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

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, 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

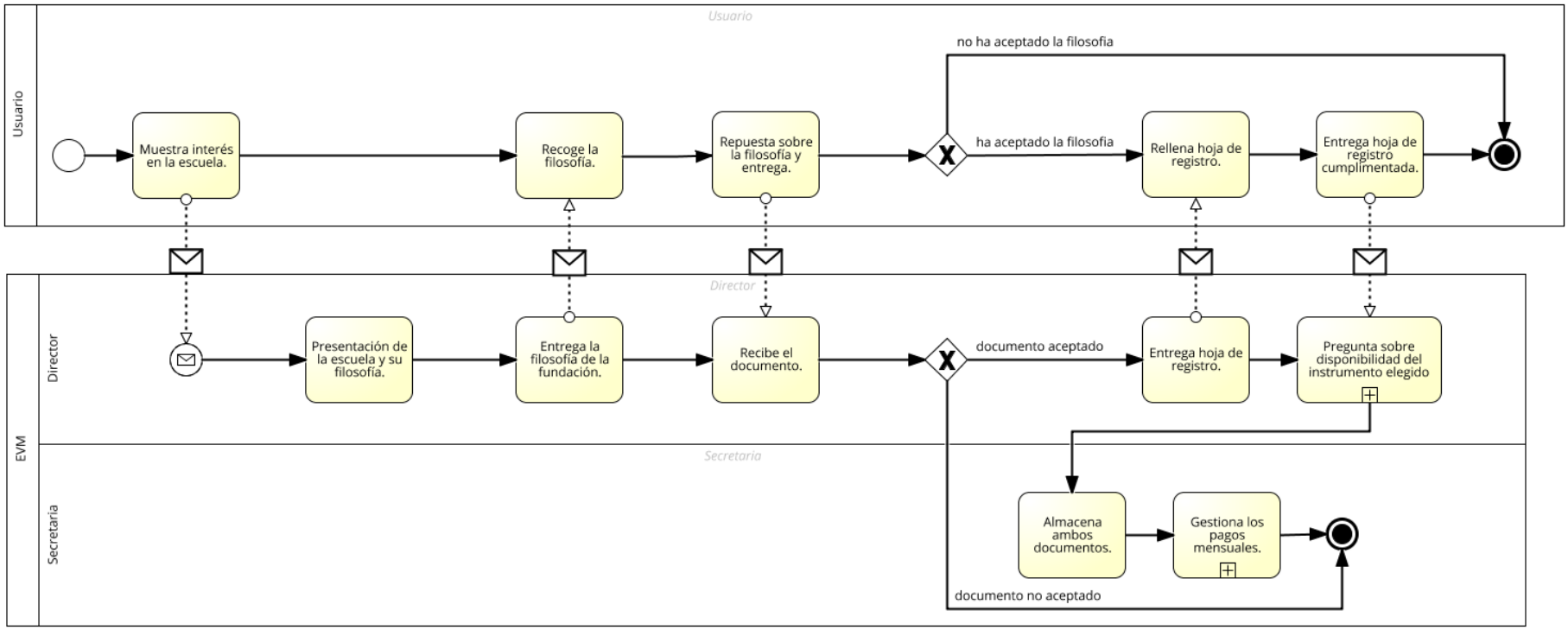


Imagen 1: BPMN registro

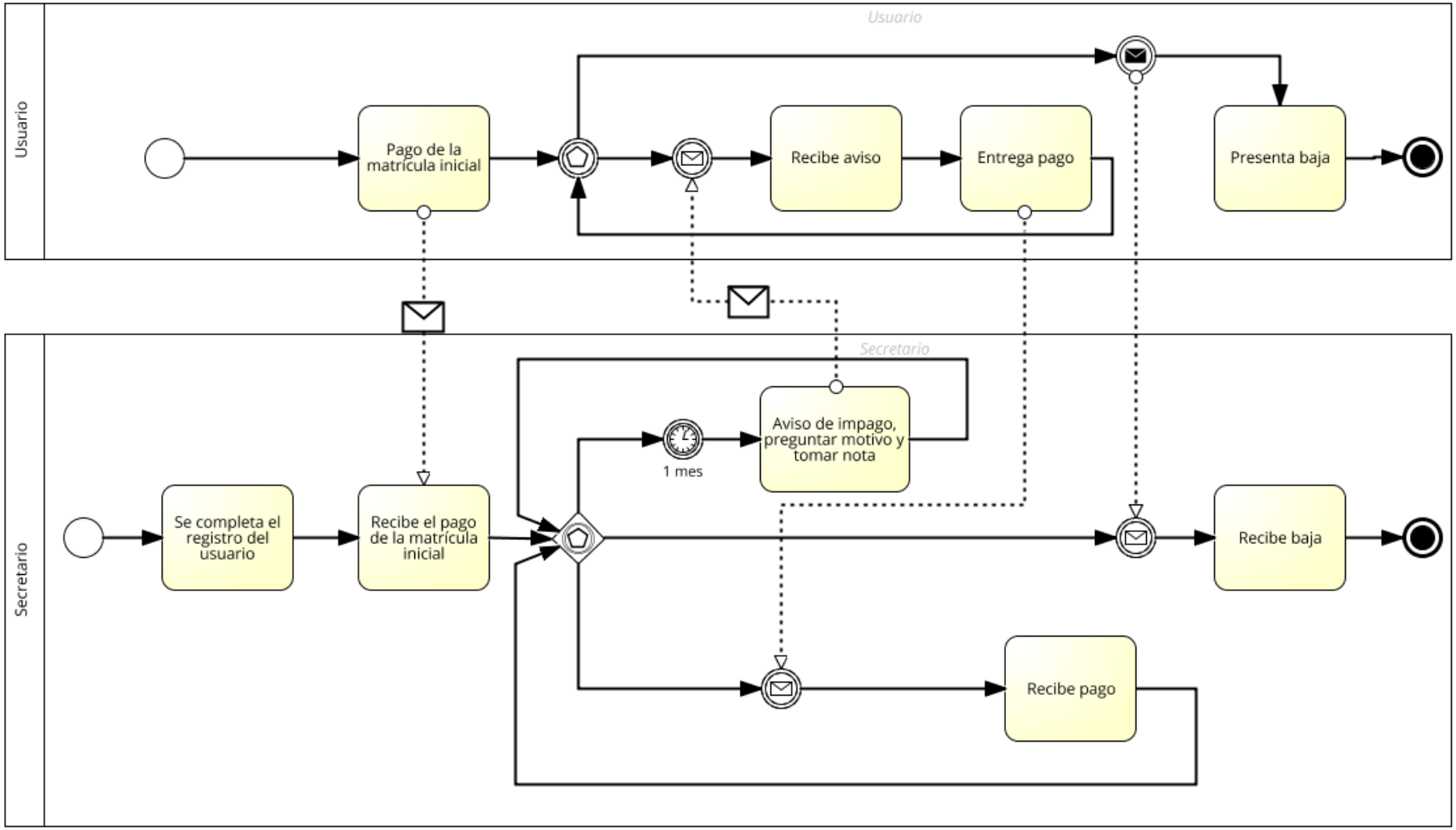


