacimiento%type, w\_direccion usuarios.direccion%type, w\_email usuarios.email%type,

w\_telefono usuarios.telefono%type, w\_derechos\_imagen usuarios.derechos\_imagen%type,salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

usuario usuarios%ROWTYPE;

w\_oid\_u usuarios.oid\_u%type;

BEGIN

/\* Insertar usuario \*/

CREAR\_USUARIO(w\_nombre, w\_apellidos,

w\_fecha\_nacimiento, w\_direccion, w\_email, w\_telefono, w\_derechos\_imagen);

/\* Seleccionar usuario y comprobar que los datos se insertaron correctamente \*/

w\_oid\_u := sec\_u.currval;

SELECT \* INTO usuario FROM usuarios WHERE oid\_u=w\_oid\_u;

IF (usuario.nombre<>w\_nombre OR usuario.apellidos<>w\_apellidos OR usuario.fecha\_nacimiento<>w\_fecha\_nacimiento

OR usuario.direccion<>w\_direccion OR usuario.email<>w\_email OR usuario.telefono<>w\_telefono

OR usuario.derechos\_imagen<>w\_derechos\_imagen) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE USUARIOS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_u usuarios.oid\_u%type, w\_nombre usuarios.nombre%type,

w\_apellidos usuarios.apellidos%type, w\_fecha\_nacimiento usuarios.fecha\_nacimiento%type, w\_direccion usuarios.direccion%type,

w\_email usuarios.email%type, w\_telefono usuarios.telefono%type, w\_derechos\_imagen usuarios.derechos\_imagen%type,

salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

usuario usuarios%ROWTYPE;

BEGIN

/\* Actualizar usuario \*/

UPDATE usuarios SET nombre=w\_nombre, apellidos=w\_apellidos, fecha\_nacimiento=w\_fecha\_nacimiento,

direccion=w\_direccion, email=w\_email, telefono=w\_telefono, derechos\_imagen=w\_derechos\_imagen WHERE oid\_u=w\_oid\_u;

/\* Seleccionar usuario y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO usuario FROM usuarios WHERE oid\_u=w\_oid\_u;

IF (usuario.nombre<>w\_nombre OR usuario.apellidos<>w\_apellidos OR usuario.fecha\_nacimiento<>w\_fecha\_nacimiento

OR usuario.direccion<>w\_direccion OR usuario.email<>w\_email OR usuario.telefono<>w\_telefono

OR usuario.derechos\_imagen<>w\_derechos\_imagen) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE USUARIOS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_u usuarios.oid\_u%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_usuarios INTEGER;

BEGIN

/\* Eliminar usuario \*/

DELETE FROM usuarios WHERE oid\_u=w\_oid\_u;

/\* Verificar que el usuario no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_usuarios FROM usuarios WHERE oid\_u=w\_oid\_u;

IF (n\_usuarios <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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 PRUEBAS\_MATRICULAS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_fecha\_matriculacion matriculas.fecha\_matriculacion%type,

w\_curso matriculas.curso%type, w\_codigo matriculas.codigo%type, w\_oid\_u matriculas.oid\_u%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_m matriculas.oid\_m%type, w\_fecha\_matriculacion matriculas.fecha\_matriculacion%type,

w\_curso matriculas.curso%type, w\_codigo matriculas.codigo%type, w\_oid\_u matriculas.oid\_u%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_m matriculas.oid\_m%type, salidaEsperada BOOLEAN);

END PRUEBAS\_MATRICULAS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_MATRICULAS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

CREAR\_ASIGNATURA('Expresion Corporal y Danza'); /\* Para que se cumpla un trigger \*/

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE MATRICULAS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_fecha\_matriculacion matriculas.fecha\_matriculacion%type,

w\_curso matriculas.curso%type, w\_codigo matriculas.codigo%type, w\_oid\_u matriculas.oid\_u%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

matricula matriculas%ROWTYPE;

w\_oid\_m matriculas.oid\_m%type;

BEGIN

/\* Insertar matricula \*/

CREAR\_MATRICULA(w\_fecha\_matriculacion, w\_curso, w\_codigo, w\_oid\_u);

/\* Seleccionar matricula y comprobar que los datos se insertaron correctamente \*/

w\_oid\_m := sec\_m.currval;

SELECT \* INTO matricula FROM matriculas WHERE oid\_m=w\_oid\_m;

IF (matricula.fecha\_matriculacion<>w\_fecha\_matriculacion OR matricula.curso<>w\_curso

OR matricula.codigo<>w\_codigo OR matricula.oid\_u<>w\_oid\_u) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE MATRICULAS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_m matriculas.oid\_m%type, w\_fecha\_matriculacion matriculas.fecha\_matriculacion%type,

w\_curso matriculas.curso%type, w\_codigo matriculas.codigo%type, w\_oid\_u matriculas.oid\_u%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

matricula matriculas%ROWTYPE;

BEGIN

/\* Actualizar matricula \*/

UPDATE matriculas SET fecha\_matriculacion=w\_fecha\_matriculacion, curso=w\_curso, codigo=w\_codigo, oid\_u=w\_oid\_u WHERE oid\_m=w\_oid\_m;

/\* Seleccionar matricula y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO matricula FROM matriculas WHERE oid\_m=w\_oid\_m;

IF (matricula.fecha\_matriculacion<>w\_fecha\_matriculacion OR matricula.curso<>w\_curso

OR matricula.codigo<>w\_codigo OR matricula.oid\_u<>w\_oid\_u) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE MATRICULAS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_m matriculas.oid\_m%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_matriculas INTEGER;

BEGIN

/\* Eliminar matricula \*/

DELETE FROM pagos WHERE oid\_m=w\_oid\_m;

DELETE FROM matriculas WHERE oid\_m=w\_oid\_m;

/\* Verificar que la matricula no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_matriculas FROM matriculas WHERE oid\_m=w\_oid\_m;

IF (n\_matriculas <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_MATRICULAS;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_FALTAS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_tipo\_falta faltas.tipo\_falta%type,

w\_fecha\_falta faltas.fecha\_falta%type, w\_justificada faltas.justificada%type, w\_oid\_m faltas.oid\_m%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_f faltas.oid\_f%type, w\_tipo\_falta faltas.tipo\_falta%type,

w\_fecha\_falta faltas.fecha\_falta%type, w\_justificada faltas.justificada%type, w\_oid\_m faltas.oid\_m%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_f faltas.oid\_f%type, salidaEsperada BOOLEAN);

END PRUEBAS\_FALTAS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_FALTAS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),1,'i101',sec\_u.currval);

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE FALTAS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_tipo\_falta faltas.tipo\_falta%type,

w\_fecha\_falta faltas.fecha\_falta%type, w\_justificada faltas.justificada%type,

w\_oid\_m faltas.oid\_m%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

falta faltas%ROWTYPE;

w\_oid\_f faltas.oid\_f%type;

BEGIN

/\* Insertar falta \*/

CREAR\_FALTA(w\_tipo\_falta, w\_fecha\_falta, w\_justificada, w\_oid\_m);

/\* Seleccionar falta y comprobar que los datos se insertaron correctamente \*/

w\_oid\_f := sec\_f.currval;

SELECT \* INTO falta FROM faltas WHERE oid\_f=w\_oid\_f;

