

Proyecto M.A.R.C

Rubén Bueno Menéndez

Manuel Jesús Jiménez Navarro

Alejandro Monteseirín Puig

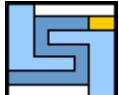
Camila Reyes Aroca

Tutor: Octavio Martín Díaz



Índice

1. Introducción al problema.....	3
2. Glosario de términos	4
3. Modelo de negocio	5
4. Visión general del sistema.....	11
5. Catálogo de requisitos.....	12
5.1. Mapa de historias de usuario.....	12
5.2. Requisitos generales / objetivos	13
5.3. Requisitos de información	14
5.4. Reglas de negocio	16
5.5. Requisitos funcionales.....	19
5.6. Requisitos no funcionales	28
6. Pruebas de aceptación	29
7. Modelo conceptual	38
7.1. Diagramas de clases UML con restricciones	38
7.2. Escenarios de prueba	43
8. Matrices de trazabilidad.....	53
9. Modelo relacional en 3FN	59
9.1. Relaciones obtenidas	59
9.2. Justificación de la estrategia de transformación de jerarquías	60
10. Modelo tecnológico.....	62



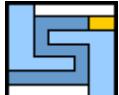
1. Introducción al problema

Este proyecto surge a partir del proyecto de Enseñanza Virtual de la Universidad de Sevilla, un proyecto diseñado para apoyar la docencia presencial por medio de recursos tecnológicos propios de cualquier formación online, que permite compartir material docente y realizar anuncios o publicar calificaciones, entre otras funciones.

Sin embargo, dicha plataforma virtual a veces puede resultar insuficiente, pues no siempre funciona correctamente y no ofrece tantas funciones como a mucho personal le gustaría.

Tras el estudio del dominio del problema, las necesidades a satisfacer, la situación actual y tras el proceso de entrevistas a varias personas pertenecientes al personal de la Escuela Técnica Superior de Ingeniería Informática, se llega a la conclusión de que es necesario una **herramienta web** que permita crear una estructura de soporte más eficaz que embarque muchas más posibilidades a la hora de la **gestión de la universidad**.

Por ello, el objetivo principal de este proyecto es la creación de dicha herramienta web que complemente a la plataforma de enseñanza virtual actual, además de satisfacer nuevas necesidades que han surgido a lo largo de los años, como por ejemplo la necesidad de herramientas que permitan la gestión de tutorías entre profesores y alumnos, controlar la asistencia a las clases o herramientas que permitan interactuar a los alumnos entre ellos.

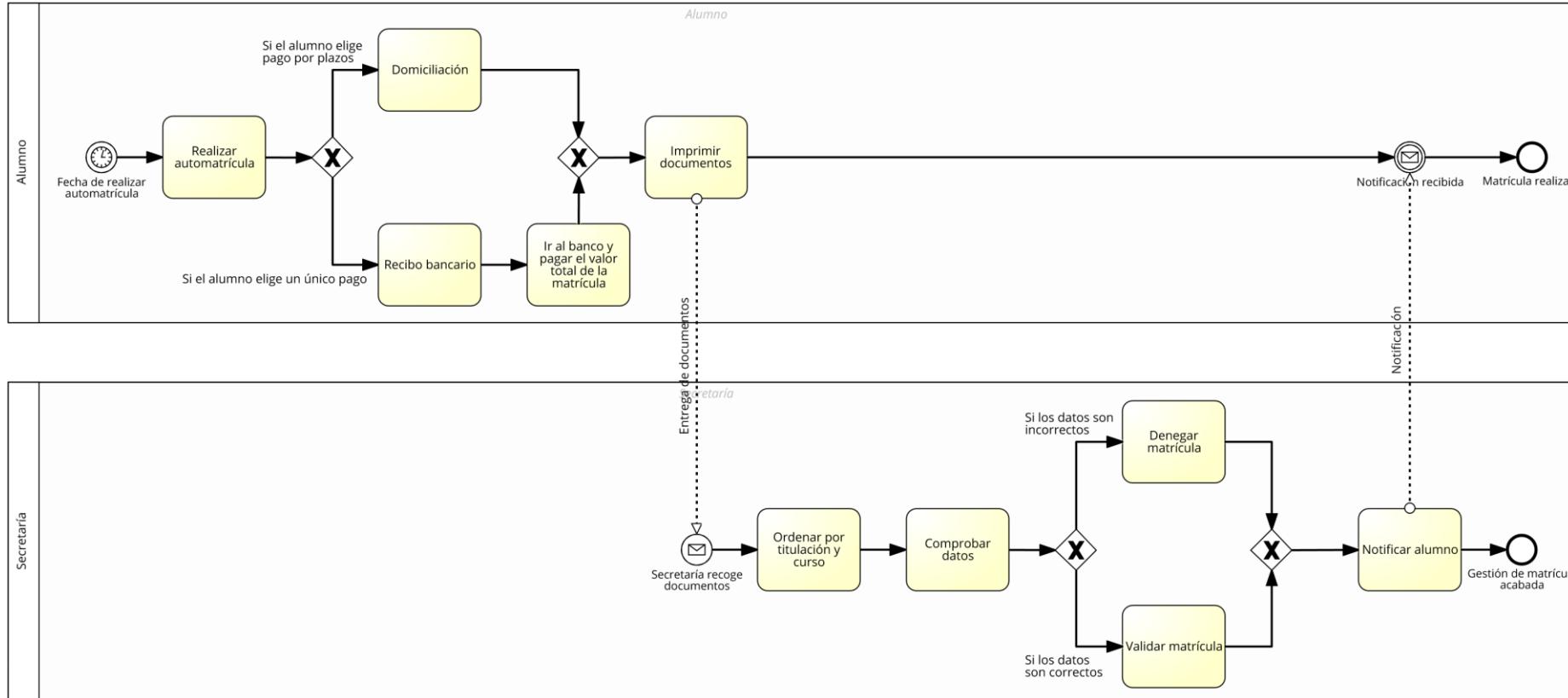


2. Glosario de términos

- **Alumno:** persona que acude a las clases impartidas por profesores en un horario determinado y realiza las actividades que estos indican.
- **Anuncio:** información, noticia o advertencia destinada al alumnado.
- **Apuntes:** resúmenes o tareas realizadas por los profesores o alumnos que podrán ser compartidos.
- **Asignatura:** materia impartida por profesores y recibida por alumnos matriculados.
- **Aula:** lugar físico en el que los profesores imparten una asignatura y los alumnos acuden.
- **Beca:** posible ayuda que recibe el alumno a la hora de matricularse. Puede ser ordinaria, de movilidad o de empresa.
- **Calendario:** registro de los días del curso organizado, con información sobre exámenes, periodo lectivo, etc.
- **Calificación:** valoración en función de la nota recibida por un alumno en una determinada asignatura.
- **Centro:** lugar físico donde los profesores y alumnos acuden para impartir clases y aprender sobre las diferentes materias respectivamente y se llevan a cabo diferentes funciones sobre dicho lugar.
- **Convocatoria:** acción que se realiza para que los alumnos puedan examinarse oficialmente de una asignatura.
- **Crédito:** unidad de valoración de una asignatura o un curso, equivalente a un determinado número de horas lectivas.
- **Curso:** tiempo señalado en cada año para asistir a las asignaturas impartidas.
- **Departamento:** unidad administrativa de docencia e investigación, formada por una o varias cátedras de intereses afines.
- **Despacho:** espacio reservado para profesores.
- **Examen:** prueba que se realiza a los alumnos para ser evaluados de las asignaturas.
- **Expediente:** conjunto de calificaciones e incidencias de los alumnos.
- **Foro:** sección en la que los alumnos y profesores pueden discutir sobre determinados asuntos relacionados con las asignaturas.
- **Grado:** titulación a la que está matriculado un alumno.
- **Grupo:** conjunto de alumnos matriculados en determinadas asignaturas en las cuales los profesores podrán controlar la asistencia de los alumnos.
- **Horario:** tiempo durante el cual se desarrolla o se realiza determinadas actividades tanto para los alumnos como para los profesores.
- **Matrícula:** inscripción realizada por un alumno a la hora de realizar un grado.
- **Nota:** puntuación adquirida entre 0 y 10 que se obtiene aplicando unos parámetros de ponderación establecidos por un profesor que determina la calificación de una asignatura.
- **Profesor:** persona que imparte clases en el centro y que enseña a sus alumnos sobre la asignatura que este imparte, pasa lista de los alumnos que acuden, además de poner los exámenes a los alumnos, proponer trabajos prácticos y ofrecerle un horario de tutorías. Podrá ser catedrático, titular, contratado doctor, colaborador, ayudante doctor, ayudante o interino.
- **Secretaría:** lugar compuesto de personas que realizan la gestión de las matrículas de los alumnos.
- **Tutoría:** tiempo dedicado por profesores al alumnado para ayudarles en un horario determinado.

3. Modelo de negocio

Matriculación del alumno (modelo BMPL)





Matriculación del alumno (descripción textual)

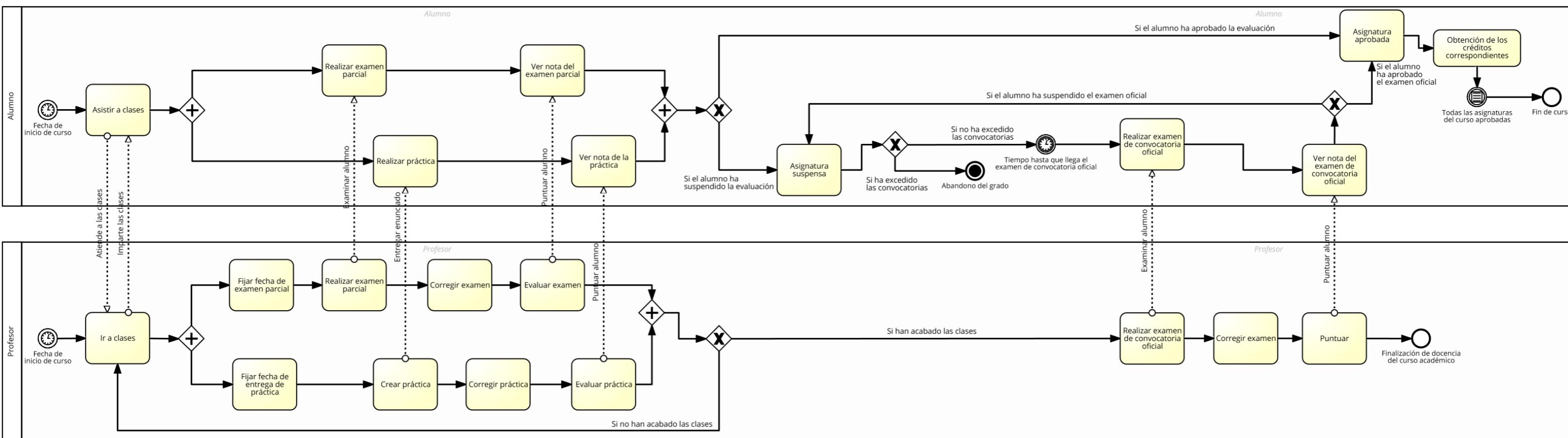


Un alumno quiere matricularse en un grado en la universidad. Para ello el alumno debe esperar a la fecha adecuada según su expediente para realizar la automatrícula. Dependiendo del tipo de pago que el alumno quiera realizar a la hora de llenar la automatrícula, tendrá que hacerlo por domiciliación (en el caso de que quiera por plazos) o por recibo bancario (en el caso de que quiera un único pago), en el caso de elegir esta última, tendrá que ir al bando a realizar el pago total de la matrícula.

Una vez está completa la automatrícula, el alumno debe imprimir los documentos y entregarlos en el buzón de secretaría. El personal de secretaría recogerá la matrícula del alumno, y para cada uno de los alumnos, organizará las matrículas por titulación y curso. Comprobará los datos y en el caso de que sean incorrectos, se le notificará al alumno, el cual tendrá que presentar los documentos necesarios en el buzón de secretaría y se repetirá el proceso. Cuando estén todos los datos correctos, se notificará al alumno que está todo bien y se validará la matrícula del alumno, quedando oficialmente matriculado en la universidad.



Curso académico del alumno (modelo BMPL)





Curso académico del alumno (descripción textual)



Un alumno quiere cursar un grado en la universidad, para ello debe estar matriculado previamente. El alumno deberá esperar al comienzo del curso.

Una vez comienza el curso, tanto los alumnos como los profesores asisten a clase a atender e impartir respectivamente. El profesor fijará una fecha de examen parcial y llegado el día examinará al alumno, el cual realizará el examen. Una vez realizado el profesor corregirá dicho examen y puntuará al alumno. Simultáneamente también se fijará una fecha de entrega de práctica y llegado el día se realizará la entrega de la práctica. Una vez entregada el profesor corregirá dicha práctica y puntuará al alumno.

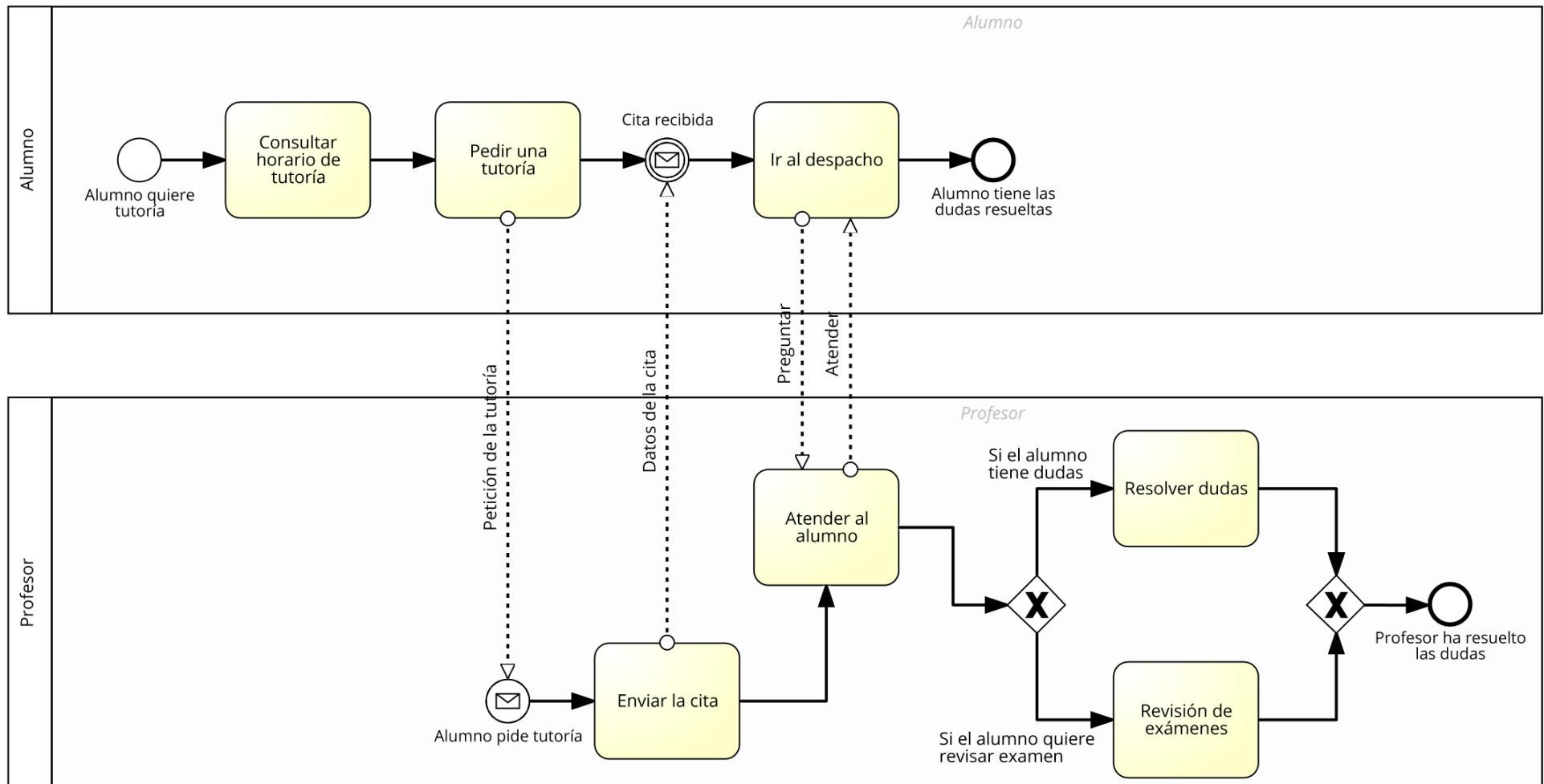
Si las clases no han acabado aún, el profesor repetirá el proceso hasta que acaben, realizando más exámenes parciales y prácticas. El alumno obtendrá la nota media de parciales y prácticas, y en el caso de estar aprobado, tendrá la asignatura aprobada.

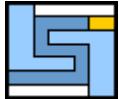
Si en caso contrario está suspenso, tendrá la asignatura suspensa y por ello tendrá que esperar a la fecha de los exámenes de convocatoria oficial, la cual realizará el profesor cuando acaban las clases y el alumno lo hará. El profesor le corregirá el examen y se lo puntuará. El alumno verá la nota de dicho examen y si ha aprobado, tendrá la asignatura aprobada. En caso contrario, si ha suspendido, tendrá que esperar de nuevo a la siguiente convocatoria para realizar el examen.

En el caso de que el alumno exceda el número de convocatorias oficiales, entonces acabará el proceso de ciclo del alumno sin haber obtenido el título del grado.

Si tiene una asignatura aprobada, obtendrá los créditos correspondientes de dicha asignatura. En el caso de que el alumno haya conseguido todos los créditos necesarios de cada asignatura de un curso, habrá superado dicho curso y acabará el proceso de ciclo del alumno para ese curso académico.

Petición de tutoría (modelo BMPL)





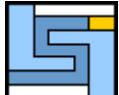
Petición de tutoría (descripción textual)



A un alumno le surgen dudas las cuáles podrá resolver un profesor. Para ello consulta el horario de tutoría del profesor y a continuación pide la cita dentro del horario disponible que más le convenga. El profesor recibirá su petición de tutoría y le enviará los datos de la cita en función de la disponibilidad que tenga.

El alumno recibirá los datos de la cita y acudirá al despacho basándose en ellos. Una vez allí, se dará una interacción entre profesor y alumno donde el alumno si tiene dudas las preguntará de manera que el profesor le atienda o bien si el alumno quiere revisar un examen, el profesor se lo facilitará y resolverá sus cuestiones.

Una vez que el alumno tiene todas sus dudas resueltas, finaliza el proceso.



4. Visión general del sistema

El entorno de trabajo engloba al profesor, el alumno y todo lo que le rodea. Nos centraremos especialmente en proceso de **matriculación del alumno**, las **clases**, los **exámenes** y las **tutorías**.

La matriculación del alumno es algo importante y primerizo, ya que, sin estar matriculado, no tiene derecho a realizar las siguientes funciones, por ello es un requisito que un alumno pueda matricularse de manera rápida y cómoda.

Las clases suelen tener un horario fijo por cuatrimestre, a excepción de las clases prácticas que pueden variar en función de los distintos subgrupos. Las asignaturas de 6 créditos tienen dos clases de 2 horas por semana durante un cuatrimestre, y las de 12 créditos durante los dos cuatrimestres.

También cabe comentar la posibilidad de pasar lista en una clase, ya sea para tener unos datos meramente estadísticos o incluso para penalizar las faltas en caso de las clases obligatorias (como algunas prácticas).

En cuanto a los exámenes podemos distinguir una división entre los parciales de evaluación continua y los de convocatoria oficial, en los primeros tanto la fecha como las preguntas son decisión del profesor además el sistema de evaluación también está a decisión del profesor, pudiendo poner varios exámenes y evaluarlos a voluntad. Las convocatorias oficiales suelen tener una fecha fija y ser gestionados por un grupo de profesores del departamento de la asignatura en cuestión.

El funcionamiento de las tutorías puede variar de un profesor a otro, pero en resumen cada profesor tiene asignada una serie de horas en el despacho en las cuales atiende a alumnos y los ayuda, ya sea con dudas de la asignatura o trámites administrativos (como asignar grupos de laboratorio o gestionar un cambio de grupo).