Imagen 2: BPMN pagos

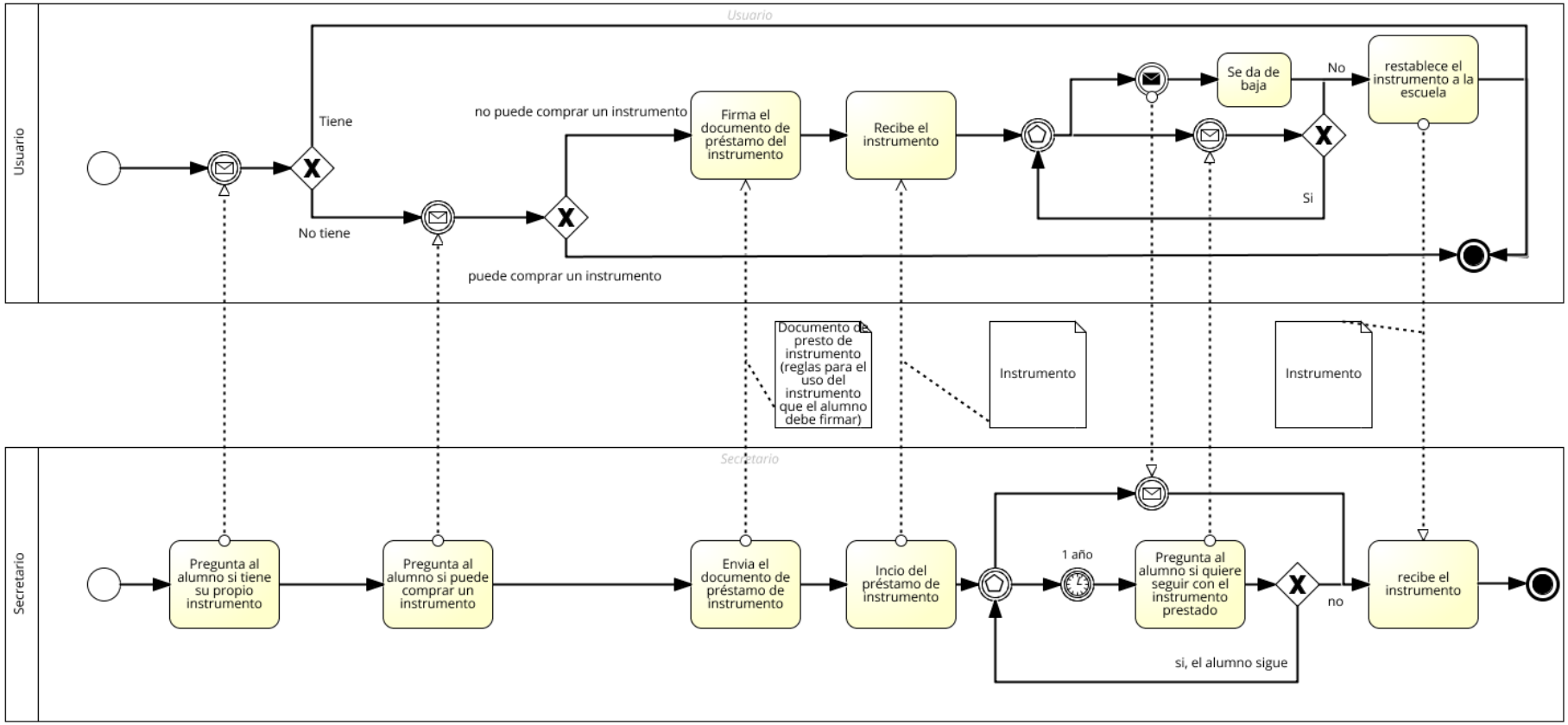


Imagen 3: BPMN préstamos

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.