IF (falta.tipo\_falta<>w\_tipo\_falta OR falta.fecha\_falta<>w\_fecha\_falta OR falta.justificada<>w\_justificada

OR falta.oid\_m<>w\_oid\_m) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE FALTAS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_f faltas.oid\_f%type, w\_tipo\_falta faltas.tipo\_falta%type,

w\_fecha\_falta faltas.fecha\_falta%type, w\_justificada faltas.justificada%type, w\_oid\_m faltas.oid\_m%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

falta faltas%ROWTYPE;

BEGIN

/\* Actualizar empleado \*/

UPDATE faltas SET oid\_f=w\_oid\_f, tipo\_falta=w\_tipo\_falta,

fecha\_falta=w\_fecha\_falta, justificada=w\_justificada, oid\_m=w\_oid\_m WHERE oid\_f=w\_oid\_f;

/\* Seleccionar falta y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO falta FROM faltas WHERE oid\_f=w\_oid\_f;

IF (falta.tipo\_falta<>w\_tipo\_falta OR falta.fecha\_falta<>w\_fecha\_falta OR falta.justificada<>w\_justificada

OR falta.oid\_m<>w\_oid\_m) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE FALTAS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_f faltas.oid\_f%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_faltas INTEGER;

BEGIN

/\* Eliminar falta \*/

DELETE FROM faltas WHERE oid\_f=w\_oid\_f;

/\* Verificar que el falta no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_faltas FROM faltas WHERE oid\_f=w\_oid\_f;

IF (n\_faltas <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_FALTAS;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_INSTRUMENTOS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_tipo instrumentos.tipo%type, w\_libre instrumentos.libre%type,

w\_nombre instrumentos.nombre%type, w\_estado\_instrumento instrumentos.estado\_instrumento%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_i instrumentos.oid\_i%type, w\_tipo instrumentos.tipo%type, w\_libre instrumentos.libre%type,

w\_nombre instrumentos.nombre%type, w\_estado\_instrumento instrumentos.estado\_instrumento%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_i instrumentos.oid\_i%type, salidaEsperada BOOLEAN);

END PRUEBAS\_INSTRUMENTOS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_INSTRUMENTOS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

DELETE FROM prestamos;

DELETE FROM instrumentos;

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE INSTRUMENTOS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_tipo instrumentos.tipo%type, w\_libre instrumentos.libre%type,

w\_nombre instrumentos.nombre%type, w\_estado\_instrumento instrumentos.estado\_instrumento%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

instrumento instrumentos%ROWTYPE;

w\_oid\_i instrumentos.oid\_i%type;

BEGIN

/\* Insertar instrumento \*/

CREAR\_INSTRUMENTO(w\_tipo, w\_libre, w\_nombre, w\_estado\_instrumento);

/\* Seleccionar instrumento y comprobar que los datos se insertaron correctamente \*/

w\_oid\_i := sec\_i.currval;

SELECT \* INTO instrumento FROM instrumentos WHERE oid\_i=w\_oid\_i;

IF (instrumento.tipo<>w\_tipo OR instrumento.libre<>w\_libre

OR instrumento.nombre<>w\_nombre OR instrumento.estado\_instrumento<>w\_estado\_instrumento) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE INSTRUMENTOS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_i instrumentos.oid\_i%type, w\_tipo instrumentos.tipo%type, w\_libre instrumentos.libre%type,

w\_nombre instrumentos.nombre%type, w\_estado\_instrumento instrumentos.estado\_instrumento%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

instrumento instrumentos%ROWTYPE;

BEGIN

/\* Actualizar instrumento \*/

UPDATE instrumentos SET tipo=w\_tipo, libre=w\_libre, nombre=w\_nombre, estado\_instrumento=w\_estado\_instrumento WHERE oid\_i=w\_oid\_i;

/\* Seleccionar instrumento y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO instrumento FROM instrumentos WHERE oid\_i=w\_oid\_i;

IF (instrumento.tipo<>w\_tipo OR instrumento.libre<>w\_libre

OR instrumento.nombre<>w\_nombre OR instrumento.estado\_instrumento<>w\_estado\_instrumento) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE INSTRUMENTOS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_i instrumentos.oid\_i%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_instrumentos INTEGER;

BEGIN

/\* Eliminar instrumento \*/

DELETE FROM instrumentos WHERE oid\_i=w\_oid\_i;

/\* Verificar que la instrumento no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_instrumentos FROM instrumentos WHERE oid\_i=w\_oid\_i;

IF (n\_instrumentos <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_INSTRUMENTOS;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_PERTENECE\_A AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_oid\_m pertenece\_a.oid\_m%type, w\_oid\_a pertenece\_a.oid\_a%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_pe pertenece\_a.oid\_pe%type, w\_oid\_m pertenece\_a.oid\_m%type,

w\_oid\_a pertenece\_a.oid\_a%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_pe pertenece\_a.oid\_pe%type, salidaEsperada BOOLEAN);

END PRUEBAS\_PERTENECE\_A;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_PERTENECE\_A AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

borrar\_matricula\_cascada;

DELETE FROM asignaturas;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),1,'i101',sec\_u.currval);

crear\_asignatura('Expresión Corporal y Danza');

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE PERTENECE\_A \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_oid\_m pertenece\_a.oid\_m%type, w\_oid\_a pertenece\_a.oid\_a%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

w\_pertenece\_a pertenece\_a%ROWTYPE;

w\_oid\_pe pertenece\_a.oid\_pe%type;

BEGIN

/\* Insertar pertenece\_a \*/

CREAR\_PERTENECE\_A(w\_oid\_m, w\_oid\_a);

/\* Seleccionar pertenece\_a y comprobar que los datos se insertaron correctamente \*/

w\_oid\_pe := sec\_pe.currval;

SELECT \* INTO w\_pertenece\_a FROM pertenece\_a WHERE oid\_pe=w\_oid\_pe;

IF (w\_pertenece\_a.oid\_m<>w\_oid\_m OR w\_pertenece\_a.oid\_a<>w\_oid\_a) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE PERTENECE\_A \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_pe pertenece\_a.oid\_pe%type, w\_oid\_m pertenece\_a.oid\_m%type,

w\_oid\_a pertenece\_a.oid\_a%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

w\_pertenece\_a pertenece\_a%ROWTYPE;

BEGIN

/\* Actualizar pertenece\_a \*/

UPDATE pertenece\_a SET oid\_m=w\_oid\_m, oid\_a=w\_oid\_a WHERE oid\_pe=w\_oid\_pe;

/\* Seleccionar pertenece\_a y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO w\_pertenece\_a FROM pertenece\_a WHERE oid\_pe=w\_oid\_pe;

IF (w\_pertenece\_a.oid\_m<>w\_oid\_m OR w\_pertenece\_a.oid\_a<>w\_oid\_a) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE PERTENECE\_A \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_pe pertenece\_a.oid\_pe%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_pertenece\_a INTEGER;

BEGIN

/\* Eliminar pertenece\_a \*/

DELETE FROM pertenece\_a WHERE oid\_pe=w\_oid\_pe;

/\* Verificar que la relacion pertenece\_a no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_pertenece\_a FROM pertenece\_a WHERE oid\_pe=w\_oid\_pe;