5. Catálogo de requisitos

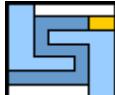
5.1. Mapa de historias de usuario

Historias de Usuario												
Gestión de matriculación y becas de alumnos			Gestión de docencia de profesorado			Gestión de clases		Gestión de tutorías		Gestión de expedientes		
RI-001 - Información sobre la matrícula	RI-002 - Información sobre el expediente académico	RI-003 - Información sobre las asignaturas	RI-004 - Información sobre los grados	RI-005 - Información sobre las becas	RI-006 - Información sobre las calificaciones	RI-007 - Información sobre los departamentos	RI-008 - Información sobre los despachos	RI-009 - Información sobre las aulas	RI-010 - Información sobre los profesores	RI-011 - Información sobre las tutorías	RI-012 - Información sobre los grupos	
RN-001 - Email del alumno	RN-007 - Calificaciones del expediente con única convocatoria y curso	RN-002 - Créditos de las asignaturas RN-008 - Asignaturas optativas de un grado con mismo número de créditos RN-009 - Asignaturas optativas deben ser cuatrimestrales	RN-010 - Cantidad de créditos optativos a cursar	RN-003 - Duración de una beca RN-004 - Cantidad mínima de la cuantía total fija de una beca	RN-011 - Valor de una nota RN-012 - Calificación debe corresponder con valor de nota	-	-	RN-005 - Capacidad de un despacho	RN-006 - Capacidad de un aula	RN-013 - Asignaturas impartidas por un profesor de un único departamento RN-014 - Dedicación de créditos de un profesor a una asignatura RN-015 - Créditos máximos que puede impartir un profesor	RN-016 - Días de tutoría RN-017 - Duración de una tutoría	-
RF-001 - Matriculación de una asignatura para un alumno RF-002 - Anulación de una asignatura para un alumno RF-004 - Garantía de alumno matriculado en una asignatura RF-006 - Informe de asignaturas de un alumno ordenadas por curso RF-014 - Informe del total de alumnos matriculados en la universidad	RF-005 - Informe de medias de calificaciones del alumno agrupados por asignatura RF-007 - Informe de expediente de un alumno RF-008 - Asociación de un departamento a una asignatura RF-028 - Informe de notas de un expediente RF-029 - Informe de nota media de un expediente	RF-032 - Informe de las asignaturas obligatorias de un grado RF-033 - Informe de las asignaturas optativas de un grado RF-034 - Informe del número de créditos optativos de un grado RF-035 - Informe del número de créditos obligatorios de un grado RF-036 - Informe del número de créditos troncales de un grado RF-037 - Informe del número de créditos totales de un grado RF-038 - Informe de los departamentos pertenecientes a un grado RF-039 - Informe de los profesores que imparten en un grado RF-040 - Informe de las asignaturas que pertenecen a un grado y curso RF-041 - Informe de las asignaturas anuales que pertenecen a un grado	RF-009 - Asociación de una beca a un alumno RF-010 - Asociación de la duración de una beca de un alumno RF-011 - Asociación de la cuantía total fija de una beca RF-012 - Asociación de la cuantía total variable de una beca RF-013 - Informe de becas obtenidas por el alumno	RF-003 - Evaluación de un alumno RF-030 - Evaluación de un examen	RF-017 - Informe de los profesores que pertenecen a un departamento RF-018 - Informe de las asignaturas que pertenecen a un departamento RF-020 - Eliminación de una asignatura a un departamento RF-021 - Asignación de un profesor a un departamento RF-022 - Eliminación de un profesor a un departamento RF-023 - Garantía de profesor asignado a una asignatura RF-024 - Garantía de que todas las asignaturas de un departamento están asignadas	RF-016 - Informe del total de espacios agrupados por el tipo RF-027 - Informe de profesores asociados a un despacho	RF-015 - Días de tutoría RF-025 - Informe de profesores agrupados por asignatura RF-043 - Informe del departamento asociado a un profesor RF-044 - Informe de la dedicación total de un profesor RF-045 - Informe de las asignaturas que imparte un profesor RF-046 - Informe de los créditos impartidos asociados a un profesor RF-047 - Asociación de una categoría a un profesor RF-050 - Adición de una asignatura y dedicación que debe impartir un profesor RF-051 - Eliminación de una asignatura impartida por un profesor	RF-026 - Informe de tutorías agrupadas por profesores RF-042 - Informe de las tutorías asociadas a un profesor RF-048 - Adición de una cita de tutoría para un profesor RF-049 - Eliminación de una cita de tutoría para un profesor	RF-019 - Informe del grupo con mayor número de alumnos			



5.2. Requisitos generales / objetivos

- **Gestión de matriculación y becas de alumnos:** se necesita una aplicación que permita llevar a cabo la matriculación de un alumno y la asignación de becas, si corresponden.
- **Gestión de docencia de profesorado:** se necesita una aplicación que permita distinguir de qué tipo de profesor se trata, cantidad de créditos que imparte, asignaturas que imparte, departamento al que pertenece, etc.
- **Gestión de clases:** se necesita una aplicación que permita asignar unas determinadas características de aulas a las clases, los grupos de alumnos que asisten, los profesores que imparten, etc.
- **Gestión de tutorías:** se necesita una aplicación que permita asignar los horarios de tutorías de cada profesor.
- **Gestión de expedientes:** se necesita una aplicación que permita registrar y mantener el expediente de un alumno, con sus correspondientes calificaciones.



5.3. Requisitos de información

RI-001 - Información sobre la matrícula:

Como alumno,

Quiero disponer de la información asociada a la matrícula: datos personales, becas asociadas, grado, pago realizado y asignaturas matriculadas,

Para poder tener control sobre la gestión de la matrícula.

RI-002 - Información sobre el expediente académico:

Como alumno,

Quiero disponer de la información asociada al expediente académico: asignaturas superadas, créditos superados, créditos que faltan por superar, calificaciones, curso académico y convocatoria,

Para poder tener control sobre la gestión del expediente.

RI-003 - Información sobre las asignaturas:

Como alumno,

Quiero disponer de la información asociada a las asignaturas: nombre, temario, fechas de exámenes, tipología, créditos, curso académico y departamento al que pertenece,

Para poder saber la importancia de cada asignatura.

RI-004 - Información sobre los grados:

Como alumno,

Quiero disponer de la información asociada a los grados: nombre, normativa académica, objetivos, competencias, salidas profesionales y académicas, sistema de garantía de calidad y plan de estudios,

Para poder saber cómo afectará durante el curso académico y en el futuro del alumno.

RI-005 - Información sobre las becas:

Como alumno,

Quiero disponer de la información asociadas a las becas: tipo de beca, cuantía total fija, cuantía total variable y duración,

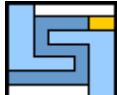
Para tener control sobre los gastos durante el curso académico.

RI-006 - Información sobre las calificaciones:

Como alumno,

Quiero disponer de la información asociada a las calificaciones: asignatura, nota media y convocatoria,

Para tener control sobre el progreso de las asignaturas.



RI-007 - Información sobre los departamentos:

Como alumno,

Quiero disponer de la información asociada a los departamentos: nombre, asignaturas impartidas por el departamento y profesores asociados al departamento,

Para poder saber cómo afectará durante el curso académico y en el futuro del alumno.

RI-008 - Información sobre los despachos:

Como alumno,

Quiero disponer de la información asociada a los despachos: módulo, planta y profesores a los que pertenecen,

Para poder acudir con facilidad.

RI-009 - Información sobre las aulas:

Como alumno,

Quiero disponer de la información asociada a las aulas: módulo, planta, capacidad y tipo de aula,

Para poder acudir con facilidad y tener una idea de cómo será.

RI-010 - Información sobre los profesores:

Como alumno,

Quiero disponer de la información asociada a los profesores: nombre completo, email, tipo de profesor, horario de tutorías y departamento al que pertenece,

Para poder contactar a un profesor fácilmente.

RI-011 - Información sobre las tutorías:

Como alumno,

Quiero disponer de la información asociada a las tutorías: profesor que la imparte, día, hora de comienzo y duración,

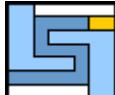
Para gestionar el tiempo y las dudas.

RI-012 - Información sobre los grupos:

Como profesor,

Quiero disponer de la información asociada a las listas de grupos: fecha, asignatura y alumno,

Para poder llevar un control de asistencia.



5.4. Reglas de negocio

RN-001 - Email del alumno:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: un alumno matriculado debe tener un email acabado en @alum.es,

Para garantizar que es un alumno matriculado en la universidad.

RN-002 - Créditos de las asignaturas:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: los créditos de una asignatura deben ser un número positivo distinto de 0,

Para garantizar que no se produzca ningún error a la hora de hacer el sumatorio total de créditos.

RN-003 - Duración de una beca:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: la duración de una beca debe ser como mínimo de 1 mes,

Para garantizar un tiempo realista de disfrute de una beca.

RN-004 - Cantidad mínima de la cuantía total fija de una beca:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: la cuantía total fija de una beca debe ser de al menos 1500€,

Para garantizar que se puedan cubrir las necesidades mínimas de un alumno.

RN-005 - Capacidad de un despacho:

Como gerente de la universidad,

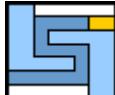
Quiero que se cumpla la siguiente regla de negocio: la capacidad máxima de profesores en un despacho debe ser 3,

Para garantizar un buen reparto de despachos y profesores.

RN-006 - Capacidad de un aula:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: la capacidad máxima de alumnos en un aula debe ser 70 si es de tipo teoría, de 30 si es de tipo práctica o 200 en caso de ser de tipo examen,



Para garantizar un buen reparto de aulas y alumnos.

RN-007 - Calificaciones del expediente con única convocatoria y curso:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: el expediente de un alumno no puede contener calificaciones para más de 1 convocatoria de una misma asignatura y curso,

Para garantizar un registro adecuado del expediente académico para cada convocatoria y curso.

RN-008 - Asignaturas optativas de un grado con mismo número de créditos:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: todas las asignaturas optativas de un grado deben tener el mismo número de créditos,

Para garantizar que no se produzca ningún error a la hora de hacer el sumatorio total de créditos.

RN-009 - Asignaturas optativas deben ser cuatrimestrales:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: todas las asignaturas optativas de un grado deben ser cuatrimestrales,

Para garantizar un reparto adecuado del tiempo dedicado a esas asignaturas.

RN-010 - Cantidad de créditos optativos a cursar:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: el número mínimo de créditos de asignaturas optativas que debe cursar un alumno debe estar comprendido entre 0 y 30,

Para garantizar que no se produzca ningún error a la hora de hacer el sumatorio total de créditos.

RN-011 - Valor de una nota:

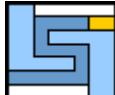
Como profesor,

Quiero que se cumpla la siguiente regla de negocio: el valor de una nota a la hora de evaluar un alumno debe estar comprendida entre 0 y 10,

Para garantizar que todas las asignaturas pueden tener una calificación en función de la nota obtenida bajo el mismo criterio.

RN-012 - Calificación debe corresponder con valor de nota:

Como profesor,



Quiero que se cumpla la siguiente regla de negocio: la calificación de una evaluación debe tener su correspondiente valor de nota,
Para garantizar que una calificación tiene sentido con el valor correspondiente de la nota.

RN-013 - Asignaturas impartidas por un profesor de un único departamento:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: un profesor no puede impartir una asignatura de otro departamento que no corresponda al suyo,
Para garantizar la estructura de docencia en la universidad.

RN-014 - Dedicación de créditos de un profesor a una asignatura:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: el número de créditos dedicados a una asignatura por un profesor debe ser mayor que 0 y menor o igual que el número de créditos de ésta,
Para garantizar el correcto reparto de créditos de un profesor.

RN-015 - Créditos máximos que puede impartir un profesor:

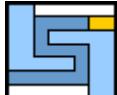
Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: un profesor puede impartir un máximo de 32 créditos o 6 en el caso de ser un profesor ayudante,
Para garantizar el correcto reparto de créditos de un profesor.

RN-016 - Días de tutoría:

Como profesor,
Quiero que se cumpla la siguiente regla de negocio: los días de la semana en donde se puede realizar una tutoría deben estar entre lunes y viernes,
Para garantizar el correcto cumplimiento de horario de docencia de un profesor.

RN-017 - Duración de una tutoría:

Como profesor,
Quiero que se cumpla la siguiente regla de negocio: la duración de una tutoría debe ser de al menos 10 minutos y de máximo 30 minutos,
Para garantizar una buena disponibilidad de tutorías para todos los alumnos.



5.5. Requisitos funcionales

RF-001 - Matriculación de una asignatura para un alumno:

Como usuario,

Quiero que el sistema permita: añadir una asignatura a un alumno,

Para poder matricular a un alumno de una asignatura.

RF-002 - Anulación de una asignatura para un alumno:

Como usuario,

Quiero que el sistema permita: eliminar una asignatura a un alumno,

Para poder anular de una asignatura a un alumno.

RF-003 - Evaluación de un alumno:

Como usuario,

Quiero que el sistema permita: añadir a un alumno una nota de una asignatura para un determinado curso y convocatoria,

Para poder evaluar de una asignatura a un alumno.

RF-004 - Garantía de alumno matriculado en una asignatura:

Como usuario,

Quiero que el sistema permita: obtener un informe que garantice que un alumno está matriculado en una determinada asignatura,

Para verificar que un alumno está matriculado en una asignatura.

RF-005 - Informe de medias de calificaciones del alumno agrupados por asignatura:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre la media de las calificaciones de un alumno agrupado por asignatura,

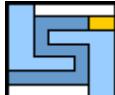
Para poder conocer las calificaciones por asignaturas de un alumno.

RF-006 - Informe de asignaturas de un alumno ordenadas por curso:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre las asignaturas de un alumno ordenadas por curso académico,

Para poder conocer las asignaturas en las que está matriculado un alumno.



RF-007 - Informe de expediente de un alumno:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre el expediente de un alumno,

Para poder conocer el avance académico de un alumno.

RF-008 - Asociación de un departamento a una asignatura:

Como usuario,

Quiero que el sistema permita: añadir en el caso de no tenerlo o actualizar un departamento a una determinada asignatura,

Para poder garantizar que la asignatura pertenece a un único departamento.

RF-009 - Asociación de una beca a un alumno:

Como usuario,

Quiero que el sistema permita: añadir una beca a un alumno que cumple las condiciones para ser becario,

Para poder asociar una beca a un alumno.

RF-010 - Asociación de la duración de una beca de un alumno:

Como usuario,

Quiero que el sistema permita: añadir en el caso de no tenerla o actualizar la duración de una determinada beca,

Para poder garantizar el tiempo que dura una beca.

RF-011 - Asociación de la cuantía total fija de una beca:

Como usuario,

Quiero que el sistema permita: añadir en el caso de no tenerla o actualizar la cuantía total fija que tiene una determinada beca,

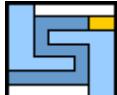
Para poder garantizar el reparto adecuado de becas.

RF-012 - Asociación de la cuantía total variable de una beca:

Como usuario,

Quiero que el sistema permita: añadir en el caso de no tenerla o actualizar la cuantía total variable que tiene una determinada beca,

Para poder garantizar el reparto adecuado de becas.



RF-013 - Informe de becas obtenidas por el alumno:

Como usuario,

Quiero que el sistema permita: obtener un informe de las becas obtenidas por el alumno hasta la fecha durante ciclo de vida universitario,

Para poder garantizar el reparto adecuado de becas.

RF-014 - Informe del total de alumnos matriculados en la universidad:

Como usuario,

Quiero que el sistema permita: obtener un informe de los alumnos matriculados en la universidad,

Para poder llevar un control de inicio, permanencia y abandono de los alumnos en la universidad.

RF-015 - Informe del total de profesores trabajando en la universidad:

Como usuario,

Quiero que el sistema permita: obtener un informe de los profesores que imparten docencia en la universidad,

Para poder saber la organización de los puestos de trabajo de cada profesor.

RF-016 - Informe del total de espacios agrupados por el tipo:

Como usuario,

Quiero que el sistema permita: obtener un informe de los espacios agrupados pos su tipo (despachos, aulas de tipo examen, aula de prácticas, etc.),

Para poder llevar un control del conteo de espacios en la universidad.

RF-017 - Informe de los profesores que pertenecen a un departamento:

Como usuario,

Quiero que el sistema permita: obtener un informe de los profesores que pertenecen a un determinado departamento,

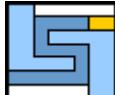
Para poder llevar un control de los profesores de cada departamento.

RF-018 - Informe de las asignaturas que pertenecen a un departamento:

Como usuario,

Quiero que el sistema permita: obtener un informe de las asignaturas que pertenecen a un determinado departamento,

Para poder llevar un control de las asignaturas de cada departamento.



RF-019 - Informe del grupo con mayor número de alumnos:

Como usuario,

Quiero que el sistema permita: obtener un informe del grupo que contenga mayor número de alumnos,

Para poder llevar un control de los grupos.

RF-020 - Eliminación de una asignatura a un departamento:

Como usuario,

Quiero que el sistema permita: eliminar una determinada asignatura de un departamento,

Para poder llevar un control de las asignaturas de cada departamento.

RF-021 - Asignación de un profesor a un departamento:

Como usuario,

Quiero que el sistema permita: añadir un determinado profesor a un departamento,

Para poder llevar un control de los profesores de cada departamento.

RF-022 - Eliminación de un profesor a un departamento:

Como usuario,

Quiero que el sistema permita: eliminar un determinado profesor a un departamento,

Para poder llevar un control de los profesores de cada departamento.

RF-023 - Garantía de profesor asignado a una asignatura:

Como usuario,

Quiero que el sistema permita: obtener un informe que garantice que al menos un profesor está asignado a una determinada asignatura,

Para poder llevar un control de profesores.

RF-024 - Garantía de que todas las asignaturas de un departamento están asignadas:

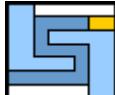
Como usuario,

Quiero que el sistema permita: obtener un informe que garantice que todas las asignaturas de un determinado departamento tienen al menos un profesor asignado,

Para poder llevar un control de profesores.

RF-025 - Informe de profesores agrupados por asignaturas:

Como usuario,



Quiero que el sistema permita: obtener un informe que muestre el número de profesores que hay asignados para cada asignatura que imparten,
Para poder llevar un control de los profesores y asignaturas.

RF-026 - Informe de tutorías agrupadas por profesores:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre el número de tutorías que hay agrupadas por profesor,
Para poder llevar un control las tutorías de los profesores.

RF-027 - Informe de profesores asociados a un despacho:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre todos los profesores que están asociados a un determinado despacho,
Para poder llevar un control de profesores y despachos.

RF-028 - Informe de notas de un expediente:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre todas las notas de un determinado expediente,
Para poder llevar un control de las notas de un expediente.

RF-029 - Informe de nota media de un expediente:

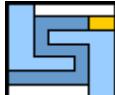
Como usuario,
Quiero que el sistema permita: obtener un informe que muestre la nota media de un determinado expediente,
Para poder llevar un control de la nota media global de un expediente.

RF-030 - Evaluación de un examen:

Como usuario,
Quiero que el sistema permita: añadir una nueva nota a un expediente,
Para poder evaluar un examen realizado.

RF-031 - Informe de la mejor nota de un expediente:

Como usuario,



Quiero que el sistema permita: obtener un informe que muestre la mejor nota obtenida de un determinado expediente,
Para poder llevar un control de un expediente.

RF-032 - Informe de las asignaturas obligatorias de un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre todas las asignaturas obligatorias de cursar asociadas a un grado,
Para poder llevar un control de las asignaturas de un grado.

RF-033 - Informe de las asignaturas optativas de un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre todas las asignaturas optativas asociadas a un grado,
Para poder llevar un control de las asignaturas de un grado.

RF-034 - Informe del número de créditos optativos de un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre el número de créditos optativos que requiere un determinado grado,
Para poder llevar un control de la cantidad de créditos de un grado.

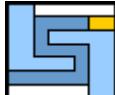
RF-035 - Informe del número de créditos obligatorios de un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre el número de créditos obligatorios que requiere un determinado grado,
Para poder llevar un control de la cantidad de créditos de un grado.

RF-036 - Informe del número de créditos troncales de un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre el número de créditos troncales que requiere un determinado grado,
Para poder llevar un control de la cantidad de créditos de un grado.

RF-037 - Informe del número de créditos totales de un grado:



Como usuario,
Quiero que el sistema permita: obtener un informe que muestre el número de créditos totales que requiere un determinado grado,
Para poder llevar un control de la cantidad de créditos de un grado.

RF-038 - Informe de los departamentos pertenecientes a un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que los departamentos pertenecientes a un determinado grado,
Para poder saber la organización de los departamentos.

RF-039 - Informe de los profesores que imparten en un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre los profesores que imparten docencia en un determinado grado,
Para poder saber la organización de los puestos de los profesores.

RF-040 - Informe de las asignaturas que pertenecen a un grado y curso:

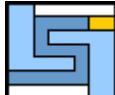
Como usuario,
Quiero que el sistema permita: obtener un informe que muestre las asignaturas pertenecientes a un grado para un determinado curso académico,
Para poder saber las asignaturas pertenecientes a un curso en concreto de un grado.

RF-041 - Informe de las asignaturas anuales que pertenecen a un grado:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre las asignaturas anuales pertenecientes a un determinado grado,
Para poder llevar un control de las asignaturas anuales de un grado.

RF-042 - Informe de las tutorías asociadas a un profesor:

Como usuario,
Quiero que el sistema permita: obtener un informe que muestre las tutorías asociadas a un determinado profesor,
Para poder llevar un control de organización de tutorías.



RF-043 - Informe del departamento asociado a un profesor:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre el departamento al que pertenece un determinado profesor,

Para poder llevar un control de profesores.

RF-044 - Informe de la dedicación total de un profesor:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre la dedicación total de créditos que puede llegar a impartir en su docencia un profesor,

Para poder llevar un control de profesores.

RF-045 - Informe de las asignaturas que imparte un profesor:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre las asignaturas impartidas por un determinado profesor,

Para poder llevar un control de profesores.

RF-046 - Informe de los créditos impartidos asociados a un profesor:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre los créditos que dedica un profesor para cada asignatura,

Para poder llevar un control de profesores.

RF-047 - Asociación de una categoría a un profesor:

Como usuario,

Quiero que el sistema permita: añadir un tipo de categoría de profesor a un determinado profesor o cambiársela en el caso de ya tenerla si es preciso,

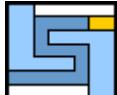
Para poder asociar una categoría a un profesor.

RF-048 - Adición de una cita de tutoría para un profesor:

Como usuario,

Quiero que el sistema permita: añadir una tutoría para un determinado día, hora y duración, de un determinado profesor,

Para poder pedir cita de una tutoría.



RF-049 - Eliminación de una cita de tutoría para un profesor:

Como usuario,

Quiero que el sistema permita: eliminar una tutoría ya pedida con anterioridad, de un determinado profesor,

Para poder anular la cita de una tutoría.

RF-050 - Adición de una asignatura y dedicación que debe impartir un profesor:

Como usuario,

Quiero que el sistema permita: añadir una asignatura con su dedicación que deberá impartir un determinado profesor,

Para poder llevar un control de profesores.

RF-051 - Eliminación de una asignatura impartida por un profesor:

Como usuario,

Quiero que el sistema permita: eliminar una asignatura ya impartida por un determinado profesor para que deje de impartirla,

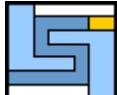
Para poder llevar un control de profesores.