PRF-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

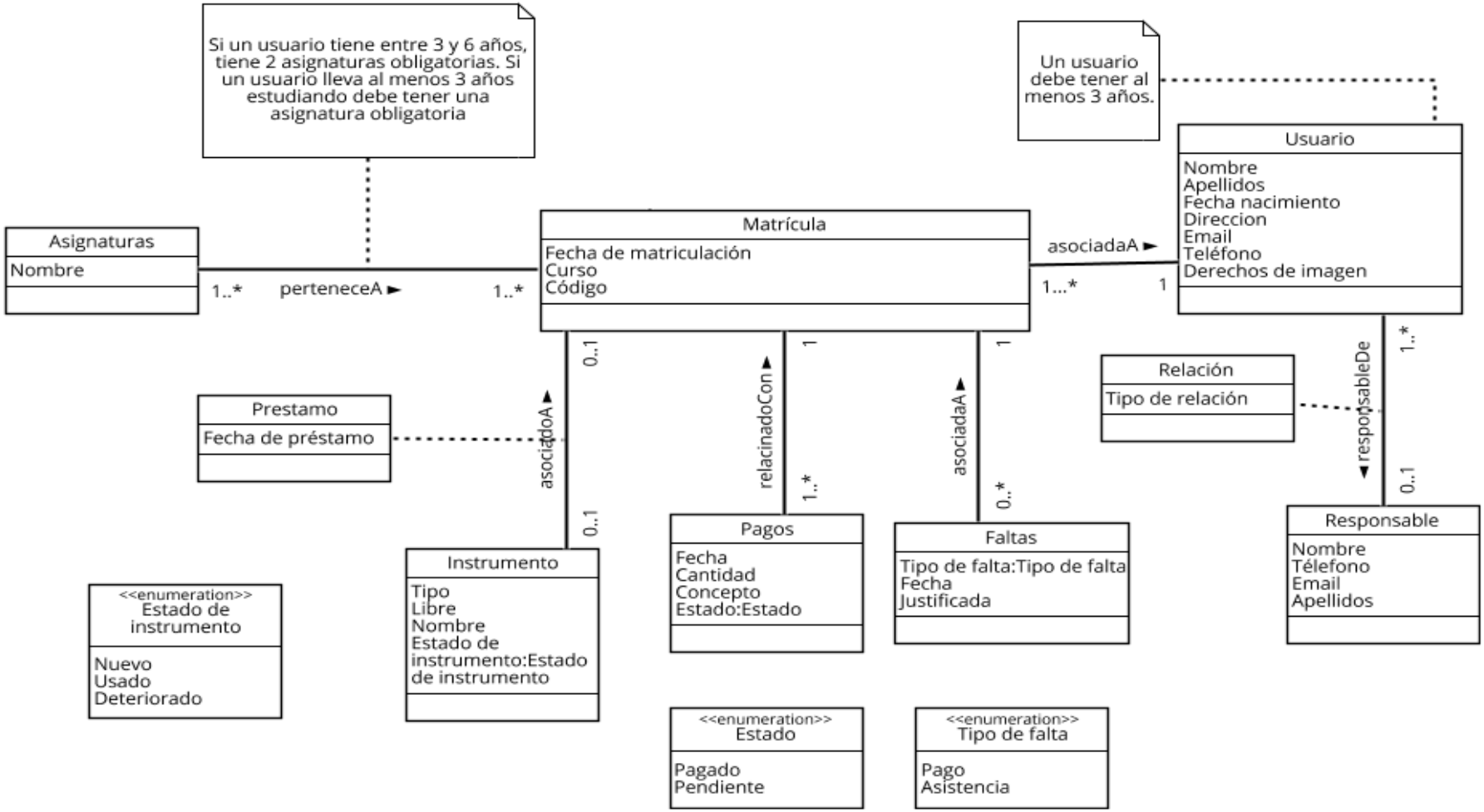

Si un usuario tiene entre 3 y 6 años, tiene 2 asignaturas obligatorias. Si un usuario lleva al menos 3 años estudiando debe tener una asignatura obligatoria
Un usuario debe tener al menos 3 años.
Usuario
Nombre
Apellidos
Fecha nacimiento
Direccion
Email
Teléfono
Derechos de imagen
Asignaturas
Nombre
1..*
perteneceA
1..*
Matrícula
Fecha de matriculación
Curso
Código
asociadaA
1...*
1
Prestamo
Fecha de préstamo
asociadoA
0..1
0..1
relacionadoCon
1
1..*
asociadaA
1
0..*
Relación
Tipo de relación
responsableDe
1..*
0..1
Responsable
Nombre
Télefono
Email
Apellidos
<<enumeration>>
Estado de instrumento
Nuevo
Usado
Deteriorado
Instrumento
Tipo
Libre
Nombre
Estado de instrumento:Estado de instrumento
Pagos
Fecha
Cantidad
Concepto
Estado:Estado
Faltas
Tipo de falta:Tipo de falta
Fecha
Justificada
<<enumeration>>
Estado
Pagado
Pendiente
<<enumeration>>
Tipo de falta
Pago
Asistencia