IF (n\_pertenece\_a <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_PERTENECE\_A;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_PRESTAMOS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_fecha\_prestamo prestamos.fecha\_prestamo%type, w\_oid\_m prestamos.oid\_m%type,

w\_oid\_i prestamos.oid\_i%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_p prestamos.oid\_p%type, w\_fecha\_prestamo prestamos.fecha\_prestamo%type,

w\_oid\_m prestamos.oid\_m%type, w\_oid\_i prestamos.oid\_i%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_p prestamos.oid\_p%type, salidaEsperada BOOLEAN);

END PRUEBAS\_PRESTAMOS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_PRESTAMOS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

borrar\_matricula\_cascada;

DELETE FROM instrumentos;

DELETE FROM prestamos;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),1,'i101',sec\_u.currval);

crear\_instrumento('Cuerda', 1, 'Violin', 'Usado');

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE PRESTAMOS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_fecha\_prestamo prestamos.fecha\_prestamo%type, w\_oid\_m prestamos.oid\_m%type,

w\_oid\_i prestamos.oid\_i%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

prestamo prestamos%ROWTYPE;

w\_oid\_p prestamos.oid\_p%type;

BEGIN

/\* Insertar prestamos \*/

CREAR\_PRESTAMO(w\_fecha\_prestamo, w\_oid\_m, w\_oid\_i);

/\* Seleccionar prestamos y comprobar que los datos se insertaron correctamente \*/

w\_oid\_p := sec\_p.currval;

SELECT \* INTO prestamo FROM prestamos WHERE oid\_p=w\_oid\_p;

IF (prestamo.fecha\_prestamo<>w\_fecha\_prestamo OR prestamo.oid\_m<>w\_oid\_m OR prestamo.oid\_i<>w\_oid\_i) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE PRESTAMOS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_p prestamos.oid\_p%type, w\_fecha\_prestamo prestamos.fecha\_prestamo%type,

w\_oid\_m prestamos.oid\_m%type, w\_oid\_i prestamos.oid\_i%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

prestamo prestamos%ROWTYPE;

BEGIN

/\* Actualizar prestamos \*/

UPDATE prestamos SET fecha\_prestamo=w\_fecha\_prestamo, oid\_m=w\_oid\_m, oid\_i=w\_oid\_i WHERE oid\_p=w\_oid\_p;

/\* Seleccionar prestamos y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO prestamo FROM prestamos WHERE oid\_p=w\_oid\_p;

IF (prestamo.fecha\_prestamo<>w\_fecha\_prestamo OR prestamo.oid\_m<>w\_oid\_m OR prestamo.oid\_i<>w\_oid\_i) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE PRESTAMOS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_p prestamos.oid\_p%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_prestamos INTEGER;

BEGIN

/\* Eliminar prestamos \*/

DELETE FROM prestamos WHERE oid\_p=w\_oid\_p;

/\* Verificar que la relacion prestamos no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_prestamos FROM prestamos WHERE oid\_p=w\_oid\_p;

IF (n\_prestamos <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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 PRUEBAS\_RESPONSABLES AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_nombre responsables.nombre%type, w\_apellidos responsables.apellidos%type,

w\_email responsables.email%type, w\_telefono responsables.telefono%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_r responsables.oid\_r%type, w\_nombre responsables.nombre%type,

w\_apellidos responsables.apellidos%type, w\_email responsables.email%type, w\_telefono responsables.telefono%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_r responsables.oid\_r%type, salidaEsperada BOOLEAN);

END PRUEBAS\_RESPONSABLES;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_RESPONSABLES AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

DELETE FROM relaciones;

DELETE FROM responsables;

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE RESPONSABLES \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_nombre responsables.nombre%type, w\_apellidos responsables.apellidos%type,

w\_email responsables.email%type, w\_telefono responsables.telefono%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

responsable responsables%ROWTYPE;

w\_oid\_r responsables.oid\_r%type;

BEGIN

/\* Insertar responsable \*/

CREAR\_RESPONSABLE(w\_nombre, w\_apellidos, w\_email, w\_telefono);

/\* Seleccionar responsable y comprobar que los datos se insertaron correctamente \*/

w\_oid\_r := sec\_r.currval;

SELECT \* INTO responsable FROM responsables WHERE oid\_r=w\_oid\_r;

IF (responsable.nombre<>w\_nombre OR responsable.apellidos<>w\_apellidos OR responsable.email<>w\_email

OR responsable.telefono<>w\_telefono) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE RESPONSABLES \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_r responsables.oid\_r%type, w\_nombre responsables.nombre%type,

w\_apellidos responsables.apellidos%type, w\_email responsables.email%type, w\_telefono responsables.telefono%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

responsable responsables%ROWTYPE;

BEGIN

/\* Actualizar responsable \*/

UPDATE responsables SET nombre=w\_nombre, apellidos=w\_apellidos, email=w\_email, telefono=w\_telefono WHERE oid\_r=w\_oid\_r;

/\* Seleccionar responsable y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO responsable FROM responsables WHERE oid\_r=w\_oid\_r;

IF (responsable.nombre<>w\_nombre OR responsable.apellidos<>w\_apellidos OR responsable.email<>w\_email

OR responsable.telefono<>w\_telefono) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE RESPONSABLES \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_r responsables.oid\_r%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_responsables INTEGER;

BEGIN

/\* Eliminar responsable \*/

DELETE FROM responsables WHERE oid\_r=w\_oid\_r;

/\* Verificar que el responsable no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_responsables FROM responsables WHERE oid\_r=w\_oid\_r;

IF (n\_responsables <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_RESPONSABLES;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_PAGOS AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_fecha\_pago pagos.fecha\_pago%type, w\_cantidad pagos.cantidad%type,

w\_concepto pagos.concepto%type, w\_estado pagos.estado%type, w\_oid\_m pagos.oid\_m%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_pa pagos.oid\_pa%type, w\_fecha\_pago pagos.fecha\_pago%type,

w\_cantidad pagos.cantidad%type, w\_concepto pagos.concepto%type, w\_estado pagos.estado%type, w\_oid\_m pagos.oid\_m%type,

salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_pa pagos.oid\_pa%type, salidaEsperada BOOLEAN);

END PRUEBAS\_PAGOS;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_PAGOS AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),1,'i101',sec\_u.currval);

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE PAGOS \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_fecha\_pago pagos.fecha\_pago%type, w\_cantidad pagos.cantidad%type,

w\_concepto pagos.concepto%type, w\_estado pagos.estado%type, w\_oid\_m pagos.oid\_m%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

pago pagos%ROWTYPE;

w\_oid\_pa pagos.oid\_pa%type;

BEGIN

/\* Insertar pago \*/

CREAR\_PAGO(w\_fecha\_pago, w\_cantidad, w\_concepto, w\_estado, w\_oid\_m);

/\* Seleccionar pago y comprobar que los datos se insertaron correctamente \*/

w\_oid\_pa := sec\_pa.currval;

SELECT \* INTO pago FROM pagos WHERE oid\_pa=w\_oid\_pa;