RF-052 - Informe de las asignaturas que se han impartido en una determinada aula:

Como usuario,

Quiero que el sistema permita: obtener un informe que muestre las asignaturas impartidas en una determinada aula un determinado día,

Para poder llevar un control de las aulas.



5.6. Requisitos no funcionales

RNF-001 - Visualización y funcionamiento correcto:

Como usuario,

Quiero que el sistema permita: visualizarse y funcionar correctamente en cualquier navegador,

Para que todo usuario pueda hacer disfrute del mismo sistema.

RNF-002 - Cumplimiento de normativa:

Como usuario,

Quiero que el sistema permita: cumplir las disposiciones recogidas en la Ley Orgánica de Datos Personales y en el Reglamento de medidas de seguridad,

Para que no haya ningún problema legal.

RNF-003 - Tiempo de respuesta:

Como usuario,

Quiero que el sistema permita: no tardar más de cinco segundos en mostrar los resultados de una búsqueda, en el caso de que se supere este plazo, el sistema detiene la búsqueda y muestra los resultados encontrados,

Para acelerar el proceso de interacción entre usuario y sistema.

RNF-004 - Disponibilidad:

Como usuario,

Quiero que el sistema permita: estar disponible las 24 horas del día, intentando que los mantenimientos duren lo menos posible y sean cuando menos interacción de usuarios haya,

Para no retrasar a un usuario que quiera interactuar con el sistema.

RNF-005 - Fácil de usar:

Como usuario,

Quiero que el sistema permita: un fácil y sencillo manejo para cualquier tipo de usuario con o sin conocimientos técnicos,

Para una comodidad de entendimiento del sistema para el usuario.

6. Pruebas de aceptación

RN-001 - Email del alumno:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: un alumno matriculado debe tener un email acabado en @alum.es,
Para garantizar que es un alumno matriculado en la universidad.

Pruebas de aceptación:

- ✓ Se matricula un alumno en la universidad con un correo acabado en @alum.es y no se recibe ningún mensaje de error.
- ✗ Se matricula un alumno en la universidad con un correo que no está acabado en @alum.es y se recibe un mensaje de error por no acabar en @alum.es.
- ✗ Se modifica la matrícula de un alumno de la universidad con un correo que no está acabado en @alum.es y se recibe un mensaje de error por no acabar en @alum.es.

RN-002 - Créditos de las asignaturas:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: los créditos de una asignatura deben ser un número positivo distinto de 0,
Para garantizar que no se produzca ningún error a la hora de hacer el sumatorio total de créditos.

Pruebas de aceptación:

- ✓ Se añade una nueva asignatura al plan de estudios académico de la universidad con un número de créditos positivo distinto de 0 y no se recibe ningún mensaje de error.
- ✗ Se añade una nueva asignatura al plan de estudios académico de la universidad con un número de créditos igual o menor a 0 y se recibe un mensaje de error porque no puede tener 0 créditos o menos.
- ✗ Se modifica una nueva asignatura del plan de estudios académico de la universidad con un número de créditos igual o menor a 0 y se recibe un mensaje de error porque no puede tener 0 créditos o menos.

RN-003 - Duración de una beca:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: la duración de una beca debe ser como mínimo de 1 mes,
Para garantizar un tiempo realista de disfrute de una beca.

Pruebas de aceptación:

- ✓ Se adjudica una beca a un alumno con una duración igual o mayor a un mes y no se produce ningún mensaje de error.
- ✗ Se adjudica una beca a un alumno con una duración menor a un mes y se produce un mensaje de error porque no puede tener una duración inferior a 1 mes.

RN-004 - Cantidad mínima de la cuantía total fija de una beca:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: la cuantía total fija de una beca debe ser de al menos 1500€,
Para garantizar que se puedan cubrir las necesidades mínimas de un alumno.

Pruebas de aceptación:

- ✓ Se adjudica una beca a un alumno con un importe de cuantía total fija de ésta igual o mayor a 1500€ y no se produce ningún mensaje de error.
- ✗ Se adjudica una beca a un alumno con un importe de cuantía total fija de ésta menor a 1500€ y se produce un mensaje de error porque no puede ser inferior a 1500€.

RN-005 - Capacidad de un despacho:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: la capacidad máxima de profesores en un despacho debe ser 3,
Para garantizar un buen reparto de despachos y profesores.

Pruebas de aceptación:

- ✓ Se registran tantos profesores como capacidad tiene un despacho y no se recibe ningún mensaje de error.
- ✗ Se intenta registrar un profesor más a ese despacho y se recibe un mensaje de registro no permitido por superar el máximo de capacidad permitida en el despacho.

RN-006 - Capacidad de un aula:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: la capacidad máxima de alumnos en un aula debe ser 70 si es de tipo teoría, de 30 si es de tipo práctica o 200 en caso de ser de tipo examen,
Para garantizar un buen reparto de aulas y alumnos.

Pruebas de aceptación:

- ✓ Se registran tantos alumnos como capacidad tiene un aula de tipo teoría, un aula de tipo práctica y un aula tipo examen y no se recibe ningún mensaje de error.
- ✗ Se intenta registrar un alumno más a esas aulas y se recibe un mensaje de registro no permitido por superar el máximo de capacidad permitida en dichas aulas.

RN-007 - Calificaciones del expediente con única convocatoria y curso:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: el expediente de un alumno no puede contener calificaciones para más de 1 convocatoria de una misma asignatura y curso,

Para garantizar un registro adecuado del expediente académico para cada convocatoria y curso.

Pruebas de aceptación:

- ✓ Se registra una calificación de un alumno para una determinada convocatoria y un determinado curso y no se recibe ningún mensaje de error.
- ✗ Se intenta registrar una calificación en un alumno que ya ha obtenido una calificación para una determinada convocatoria y un determinado curso y se recibe un mensaje de error porque no se puede tener otra calificación para la misma convocatoria y curso.

RN-008 - Asignaturas optativas de un grado con mismo número de créditos:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: todas las asignaturas optativas de un grado deben tener el mismo número de créditos,

Para garantizar que no se produzca ningún error a la hora de hacer el sumatorio total de créditos.

Pruebas de aceptación:

- ✓ Se añaden una serie de asignaturas optativas al plan de estudios académico de un grado donde cada una de ellas tienen el mismo número de créditos y no se recibe ningún mensaje de error.
- ✗ Se intentan añadir una serie de asignaturas optativas al plan de estudios académico de un grado donde tienen diferentes números de créditos y se recibe un mensaje de error porque todas las asignaturas optativas del grado deben tener el mismo número de créditos optativos.
- ✗ Se intenta modificar el número de créditos optativos de una asignatura optativa de manera que tenga una cantidad diferente al resto de asignaturas optativas del grado y se recibe un mensaje de error porque todas las asignaturas optativas del grado deben tener el mismo número de créditos optativos.

RN-009 - Asignaturas optativas deben ser cuatrimestrales:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: todas las asignaturas optativas de un grado deben ser cuatrimestrales,
Para garantizar un reparto adecuado del tiempo dedicado a esas asignaturas.

Pruebas de aceptación:

- ✓ Se añade una asignatura optativa al plan de estudios académico de un grado donde tiene una duración cuatrimestral y no se recibe ningún mensaje de error.
- ✗ Se intenta añadir una asignatura optativa al plan de estudios académico de un grado donde tiene una duración anual y se recibe un mensaje de error porque una asignatura optativa debe tener una duración cuatrimestral.
- ✗ Se intenta modificar la duración de una asignatura optativa de manera que sea anual y se recibe un mensaje de error porque una asignatura optativa debe tener una duración cuatrimestral.

RN-010 - Cantidad de créditos optativos a cursar:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: el número mínimo de créditos de asignaturas optativas que debe cursar un alumno debe estar comprendido entre 0 y 30,
Para garantizar que no se produzca ningún error a la hora de hacer el sumatorio total de créditos.

Pruebas de aceptación:

- ✓ Se matricula un alumno de una serie de asignaturas optativas de manera que la suma de ellas no supere los 30 créditos optativos y no se recibe ningún mensaje de error.
- ✗ Se intenta matricular a un alumno de una serie de asignaturas optativas de manera que la suma de ellas supera los 30 créditos optativos y se recibe un mensaje de error porque el máximo de créditos optativos es de 30.

RN-011 - Valor de una nota:

Como profesor,

Quiero que se cumpla la siguiente regla de negocio: el valor de una nota a la hora de evaluar un alumno debe estar comprendida entre 0 y 10,

Para garantizar que todas las asignaturas pueden tener una calificación en función de la nota obtenida bajo el mismo criterio.

Pruebas de aceptación:

- ✓ Se evalúa un alumno con una nota de puntuación comprendida entre 0 y 10 y no se recibe ningún mensaje de error.
- ✗ Se intenta evaluar un alumno con una nota de puntuación menor a 0 o mayor a 10 y se recibe un mensaje de error porque un alumno debe estar evaluado con una nota de puntuación comprendida entre 0 y 10.
- ✗ Se intenta modificar la nota de puntuación de un alumno de manera que sea menor a 0 o mayor a 10 y se recibe un mensaje de error porque un alumno debe estar evaluado con una nota de puntuación comprendida entre 0 y 10.

RN-012 - Calificación debe corresponder con valor de nota:

Como profesor,

Quiero que se cumpla la siguiente regla de negocio: la calificación de una evaluación debe tener su correspondiente valor de nota,

Para garantizar que una calificación tiene sentido con el valor correspondiente de la nota.

Pruebas de aceptación:

- ✓ Se le asigna una calificación correspondiente al valor de la nota de un alumno y no se recibe ningún mensaje de error.
- ✗ Se intenta asignar una calificación que no corresponde al valor de la nota de un alumno y se recibe un mensaje de error porque un alumno debe tener una calificación que corresponda al valor de la nota.
- ✗ Se intenta modificar la calificación de la nota de puntuación de un alumno de manera que no corresponda al valor de la nota y se recibe un mensaje de error porque un alumno debe tener una calificación que corresponda al valor de la nota.

RN-013 - Asignaturas impartidas por un profesor de un único departamento:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: un profesor no puede impartir una asignatura de otro departamento que no corresponda al suyo,

Para garantizar la estructura de docencia en la universidad.

Pruebas de aceptación:

- ✓ Se asigna una asignatura de un determinado departamento a un profesor que pertenece a ese mismo departamento para que imparta docencia de ella y no se produce ningún mensaje de error.
- ✗ Se intenta asignar una asignatura de un determinado departamento a un profesor que no pertenece a ese departamento y se produce un mensaje de error porque no puede impartir un profesor una asignatura que pertenezca a un departamento que no es el suyo.
- ✗ Se intenta cambiar el departamento al que pertenece un profesor teniendo asignaturas que imparte del anterior departamento y se produce un mensaje de error porque no puede impartir un profesor una asignatura que pertenezca a un departamento que no es el suyo.

RN-014 - Dedicación de créditos de un profesor a una asignatura:

Como gerente de la universidad,

Quiero que se cumpla la siguiente regla de negocio: el número de créditos dedicados a una asignatura por un profesor debe ser mayor que 0 y menor o igual que el número de créditos de ésta,

Para garantizar el correcto reparto de créditos de un profesor.

Pruebas de aceptación:

- ✓ Se asigna a un profesor una dedicación de créditos mayor a 0 y menor o igual al número de créditos de la asignatura a la que va a dedicarlos y no se produce ningún mensaje de error.
- ✗ Se intenta asignar a un profesor una dedicación de créditos superior al número de créditos que tiene la asignatura a la que va a dedicarlos y se produce un mensaje de error porque un profesor no puede dedicar más créditos de los que tiene la propia asignatura.
- ✗ Se intenta modificar la dedicación de créditos que está impartiendo un profesor a una asignatura de manera que imparte más créditos de los que tiene la propia asignatura y se produce un mensaje de error porque un profesor no puede dedicar más créditos de los que tiene la propia asignatura.

RN-015 - Créditos máximos que puede impartir un profesor:

Como gerente de la universidad,
Quiero que se cumpla la siguiente regla de negocio: un profesor puede impartir un máximo de 32 créditos o 6 en el caso de ser un profesor ayudante,
Para garantizar el correcto reparto de créditos de un profesor

Pruebas de aceptación:

- ✓ Se asigna a un profesor una dedicación de créditos máximo de 32 créditos o 6 en caso de ser un profesor ayudante y no se produce ningún mensaje de error.
- ✗ Se intenta asignar a un profesor una dedicación de créditos superior a la que puede dedicar y se produce un mensaje de error porque un profesor no puede dedicar más créditos de los que tiene impuestos como máximo.
- ✗ Se intenta modificar la dedicación de créditos que está impartiendo un profesor de manera que supere la dedicación máxima y se produce un mensaje de error porque un profesor no puede dedicar más créditos de los que tiene impuestos como máximo.

RN-016 - Días de tutoría:

Como profesor,
Quiero que se cumpla la siguiente regla de negocio: los días de la semana en donde se puede realizar una tutoría deben estar entre lunes y viernes,
Para garantizar el correcto cumplimiento de horario de docencia de un profesor.

Pruebas de aceptación:

- ✓ Se asigna a un alumno una tutoría cuyo día de la semana está comprendido entre lunes y viernes y no se produce ningún mensaje de error.
- ✗ Se intenta asignar a un alumno una tutoría cuyo día de la semana es sábado o domingo y se produce un mensaje de error porque no se puede impartir una tutoría los fines de semana.
- ✗ Se intenta cambiar la cita de tutoría que tenía un alumno a un día que corresponde a sábado o domingo y se produce un mensaje de error porque no se puede impartir una tutoría los fines de semana.

RN-017 - Duración de una tutoría:

Como profesor,

Quiero que se cumpla la siguiente regla de negocio: la duración de una tutoría debe ser de al menos 10 minutos y de máximo 30 minutos,

Para garantizar una buena disponibilidad de tutorías para todos los alumnos.

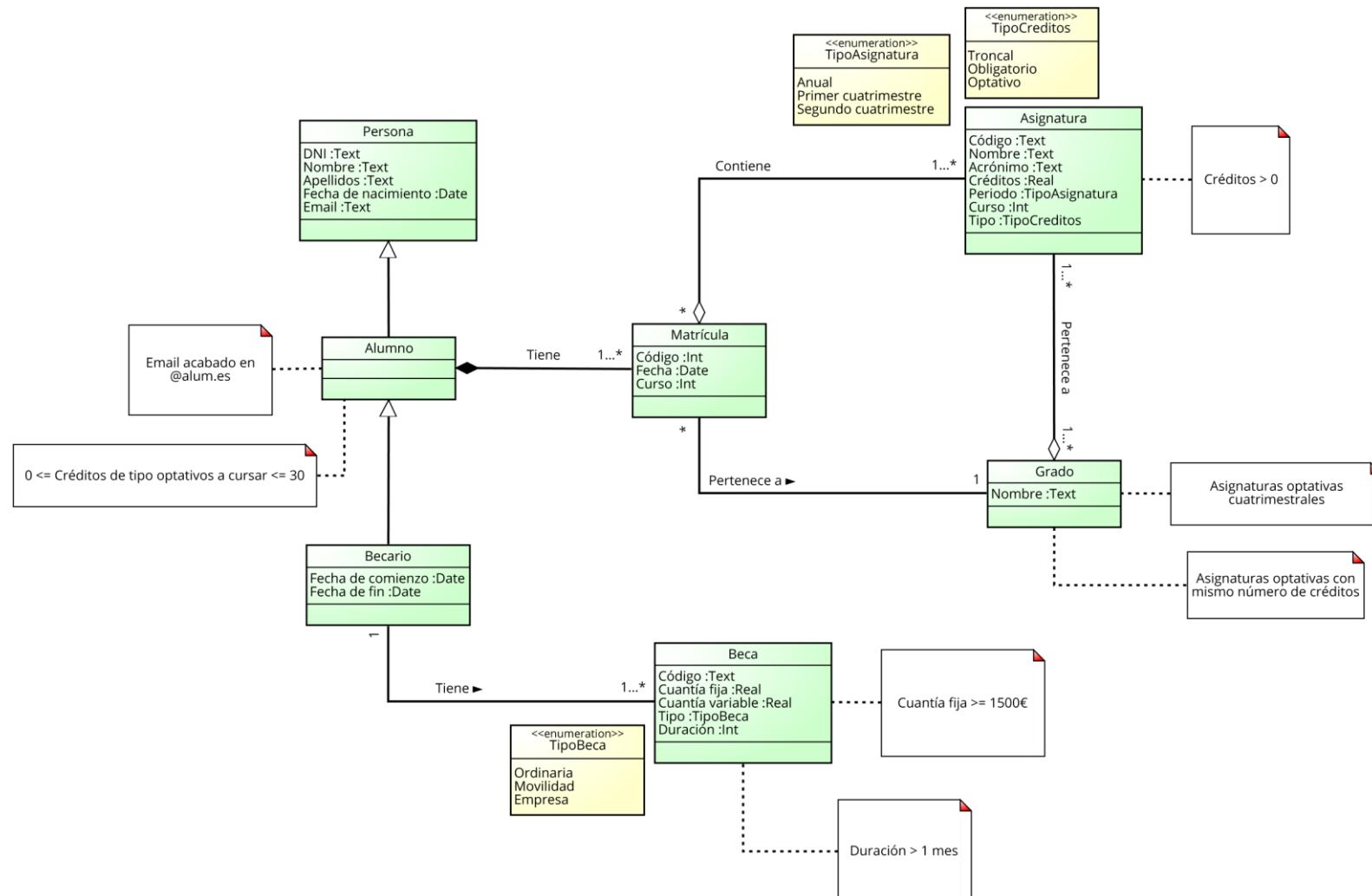
Pruebas de aceptación:

- ✓ Se le asigna a un alumno una tutoría que tiene una duración comprendida entre 10 y 30 minutos y no se produce ningún mensaje de error.
- ✗ Se le asigna a un alumno una tutoría que tiene una duración menor a 10 minutos o mayor a 30 minutos y se produce un mensaje de error porque la duración de una tutoría debe estar comprendida entre 10 y 30 minutos.