Imagen 4: UML

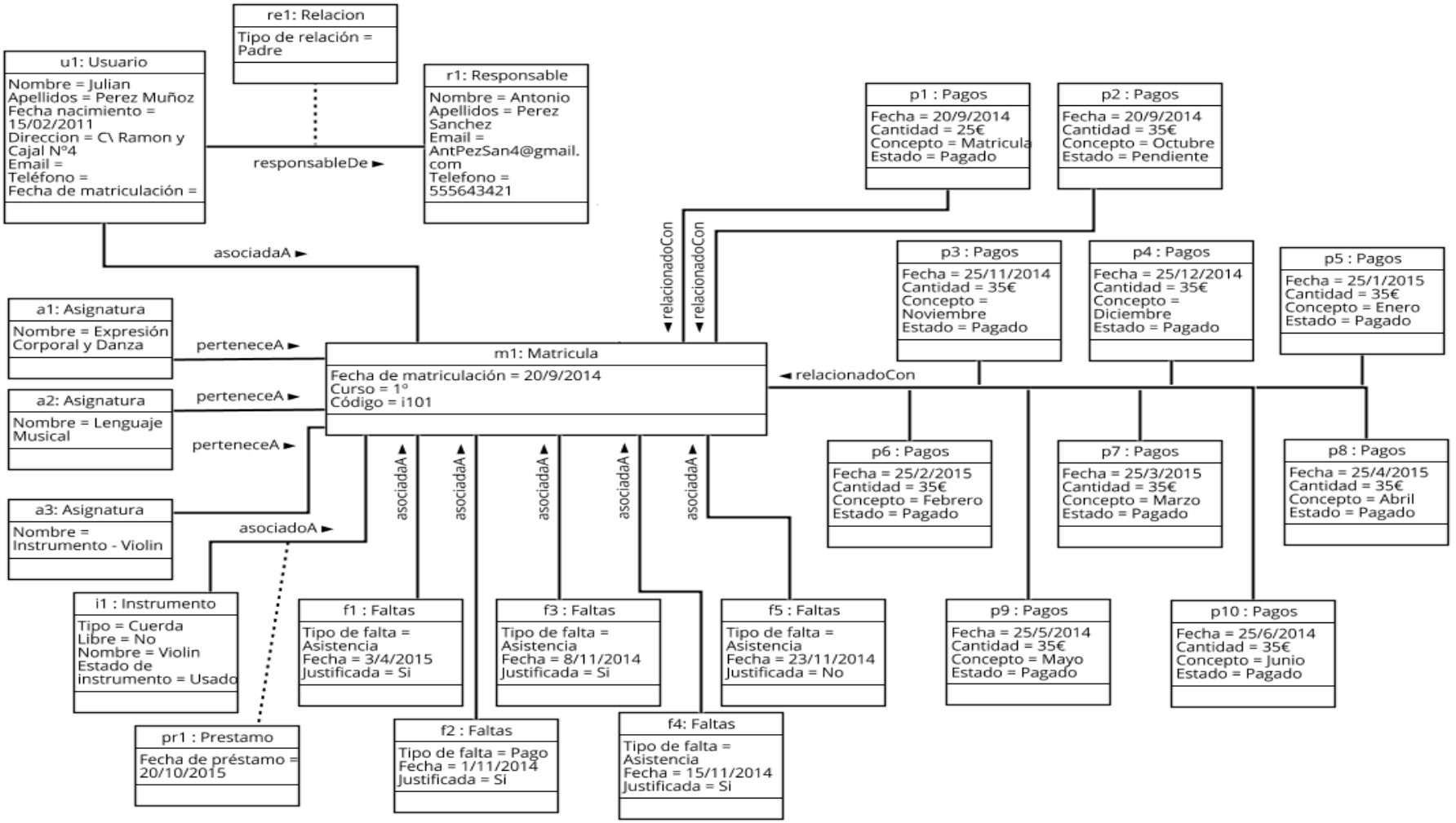


Imagen 5: Escenario de pruebas 1

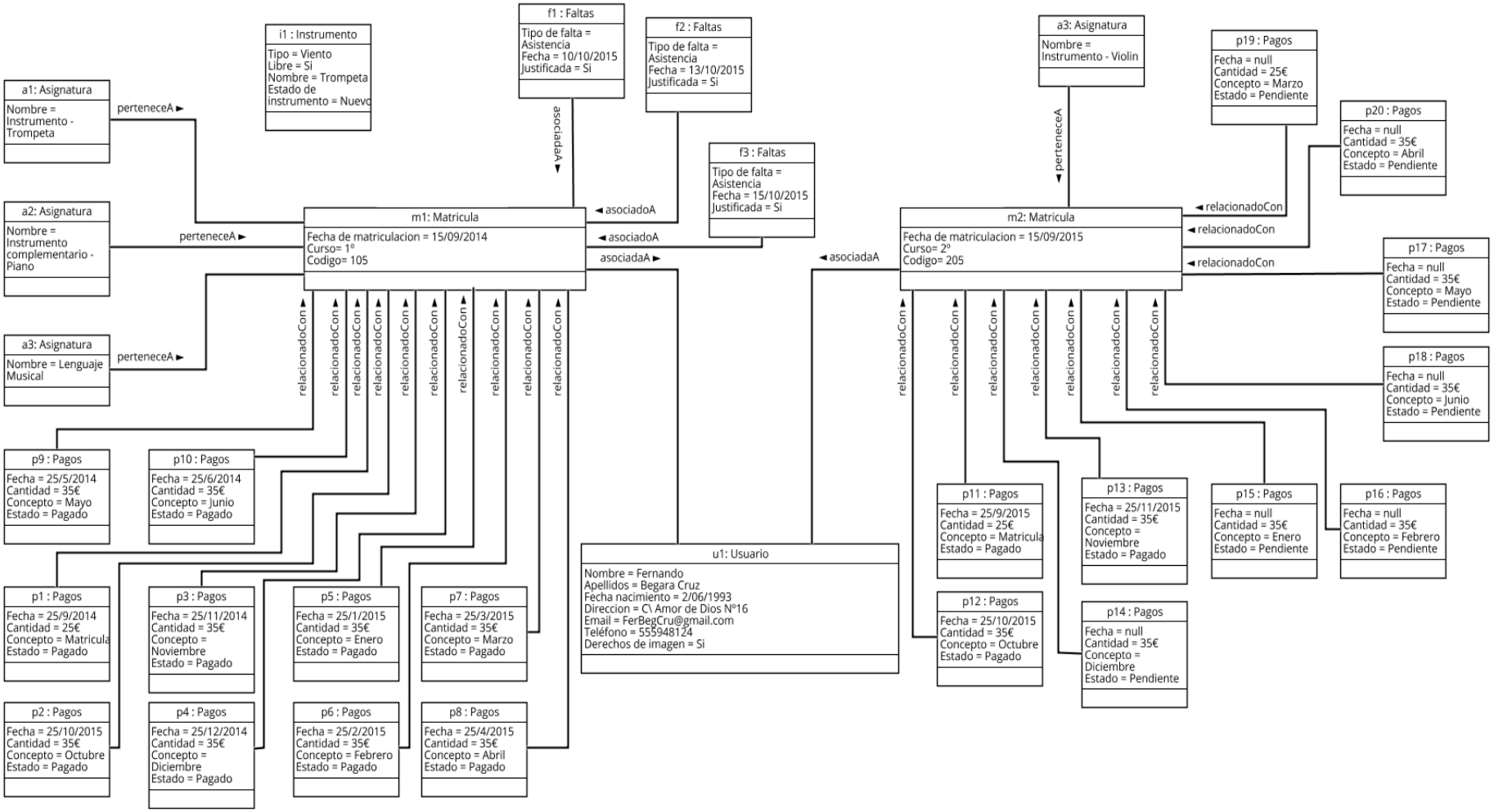


Imagen 6: Escenario de pruebas 2

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

18 de Noviembre de 2015

Siendo las 19:00 hrs. del presente día, se reunieron en las instalaciones de Espacio Vida & Música, Myriam Olmo González, encargada de la logística de la escuela y Carlos Muñoz de Sousa, estudiante de la Universidad de Sevilla.

Se revisó el plan de actuación de nuevos alumnos matriculados y el procedimiento a seguir en la base de datos del centro.

Proporcionando al sujeto ejemplo ilustrativo de los documentos usados.

No habiendo otro asunto que tratar, termina la sesión.

Myriam Olmo González

Carlos Muñoz de Sousa

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.

ACTA

28 de Diciembre de 2015

Siendo las 19:30h del presente día, se reunieron en el despacho de Espacio, Vida & Música, Myriam Olmo González, encargada de la logística de la escuela y Carlos Muñoz de Souza, estudiante de la Universidad de Sevilla.

Se habló sobre la organización de los grupos por curso, y de las asignaturas comunes que se imparten en ellas.

Proporcionando al sujeto ejemplo ilustrativo de los documentos que almacenan esta información.

No habiendo otro asunto que tratar, termina la sesión.