IF (pago.fecha\_pago<>w\_fecha\_pago OR pago.cantidad<>w\_cantidad

OR pago.concepto<>w\_concepto OR pago.estado<>w\_estado OR pago.oid\_m<>w\_oid\_m) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE PAGOS \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_pa pagos.oid\_pa%type, w\_fecha\_pago pagos.fecha\_pago%type,

w\_cantidad pagos.cantidad%type, w\_concepto pagos.concepto%type, w\_estado pagos.estado%type, w\_oid\_m pagos.oid\_m%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

pago pagos%ROWTYPE;

BEGIN

/\* Actualizar pago \*/

UPDATE pagos SET fecha\_pago=w\_fecha\_pago, cantidad=w\_cantidad, concepto=w\_concepto, estado=w\_estado, oid\_m=w\_oid\_m WHERE oid\_pa=w\_oid\_pa;

/\* Seleccionar pago y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO pago FROM pagos WHERE oid\_pa=w\_oid\_pa;

IF (pago.fecha\_pago<>w\_fecha\_pago OR pago.cantidad<>w\_cantidad

OR pago.concepto<>w\_concepto OR pago.estado<>w\_estado OR pago.oid\_m<>w\_oid\_m) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE PAGOS \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_pa pagos.oid\_pa%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_pagos INTEGER;

BEGIN

/\* Eliminar pago \*/

DELETE FROM pagos WHERE oid\_pa=w\_oid\_pa;

/\* Verificar que la pago no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_pagos FROM pagos WHERE oid\_pa=w\_oid\_pa;

IF (n\_pagos <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_RELACIONES AS

PROCEDURE inicializar;

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_oid\_u relaciones.oid\_u%type, w\_oid\_r relaciones.oid\_r%type,

w\_tipo\_relacion relaciones.tipo\_relacion%type, salidaEsperada BOOLEAN);

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_rel relaciones.oid\_rel%type, w\_oid\_u relaciones.oid\_u%type,

w\_oid\_r relaciones.oid\_r%type, w\_tipo\_relacion relaciones.tipo\_relacion%type, salidaEsperada BOOLEAN);

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_rel relaciones.oid\_rel%type, salidaEsperada BOOLEAN);

END PRUEBAS\_RELACIONES;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_RELACIONES AS

/\* INICIALIZACIÓN \*/

PROCEDURE inicializar AS

BEGIN

/\* Borrar contenido de la tabla \*/

DELETE FROM relaciones;

borrar\_matricula\_cascada;

DELETE FROM usuarios;

DELETE FROM responsables;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_responsable('Antonio', 'Perez Sanchez', 'AntPezSan4@gmail.com', '555643421');

END inicializar;

/\* PRUEBA PARA LA INSERCIÓN DE RELACIONES \*/

PROCEDURE insertar (nombre\_prueba VARCHAR2, w\_oid\_u relaciones.oid\_u%type, w\_oid\_r relaciones.oid\_r%type,

w\_tipo\_relacion relaciones.tipo\_relacion%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

relacion relaciones%ROWTYPE;

w\_oid\_rel relaciones.oid\_rel%type;

BEGIN

/\* Insertar relacion \*/

CREAR\_RELACION(w\_oid\_u, w\_oid\_r, w\_tipo\_relacion);

/\* Seleccionar relacion y comprobar que los datos se insertaron correctamente \*/

w\_oid\_rel := sec\_rel.currval;

SELECT \* INTO relacion FROM relaciones WHERE oid\_rel=w\_oid\_rel;

IF (relacion.oid\_u<>w\_oid\_u OR relacion.oid\_r<>w\_oid\_r

OR relacion.tipo\_relacion<>w\_tipo\_relacion) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ACTUALIZACIÓN DE RELACIONES \*/

PROCEDURE actualizar (nombre\_prueba VARCHAR2, w\_oid\_rel relaciones.oid\_rel%type, w\_oid\_u relaciones.oid\_u%type,

w\_oid\_r relaciones.oid\_r%type, w\_tipo\_relacion relaciones.tipo\_relacion%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

relacion relaciones%ROWTYPE;

BEGIN

/\* Actualizar relacion \*/

UPDATE relaciones SET oid\_u=w\_oid\_u, oid\_r=w\_oid\_r, tipo\_relacion=w\_tipo\_relacion WHERE oid\_rel=w\_oid\_rel;

/\* Seleccionar relacion y comprobar que los campos se actualizaron correctamente \*/

SELECT \* INTO relacion FROM relaciones WHERE oid\_rel=w\_oid\_rel;

IF (relacion.oid\_u<>w\_oid\_u OR relacion.oid\_r<>w\_oid\_r

OR relacion.tipo\_relacion<>w\_tipo\_relacion) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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;

/\* PRUEBA PARA LA ELIMINACIÓN DE RELACIONES \*/

PROCEDURE eliminar (nombre\_prueba VARCHAR2,w\_oid\_rel relaciones.oid\_rel%type, salidaEsperada BOOLEAN) AS

salida BOOLEAN := true;

n\_relaciones INTEGER;

BEGIN

/\* Eliminar relacion \*/

DELETE FROM relaciones WHERE oid\_rel=w\_oid\_rel;

/\* Verificar que la relacion no se encuentra en la BD \*/

SELECT COUNT(\*) INTO n\_relaciones FROM relaciones WHERE oid\_rel=w\_oid\_rel;

IF (n\_relaciones <> 0) THEN

salida := false;

END IF;

COMMIT WORK;

/\* Mostrar resultado de la prueba \*/

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\_RELACIONES;

/

CREATE OR REPLACE PACKAGE PRUEBAS\_REQUISITOS\_FUNCIONALES AS

PROCEDURE RF1;

PROCEDURE RF2;

PROCEDURE RF7;

PROCEDURE RF8;

PROCEDURE RF9;

PROCEDURE RF10;

PROCEDURE RF11;

PROCEDURE RF12;

END PRUEBAS\_REQUISITOS\_FUNCIONALES;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS\_REQUISITOS\_FUNCIONALES AS

PROCEDURE RF1 AS

BEGIN

DELETE FROM prestamos;

DELETE FROM instrumentos;

crear\_instrumento('Cuerda', 1, 'Violin', 'Usado');

crear\_instrumento('Cuerda', 1, 'Violin', 'Usado');

crear\_instrumento('Cuerda', 0, 'Violin', 'Usado');

crear\_instrumento('Cuerda', 0, 'Guitarra', 'Usado');

crear\_instrumento('Cuerda', 1, 'Guitarra', 'Deteriorado');

DBMS\_OUTPUT.put\_line('Prueba de INSTRUMENTOS\_LIBRES:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

instrumentos\_libres;

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF1;

PROCEDURE RF2 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

DELETE FROM responsables;

crear\_responsable('Antonio', 'Perez Sanchez', 'AntPezSan4@gmail.com', '555643421');

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_relacion(sec\_u.currval, sec\_r.currval, 'Padre');

DBMS\_OUTPUT.put\_line('Prueba de RESPONSABLE\_DEL\_USUARIO:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

RESPONSABLE\_DEL\_USUARIO(sec\_u.currval);

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF2;

PROCEDURE RF7 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/13','DD/MM/RR'),1,'i101',sec\_u.currval);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),2,'i201',sec\_u.currval);

crear\_matricula(to\_date('20/09/15','DD/MM/RR'),3,'i301',sec\_u.currval);

DBMS\_OUTPUT.put\_line('Prueba de MATRICULAS\_EN\_VIGOR:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