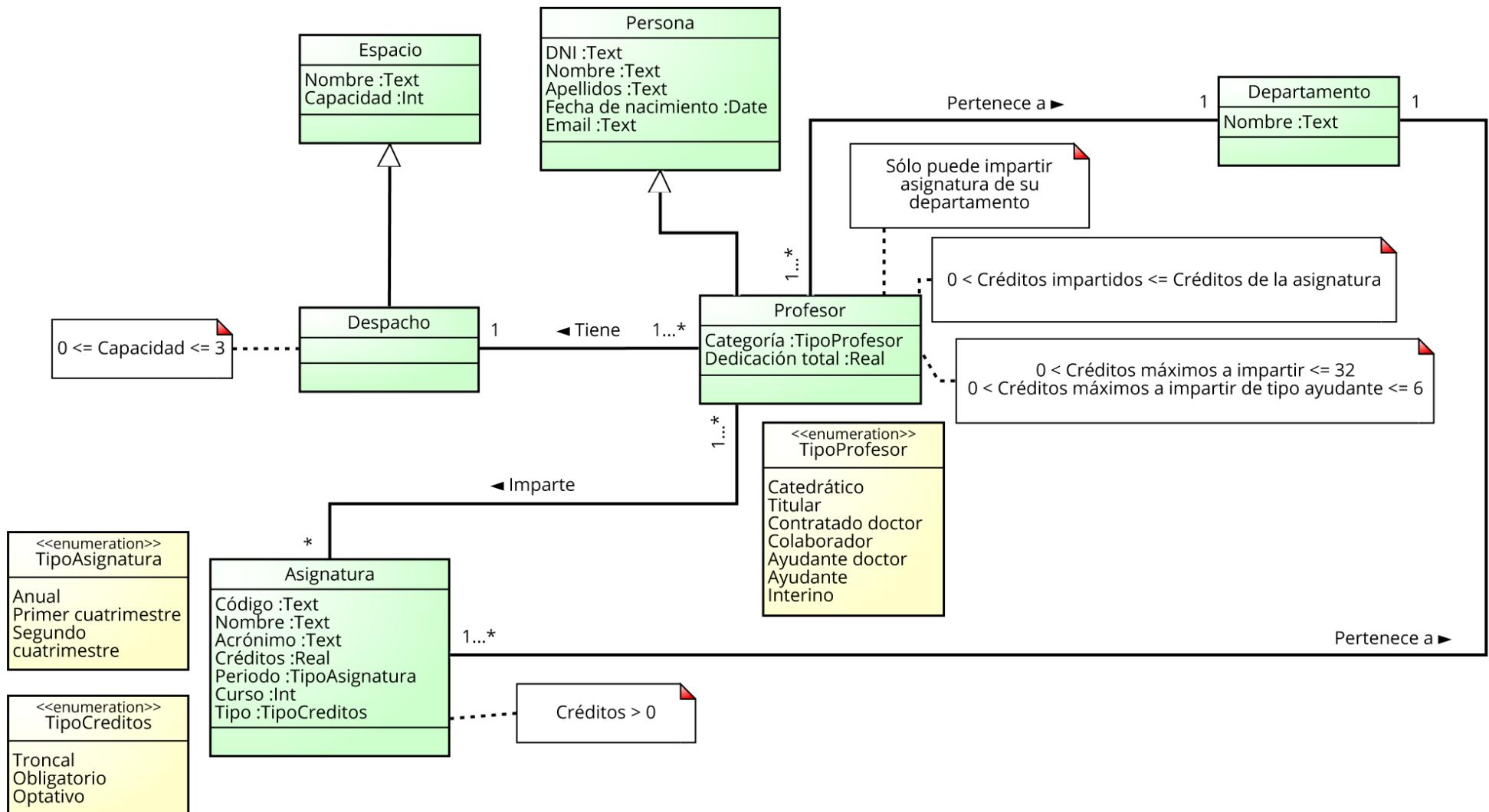
7. Modelo conceptual

7.1. Diagramas de clases UML con restricciones

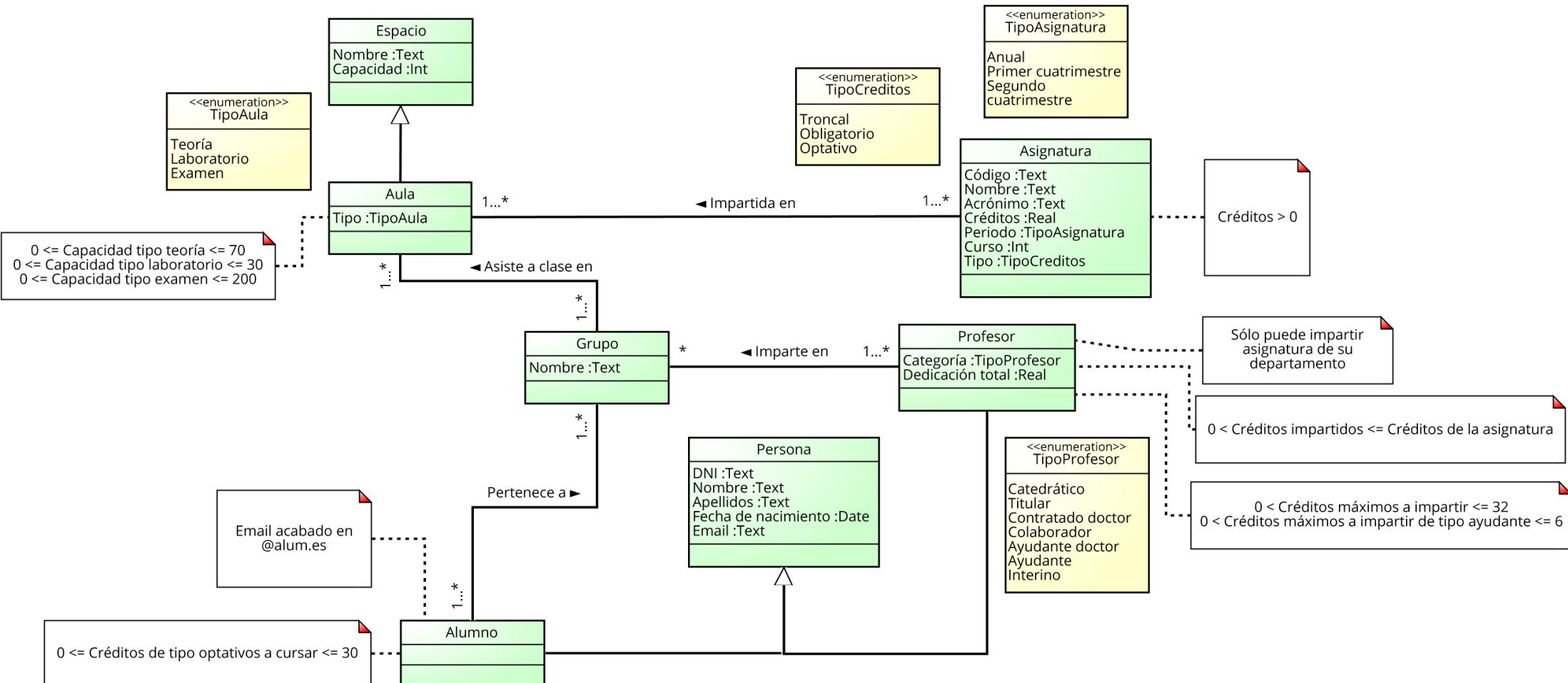
Modelado de matrículas y becas



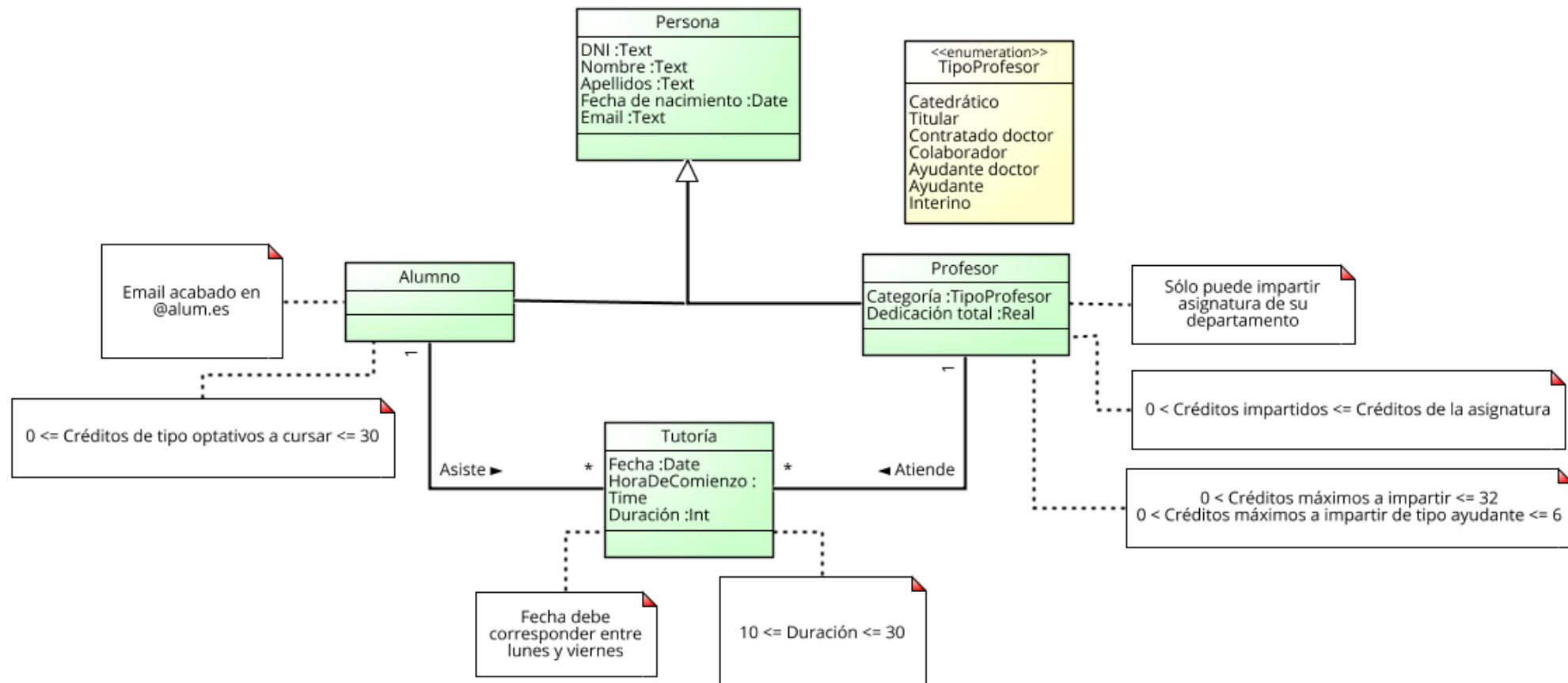
Modelado de docencia de profesorado



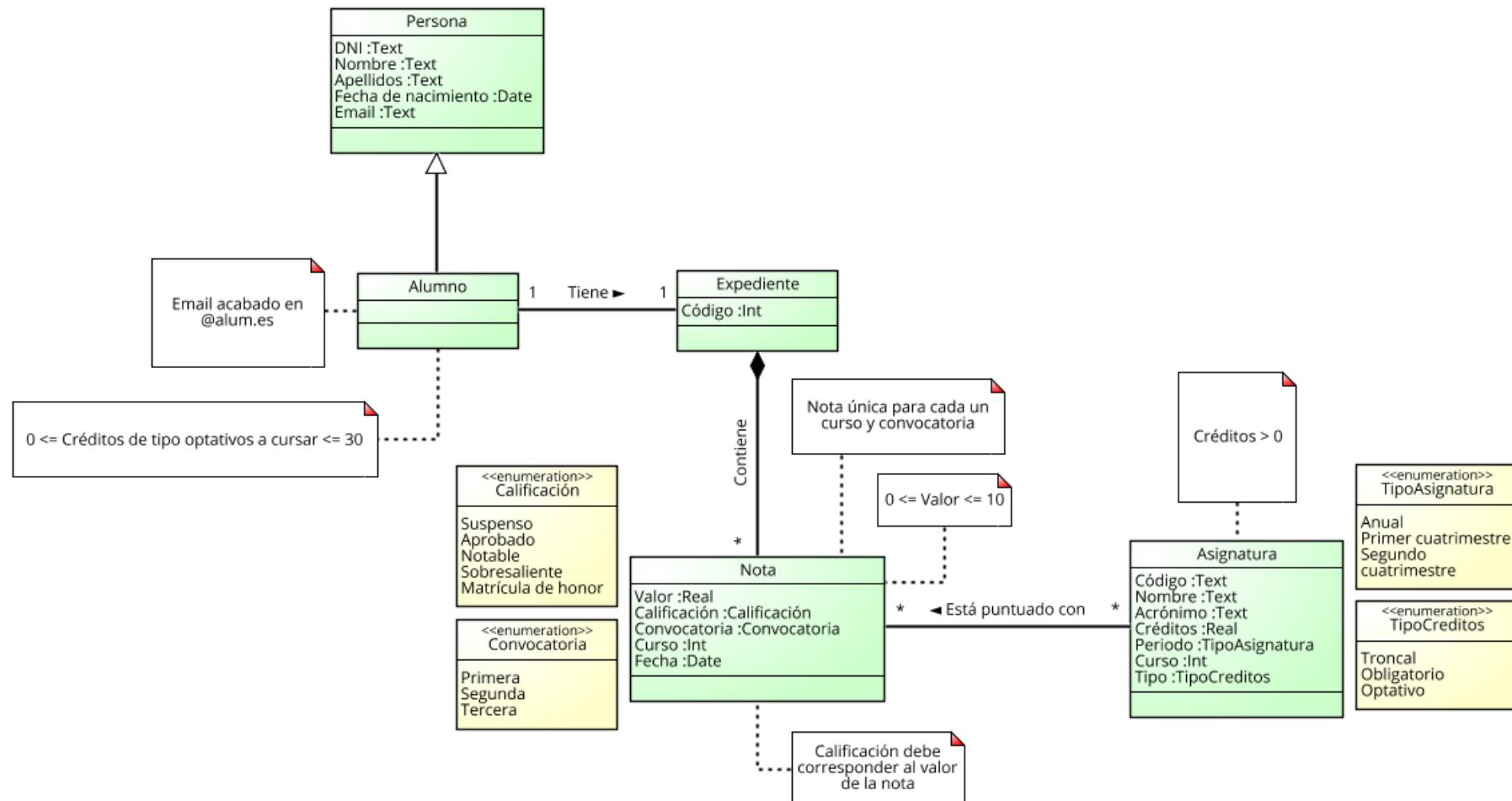
Modelado de clases



Modelado de tutorías



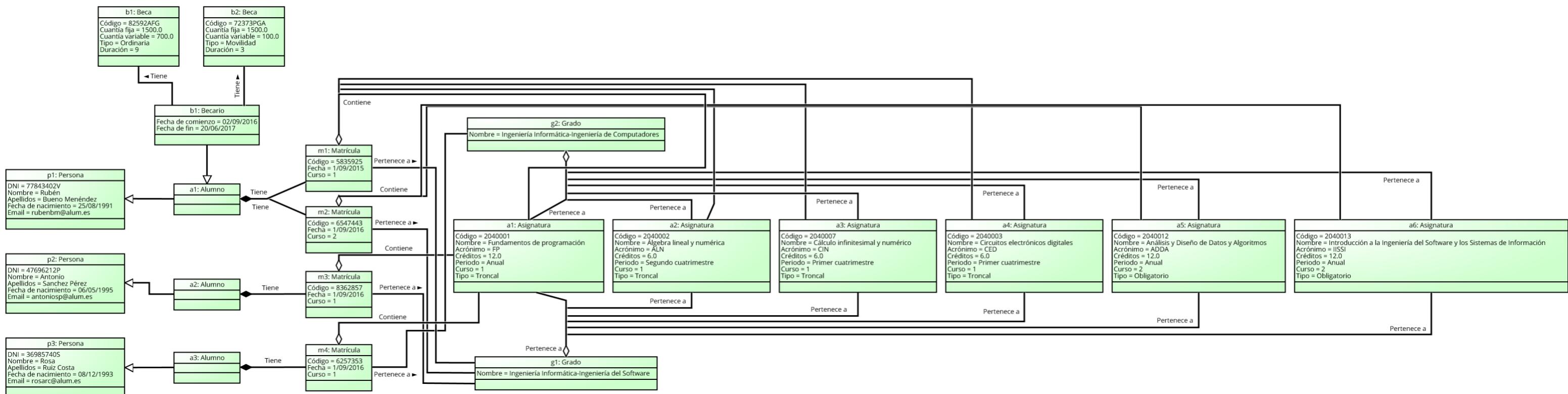
Modelado de expedientes





7.2. Escenarios de prueba

Escenario de prueba de matrículas y becas (diagrama de objetos UML)





Escenario de prueba de matrículas y becas (descripción textual)

En dicho escenario de prueba tenemos 3 personas (p1, p2, p3) que vienen a ser 3 alumnos (a1, a2, a3) de los cuales el alumno a1 es becario (b1). El becario b1 tiene 2 becas diferentes (b1, b2).

El alumno a1 tiene 2 matrículas (m1, m2). La matrícula m1 contiene las asignaturas a1, a2, a3 y a4. La matrícula m2 contiene las asignaturas a5 y a6. Tanto la matrícula m1 como la m2 pertenecen al grado g1.

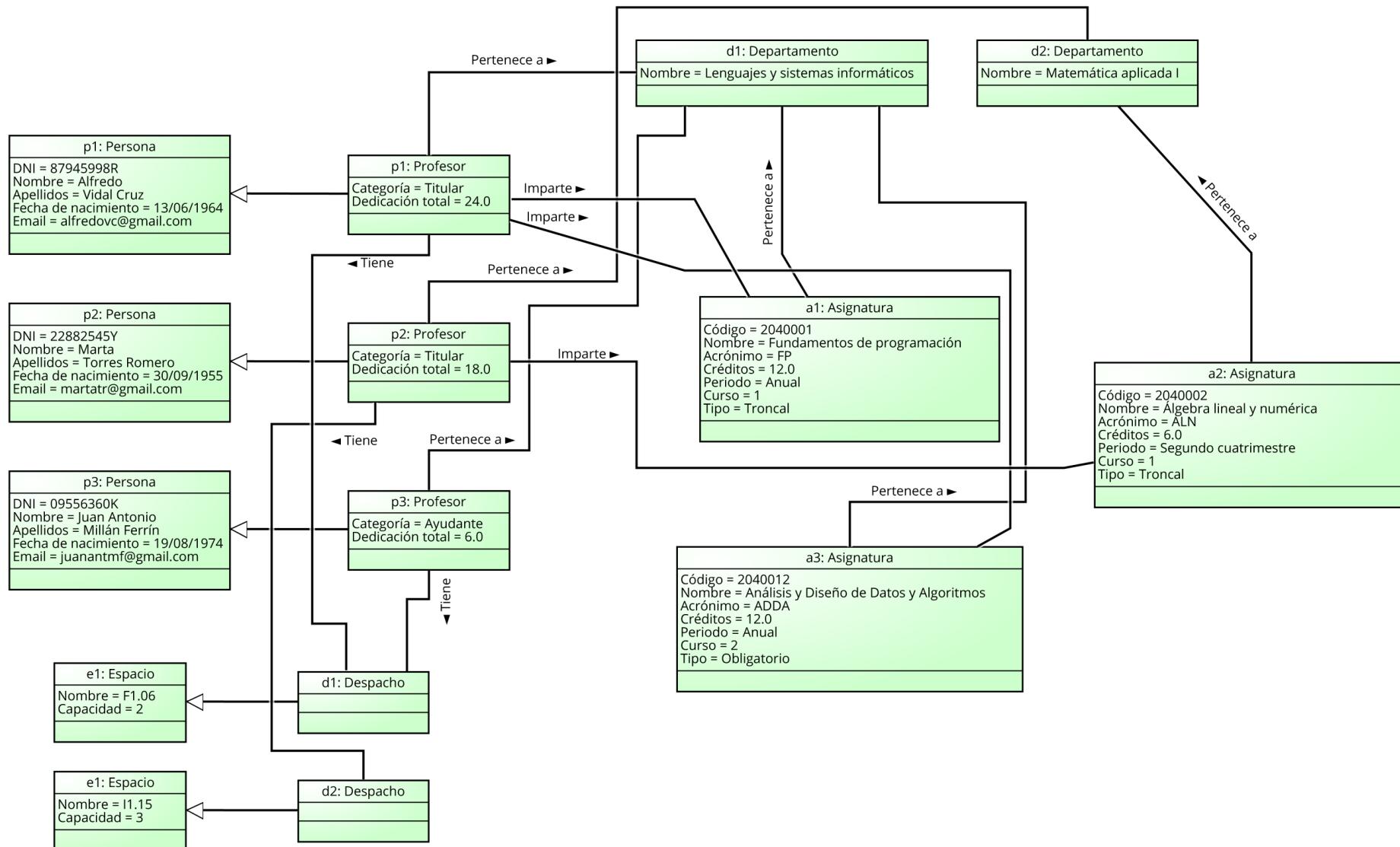
El alumno a2 tiene 1 matrícula (m3). Dicha matrícula contiene la asignatura a1 y pertenece al grado g1.

El alumno a3 tiene 1 matrícula (m4). Dicha matrícula contiene la asignatura a1 y pertenece al grado g2.

Las asignaturas a1, a2, a3, a4, a5 y a6 pertenecen tanto al grado g1 como el grado g2.



Escenario de prueba de docencia de profesorado (diagrama de objetos UML)





Escenario de prueba de docencia de profesorado (descripción textual)

En dicho escenario de prueba tenemos 3 personas (p_1, p_2, p_3) que vienen a ser 3 profesores (p_1, p_2, p_3). Tenemos también 2 espacios (e_1, e_2) que vienen a ser 2 despachos (d_1, d_2).

Tanto el profesor p_1 como el profesor p_3 tienen adjudicado el despacho d_1 . El profesor p_2 tiene adjudicado el despacho d_2 .

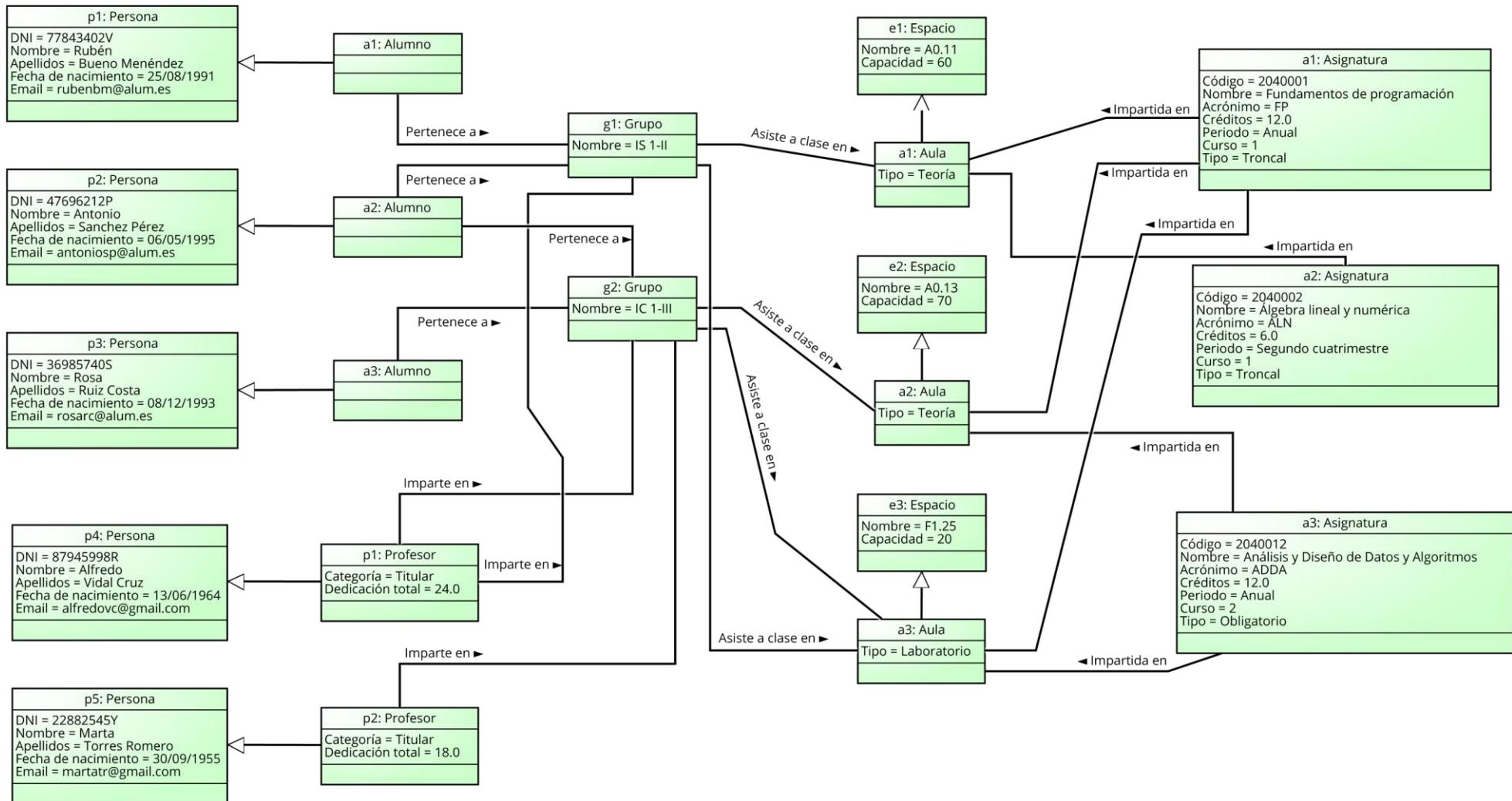
Tanto el profesor p_1 como el profesor p_3 pertenecen al departamento d_1 . El profesor p_2 pertenece al departamento d_2 .