Imagen 7: Folleto promocional

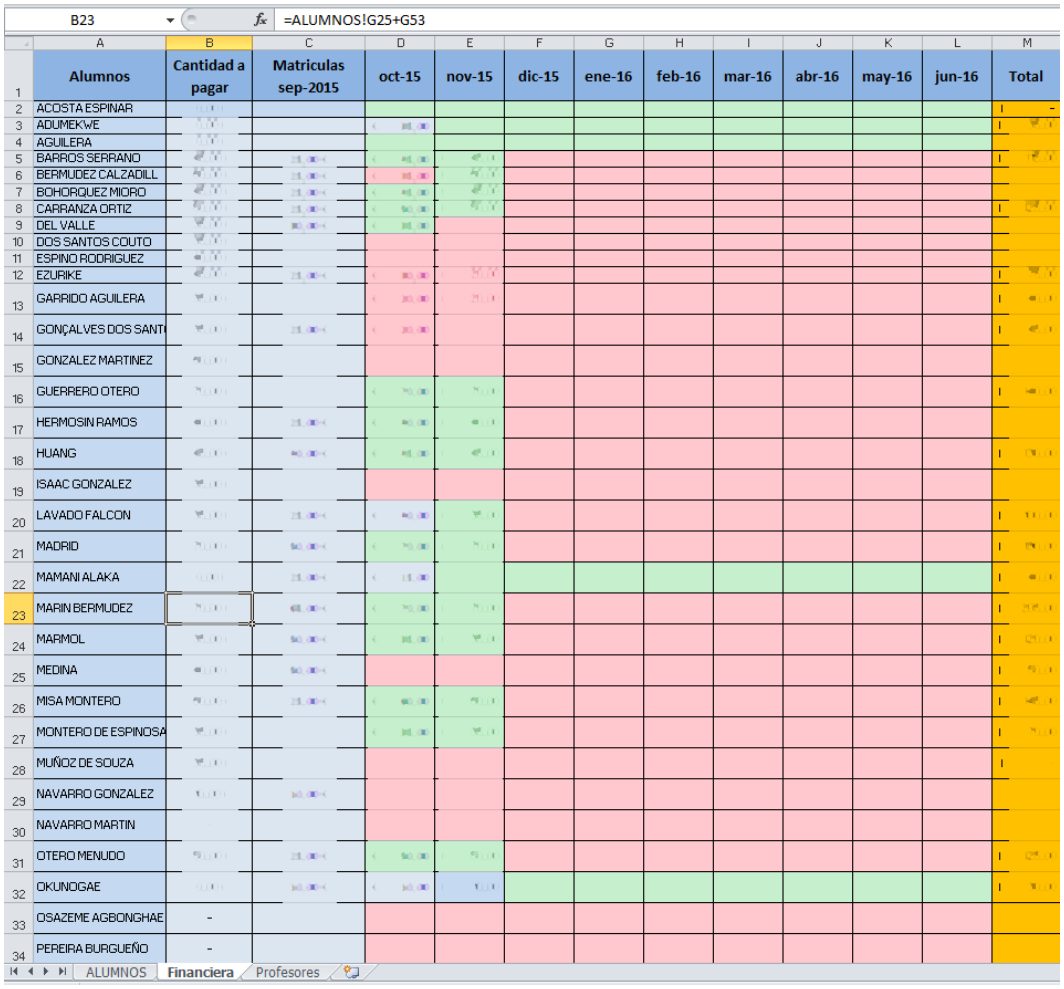
G3 fx 60

Barra de fórmulas

NOMBRE DE FAMILIA	Padres/Responsables del alumno	Fecha de nacimiento	Instrumento	Ayuda Financiera Total	Profesores del alumno	Presencia a clase		Derechos de imagen	Dirección	Instrumento	Teléfono	E-Mail
						INSTRUMENTO	GRUPAL					
MISA MONTERO	Jorge Misa Morales y Alicia Montero de Espinosa	15/07/2000	Trompa	[illegible]	ANTONIO		Miércoles	SI	[illegible]	PROPIO	[illegible]	[illegible]
OKUNOGAE	Framk y Faith Okunogae	19/08/2009	Violín	[illegible]	ESTER			SI	[illegible]	PRESTADO	[illegible]	
MUÑOZ DE SOUZA	Elaine Cristina de Souza Pereira y Jose Manuel Redondo	05/04/1996	Viola	[illegible]	ESTER		Miércoles	SI	[illegible]	PRESTADO	[illegible]	[illegible]
STEVE	-	14/05/1986	Trombón	[illegible]	ANTONIO		Miércoles	SI	[illegible]	PRESTADO		
ADUMEKWE	Victor y Lilian Adumekwe	28/09/2009	Violín	[illegible]	ESTER			Si	[illegible]	PRESTADO	[illegible]	
VILLAVERDE PIZARRO	Antonio Villaverde y Carmen Mª Pizarro	02/11/2010	Violín	[illegible]	REBECA		Martes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
		26/09/2003	Violoncello		CARLOS		Martes	SI		PRESTADO		
GONÇALVES DOS SANTOS	Monica Gonçalves y Jose Eduardo	25/04/2002	Violín	[illegible]	DAVID		Miércoles	SI	[illegible]	PROPIO	[illegible]	
ROSALES SALAZAR	Wilson Rosales y Jenny Salazar	25/11/2004	Violín	[illegible]	ESTER		Martes	Si	[illegible]	-	[illegible]	
BARROS SERRANO	André Alexandre Serrano	21/04/2003	Violoncello	[illegible]	CARLOS		Martes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
	Suellen Alves de Barros	28/08/2004	Violoncello		CARLOS		Martes	SI		PRESTADO		
VILLARROEL GUZMAN	Franz Villarroel y Janete Guzmán	01/12/2000	Violín	[illegible]	ESTER		Miércoles	SI	[illegible]	PRESTADO	[illegible]	[illegible]
					DAVID		Miércoles	SI		PRESTADO		[illegible]
WEI LIU	Jun Wei Liu	03/02/2010	Violín	[illegible]	-		Lunes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
	Xufen Wang	01/10/1997	Violín		-		Lunes	SI		PRESTADO		
MARMOL	Juan Marmol y Mª Mercedes	21/11/2002	Violín	[illegible]	ESTER		Martes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
RODRIGUEZ RUDA	Juan Carlos Rodriguez y Antonia Mª Ruda	30/11/1999	Violín y Percusión	[illegible]	ESTER Y DAVID		Miércoles	SI	[illegible]	PROPIO	[illegible]	[illegible]
MADRID	Victoria Madrid	25/04/2009	Trompeta	[illegible]	ANTONIO		Martes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