MATRICULAS\_EN\_VIGOR;

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF7;

PROCEDURE RF8 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/13','DD/MM/RR'),1,'i101',sec\_u.currval);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),2,'i201',sec\_u.currval);

crear\_matricula(to\_date('20/09/15','DD/MM/RR'),3,'i301',sec\_u.currval);

DBMS\_OUTPUT.put\_line('Prueba de MATRICULAS\_POR\_CURSO:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

MATRICULAS\_POR\_CURSO;

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF8;

PROCEDURE RF9 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM relaciones;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/14','DD/MM/RR'),1,'i101', sec\_u.currval);

crear\_matricula(to\_date('20/09/15','DD/MM/RR'),1,'i201', sec\_u.currval);

DBMS\_OUTPUT.put\_line('Prueba de PAGOS\_DEL\_USUARIO:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

PAGOS\_DEL\_USUARIO(sec\_u.currval);

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF9;

PROCEDURE RF10 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM instrumentos;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/15','DD/MM/RR'),1,'i101', sec\_u.currval);

crear\_instrumento('Cuerda', 1, 'Violin', 'Usado');

crear\_prestamo(to\_date('20/09/15','DD/MM/RR'), sec\_m.currval, sec\_i.currval);

DBMS\_OUTPUT.put\_line('Prueba de USUARIOS\_CON\_PRESTAMOS:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

usuarios\_con\_prestamos;

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF10;

PROCEDURE RF11 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM asignaturas;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/15','DD/MM/RR'),1,'i101', sec\_u.currval);

crear\_asignatura('asignatura1');

crear\_pertenece\_a(sec\_m.currval, sec\_a.currval);

crear\_asignatura('asignatura2');

crear\_pertenece\_a(sec\_m.currval, sec\_a.currval);

crear\_asignatura('asignatura3');

crear\_pertenece\_a(sec\_m.currval, sec\_a.currval);

DBMS\_OUTPUT.put\_line('Prueba de ASIGNATURAS\_DEL\_USUARIO:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

asignaturas\_del\_usuario(sec\_u.currval);

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF11;

PROCEDURE RF12 AS

BEGIN

borrar\_matricula\_cascada;

DELETE FROM asignaturas;

DELETE FROM instrumentos;

DELETE FROM usuarios;

crear\_usuario('Julian', 'Perez Muñoz', to\_date('15/02/05', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1);

crear\_matricula(to\_date('20/09/15','DD/MM/RR'),1,'i101', sec\_u.currval);

crear\_falta('Asistencia', to\_date('15/11/15', 'DD/MM/RR'),1,sec\_m.currval);

crear\_falta('Pago', to\_date('1/12/15', 'DD/MM/RR'),1,sec\_m.currval);

crear\_falta('Asistencia', to\_date('4/12/15', 'DD/MM/RR'),0,sec\_m.currval);

crear\_falta('Asistencia', to\_date('11/12/15', 'DD/MM/RR'),0,sec\_m.currval);

DBMS\_OUTPUT.put\_line('Prueba de FALTAS\_DEL\_USUARIO:');

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

faltas\_del\_usuario(sec\_u.currval);

DBMS\_OUTPUT.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END RF12;

END PRUEBAS\_REQUISITOS\_FUNCIONALES;

/

/\* Activar salida de texto por pantalla \*/

SET SERVEROUTPUT ON;

DECLARE

oid\_u usuarios.oid\_u%type;

oid\_r responsables.oid\_r%type;

oid\_rel relaciones.oid\_rel%type;

oid\_m matriculas.oid\_m%type;

oid\_f faltas.oid\_f%type;

oid\_pa pagos.oid\_pa%type;

oid\_i instrumentos.oid\_i%type;

oid\_i2 instrumentos.oid\_i%type;

oid\_p prestamos.oid\_p%type;

oid\_a asignaturas.oid\_a%type;

oid\_pe pertenece\_a.oid\_pe%type;

BEGIN

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRUEBAS DE LAS OPERACIONES SOBRE LAS TABLAS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF1;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF2;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF7;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF8;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF9;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF10;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF11;

PRUEBAS\_REQUISITOS\_FUNCIONALES.RF12;

/\* Usuarios \*/

PRUEBAS\_USUARIOS.INICIALIZAR;

PRUEBAS\_USUARIOS.INSERTAR('Prueba 1 - Inserción usuario','Julian', 'Perez Muñoz', to\_date('15/02/12', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1, true);

oid\_u := sec\_u.currval;

PRUEBAS\_USUARIOS.INSERTAR('Prueba 2 - Inserción usuario menor a 3 años','Julian', 'Perez Muñoz', to\_date('15/02/15', 'dd/mm/rr'), 'C\ Ramon y Cajal Nº4', null, null, 1, false);

PRUEBAS\_USUARIOS.INSERTAR('Prueba 3 - Inserción usuario con nombre null', null, 'Perez Muñoz', to\_date('15/02/11', 'dd/mm/rr'), 'C\ Ramon y Cajal Nº4', null, null, 1, false);

PRUEBAS\_USUARIOS.INSERTAR('Prueba 4 - Inserción usuario con apellidos null','Julian', null, to\_date('15/02/11', 'dd/mm/rr'), 'C\ Ramon y Cajal Nº4', null, null, 1, false);

PRUEBAS\_USUARIOS.INSERTAR('Prueba 5 - Inserción usuario con fecha de nacimiento null','Julian', 'Perez Muñoz', null, 'C\ Ramon y Cajal Nº4', null, null, 1, false);

PRUEBAS\_USUARIOS.INSERTAR('Prueba 6 - Inserción usuario con direccion null','Julian', 'Perez Muñoz', to\_date('15/02/11', 'dd/mm/rr'), null, null, null, 1, false);

PRUEBAS\_USUARIOS.INSERTAR('Prueba 7 - Inserción usuario con derechos de imagen null','Julian', 'Perez Muñoz', to\_date('15/02/11', 'dd/mm/rr'), 'C\ Ramon y Cajal Nº4', null, null, null, false);

PRUEBAS\_USUARIOS.ACTUALIZAR('Prueba 8 - Actualización nombre usuario', oid\_u, 'Julio', 'Perez Muñoz', to\_date('15/02/11', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1, true);

PRUEBAS\_USUARIOS.ACTUALIZAR('Prueba 9 - Actualización nombre usuario null', oid\_u, null, 'Perez Muñoz', to\_date('15/02/11', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1, false);

PRUEBAS\_USUARIOS.ACTUALIZAR('Prueba 10 - Actualización edad mayor a 3 años', oid\_u, 'Julio', 'Perez Muñoz', to\_date('15/02/07', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1, true);

PRUEBAS\_USUARIOS.ACTUALIZAR('Prueba 11 - Actualización edad menor a 3 años', oid\_u, 'Julio', 'Perez Muñoz', to\_date('15/02/15', 'DD/MM/RR'), 'C\ Ramon y Cajal Nº4', null, null, 1, false);

PRUEBAS\_USUARIOS.ELIMINAR('Prueba 12 - Eliminación usuario', oid\_u, true);

/\* Responsables \*/

PRUEBAS\_RESPONSABLES.INICIALIZAR;