El profesor p_1 imparte los 12 créditos de la asignatura a_1 y los 12 créditos de la asignatura a_3 . El profesor p_2 imparte los 6 créditos de la asignatura a_2 . El profesor p_3 no impartirá ninguna asignatura.

Las asignaturas a_1 y a_3 pertenecen al departamento d_1 y la asignatura a_2 pertenece al departamento d_2 . De esta manera se cumple que los profesores pertenecen al mismo departamento que pertenecen las asignaturas que imparten, en el caso de hacerlo.



Escenario de prueba de clases (diagrama de objetos UML)





Escenario de prueba de clases (descripción textual)

En dicho escenario de prueba tenemos 5 personas (p1, p2, p3, p4, p5), de las cuales, 3 son alumnos (a1, a2, a3) y 2 son profesores (p1, p2).

El alumno a1 pertenece al g1. El alumno a2 pertenece a los grupos g1 y g2. El alumno a3 pertenece al grupo g2.

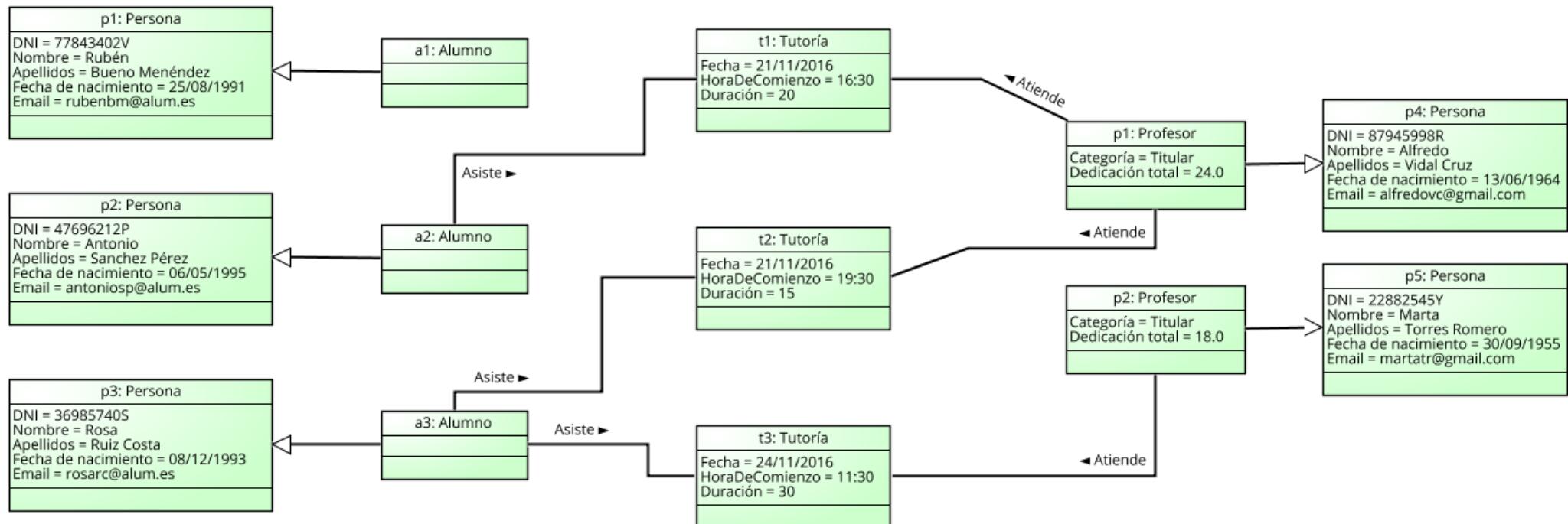
El profesor p1 imparte clases tanto en el grupo g1 como en el grupo g2. El profesor p2 imparte clases solamente en el grupo g2.

Tenemos 3 espacios diferentes (e1, e2, e3), que vienen a ser 3 aulas (a1, a2, a3).

El grupo g1 asiste a clases en el aula a1 y a3. El grupo g2 asiste a clases en el aula a2 y a3.

Tenemos 3 asignaturas (a1, a2, a3). La asignatura a1 es impartida en las aulas a1, a2 y a3. La asignatura a2 es impartida solamente en el aula a1. La asignatura a3 es impartida tanto en el aula a2 como en el aula a3.

Escenario de prueba de tutorías (diagrama de objetos UML)





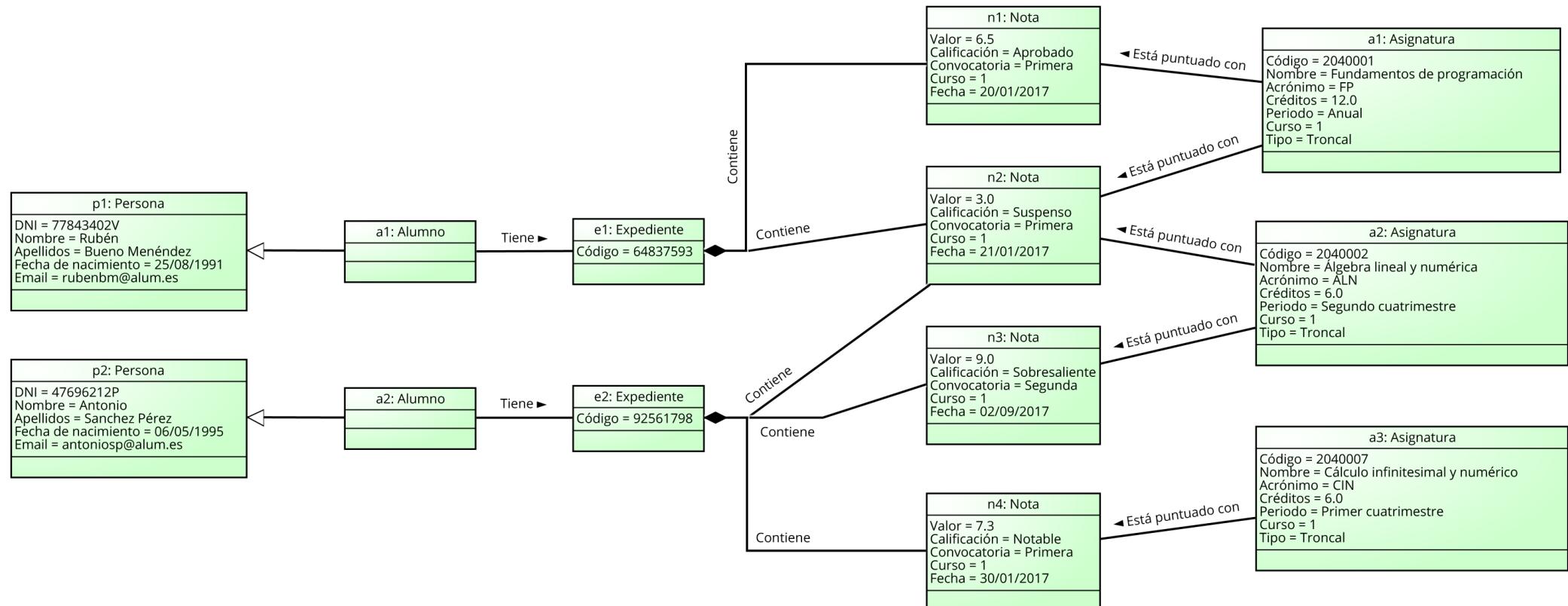
Escenario de prueba de tutorías (descripción textual)

En dicho escenario de prueba tenemos 5 personas (p1, p2, p3, p4, p5), de las cuales, 3 son alumnos (a1, a2, a3) y 2 son profesores (p1, p2).

El alumno a1 no va a asistir a ninguna tutoría. El alumno a2 asiste a la tutoría t1 que será atendida por el profesor p1.

El alumno a3 asiste un día a la tutoría t2 y otro día a la tutoría t3. La tutoría t2 es atendida por el profesor p1 y la tutoría t3 es atendida por el profesor p2.

Escenario de prueba de expedientes (diagrama de objetos UML)





Escenario de prueba de expedientes (descripción textual)



En dicho escenario de prueba tenemos 2 personas (p1, p2), que vienen a ser 2 alumnos (a1, a2).

El alumno a1 tiene un único expediente e1. El alumno a2 tiene un único expediente e2. El expediente e1 contiene 2 notas (n1, n2) y el expediente e2 contiene 3 notas (n2, n3, n4).

Tenemos 3 asignaturas (a1, a2, a3). La asignatura a1 está puntuada con las notas n1 y n2, la asignatura a2 está puntuada con las notas n2 y n3, y la asignatura a3 está puntuada con la nota n4.



8. Matrices de trazabilidad

**Matriz de trazabilidad de Clases → Requisitos de información**

Clases \ R. de Inf.	RI-001 - Información sobre la matrícula	RI-002 - Información sobre el expediente académico	RI-003 - Información sobre las asignaturas	RI-004 - Información sobre los grados	RI-005 - Información sobre las becas	RI-006 - Información sobre las calificaciones	RI-007 - Información sobre los departamentos	RI-008 - Información sobre los despachos	RI-009 - Información sobre las aulas	RI-010 - Información sobre los profesores	RI-011 - Información sobre las tutorías	RI-012 - Información sobre los grupos
Clases	RI-001 - Información sobre la matrícula	RI-002 - Información sobre el expediente académico	RI-003 - Información sobre las asignaturas	RI-004 - Información sobre los grados	RI-005 - Información sobre las becas	RI-006 - Información sobre las calificaciones	RI-007 - Información sobre los departamentos	RI-008 - Información sobre los despachos	RI-009 - Información sobre las aulas	RI-010 - Información sobre los profesores	RI-011 - Información sobre las tutorías	RI-012 - Información sobre los grupos
Asignatura			✓									
Alumno												✓
Aula									✓			
Beca					✓							
Becario					✓							
Departamento							✓					
Despacho								✓				
Espacio									✓	✓		
Expediente		✓				✓						
Grado				✓								
Grupo												✓
Matrícula	✓											
Nota						✓						
Persona										✓		✓
Profesor										✓		
Tutoría											✓	



Matriz de trazabilidad de Clases → Reglas de negocio

R. de Neg. Clases	RN-001 - Email del alumno	RN-002 - Créditos de las asignaturas	RN-003 - Duración de una beca	RN-004 - Cantidad mínima de la cuantía total fija de una beca	RN-005 - Capacidad de un despacho	RN-006 - Capacidad de un aula	RN-007 - Calificaciones del expediente con única convocatoria y curso	RN-008 - Asignaturas optativas de un grado con mismo número de créditos	RN-009 - Asignaturas optativas deben ser cuatrimestrales	RN-010 - Cantidad de créditos optativos a cursar	RN-011 - Valor de una nota	RN-012 - Calificación debe corresponder con valor de nota	RN-013 - Asignaturas impartidas por un profesor de un único departamento	RN-014 - Dedicación de créditos de un profesor a una asignatura	RN-015 - Créditos máximos que puede impartir un profesor	RN-016 - Días de tutoría	RN-017 - Duración de una tutoría
Asignatura		✓						✓	✓						✓		
Alumno	✓																
Aula						✓											
Beca			✓	✓													
Becario	✓		✓														
Departamento													✓				
Despacho					✓												
Espacio					✓	✓											
Expediente							✓										
Grado										✓							
Grupo																	
Matrícula																	
Nota											✓	✓					
Persona	✓												✓	✓	✓		
Profesor													✓	✓	✓		
Tutoría															✓	✓	



Matriz de trazabilidad de Asociaciones → Requisitos de información

R. de Inf. Asociaciones	RI-001 - Información sobre la matrícula	RI-002 - Información sobre el expediente académico	RI-003 - Información sobre las asignaturas	RI-004 - Información sobre los grados	RI-005 - Información sobre las becas	RI-006 - Información sobre las calificaciones	RI-007 - Información sobre los departamentos	RI-008 - Información sobre los despachos	RI-009 - Información sobre las aulas	RI-010 - Información sobre los profesores	RI-011 - Información sobre las tutorías	RI-012 - Información sobre los grupos
asisteAClaseEnAula									✓			✓
asisteTutoría											✓	
atiendeTutoría										✓	✓	
contieneAsignatura	✓		✓									
contieneNota		✓				✓						
estáPuntuadaCon			✓			✓						
imparteAsignatura			✓							✓		
imparteEnGrupo										✓		✓
impartidaEnAula									✓	✓		
perteneceADepartamento			✓				✓			✓		
perteneceAGrado			✓	✓								✓
perteneceAGrupo												✓
tieneBeca					✓							
tieneDespacho								✓		✓		
tieneExpediente		✓										
tieneMatrícula	✓											



Matriz de trazabilidad de Asociaciones → Reglas de negocio

R. de Neg. Asociaciones	RN-001 - Email del alumno	RN-002 - Créditos de las asignaturas	RN-003 - Duración de una beca	RN-004 - Cantidad mínima de la cuantía total fija de una beca	RN-005 - Capacidad de un despacho	RN-006 - Capacidad de un aula	RN-007 - Calificaciones del expediente con única convocatoria y curso	RN-008 - Asignaturas optativas de un grado con mismo número de créditos	RN-009 - Asignaturas optativas deben ser cuatrimestrales	RN-010 - Cantidad de créditos optativos a cursar	RN-011 - Valor de una nota	RN-012 - Calificación debe corresponder con valor de nota	RN-013 - Asignaturas impartidas por un profesor de un único departamento	RN-014 - Dedicación de créditos de un profesor a una asignatura	RN-015 - Créditos máximos que puede impartir un profesor	RN-016 - Días de tutoría	RN-017 - Duración de una tutoría
asisteAClaseEnAula						✓											
asisteTutoría																✓	✓
atiendeTutoría																✓	✓
contieneAsignatura		✓						✓	✓								
contieneNota							✓				✓	✓					
estáPuntuadaCon						✓					✓	✓					
imparteAsignatura														✓	✓	✓	
imparteEnGrupo															✓	✓	
impartidaEnAula																	
perteneceADepartamento															✓		
perteneceAGrado								✓	✓	✓							
perteneceAGrupo																	
tieneBeca			✓	✓													
tieneDespacho																	
tieneExpediente							✓				✓						
tieneMatrícula																	



Matriz de trazabilidad de Restricciones → Requisitos de información

R. de Inf. Restricciones	RI-001 - Información sobre la matrícula	RI-002 - Información sobre el expediente académico	RI-003 - Información sobre las asignaturas	RI-004 - Información sobre los grados	RI-005 - Información sobre las becas	RI-006 - Información sobre las calificaciones	RI-007 - Información sobre los departamentos	RI-008 - Información sobre los despachos	RI-009 - Información sobre las aulas	RI-010 - Información sobre los profesores	RI-011 - Información sobre las tutorías	RI-012 - Información sobre los grupos
0 < Créditos impartidos ≤ Créditos de la asignatura			✓							✓		
0 < Créditos máximos a impartir ≤ 32			✓							✓		
0 < Créditos máximos a impartir de tipo ayudante ≤ 6			✓							✓		
0 ≤ Capacidad ≤ 3								✓				
0 ≤ Capacidad tipo examen ≤ 200									✓			
0 ≤ Capacidad tipo laboratorio ≤ 30									✓			
0 ≤ Capacidad tipo teoría ≤ 70									✓			
0 ≤ Créditos de tipo optativos a cursar ≤ 30	✓		✓									
0 ≤ Valor ≤ 10			✓				✓					
10 ≤ Duración ≤ 30											✓	
Asignaturas optativas con mismo número de créditos				✓								
Asignaturas optativas cuatrimestrales				✓								
Créditos > 0				✓								
Cuantía fija ≥ 1500 €						✓						
Duración > 1 mes						✓						
Email acabado en @alum.es												✓
Fecha debe corresponder entre lunes y viernes											✓	
Nota única para un curso y convocatoria			✓									
Sólo puede impartir una asignatura de su departamento			✓				✓			✓		
Valor de nota debe corresponder con calificación						✓						



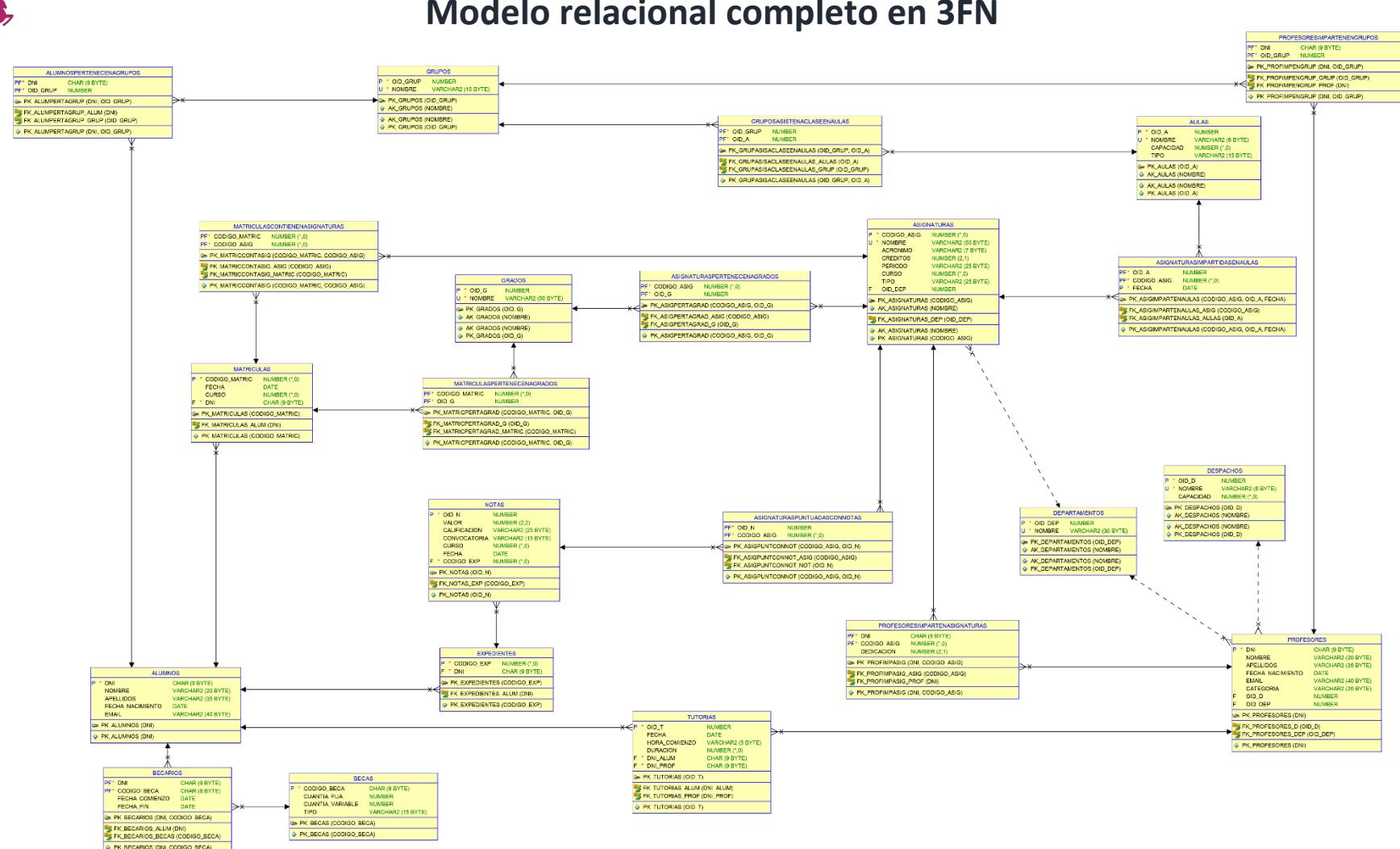
Matriz de trazabilidad de Restricciones → Reglas de negocio

R. de Neg. Restricciones	RN-001 - Email del alumno	RN-002 - Créditos de las asignaturas	RN-003 - Duración de una beca	RN-004 - Cantidad mínima de la cuantía total fija de una beca	RN-005 - Capacidad de un despacho	RN-006 - Capacidad de un aula	RN-007 - Calificaciones del expediente con única convocatoria y curso	RN-008 - Asignaturas optativas de un grado con mismo número de créditos	RN-009 - Asignaturas optativas deben ser cuatrimestrales	RN-010 - Cantidad de créditos optativos a cursar	RN-011 - Valor de una nota	RN-012 - Calificación debe corresponder con valor de nota	RN-013 - Asignaturas impartidas por un profesor de un único departamento	RN-014 - Dedicación de créditos de un profesor a una asignatura	RN-015 - Créditos máximos que puede impartir un profesor	RN-016 - Días de tutoría	RN-017 - Duración de una tutoría
0 < Créditos impartidos ≤ Créditos de la asignatura															✓		
0 < Créditos máximos a impartir ≤ 32																✓	
0 < Créditos máximos a impartir de tipo ayudante ≤ 6																✓	
0 ≤ Capacidad ≤ 3					✓												
0 ≤ Capacidad tipo examen ≤ 200						✓											
0 ≤ Capacidad tipo laboratorio ≤ 30						✓											
0 ≤ Capacidad tipo teoría ≤ 70						✓											
0 ≤ Créditos de tipo optativos a cursar ≤ 30											✓						
0 ≤ Valor ≤ 10												✓					
10 ≤ Duración ≤ 30																	✓
Asignaturas optativas con mismo número de créditos								✓									
Asignaturas optativas cuatrimestrales									✓								
Créditos > 0		✓															
Cuantía fija ≥ 1500 €				✓													
Duración > 1 mes			✓														
Email acabado en @alum.es	✓																
Fecha debe corresponder entre lunes y viernes																	✓
Nota única para un curso y convocatoria							✓										
Sólo puede impartir una asignatura de su departamento													✓				
Valor de nota debe corresponder con calificación											✓						