		29/08/1947	Flauta		MYRIAM		Martes	SI		PROPIO		
CARRANZA ORTIZ	Juan Antonio Carranza Andrade y Rosario Ortiz Aguilar	29/12/2006	Violín	[illegible]	ESTER		Martes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
		28/10/2001	Violoncello		CARLOS		Martes	SI		PRESTADO		
		08/09/2012	Violín		-		Lunes	SI		PRESTADO		
MARIN BERMUDEZ	Fernando Marín Bermudez (ABUELO) y Sandra Marín de Sard	02/01/2011	Violín	[illegible]			Lunes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
		19/01/1957	Batería		DAVID		Lunes	SI		-		
ROSSI FERNANDEZ	José María Rossi y Ana María Fernandez	29/06/2003	Violín	[illegible]	JESUS		Miércoles	SI	[illegible]	PRESTADO	[illegible]	[illegible]
LAVADO FALCON	Auxi Lavado Zafra	16/06/2003	Violín	[illegible]	REBECA		Martes	SI	[illegible]	PRESTADO	[illegible]	[illegible]
OTERO MENUDO	José Manuel Otero	28/02/2008	Violín	[illegible]	ESTER		Miércoles	SI	[illegible]	PRESTADO	[illegible]	[illegible]
	José Manuel Otero	09/05/2010	Violín		REBECA		Jueves	SI		PRESTADO		

ALUMNOS | Financiera | Profesores

Imagen 8: Tabla de Excel con los registros de los alumnos



B23 =ALUMNOS!G25+G53

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Alumnos	Cantidad a pagar	Matriculas sep-2015	oct-15	nov-15	dic-15	ene-16	feb-16	mar-16	abr-16	may-16	jun-16	Total
2	ACOSTA ESPINAR												-
3	ADUMEKWE												
4	AGUILERA												
5	BARROS SERRANO												
6	BERMUDEZ CALZADILL												
7	BOHORQUEZ MIORO												
8	CARRANZA ORTIZ												
9	DEL VALLE												
10	DOS SANTOS COUTO												
11	ESPINO RODRIGUEZ												
12	EZURIKE												
13	GARRIDO AGUILERA												
14	GONÇALVES DOS SANT												
15	GONZALEZ MARTINEZ												
16	GUERRERO OTERO												
17	HERMOSIN RAMOS												
18	HUANG												
19	ISAAC GONZALEZ												
20	LAVADO FALCON												
21	MADRID												
22	MAMANI ALAKA												
23	MARIN BERMUDEZ												
24	MARMOL												
25	MEDINA												
26	MISA MONTERO												
27	MONTERO DE ESPINOSA												
28	MUÑOZ DE SOUZA												
29	NAVARRO GONZALEZ												
30	NAVARRO MARTIN												
31	OTERO MENUDO												
32	OKUNOGAE												
33	OSAZEME AGBONGHAE	-											
34	PEREIRA BURGUEÑO	-											

ALUMNOS | Financiera | Profesores

Imagen 9: Tabla de Excel con los registros de pagos