PRUEBAS\_RESPONSABLES.INSERTAR('Prueba 13 - Inserción responsable','Antonio', 'Perez Sanchez', 'AntPezSan4@gmail.com', '555643421', true);

oid\_r := sec\_r.currval;

PRUEBAS\_RESPONSABLES.INSERTAR('Prueba 14 - Inserción responsable con nombre a null',null, 'Perez Sanchez', 'AntPezSan4@gmail.com', '555643421', false);

PRUEBAS\_RESPONSABLES.INSERTAR('Prueba 15 - Inserción responsable con telefono a null','Antonio', 'Perez Sanchez', 'AntPezSan4@gmail.com', null, false);

PRUEBAS\_RESPONSABLES.ACTUALIZAR('Prueba 16 - Actualización nombre responsable', oid\_r, 'Juan', 'Perez Sanchez', 'AntPezSan4@gmail.com', '555643421', true);

PRUEBAS\_RESPONSABLES.ACTUALIZAR('Prueba 17 - Actualización responsable con nombre a null', oid\_r, null, 'Perez Sanchez', 'AntPezSan4@gmail.com', '555643421', false);

PRUEBAS\_RESPONSABLES.ACTUALIZAR('Prueba 18 - Actualización responsable con telefono a null', oid\_r,'Antonio', 'Perez Sanchez', 'AntPezSan4@gmail.com', null, false);

PRUEBAS\_RESPONSABLES.ELIMINAR('Prueba 19 - Eliminación responsable', oid\_r, true);

/\* Relaciones \*/

PRUEBAS\_RELACIONES.INICIALIZAR;

oid\_u := sec\_u.currval;

oid\_r := sec\_r.currval;

PRUEBAS\_RELACIONES.INSERTAR('Prueba 20 - Inserción relación', oid\_u, oid\_r, 'Padre', true);

oid\_rel := sec\_rel.currval;

PRUEBAS\_RELACIONES.INSERTAR('Prueba 21 - Inserción relación con tipo null', oid\_u, oid\_r, null, false);

PRUEBAS\_RELACIONES.ACTUALIZAR('Prueba 22 - Actualización relación',oid\_rel, oid\_u, oid\_r, 'Tio', true);

PRUEBAS\_RELACIONES.ACTUALIZAR('Prueba 23 - Actualización relación con tipo null', oid\_rel, oid\_u, oid\_r, null, false);

PRUEBAS\_RELACIONES.ELIMINAR('Prueba 24 - Eliminación relacion', oid\_rel, true);

/\* Matriculas \*/

PRUEBAS\_MATRICULAS.INICIALIZAR;

oid\_u := sec\_u.currval;

PRUEBAS\_MATRICULAS.INSERTAR('Prueba 25 - Inserción matrícula', to\_date('20/09/14','DD/MM/RR'),1,'i101',oid\_u, true);

oid\_m := sec\_m.currval;

PRUEBAS\_MATRICULAS.INSERTAR('Prueba 26 - Inserción matrícula con codigo null', to\_date('20/09/14','DD/MM/RR'),1,null,oid\_u, false);

PRUEBAS\_MATRICULAS.ACTUALIZAR('Prueba 27 - Actualización fecha matrícula', oid\_m, to\_date('21/09/14','DD/MM/RR'),1,'i101',oid\_u, true);

PRUEBAS\_MATRICULAS.ACTUALIZAR('Prueba 28 - Actualización matrícula con codigo null', oid\_m, to\_date('20/09/14','DD/MM/RR'),1,null,oid\_u, false);

PRUEBAS\_MATRICULAS.ELIMINAR('Prueba 29 - Eliminación matricula', oid\_m, true);

/\* Faltas \*/

PRUEBAS\_FALTAS.INICIALIZAR;

oid\_m := sec\_m.currval;

PRUEBAS\_FALTAS.INSERTAR('Prueba 30 - Inserción falta','Asistencia', to\_date('03/04/15', 'dd/mm/rr'), 0, oid\_m, true);

oid\_f := sec\_f.currval;

PRUEBAS\_FALTAS.INSERTAR('Prueba 31 - Inserción falta de tipo no enumerado','Modales', to\_date('03/04/15', 'dd/mm/rr'), 0, oid\_m, false);

PRUEBAS\_FALTAS.ACTUALIZAR('Prueba 32 - Actualización justificacion falta', oid\_f,'Asistencia', to\_date('03/04/15', 'dd/mm/rr'), 1, oid\_m, true);

PRUEBAS\_FALTAS.ACTUALIZAR('Prueba 33 - Actualización falta con tipo no enumerado', oid\_f,'Modales', to\_date('03/04/15', 'dd/mm/rr'), 1, oid\_m, false);

PRUEBAS\_FALTAS.ELIMINAR('Prueba 34 - Eliminación falta', oid\_f, true);

/\* Pagos \*/

PRUEBAS\_PAGOS.INICIALIZAR;

oid\_m := sec\_m.currval;

PRUEBAS\_PAGOS.INSERTAR('Prueba 35 - Inserción pago', to\_date('20/09/14','DD/MM/RR'), 25, 'Matricula', 'Pagado', oid\_m, true);

oid\_pa := sec\_pa.currval;

PRUEBAS\_PAGOS.INSERTAR('Prueba 36 - Inserción pago con cantidad null', to\_date('20/09/14','DD/MM/RR'), null, 'Matricula', 'Pagado', oid\_m, false);

PRUEBAS\_PAGOS.INSERTAR('Prueba 37 - Inserción pago con concepto null', to\_date('20/09/14','DD/MM/RR'), 25, null, 'Pagado', oid\_m, false);

PRUEBAS\_PAGOS.INSERTAR('Prueba 38 - Inserción pago con estado no enumerado', to\_date('20/09/14','DD/MM/RR'), 25, 'Matricula', 'Perdonado', oid\_m, false);

PRUEBAS\_PAGOS.ACTUALIZAR('Prueba 39 - Actualización cantidad pago', oid\_pa, to\_date('20/09/14','DD/MM/RR'), 20, 'Matricula', 'Pagado', oid\_m, true);

PRUEBAS\_PAGOS.ACTUALIZAR('Prueba 40 - Actualización pago con cantidad null', oid\_pa, to\_date('20/09/14','DD/MM/RR'), null, 'Matricula', 'Pagado', oid\_m, false);

PRUEBAS\_PAGOS.ACTUALIZAR('Prueba 41 - Actualización pago con concepto null', oid\_pa, to\_date('20/09/14','DD/MM/RR'), 25, null, 'Pagado', oid\_m, false);

PRUEBAS\_PAGOS.ACTUALIZAR('Prueba 42 - Actualización pago con estado no enumerado', oid\_pa, to\_date('20/09/14','DD/MM/RR'), 25, 'Matricula', 'Perdonado', oid\_m, false);

PRUEBAS\_PAGOS.ELIMINAR('Prueba 43 - Eliminación pago', oid\_pa, true);

/\* Instrumentos \*/

PRUEBAS\_INSTRUMENTOS.INICIALIZAR;

PRUEBAS\_INSTRUMENTOS.INSERTAR('Prueba 44 - Inserción instrumento', 'Cuerda', 1, 'Violin', 'Usado', true);

oid\_i := sec\_i.currval;