9. Modelo relacional en 3FN

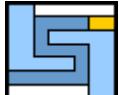
9.1. Relaciones obtenidas





9.2. Justificación de la estrategia de transformación de jerarquías

Relaciones obtenidas					
Tablas	Atributos	Primary Keys	Alternate Keys	Foreign Keys	
Alumnos	DNI, Nombre, Apellidos, Fecha_Nacimiento, Email	DNI	-	-	
AlumnosPertenecenAGrupos	DNI, OID_Grup	DNI, OID_Grup	-	DNI / Alumnos	OID_Grup / Grupos
Asignaturas	Codigo_Asig, Nombre, Acronimo, Creditos, Periodo, Curso, Tipo, OID_Dep	Codigo_Asig	Nombre	OID_Dep / Departamentos	
AsignaturasPuntuadasConNotas	Codigo_Asig, OID_N	Codigo_Asig, OID_N	-	Codigo_Asig / Asignaturas	OID_N / Notas
AsignaturasImpartidasEnAulas	Codigo_Asig, OID_A, Fecha	Codigo_Asig, OID_A, Fecha	-	Codigo_Asig / Asignaturas	OID_A / Aulas
AsignaturasPertenecenAGrados	Codigo_Asig, OID_G	Codigo_Asig, OID_G	-	Codigo_Asig / Asignaturas	OID_G / Grados
Aulas	OID_A, Nombre, Capacidad, Tipo	OID_A	Nombre	-	
Becarios	DNI, Codigo_Beca, Fecha_Comienzo, Fecha_Fin	DNI, Codigo_Beca	-	DNI / Alumnos	Codigo_Beca / Becas
Becas	Codigo_Beca, Cuantia_Fija, Cuantia_Variable, Tipo	Codigo_Beca	-	-	
Departamentos	OID_Dep, Nombre	OID_Dep	Nombre	-	
Despachos	OID_D, Nombre, Capacidad	OID_D	Nombre	-	
Expedientes	Codigo_Exp, DNI	Codigo_Exp	-	DNI / Alumnos	
Grados	OID_G, Nombre	OID_G	Nombre	-	
Grupos	OID_Grup, Nombre	OID_Grup	Nombre	-	
GruposAsistenAClaseEnAulas	OID_Grup, OID_A	OID_Grup, OID_A	-	OID_Grup / Grupos	OID_A / Aulas
Matriculas	Codigo_Matric, Fecha, Curso, DNI	Codigo_Matric	-	DNI / Alumnos	
MatriculasContienenAsignaturas	Codigo_Matric, Codigo_Asig	Codigo_Matric, Codigo_Asig	-	Codigo_Matric / Matriculas	Codigo_Asig / Asignaturas
MatriculasPertenecenAGrados	Codigo_Matric, OID_G	Codigo_Matric, OID_G	-	Codigo_Matric / Matriculas	OID_G / Grados
Notas	OID_N, Valor, Calificacion, Convocatoria, Curso, Fecha, Codigo_Exp	OID_N	-	Codigo_Exp / Expedientes	
Profesores	DNI, Nombre, Apellidos, Fecha_Nacimiento, Email, Categoria, OID_D, OID_Dep	DNI	-	OID_D / Despachos	OID_Dep / Departamentos
ProfesoresImpartenAsignaturas	DNI, Codigo_Asig, Dedicacion	DNI, Codigo_Asig	-	DNI / Profesores	Codigo_Asig / Asignaturas
ProfesoresImpartenEnGrupos	DNI, OID_Grup	DNI, OID_Grup	-	DNI / Profesores	OID_Grup / Grupos
Tutorias	OID_T, Fecha, Hora_Comienzo, Duracion, DNI_Alum, DNI_Prof	OID_T	-	DNI / Profesores	DNI / Alumnos



Transformación de entidades:

Al realizar la transformación del modelo conceptual al modelo relacional, para cada entidad se ha trasformado en una relación cuyo nombre ha pasado a estar en plural, con sus correspondientes atributos necesarios.

En aquellas relaciones donde hubiese atributos que sean claves semánticas sencillas, se han usado para definir las claves primarias. En el resto, se han creado unas OIDs para definir como clave primaria.

En aquellas relaciones donde hubiese atributos que sean claves semánticas y no se hayan definido como claves primarias, se han definidos claves alternativas.

Para el caso de los enumerados, se creará una restricción asociado al atributo de su correspondiente tabla.

Transformación de asociaciones:

En el caso de las asociaciones con cardinalidad 1:N, se ha creado un atributo en la tabla con participación [N] que será una clave ajena apuntando al atributo con clave primaria de la tabla con participación [1].

En el caso de las asociaciones con cardinalidad 1:1, se ha creado un atributo en una de las tablas que será una clave ajena apuntando al atributo con clave primaria de la otra tabla.

En el caso de las asociaciones con cardinalidad N:M, se ha creado una tabla auxiliar que relate ambas entidades, la cual tendrá atributos que serán una clave primaria compuesta y ajenas que apuntan al atributo con clave primaria de cada tabla de cada lado de la asociación.

Transformación de clasificaciones:

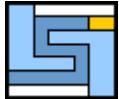
El modelo conceptual cuenta con entidades que son heredadas de otras entidades jerárquicamente superiores:

- En el caso de *Persona*, que hereda a *Alumno* y *Profesor*, se trata de una clasificación disjunta y completa, luego la estrategia más viable es crear una tabla *Alumnos* y otra tabla *Profesores*.
- En el caso de *Espacio*, que hereda a *Despacho* y *Aula*, se trata de una clasificación disjunta y completa, luego la estrategia más viable es crear una tabla *Despachos* y otra tabla *Aulas*.
- En el caso de *Alumno*, que hereda a *Becario*, se trata de una clasificación disjunta e incompleta, luego la estrategia más viable es crear una tabla *Alumnos* y otra tabla *Becarios*.

Normalización del modelo relacional:

Cumpliendo todas las transformaciones anteriores se obtiene el modelo relacional en 3FN. La justificación es:

- Está en 1FN porque los atributos son monovaluados.
- Está en 2FN porque está en 1FN y todos los atributos que no forman parte de ninguna clave candidata son completamente dependientes de las claves candidatas de la relación.
- Está en 3FN porque está en 2FN y no existen dependencias transitivas.



10. Modelo tecnológico

Tablas con restricciones

```
--Creación de tablas
CREATE TABLE Alumnos(
    DNI CHAR(9),
    Nombre VARCHAR2(20),
    Apellidos VARCHAR2(35),
    Fecha_Nacimiento DATE,
    Email VARCHAR2(40)
);

CREATE TABLE AlumnosPertenecenAGrupos (
    DNI CHAR(9),
    OID_Grup NUMBER
);

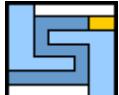
CREATE TABLE Asignaturas(
    Codigo_Asig INTEGER,
    Nombre VARCHAR2(100),
    Acronimo VARCHAR2(7),
    Creditos NUMBER,
    Periodo VARCHAR2(25),
    Curso INTEGER,
    Tipo VARCHAR2(25),
    OID_Dep NUMBER
);

CREATE TABLE AsignaturasPuntuadasConNotas (
    Codigo_Asig INTEGER,
    OID_N NUMBER
);

CREATE TABLE AsignaturasImpartidasEnAulas (
    Codigo_Asig INTEGER,
    OID_A NUMBER,
    Fecha DATE
);

CREATE TABLE AsignaturasPertenecenAGrados (
    Codigo_Asig INTEGER,
    OID_G NUMBER
);

CREATE TABLE Aulas (
    OID_A NUMBER,
    Nombre VARCHAR2(6),
    Capacidad INTEGER,
    Tipo VARCHAR2(15)
);
```



```
CREATE TABLE Becarios(
    DNI CHAR(9),
    Codigo_Beca CHAR(8),
    Fecha_Comienzo DATE,
    Fecha_Fin DATE
);

CREATE TABLE Becas(
    Codigo_Beca CHAR(8),
    Cuantia_Fija NUMBER,
    Cuantia_Variable NUMBER,
    Tipo VARCHAR2(15)
);

CREATE TABLE Departamentos(
    OID_Dep NUMBER,
    Nombre VARCHAR2(50)
);

CREATE TABLE Despachos(
    OID_D NUMBER,
    Nombre VARCHAR2(6),
    Capacidad INTEGER
);

CREATE TABLE Expedientes(
    Codigo_Exp INTEGER,
    DNI CHAR(9)
);

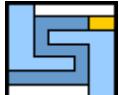
CREATE TABLE Grados(
    OID_G NUMBER,
    Nombre VARCHAR2(100)
);

CREATE TABLE Grupos(
    OID_Grup NUMBER,
    Nombre VARCHAR2(10)
);

CREATE TABLE GruposAsistenAClaseEnAulas(
    OID_Grup NUMBER,
    OID_A NUMBER
);

CREATE TABLE Matriculas(
    Codigo_Matric INTEGER,
    Fecha DATE,
    Curso INTEGER,
    DNI CHAR(9)
);

CREATE TABLE MatriculasContienenAsignaturas(
    Codigo_Matric INTEGER,
    Codigo_Asig INTEGER
);
```



```
CREATE TABLE MatriculasPertenecenAGrados (
   Codigo_Matric INTEGER,
   OID_G NUMBER
);

CREATE TABLE Notas (
   OID_N NUMBER,
   Valor NUMBER,
   Calificacion VARCHAR2(25),
   Convocatoria VARCHAR2(15),
   Curso INTEGER,
   Fecha DATE,
   Codigo_Exp INTEGER
);

CREATE TABLE Profesores (
   DNI CHAR(9),
   Nombre VARCHAR2(20),
   Apellidos VARCHAR2(35),
   Fecha_Nacimiento DATE,
   Email VARCHAR2(40),
   Categoria VARCHAR2(20),
   OID_D NUMBER,
   OID_Dep NUMBER
);

CREATE TABLE ProfesoresImpartenAsignaturas (
   DNI CHAR(9),
   Codigo_Asig INTEGER,
   Dedicacion NUMBER
);

CREATE TABLE ProfesoresImpartenEnGrupos (
   DNI CHAR(9),
   OID_Grup NUMBER
);

CREATE TABLE Tutorias (
   OID_T NUMBER,
   Fecha DATE,
   Hora_Comienzo VARCHAR2(5),
   Duracion INTEGER,
   DNI_Alum CHAR(9),
   DNI_Prof CHAR(9)
);
```



```
--Restricciones NOT NULL
ALTER TABLE Alumnos MODIFY (DNI NOT NULL);
ALTER TABLE AlumnosPertenecenAGrupos MODIFY (DNI NOT NULL);
ALTER TABLE AlumnosPertenecenAGrupos MODIFY (OID_Grupo NOT NULL);
ALTER TABLE Asignaturas MODIFY (Codigo_Asig NOT NULL);
ALTER TABLE Asignaturas MODIFY (Nombre NOT NULL);
ALTER TABLE AsignaturasPuntuadasConNotas MODIFY (Codigo_Asig NOT NULL);
ALTER TABLE AsignaturasPuntuadasConNotas MODIFY (OID_N NOT NULL);
ALTER TABLE AsignaturasImpartidasEnAulas MODIFY (Codigo_Asig NOT NULL);
ALTER TABLE AsignaturasImpartidasEnAulas MODIFY (OID_A NOT NULL);
ALTER TABLE AsignaturasImpartidasEnAulas MODIFY (Fecha NOT NULL);
ALTER TABLE AsignaturasPertenecenAGrados MODIFY (Codigo_Asig NOT NULL);
ALTER TABLE AsignaturasPertenecenAGrados MODIFY (OID_G NOT NULL);
ALTER TABLE Aulas MODIFY (OID_A NOT NULL);
ALTER TABLE Aulas MODIFY (Nombre NOT NULL);
ALTER TABLE Becarios MODIFY (DNI NOT NULL);
ALTER TABLE Becarios MODIFY (Codigo_Beca NOT NULL);
ALTER TABLE Becas MODIFY (Codigo_Beca NOT NULL);
ALTER TABLE Departamentos MODIFY (OID_Dep NOT NULL);
ALTER TABLE Departamentos MODIFY (Nombre NOT NULL);
ALTER TABLE Despachos MODIFY (OID_D NOT NULL);
ALTER TABLE Despachos MODIFY (Nombre NOT NULL);
ALTER TABLE Expedientes MODIFY (Codigo_Exp NOT NULL);
ALTER TABLE Expedientes MODIFY (DNI NOT NULL);
ALTER TABLE Grados MODIFY (OID_G NOT NULL);
ALTER TABLE Grados MODIFY (Nombre NOT NULL);
ALTER TABLE Grupos MODIFY (OID_Grupo NOT NULL);
ALTER TABLE Grupos MODIFY (Nombre NOT NULL);
ALTER TABLE GruposAsistenAClaseEnAulas MODIFY (OID_Grupo NOT NULL);
ALTER TABLE GruposAsistenAClaseEnAulas MODIFY (OID_A NOT NULL);
ALTER TABLE Matriculas MODIFY (Codigo_Matric NOT NULL);
ALTER TABLE Matriculas MODIFY (DNI NOT NULL);
ALTER TABLE MatriculasContienenAsignaturas MODIFY (Codigo_Matric NOT NULL);
ALTER TABLE MatriculasContienenAsignaturas MODIFY (Codigo_Asig NOT NULL);
ALTER TABLE MatriculasPertenecenAGrados MODIFY (Codigo_Matric NOT NULL);
ALTER TABLE MatriculasPertenecenAGrados MODIFY (OID_G NOT NULL);
ALTER TABLE Notas MODIFY (OID_N NOT NULL);
ALTER TABLE Notas MODIFY (Codigo_Exp NOT NULL);
ALTER TABLE Profesores MODIFY (DNI NOT NULL);
ALTER TABLE ProfesoresImpartenAsignaturas MODIFY (DNI NOT NULL);
ALTER TABLE ProfesoresImpartenAsignaturas MODIFY (Codigo_Asig NOT NULL);
ALTER TABLE ProfesoresImpartenEnGrupos MODIFY (DNI NOT NULL);
ALTER TABLE ProfesoresImpartenEnGrupos MODIFY (OID_Grupo NOT NULL);
ALTER TABLE Tutorias MODIFY (OID_T NOT NULL);
ALTER TABLE Tutorias MODIFY (DNI_Alumno NOT NULL);
ALTER TABLE Tutorias MODIFY (DNI_Profesor NOT NULL);
```



```
--Primary Keys
ALTER TABLE Alumnos ADD CONSTRAINT PK_Alumnos PRIMARY KEY (DNI);
ALTER TABLE AlumnosPertenecenAGrupos ADD CONSTRAINT PK_AlumPertAGrup PRIMARY KEY (DNI, OID_Grup);
ALTER TABLE Asignaturas ADD CONSTRAINT PK_Asignaturas PRIMARY KEY (Codigo_Asig);
ALTER TABLE AsignaturasPuntuadasConNotas ADD CONSTRAINT PK_AsigPuntConNot PRIMARY KEY (Codigo_Asig, OID_N);
ALTER TABLE AsignaturasImpartidasEnAulas ADD CONSTRAINT PK_AsigImpartEnAulas PRIMARY KEY (Codigo_Asig, OID_A,
Fecha);
ALTER TABLE AsignaturasPertenecenAGrados ADD CONSTRAINT PK_AsigPertAGrad PRIMARY KEY (Codigo_Asig, OID_G);
ALTER TABLE Aulas ADD CONSTRAINT PK_Aulas PRIMARY KEY (OID_A);
ALTER TABLE Becarios ADD CONSTRAINT PK_Becarios PRIMARY KEY (DNI, Codigo_Beca);
ALTER TABLE Becas ADD CONSTRAINT PK_Becas PRIMARY KEY (Codigo_Beca);
ALTER TABLE Departamentos ADD CONSTRAINT PK_Departamentos PRIMARY KEY (OID_Dep);
ALTER TABLE Despachos ADD CONSTRAINT PK_Despachos PRIMARY KEY (OID_D);
ALTER TABLE Expedientes ADD CONSTRAINT PK_Expedientes PRIMARY KEY (Codigo_Exp);
ALTER TABLE Grados ADD CONSTRAINT PK_Grados PRIMARY KEY (OID_G);
ALTER TABLE Grupos ADD CONSTRAINT PK_Grupos PRIMARY KEY (OID_Grup);
ALTER TABLE GruposAsistenAClaseEnAulas ADD CONSTRAINT PK_GrupAsisAClaseEnAulas PRIMARY KEY (OID_Grup, OID_A);
ALTER TABLE Matriculas ADD CONSTRAINT PK_Matriculas PRIMARY KEY (Codigo_Matric);
ALTER TABLE MatriculasContienenAsignaturas ADD CONSTRAINT PK_MatricContAsig PRIMARY KEY (Codigo_Matric,
Codigo_Asig);
ALTER TABLE MatriculasPertenecenAGrados ADD CONSTRAINT PK_MatricPertAGrad PRIMARY KEY (Codigo_Matric, OID_G);
ALTER TABLE Notas ADD CONSTRAINT PK_Notas PRIMARY KEY (OID_N);
ALTER TABLE Profesores ADD CONSTRAINT PK_Profesores PRIMARY KEY (DNI);
ALTER TABLE ProfesoresImpartenAsignaturas ADD CONSTRAINT PK_ProfImpAsig PRIMARY KEY (DNI, Codigo_Asig);
ALTER TABLE ProfesoresImpartenEnGrupos ADD CONSTRAINT PK_ProfImpEnGrup PRIMARY KEY (DNI, OID_Grup);
ALTER TABLE Tutorias ADD CONSTRAINT PK_Tutorias PRIMARY KEY (OID_T);
```

```
--Alternate Keys
ALTER TABLE Asignaturas ADD CONSTRAINT AK_Asignaturas UNIQUE (Nombre);
ALTER TABLE Aulas ADD CONSTRAINT AK_Aulas UNIQUE (Nombre);
ALTER TABLE Departamentos ADD CONSTRAINT AK_Departamentos UNIQUE (Nombre);
ALTER TABLE Despachos ADD CONSTRAINT AK_Despachos UNIQUE (Nombre);
ALTER TABLE Grados ADD CONSTRAINT AK_Grados UNIQUE (Nombre);
ALTER TABLE Grupos ADD CONSTRAINT AK_Grupos UNIQUE (Nombre);
```