PRUEBAS\_INSTRUMENTOS.INSERTAR('Prueba 45 - Inserción instrumento prestado', 'Cuerda', 0, 'Guitarra', 'Deteriorado', true);

PRUEBAS\_INSTRUMENTOS.INSERTAR('Prueba 46 - Inserción instrumento de tipo no enumerado', 'Cuerda', 1, 'Violin', 'Mojado', false);

PRUEBAS\_INSTRUMENTOS.ACTUALIZAR('Prueba 47 - Actualización instrumento libre', oid\_i, 'Cuerda', 1, 'Violin', 'Usado', true);

PRUEBAS\_INSTRUMENTOS.ACTUALIZAR('Prueba 48 - Actualización estado instrumento', oid\_i, 'Cuerda', 1, 'Violin', 'Deteriorado', true);

PRUEBAS\_INSTRUMENTOS.ACTUALIZAR('Prueba 49 - Actualización estado instrumento de tipo no enumerado', oid\_i, 'Cuerda', 1, 'Violin', 'Mojado', false);

PRUEBAS\_INSTRUMENTOS.ELIMINAR('Prueba 50 - Eliminación instrumento', oid\_i, true);

/\* Prestamos \*/

PRUEBAS\_PRESTAMOS.INICIALIZAR;

oid\_m := sec\_m.currval;

oid\_i := sec\_i.currval;

crear\_instrumento('Cuerda', 0, 'Violin', 'Usado');

oid\_i2 := sec\_i.currval;

PRUEBAS\_PRESTAMOS.INSERTAR('Prueba 51 - Inserción prestamo', to\_date('03/04/15', 'dd/mm/rr'), oid\_m, oid\_i,true);

oid\_p := sec\_p.currval;

PRUEBAS\_PRESTAMOS.INSERTAR('Prueba 52 - Inserción prestamo de un instrumento que no esta libre', to\_date('03/04/15', 'dd/mm/rr'), oid\_m, oid\_i2,false);

PRUEBAS\_PRESTAMOS.ACTUALIZAR('Prueba 53 - Actualización de la fecha de prestamo de un instrumento que no esta libre', oid\_p, to\_date('04/04/15', 'dd/mm/rr'), oid\_m, oid\_i,false);

PRUEBAS\_PRESTAMOS.ACTUALIZAR('Prueba 54 - Actualización prestamo de un instrumento que no esta libre', oid\_p, to\_date('03/04/15', 'dd/mm/rr'), oid\_m, oid\_i2,false);

PRUEBAS\_PRESTAMOS.ELIMINAR('Prueba 55 - Eliminación prestamo', oid\_p, true);

/\* Asignaturas \*/

PRUEBAS\_ASIGNATURAS.INICIALIZAR;

PRUEBAS\_ASIGNATURAS.INSERTAR('Prueba 56 - Inserción asignatura','Expresión corporal y danza', true);

oid\_a := sec\_a.currval;

PRUEBAS\_ASIGNATURAS.INSERTAR('Prueba 57 - Inserción asignatura con nombre null',null, false);

PRUEBAS\_ASIGNATURAS.ACTUALIZAR('Prueba 58 - Actualización nombre asignatura', oid\_a, 'Lenguaje musical', true);

PRUEBAS\_ASIGNATURAS.ACTUALIZAR('Prueba 59 - Actualización asignatura con nombre null', oid\_a, null, false);

PRUEBAS\_ASIGNATURAS.ELIMINAR('Prueba 60 - Eliminación asignatura', oid\_a, true);

/\* Pertenece\_A \*/

PRUEBAS\_PERTENECE\_A.INICIALIZAR;

oid\_m := sec\_m.currval;

oid\_a := sec\_a.currval;

PRUEBAS\_PERTENECE\_A.INSERTAR('Prueba 61 - Inserción pertenece a', oid\_m, oid\_a, true);

oid\_pe := sec\_pe.currval;

PRUEBAS\_PERTENECE\_A.ACTUALIZAR('Prueba 62 - Actualización pertenece a con oid asignatura a null', oid\_pe, null, oid\_a, true);

PRUEBAS\_PERTENECE\_A.ELIMINAR('Prueba 63 - Eliminación pertenece\_a', oid\_pe, true);

END;

# Anexo

**Acta 1**

Temas a tratar

* Presentación del proyecto a realizar al director de la escuela
* Entrevista acerca del funcionamiento general de la escuela

Conclusiones

* Conseguimos información general sobre la filosofía y proceso de registro de la escuela.
* Necesitamos otra entrevista con la secretaria para obtener información concreta sobre el registro de los alumnos.

**Acta 2**

Temas a tratar

* Obtener información detallada sobre los registros y la administración general de los alumnos.

Conclusiones

* Conseguimos información concreta sobre el registro de los alumnos.
* Necesitamos otra entrevista con la secretaria para obtener información concreta sobre la administración de pagos y espacios de la escuela.

****

Imagen 4: Segunda Reunión

**Acta 3**

Temas a tratar

* Obtener información detallada sobre los pagos e impagos y la administración de los instrumentos prestados, así como del control de faltas.

Conclusiones

* Conseguimos información concreta sobre los temas tratados.



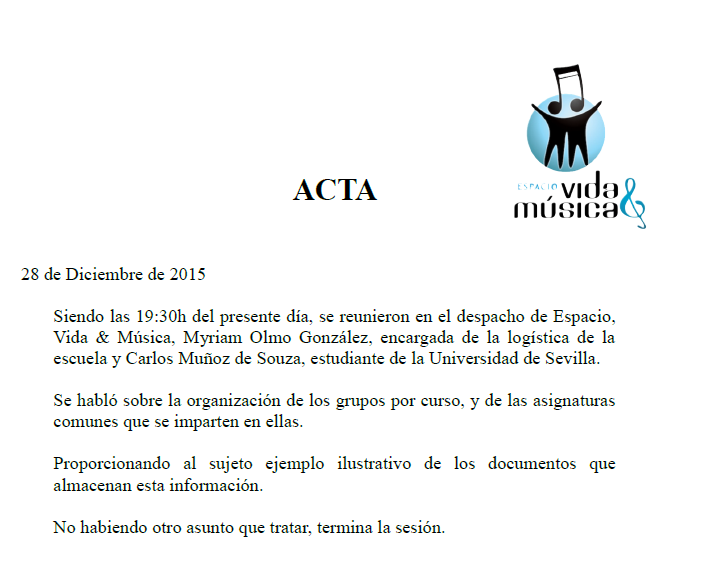
**Acta 4**

Temas a tratar

* Obtener información detallada sobre la organización de los grupos por curso.

Conclusiones

* Los únicos grupos existentes es el único de cada curso.