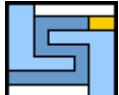


--Foreign Keys

```
ALTER TABLE AlumnosPertenecenAGrupos ADD CONSTRAINT FK_AlumPertAGrup_Alum FOREIGN KEY (DNI) REFERENCES Alumnos (DNI) ON DELETE CASCADE;
ALTER TABLE AlumnosPertenecenAGrupos ADD CONSTRAINT FK_AlumPertAGrup_Grup FOREIGN KEY (OID_Grup) REFERENCES Grupos (OID_Grup) ON DELETE CASCADE;
ALTER TABLE Asignaturas ADD CONSTRAINT FK_Asignaturas_Dep FOREIGN KEY (OID_Dep) REFERENCES Departamentos (OID_Dep) ON DELETE CASCADE;
ALTER TABLE AsignaturasPuntuadasConNotas ADD CONSTRAINT FK_AsigPuntConNot_Asig FOREIGN KEY (Codigo_Asig) REFERENCES Asignaturas (Codigo_Asig) ON DELETE CASCADE;
ALTER TABLE AsignaturasPuntuadasConNotas ADD CONSTRAINT FK_AsigPuntConNot_Not FOREIGN KEY (OID_N) REFERENCES Notas (OID_N) ON DELETE CASCADE;
ALTER TABLE AsignaturasImpartidasEnAulas ADD CONSTRAINT FK_AsigImpartEnAulas_Asig FOREIGN KEY (Codigo_Asig) REFERENCES Asignaturas (Codigo_Asig) ON DELETE CASCADE;
ALTER TABLE AsignaturasImpartidasEnAulas ADD CONSTRAINT FK_AsigImpartEnAulas_Aulas FOREIGN KEY (OID_A) REFERENCES Aulas (OID_A) ON DELETE CASCADE;
ALTER TABLE AsignaturasPertenecenAGrados ADD CONSTRAINT FK_AsigPertAGrad_Asig FOREIGN KEY (Codigo_Asig) REFERENCES Asignaturas (Codigo_Asig) ON DELETE CASCADE;
ALTER TABLE AsignaturasPertenecenAGrados ADD CONSTRAINT FK_AsigPertAGrad_G FOREIGN KEY (OID_G) REFERENCES Grados (OID_G) ON DELETE CASCADE;
ALTER TABLE Becarios ADD CONSTRAINT FK_Becarios_Alum FOREIGN KEY (DNI) REFERENCES Alumnos (DNI) ON DELETE CASCADE;
ALTER TABLE Becarios ADD CONSTRAINT FK_Becarios_Becas FOREIGN KEY (Codigo_Beca) REFERENCES Becas (Codigo_Beca) ON DELETE CASCADE;
ALTER TABLE Expedientes ADD CONSTRAINT FK_Expedientes_Alum FOREIGN KEY (DNI) REFERENCES Alumnos (DNI) ON DELETE CASCADE;
ALTER TABLE GruposAsistenAClaseEnAulas ADD CONSTRAINT FK_GrupAsisAClaseEnAulas_Grup FOREIGN KEY (OID_Grup) REFERENCES Grupos (OID_Grup) ON DELETE CASCADE;
ALTER TABLE GruposAsistenAClaseEnAulas ADD CONSTRAINT FK_GrupAsisAClaseEnAulas_Aulas FOREIGN KEY (OID_A) REFERENCES Aulas (OID_A) ON DELETE CASCADE;
ALTER TABLE Matriculas ADD CONSTRAINT FK_Matriculas_Alum FOREIGN KEY (DNI) REFERENCES Alumnos (DNI) ON DELETE CASCADE;
ALTER TABLE MatriculasContienenAsignaturas ADD CONSTRAINT FK_MatricContAsig_Matric FOREIGN KEY (Codigo_Matric) REFERENCES Matriculas (Codigo_Matric) ON DELETE CASCADE;
ALTER TABLE MatriculasContienenAsignaturas ADD CONSTRAINT FK_MatricContAsig_Asig FOREIGN KEY (Codigo_Asig) REFERENCES Asignaturas (Codigo_Asig) ON DELETE CASCADE;
ALTER TABLE MatriculasPertenecenAGrados ADD CONSTRAINT FK_MatricPertAGrad_Matric FOREIGN KEY (Codigo_Matric) REFERENCES Matriculas (Codigo_Matric) ON DELETE CASCADE;
ALTER TABLE MatriculasPertenecenAGrados ADD CONSTRAINT FK_MatricPertAGrad_G FOREIGN KEY (OID_G) REFERENCES Grados (OID_G) ON DELETE CASCADE;
ALTER TABLE Notas ADD CONSTRAINT FK_Notas_Exp FOREIGN KEY (Codigo_Exp) REFERENCES Expedientes (Codigo_Exp) ON DELETE CASCADE;
ALTER TABLE Profesores ADD CONSTRAINT FK_Profesores_D FOREIGN KEY (OID_D) REFERENCES Despachos (OID_D) ON DELETE CASCADE;
ALTER TABLE Profesores ADD CONSTRAINT FK_Profesores_Dep FOREIGN KEY (OID_Dep) REFERENCES Departamentos (OID_Dep) ON DELETE CASCADE;
ALTER TABLE ProfesoresImpartenAsignaturas ADD CONSTRAINT FK_ProfImpAsig_Prof FOREIGN KEY (DNI) REFERENCES Profesores (DNI) ON DELETE CASCADE;
ALTER TABLE ProfesoresImpartenAsignaturas ADD CONSTRAINT FK_ProfImpAsig_Asig FOREIGN KEY (Codigo_Asig) REFERENCES Asignaturas (Codigo_Asig) ON DELETE CASCADE;
ALTER TABLE ProfesoresImpartenEnGrupos ADD CONSTRAINT FK_ProfImpEnGrup_Prof FOREIGN KEY (DNI) REFERENCES Profesores (DNI) ON DELETE CASCADE;
ALTER TABLE ProfesoresImpartenEnGrupos ADD CONSTRAINT FK_ProfImpEnGrup_Grup FOREIGN KEY (OID_Grup) REFERENCES Grupos (OID_Grup) ON DELETE CASCADE;
ALTER TABLE Tutorias ADD CONSTRAINT FK_Tutorias_Alum FOREIGN KEY (DNI_Alum) REFERENCES Alumnos (DNI) ON DELETE CASCADE;
ALTER TABLE Tutorias ADD CONSTRAINT FK_Tutorias_Prof FOREIGN KEY (DNI_Prof) REFERENCES Profesores (DNI) ON DELETE CASCADE;
```



```
--Restricciones de tablas
ALTER TABLE Alumnos ADD CONSTRAINT CK_DNI_Alumnos CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE AlumnosPertenecenAGrupos ADD CONSTRAINT CK_DNI_AlumPertAGrup CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Asignaturas ADD CONSTRAINT CK_Cod_Asig_Asignaturas CHECK (REGEXP_LIKE(Codigo_Asig, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE Asignaturas ADD CONSTRAINT CK_Periodo_Asignaturas CHECK (Periodo IN ('Anual', 'Primer cuatrimestre', 'Segundo cuatrimestre'));
ALTER TABLE Asignaturas ADD CONSTRAINT CK_Curso_Asignaturas CHECK (Curso IN ('1', '2', '3', '4'));
ALTER TABLE Asignaturas ADD CONSTRAINT CK_Tipo_Asignaturas CHECK (Tipo IN ('Troncal', 'Obligatorio', 'Optativo'));
ALTER TABLE AsignaturasPuntuadasConNotas ADD CONSTRAINT CK_Cod_Asig_AsigPuntConNot CHECK (REGEXP_LIKE(Codigo_Asig, '[0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE AsignaturasImpartidasEnAulas ADD CONSTRAINT CK_Cod_Asig_AsigImpartEnAulas CHECK (REGEXP_LIKE(Codigo_Asig, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE AsignaturasPertenecenAGrados ADD CONSTRAINT CK_Cod_Asig_AsigPertAGrad CHECK (REGEXP_LIKE(Codigo_Asig, '[0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE Aulas ADD CONSTRAINT CK_Tipo_Aulas CHECK (Tipo IN ('Teoría', 'Laboratorio', 'Examen'));
ALTER TABLE Becarios ADD CONSTRAINT CK_DNI_Becarios CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Becarios ADD CONSTRAINT CK_Codigo_Becas_Becarios CHECK (REGEXP_LIKE(Codigo_Beca, '[0-9][0-9][0-9][0-9][A-Z][A-Z][A-Z]'));
ALTER TABLE Becarios ADD CONSTRAINT CK_Fechas_Becarios CHECK (TO_DATE(Fecha_Comienzo, 'DD/MM/YYYY') < TO_DATE(Fecha_Fin, 'DD/MM/YYYY'));
ALTER TABLE Becas ADD CONSTRAINT CK_Codigo_Becas CHECK (REGEXP_LIKE(Codigo_Beca, '[0-9][0-9][0-9][0-9][0-9][A-Z][A-Z][A-Z]'));
ALTER TABLE Becas ADD CONSTRAINT CK_Tipo_Becas CHECK (Tipo IN ('Ordinaria', 'Movilidad', 'Empresa'));
ALTER TABLE Expedientes ADD CONSTRAINT CK_Codigo_Expedientes CHECK (REGEXP_LIKE(Codigo_Exp, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE Expedientes ADD CONSTRAINT CK_DNI_Expedientes CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Matriculas ADD CONSTRAINT CK_Cod_Matric_Matriculas CHECK (REGEXP_LIKE(Codigo_Matric, '[0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE Matriculas ADD CONSTRAINT CK_Curso_Matriculas CHECK (Curso IN ('1', '2', '3', '4'));
ALTER TABLE MatriculasContienenAsignaturas ADD CONSTRAINT CK_Cod_Matric_MatricContAsig CHECK (REGEXP_LIKE(Codigo_Matric, '[0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE MatriculasContienenAsignaturas ADD CONSTRAINT CK_Cod_Asig_MatricContAsig CHECK (REGEXP_LIKE(Codigo_Asig, '[0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE MatriculasPertenecenAGrados ADD CONSTRAINT CK_Cod_Matric_MatricPertAGrad CHECK (REGEXP_LIKE(Codigo_Matric, '[0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE Notas ADD CONSTRAINT CK_Calificacion_Notas CHECK (Calificacion IN ('Suspensos', 'Aprobado', 'Notable', 'Sobresaliente', 'Matrícula de honor'));
ALTER TABLE Notas ADD CONSTRAINT CK_Convocatoria_Notas CHECK (Convocatoria IN ('Primera', 'Segunda', 'Tercera'));
ALTER TABLE Notas ADD CONSTRAINT CK_Curso_Notas CHECK (Curso IN ('1', '2', '3', '4'));
ALTER TABLE Profesores ADD CONSTRAINT CK_DNI_Profesores CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Profesores ADD CONSTRAINT CK_Categoría_Profesores CHECK (Categoria IN ('Catedrático', 'Titular', 'Contratado doctor', 'Colaborador', 'Ayudante doctor', 'Ayudante', 'Interino'));
ALTER TABLE ProfesoresImpartenAsignaturas ADD CONSTRAINT CK_DNI_ProfImpAsig CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE ProfesoresImpartenAsignaturas ADD CONSTRAINT CK_Cod_Asig_ProfImpAsig CHECK (REGEXP_LIKE(Codigo_Asig, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'));
ALTER TABLE ProfesoresImpartenEnGrupos ADD CONSTRAINT CK_DNI_ProfImpEnGrup CHECK (REGEXP_LIKE(DNI, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Tutorias ADD CONSTRAINT CK_Hora_Comienzo_Tutorias CHECK (REGEXP_LIKE(Hora_Comienzo, '[0-2][0-9]:[0-5][0-9]'));
ALTER TABLE Tutorias ADD CONSTRAINT CK_DNI_Alum_Tutorias CHECK (REGEXP_LIKE(DNI_Alum, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
ALTER TABLE Tutorias ADD CONSTRAINT CK_DNI_Prof_Tutorias CHECK (REGEXP_LIKE(DNI_Prof, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));
```



Secuencias y Triggers asociados a secuencias



--Secuencias

```
CREATE SEQUENCE SEC_Aulas
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```

```
CREATE SEQUENCE SEC_Departamentos
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```

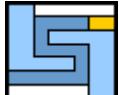
```
CREATE SEQUENCE SEC_Despachos
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```

```
CREATE SEQUENCE SEC_Grados
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```

```
CREATE SEQUENCE SEC_Grupos
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```

```
CREATE SEQUENCE SEC_Notas
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```

```
CREATE SEQUENCE SEC_Tutorias
INCREMENT BY 1
START WITH 1
MAXVALUE 99999;
```



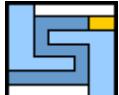
```
--Triggers asociados a secuencias
CREATE OR REPLACE TRIGGER TR_SEC_Aulas
BEFORE INSERT ON Aulas
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Aulas.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_A := valorSec;
END;
/

CREATE OR REPLACE TRIGGER TR_SEC_Departamentos
BEFORE INSERT ON Departamentos
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Departamentos.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_Dep := valorSec;
END;
/

CREATE OR REPLACE TRIGGER TR_SEC_Despachos
BEFORE INSERT ON Despachos
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Despachos.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_D := valorSec;
END;
/

CREATE OR REPLACE TRIGGER TR_SEC_Grados
BEFORE INSERT ON Grados
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Grados.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_G := valorSec;
END;
/

CREATE OR REPLACE TRIGGER TR_SEC_Grupos
BEFORE INSERT ON Grupos
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Grupos.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_Grup := valorSec;
END;
/
```



```
CREATE OR REPLACE TRIGGER TR_SEC_Notas
BEFORE INSERT ON Notas
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Notas.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_N := valorSec;
END;
/

CREATE OR REPLACE TRIGGER TR_SEC_Tutorias
BEFORE INSERT ON Tutorias
FOR EACH ROW
DECLARE
    valorSec NUMBER := 0;
BEGIN
    SELECT SEC_Tutorias.NEXTVAL INTO valorSec FROM DUAL;
    :NEW.OID_T := valorSec;
END;
/
```



Procedimientos y funciones asociadas a las reglas funcionales

```
--Procedimientos y funciones asociadas a las reglas funcionales
--RF-001
CREATE OR REPLACE PROCEDURE PR_Matricular_Asig (v_codigo_matric IN MatriculasContienenAsignaturas.Codigo_Matric%TYPE, v_codigo_asig IN
MatriculasContienenAsignaturas.Codigo_Asig%TYPE)
IS
BEGIN
    INSERT INTO MatriculasContienenAsignaturas (Codigo_Matric, Codigo_Asig) VALUES (v_codigo_matric, v_codigo_asig);
    COMMIT;
END;
/

--RF-002
CREATE OR REPLACE PROCEDURE PR_Anular_Asig (v_codigo_matric IN MatriculasContienenAsignaturas.Codigo_Matric%TYPE, v_codigo_asig IN
MatriculasContienenAsignaturas.Codigo_Asig%TYPE)
IS
BEGIN
    DELETE FROM MatriculasContienenAsignaturas WHERE Codigo_Matric = v_codigo_matric AND Codigo_Asig = v_codigo_asig;
    COMMIT;
END;
/

--RF-003
CREATE OR REPLACE PROCEDURE PR_Evaluar_Alum (v_codigo_asig IN AsignaturasPuntuadasConNotas.Codigo_Asig%TYPE, v_OID_N IN AsignaturasPuntuadasConNotas.OID_N%TYPE)
IS
BEGIN
    INSERT INTO AsignaturasPuntuadasConNotas (Codigo_Asig, OID_N) VALUES (v_codigo_asig, v_OID_N);
    COMMIT;
END;
/

--RF-004
CREATE OR REPLACE PROCEDURE PR_Alum_Matriculado_En_Asig (v_DNI IN Alumnos.DNI%TYPE, v_codigo_asig IN Asignaturas.Codigo_Asig%TYPE)
IS
    v_EstaMatriculado NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_EstaMatriculado FROM MatriculasContienenAsignaturas WHERE Codigo_Asig = v_codigo_asig AND Codigo_Matric IN (SELECT Codigo_Matric FROM Matriculas
WHERE DNI = v_DNI);
    IF v_EstaMatriculado = 1 THEN
        DBMS_OUTPUT.PUT_LINE('El alumno está matriculado en dicha asignatura.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('El alumno no está matriculado en dicha asignatura.');
    END IF;
END;
/
```



```
--RF-005
CREATE OR REPLACE PROCEDURE PR_CalfMed_Agrup_Por_Asig (v_DNI IN Alumnos.DNI%TYPE)
IS
  CURSOR C IS
    SELECT Nombre, AVG(Valor) AS Nota_Media FROM Asignaturas NATURAL JOIN AsignaturasPuntuadasConNotas NATURAL JOIN Notas WHERE Código_Exp IN (SELECT Código_Exp FROM Expedientes WHERE DNI = v_DNI) GROUP BY Nombre;
    v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Asignaturas:', 100) || RPAD('Nota media:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 125, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Nota_Media, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/

--RF-006
CREATE OR REPLACE PROCEDURE PR_Asig_Orden_Por_Curso (v_DNI IN Alumnos.DNI%TYPE)
IS
  CURSOR C IS
    SELECT Nombre FROM Asignaturas NATURAL JOIN MatriculasContienenAsignaturas WHERE Código_Matric IN (SELECT Código_Matric FROM Matriculas WHERE DNI = v_DNI) ORDER BY Curso;
    v_NombreAsig Asignaturas.Nombre%TYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_NombreAsig;
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(v_NombreAsig);
    FETCH C INTO v_NombreAsig;
  END LOOP;
  CLOSE C;
END;
/

--RF-007
CREATE OR REPLACE PROCEDURE PR_Expediente_Alumno (v_DNI IN Alumnos.DNI%TYPE)
IS
  CURSOR C IS
    SELECT Nombre, Valor, Calificación, Convocatoria, Curso, Fecha FROM Notas NATURAL JOIN Expedientes NATURAL JOIN AsignaturasPuntuadasConNotas NATURAL JOIN Asignaturas WHERE Código_Exp IN (SELECT Código_Exp FROM Expedientes WHERE DNI = v_DNI);
    v_Notas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Notas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Asignaturas:', 25) || RPAD('Nota:', 25) || RPAD('Calificación', 25) || RPAD('Convocatoria:', 25) || RPAD('Curso:', 25) || RPAD('Fecha:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 135, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Notas.Nombre, 25) || RPAD(v_Notas.Valor, 25) || RPAD(v_Notas.Calificación, 25) || RPAD(v_Notas.Convocatoria, 25) || RPAD(v_Notas.Curso, 25) || RPAD(v_Notas.Fecha, 25));
    FETCH C INTO v_Notas;
  END LOOP;
  CLOSE C;
END;
/
```



```
--RF-008
CREATE OR REPLACE PROCEDURE PR_Asociar_Asignatura_Dep (v_codigo_asig IN Asignaturas.Codigo_Asig%TYPE, v_NombreDepartamento IN Departamentos.Nombre%TYPE)
IS
    v_OID_Dep Asignaturas.OID_Dep%TYPE;
BEGIN
    SELECT OID_Dep INTO v_OID_Dep FROM Departamentos WHERE Nombre = v_NombreDepartamento;
    UPDATE Asignaturas SET OID_Dep = v_OID_Dep WHERE Codigo_Asig = v_codigo_asig;
    COMMIT;
END;
/

--RF-009
CREATE OR REPLACE PROCEDURE PR_Asociar_Beca (v_codigo_beca IN Becarios.Codigo_Beca%TYPE, v_DNI IN Becarios.DNI%TYPE, v_Fecha_Comienzo IN Becarios.Fecha_Comienzo%TYPE,
v_Fecha_Fin IN Becarios.Fecha_Fin%TYPE)
IS
BEGIN
    INSERT INTO Becarios (DNI, Codigo_Beca, Fecha_Comienzo, Fecha_Fin) VALUES (v_DNI, v_Codigo_Beca, v_Fecha_Comienzo, v_Fecha_Fin);
    COMMIT;
END;
/

--RF-010
CREATE OR REPLACE PROCEDURE PR_Asociar_Duracion_Beca (v_codigo_beca IN Becas.Codigo_Beca%TYPE, v_DNI IN Becarios.DNI%TYPE, v_Fecha_Fin IN Becarios.Fecha_Fin%TYPE)
IS
    v_Existe NUMBER;
    v_FechaComienzo DATE := SYSDATE;
BEGIN
    SELECT COUNT(*) INTO v_Existe FROM Becarios WHERE DNI = v_DNI AND Codigo_Beca = v_codigo_beca;
    IF v_Existe = 1 THEN
        UPDATE Becarios SET Fecha_Fin = v_Fecha_Fin WHERE DNI = v_DNI AND Codigo_Beca = v_codigo_beca;
    ELSE
        INSERT INTO Becarios (DNI, Codigo_Beca, Fecha_Comienzo, Fecha_Fin) VALUES (v_DNI, v_Codigo_Beca, v_FechaComienzo, v_Fecha_Fin);
    END IF;
    COMMIT;
END;
/

--RF-011
CREATE OR REPLACE PROCEDURE PR_Asociar_Cuantia_Fija_Beca (v_codigo_beca IN Becas.Codigo_Beca%TYPE, v_cuantia_fija IN Becas.Cuantia_Fija%TYPE)
IS
BEGIN
    UPDATE Becas SET Cuantia_Fija = v_cuantia_fija WHERE Codigo_Beca = v_codigo_beca;
    COMMIT;
END;
/

--RF-012
CREATE OR REPLACE PROCEDURE PR_Asociar_Cuantia_Var_Beca (v_codigo_beca IN Becas.Codigo_Beca%TYPE, v_cuantia_variable IN Becas.Cuantia_Variable%TYPE)
IS
BEGIN
    UPDATE Becas SET Cuantia_Variable = v_cuantia_variable WHERE Codigo_Beca = v_codigo_beca;
    COMMIT;
END;
/
```



```
--RF-013
CREATE OR REPLACE PROCEDURE PR_Becas_Alumno (v_DNI IN Becarios.DNI%TYPE)
IS
  CURSOR C IS
    SELECT Código_Beca, Cuantia_Fija, Cuantia_Variable, Tipo, Fecha_Comienzo, Fecha_Fin FROM Becas NATURAL JOIN BECARIOS WHERE DNI = v_DNI;
  v_Becas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Becas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Código de la beca:', 25) || RPAD('Cuantía fija:', 25) || RPAD('Cuantía variable:', 25) || RPAD('Tipo:', 25) || RPAD('Fecha de comienzo:', 25) ||
RPAD('Fecha de fin:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 140, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Becas.Código_Beca, 25) || RPAD(v_Becas.Cuantia_Fija, 25) || RPAD(v_Becas.Cuantia_Variable, 25) || RPAD(v_Becas.Tipo, 25) || RPAD(v_Becas.Fecha_Comienzo, 25) || RPAD(v_Becas.Fecha_Fin, 25));
    FETCH C INTO v_Becas;
  END LOOP;
  CLOSE C;
END;
/

--RF-014
CREATE OR REPLACE PROCEDURE PR_Alumnos_Matriculados
IS
  CURSOR C IS
    SELECT DNI, Nombre, Apellidos, Fecha_Nacimiento, Email FROM Alumnos;
  v_Alumnos C%ROWTYPE;
  v_TotalAlumnos NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_TotalAlumnos FROM Alumnos;
  DBMS_OUTPUT.PUT_LINE(v_TotalAlumnos || ' alumnos matriculados en la universidad.' || CHR(13) || CHR(13));
  OPEN C;
  FETCH C INTO v_Alumnos;
  DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha de nacimiento:', 25) || RPAD('Email:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Alumnos.DNI, 25) || RPAD(v_Alumnos.Nombre, 25) || RPAD(v_Alumnos.Apellidos, 25) || RPAD(v_Alumnos.Fecha_Nacimiento, 25) || RPAD(v_Alumnos.Email, 25));
    FETCH C INTO v_Alumnos;
  END LOOP;
  CLOSE C;
END;
/
```