****

Imagen : Folleto promocional

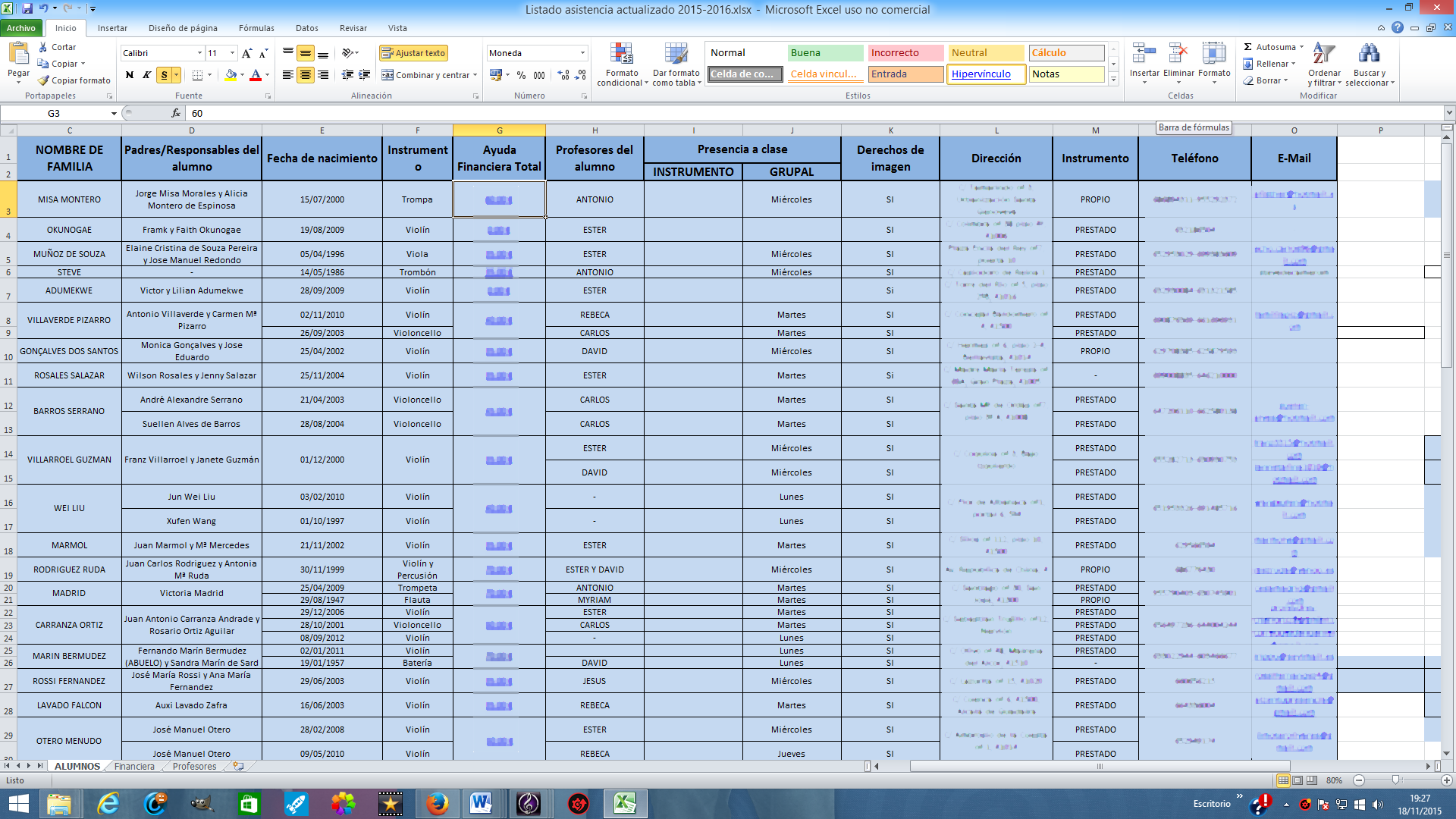


Imagen : Tabla de Excel con los registros de los alumnos

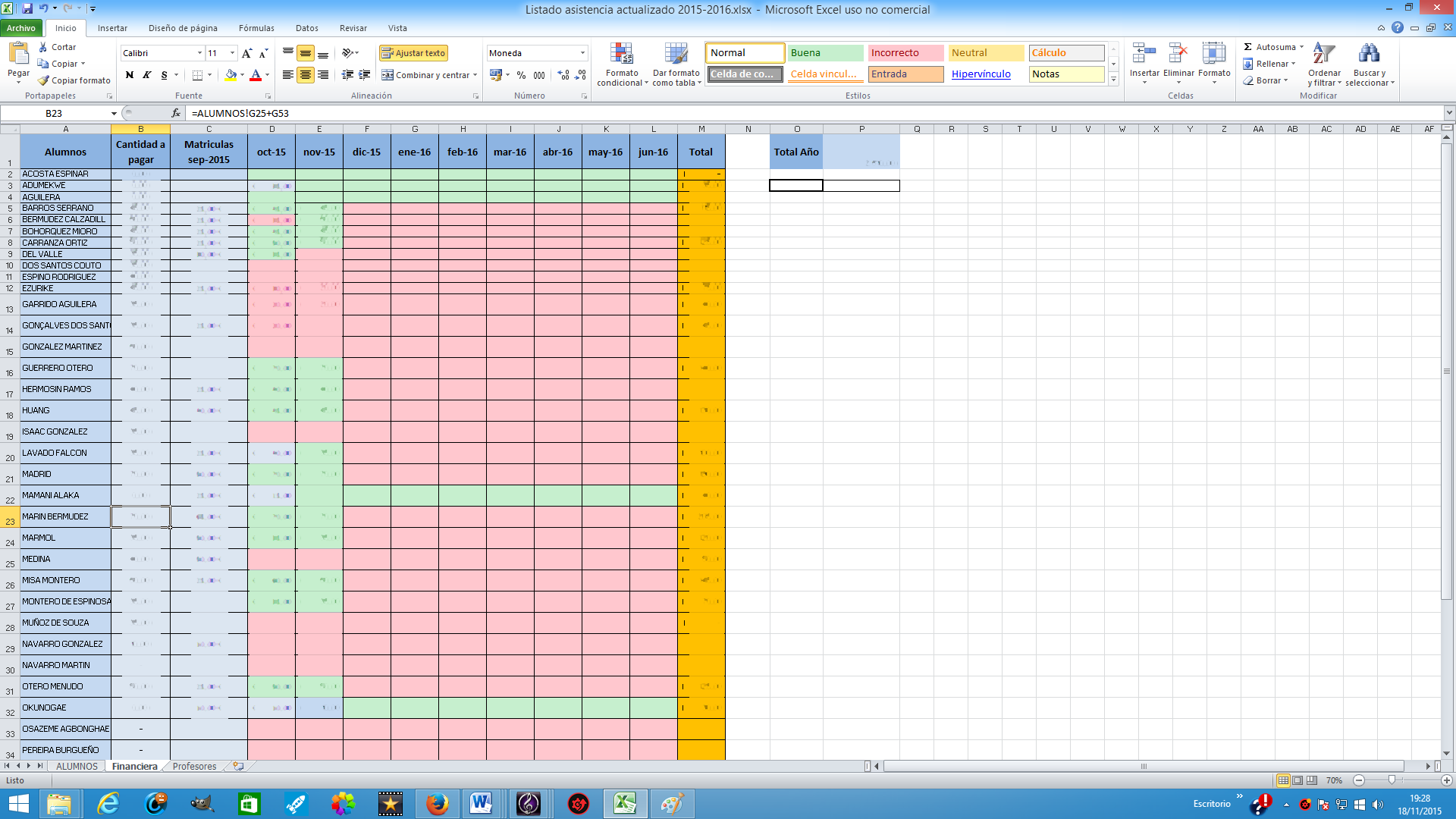


Imagen : Tabla de Excel con los registros de pagos