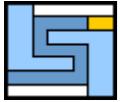


```
--RF-015
CREATE OR REPLACE PROCEDURE PR_Profesores_Impartiendo
IS
CURSOR C IS
    SELECT DNI, Nombre, Apellidos, Fecha_Nacimiento, Email, Categoria FROM Profesores;
    v_Profesores C%ROWTYPE;
    v_TotalProfesores NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_TotalProfesores FROM Profesores;
    DBMS_OUTPUT.PUT_LINE(v_TotalProfesores || ' profesores impartiendo en la universidad.' || CHR(13) || CHR(13));
    OPEN C;
    FETCH C INTO v_Profesores;
    DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha de nacimiento:', 25) || RPAD('Email:', 25) || RPAD('Categoria:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 140, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Profesores.DNI, 25) || RPAD(v_Profesores.Nombre, 25) || RPAD(v_Profesores.Apellidos, 25) || RPAD(v_Profesores.Fecha_Nacimiento, 25) || RPAD(v_Profesores.Email, 25) || RPAD(v_Profesores.Categoría, 25));
        FETCH C INTO v_Profesores;
    END LOOP;
    CLOSE C;
END;
/

--RF-016
CREATE OR REPLACE PROCEDURE PR_Espacios_AgrupPor_Tipo
IS
CURSOR C1 IS
    SELECT Nombre, Capacidad FROM Aulas WHERE Tipo = 'Teoría';
CURSOR C2 IS
    SELECT Nombre, Capacidad FROM Aulas WHERE Tipo = 'Laboratorio';
CURSOR C3 IS
    SELECT Nombre, Capacidad FROM Aulas WHERE Tipo = 'Examen';
CURSOR C4 IS
    SELECT Nombre, Capacidad FROM Despachos;
v_EspaciosT C1%ROWTYPE;
v_EspaciosL C2%ROWTYPE;
v_EspaciosE C3%ROWTYPE;
v_EspaciosD C4%ROWTYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 17, '-'));
    DBMS_OUTPUT.PUT_LINE('|Aulas de teoría|');
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 17, '-') || CHR(13) || CHR(13));
    OPEN C1;
    FETCH C1 INTO v_EspaciosT;
    DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Capacidad:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 40, '-'));
    WHILE C1%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_EspaciosT.Nombre, 25) || RPAD(v_EspaciosT.Capacidad, 25));
        FETCH C1 INTO v_EspaciosT;
    END LOOP;
    CLOSE C1;
    DBMS_OUTPUT.PUT_LINE(CHR(13) || CHR(13));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 22, '-'));
    DBMS_OUTPUT.PUT_LINE('|Aulas de laboratorio|');
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 22, '-') || CHR(13) || CHR(13));
    OPEN C2;
    FETCH C2 INTO v_EspaciosL;
    DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Capacidad:', 25));
```



```
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 40, ' '));
WHILE C2%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE( RPAD(v_EspaciosL.Nombre, 25) || RPAD(v_EspaciosL.Capacidad, 25));
    FETCH C2 INTO v_EspaciosL;
END LOOP;
CLOSE C2;
DBMS_OUTPUT.PUT_LINE( CHR(13) || CHR(13));
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 17, ' '));
DBMS_OUTPUT.PUT_LINE( '|Aulas de examen|');
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 17, ' ') || CHR(13) || CHR(13));
OPEN C3;
FETCH C3 INTO v_EspaciosE;
DBMS_OUTPUT.PUT_LINE( RPAD('Nombre:', 25) || RPAD('Capacidad:', 25));
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 40, ' '));
WHILE C3%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE( RPAD(v_EspaciosE.Nombre, 25) || RPAD(v_EspaciosE.Capacidad, 25));
    FETCH C3 INTO v_EspaciosE;
END LOOP;
CLOSE C3;
DBMS_OUTPUT.PUT_LINE( CHR(13) || CHR(13));
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 11, ' '));
DBMS_OUTPUT.PUT_LINE( '|Despachos|');
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 11, ' ') || CHR(13) || CHR(13));
OPEN C4;
FETCH C4 INTO v_EspaciosD;
DBMS_OUTPUT.PUT_LINE( RPAD('Nombre:', 25) || RPAD('Capacidad:', 25));
DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 40, ' '));
WHILE C4%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE( RPAD(v_EspaciosD.Nombre, 25) || RPAD(v_EspaciosD.Capacidad, 25));
    FETCH C4 INTO v_EspaciosD;
END LOOP;
CLOSE C4;
END;
/
--RF-017
CREATE OR REPLACE PROCEDURE PR_Profesores_Departamento (v_NombreDepartamento IN Departamentos.Nombre%TYPE)
IS
    CURSOR C IS
        SELECT DNI, Nombre, Apellidos, Fecha_Nacimiento, Email, Categoria FROM Profesores WHERE OID_Dep IN (SELECT OID_DEP FROM Departamentos WHERE Nombre =
v_NombreDepartamento);
    v_Profesores C%ROWTYPE;
BEGIN
    OPEN C;
    FETCH C INTO v_Profesores;
    DBMS_OUTPUT.PUT_LINE( RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha de nacimiento:', 25) || RPAD('Email:', 25) || RPAD('Categoria:', 25));
    DBMS_OUTPUT.PUT_LINE( LPAD( ' ', 140, ' '));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE( RPAD(v_Profesores.DNI, 25) || RPAD(v_Profesores.Nombre, 25) || RPAD(v_Profesores.Apellidos, 25) || RPAD(v_Profesores.Fecha_Nacimiento, 25) || RPAD(v_Profesores.Email, 25) || RPAD(v_Profesores.Categoría, 25));
        FETCH C INTO v_Profesores;
    END LOOP;
    CLOSE C;
END;
/
```

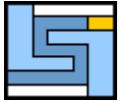


```
--RF-018
CREATE OR REPLACE PROCEDURE PR_Asignaturas_Departamento (v_NombreDepartamento IN Departamentos.Nombre%TYPE)
IS
  CURSOR C IS
    SELECT Código_Asig, Nombre, Acronimo, Creditos, Periodo, Curso, Tipo FROM Asignaturas WHERE OID_Dep IN (SELECT OID_DEP FROM Departamentos WHERE Nombre =
v_NombreDepartamento);
    v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Código:', 25) || RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Periodo:', 25) || RPAD('Curso:', 25) ||
RPAD('Tipo:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 240, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Código_Asig, 25) || RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acrónimo, 25) || RPAD(v_Asignaturas.Creditos, 25) ||
RPAD(v_Asignaturas.Periodo, 25) || RPAD(v_Asignaturas.Curso, 25) || RPAD(v_Asignaturas.Tipo, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/

--RF-019
CREATE OR REPLACE FUNCTION FN_Grupo_Mas_Alumnos
RETURN VARCHAR2
IS
  v_nombreGrupo Grupos.Nombre%TYPE;
BEGIN
  SELECT Nombre INTO v_nombreGrupo FROM Grupos WHERE OID_Grup IN (SELECT OID_Grup FROM AlumnosPertenecenAGrupos GROUP BY OID_Grup HAVING COUNT(DNI) = (SELECT
MAX(COUNT(DNI)) FROM AlumnosPertenecenAGrupos GROUP BY OID_Grup));
  RETURN (v_nombreGrupo);
END;
/

--RF-020
CREATE OR REPLACE PROCEDURE PR_Eliminar_Asignatura_Dep (v_codigo_asig IN Asignaturas.Código_Asig%TYPE)
IS
BEGIN
  UPDATE Asignaturas SET OID_Dep = NULL WHERE Código_Asig = v_codigo_asig;
  COMMIT;
END;
/

--RF-021
CREATE OR REPLACE PROCEDURE PR_Asociar_Profesor_Dep (v_DNI IN Profesores.DNI%TYPE, v_NombreDepartamento IN Departamentos.Nombre%TYPE)
IS
  v_OID_Dep Profesores.OID_Dep%TYPE;
BEGIN
  SELECT OID_Dep INTO v_OID_Dep FROM Departamentos WHERE Nombre = v_NombreDepartamento;
  UPDATE Profesores SET OID_Dep = v_OID_Dep WHERE DNI = v_DNI;
  COMMIT;
END;
/
```



```
--RF-022
CREATE OR REPLACE PROCEDURE PR_Eliminar_Profesor_Dep (v_DNI IN Profesores.DNI%TYPE)
IS
BEGIN
    UPDATE Profesores SET OID_Dep = NULL WHERE DNI = v_DNI;
    COMMIT;
END;
/

--RF-023
CREATE OR REPLACE PROCEDURE PR_Prof_Imparte_Asig (v_codigo_asig IN Asignaturas.Codigo_Asig%TYPE)
IS
    v_ImparteProfesor NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_ImparteProfesor FROM ProfesoresImpartenAsignaturas WHERE Codigo_Asig = v_codigo_asig;
    IF v_ImparteProfesor >= 1 THEN
        DBMS_OUTPUT.PUT_LINE('La asignatura es impartida al menos por un profesor.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('La asignatura no es impartida por ningn profesor.');
    END IF;
END;
/

--RF-024
CREATE OR REPLACE PROCEDURE PR_Prof_Imparte_Asigs_Dep (v_NombreDepartamento IN Departamentos.Nombre%TYPE)
IS
    CURSOR C IS
        SELECT Codigo_Asig FROM Asignaturas WHERE OID_Dep IN (SELECT OID_Dep FROM Departamentos WHERE Nombre = v_NombreDepartamento);
    v_Asignaturas C%ROWTYPE;
    v_ImparteProfesor NUMBER;
    v_TodasAsignadas BOOLEAN := TRUE;
BEGIN
    OPEN C;
    FETCH C INTO v_Asignaturas;
    WHILE C%FOUND LOOP
        SELECT COUNT(*) INTO v_ImparteProfesor FROM ProfesoresImpartenAsignaturas WHERE Codigo_Asig = v_Asignaturas.Codigo_Asig;
        IF v_ImparteProfesor = 0 THEN
            v_TodasAsignadas := FALSE;
        END IF;
        FETCH C INTO v_Asignaturas;
    END LOOP;
    CLOSE C;
    IF v_TodasAsignadas = TRUE THEN
        DBMS_OUTPUT.PUT_LINE('Todas las asignaturas del departamento tienen al menos un profesor asignado.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No todas las asignaturas del departamento tienen al menos un profesor asignado.');
    END IF;
END;
/
```



```
--RF-025
CREATE OR REPLACE PROCEDURE PR_NumProf_AgrupadosPor_Asig
IS
  CURSOR C IS
    SELECT Nombre, COUNT(DNI) AS NumProfesores FROM ProfesoresImpartenAsignaturas NATURAL JOIN Asignaturas GROUP BY Nombre;
    v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Asignatura:', 100) || RPAD('Número de profesores:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.NumProfesores, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/

--RF-026
CREATE OR REPLACE PROCEDURE PR_NumTut_AgrupadosPor_Prof
IS
  CURSOR C IS
    SELECT DNI_Prof, COUNT(OID_T) AS NumTutorias FROM Tutorias GROUP BY DNI_Prof;
    v_Profesores C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Profesores;
  DBMS_OUTPUT.PUT_LINE(RPAD('DNI del profesor:', 25) || RPAD('Número de tutorias:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 47, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Profesores.DNI_Prof, 25) || RPAD(v_Profesores.NumTutorias, 25));
    FETCH C INTO v_Profesores;
  END LOOP;
  CLOSE C;
END;
/

--RF-027
CREATE OR REPLACE PROCEDURE PR_Profesores_Despacho (v_NombreDespacho IN Despachos.Nombre%TYPE)
IS
  CURSOR C IS
    SELECT DNI, Nombre, Apellidos, Fecha_Nacimiento, Email, Categoria FROM Profesores WHERE OID_D IN (SELECT OID_D FROM Despachos WHERE Nombre = v_NombreDespacho);
    v_Profesores C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Profesores;
  DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha de nacimiento:', 25) || RPAD('Email:', 25) || RPAD('Categoria:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 140, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Profesores.DNI, 25) || RPAD(v_Profesores.Nombre, 25) || RPAD(v_Profesores.Apellidos, 25) || RPAD(v_Profesores.Fecha_Nacimiento, 25) || RPAD(v_Profesores.Email, 25) || RPAD(v_Profesores.Categoria, 25));
    FETCH C INTO v_Profesores;
  END LOOP;
  CLOSE C;
END;
/
```



```
--RF-028
CREATE OR REPLACE PROCEDURE PR_Notas_Expediente (v_codigo_exp IN Expedientes.Codigo_exp%TYPE)
IS
  CURSOR C IS
    SELECT Valor, Calificacion, Convocatoria, Curso, Fecha FROM Notas WHERE Codigo_exp = v_codigo_exp;
  v_Notas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Notas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Valor:', 25) || RPAD('Calificación', 25) || RPAD('Convocatoria:', 25) || RPAD('Curso:', 25) || RPAD('Fecha:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 140, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Notas.Valor, 25) || RPAD(v_Notas.Calificacion, 25) || RPAD(v_Notas.Convocatoria, 25) || RPAD(v_Notas.Curso, 25) || RPAD(v_Notas.Fecha, 25));
    FETCH C INTO v_Notas;
  END LOOP;
  CLOSE C;
END;
/

--RF-029
CREATE OR REPLACE FUNCTION FN_NotaMedia_Expediente (v_codigo_exp IN Expedientes.Codigo_exp%TYPE)
RETURN NUMBER
IS
  v_notaMedia Notas.Valor%TYPE;
BEGIN
  SELECT AVG(Valor) INTO v_notaMedia FROM Notas WHERE Codigo_exp = v_codigo_exp;
  RETURN (v_notaMedia);
END;
/

--RF-030
CREATE OR REPLACE PROCEDURE PR_Evaluar_Examen (v_Valor IN Notas.Valor%TYPE, v_Calificacion IN Notas.Calificacion%TYPE, v_Convocatoria IN Notas.Convocatoria%TYPE, v_Curso IN Notas.Curso%TYPE, v_Fecha IN Notas.Fecha%TYPE, v_codigo_exp IN Notas.Codigo_exp%TYPE)
IS
BEGIN
  INSERT INTO Notas (Valor, Calificacion, Convocatoria, Curso, Fecha, Codigo_Exp) VALUES (v_Valor, v_Calificacion, v_Convocatoria, v_Curso, v_Fecha, v_codigo_exp);
  COMMIT;
END;
/

--RF-031
CREATE OR REPLACE FUNCTION FN_MejorNota_Expediente (v_codigo_exp IN Expedientes.Codigo_exp%TYPE)
RETURN NUMBER
IS
  v_mejorNota Notas.Valor%TYPE;
BEGIN
  SELECT MAX(Valor) INTO v_mejorNota FROM Notas WHERE Codigo_exp = v_codigo_exp;
  RETURN (v_mejorNota);
END;
/
```



```
--RF-032
CREATE OR REPLACE PROCEDURE PR_AsignaturasOblig_Grado (v_Grado IN Grados.Nombre%TYPE)
IS
  CURSOR C IS
    SELECT Nombre, Acronimo, Creditos, Periodo, Curso, Tipo FROM Asignaturas WHERE Código_Asig IN (Select Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN
(SELECT OID_G FROM Grados WHERE Nombre = v_Grado)) AND Tipo = 'Obligatorio' OR Tipo = 'Troncal';
    v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Periodo:', 25) || RPAD('Curso:', 25) || RPAD('Tipo:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 215, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acronimo, 25) || RPAD(v_Asignaturas.Creditos, 25) || RPAD(v_Asignaturas.Periodo, 25) ||
RPAD(v_Asignaturas.Curso, 25) || RPAD(v_Asignaturas.Tipo, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/

--RF-033
CREATE OR REPLACE PROCEDURE PR_AsignaturasOpt_Grado (v_Grado IN Grados.Nombre%TYPE)
IS
  CURSOR C IS
    SELECT Nombre, Acronimo, Creditos, Periodo, Curso, Tipo FROM Asignaturas WHERE Código_Asig IN (Select Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN
(SELECT OID_G FROM Grados WHERE Nombre = v_Grado)) AND Tipo = 'Optativo';
    v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Periodo:', 25) || RPAD('Curso:', 25) || RPAD('Tipo:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 215, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acronimo, 25) || RPAD(v_Asignaturas.Creditos, 25) || RPAD(v_Asignaturas.Periodo, 25) ||
RPAD(v_Asignaturas.Curso, 25) || RPAD(v_Asignaturas.Tipo, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/

--RF-034
CREATE OR REPLACE FUNCTION FN_CreditosOptativos_Grado (v_Grado IN Grados.Nombre%TYPE)
RETURN NUMBER
IS
  v_numCreditos Asignaturas.Creditos%TYPE;
BEGIN
  SELECT SUM(Creditos) INTO v_numCreditos FROM Asignaturas WHERE Código_Asig IN (SELECT Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT OID_G FROM
Grados WHERE Nombre = v_Grado)) AND Tipo = 'Optativo';
  RETURN (v_numCreditos);
END;
/
```



```
--RF-035
CREATE OR REPLACE FUNCTION FN_CreditosObligatorios_Grado (v_Grado IN Grados.Nombre%TYPE)
RETURN NUMBER
IS
    v_numCreditos Asignaturas.Creditos%TYPE;
BEGIN
    SELECT SUM(Creditos) INTO v_numCreditos FROM Asignaturas WHERE Código_Asig IN (SELECT Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT OID_G FROM
Grados WHERE Nombre = v_Grado)) AND Tipo = 'Obligatorio';
    RETURN (v_numCreditos);
END;
/

--RF-036
CREATE OR REPLACE FUNCTION FN_CreditosTroncales_Grado (v_Grado IN Grados.Nombre%TYPE)
RETURN NUMBER
IS
    v_numCreditos Asignaturas.Creditos%TYPE;
BEGIN
    SELECT SUM(Creditos) INTO v_numCreditos FROM Asignaturas WHERE Código_Asig IN (Select Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT OID_G FROM
Grados WHERE Nombre = v_Grado)) AND Tipo = 'Troncal';
    RETURN (v_numCreditos);
END;
/

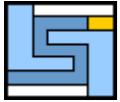
--RF-037
CREATE OR REPLACE FUNCTION FN_CreditosTotales_Grado (v_Grado IN Grados.Nombre%TYPE)
RETURN NUMBER
IS
    v_numCreditos Asignaturas.Creditos%TYPE;
BEGIN
    SELECT SUM(Creditos) INTO v_numCreditos FROM Asignaturas WHERE Código_Asig IN (Select Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT OID_G FROM
Grados WHERE Nombre = v_Grado));
    RETURN (v_numCreditos);
END;
/

--RF-038
CREATE OR REPLACE PROCEDURE PR_Departamentos_Grado (v_Grado IN Grados.Nombre%TYPE)
IS
    CURSOR C IS
        SELECT Nombre FROM Departamentos WHERE OID_Dep IN (SELECT OID_Dep FROM Asignaturas NATURAL JOIN AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT OID_G FROM Grados
WHERE Nombre = v_Grado));
    v_Departamentos C%ROWTYPE;
BEGIN
    OPEN C;
    FETCH C INTO v_Departamentos;
    DBMS_OUTPUT.PUT_LINE(RPAD('Departamento:', 75));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 75, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Departamentos.Nombre, 75));
        FETCH C INTO v_Departamentos;
    END LOOP;
    CLOSE C;
END;
/
```



```
--RF-039
CREATE OR REPLACE PROCEDURE PR_Profesores_Grado (v_Grado IN Grados.Nombre%TYPE)
IS
  CURSOR C IS
    SELECT DISTINCT DNI, Nombre, Apellidos, Fecha_Nacimiento, Email, Categoria FROM Profesores NATURAL JOIN ProfesoresImpartenAsignaturas WHERE Código_Asig IN (SELECT
Código_Asig FROM Asignaturas NATURAL JOIN AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT OID_G FROM Grados WHERE Nombre = v_Grado));
  v_Profesores C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Profesores;
  DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha de nacimiento:', 25) || RPAD('Email:', 25) || RPAD('Categoria:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 140, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Profesores.DNI, 25) || RPAD(v_Profesores.Nombre, 25) || RPAD(v_Profesores.Apellidos, 25) || RPAD(v_Profesores.Fecha_Nacimiento, 25) ||
RPAD(v_Profesores.Email, 25) || RPAD(v_Profesores.Categoría, 25));
    FETCH C INTO v_Profesores;
  END LOOP;
  CLOSE C;
END;
/

--RF-040
CREATE OR REPLACE PROCEDURE PR_Asignaturas_Grado_Curso (v_Grado IN Grados.Nombre%TYPE, v_Curso IN Asignaturas.Curso%TYPE)
IS
  CURSOR C IS
    SELECT Nombre, Acronimo, Creditos, Periodo, Tipo FROM Asignaturas WHERE Código_Asig IN (SELECT Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT
OID_G FROM Grados WHERE Nombre = v_Grado)) AND Curso = v_Curso;
  v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Periodo:', 25) || RPAD('Tipo:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 200, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acrónimo, 25) || RPAD(v_Asignaturas.Creditos, 25) || RPAD(v_Asignaturas.Periodo, 25) ||
RPAD(v_Asignaturas.Tipo, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/
```



```
--RF-041
CREATE OR REPLACE PROCEDURE PR_AsignaturasAnuales_Grado (v_Grado IN Grados.Nombre%TYPE)
IS
  CURSOR C IS
    SELECT Nombre, Acronimo, Creditos, Curso, Tipo FROM Asignaturas WHERE Código_Asig IN (SELECT Código_Asig FROM AsignaturasPertenecenAGrados WHERE OID_G IN (SELECT
OID_G FROM Grados WHERE Nombre = v_Grado)) AND Periodo = 'Anual';
    v_Asignaturas C%ROWTYPE;
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Curso:', 25) || RPAD('Tipo:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 200, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acrónimo, 25) || RPAD(v_Asignaturas.Creditos, 25) || RPAD(v_Asignaturas.Curso, 25) ||
RPAD(v_Asignaturas.Tipo, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END;
/

--RF-042
CREATE OR REPLACE PROCEDURE PR_Tutorias_Profesor (v_DNI IN Profesores.DNI%TYPE)
IS
  CURSOR C IS
    SELECT Fecha, Hora_comienzo, Duracion, DNI_Alum FROM Tutorias WHERE DNI_Prof = v_DNI;
    v_Tutorias C%ROWTYPE;

BEGIN
  OPEN C;
  FETCH C INTO v_Tutorias;
  DBMS_OUTPUT.PUT_LINE(RPAD('Fecha', 25) || RPAD('Hora de comienzo', 25) || RPAD('Duración', 25) || RPAD('DNI del Alumno que atiende', 27));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 105, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Tutorias.Fecha, 25) || RPAD(v_Tutorias.Hora_comienzo, 25) || RPAD(v_Tutorias.Duracion, 25) || RPAD(v_Tutorias.Dni_Alum, 27));
    FETCH C INTO v_Tutorias;
  END LOOP;
  CLOSE C;
END;
/

--RF-043
CREATE OR REPLACE FUNCTION FN_Departamento_Profesor (v_DNI IN Profesores.DNI%TYPE)
RETURN VARCHAR2
IS
  v_Departamento Departamentos.Nombre%TYPE;
BEGIN
  SELECT Nombre INTO v_Departamento FROM Departamentos WHERE OID_DEP IN (SELECT OID_DEP FROM Profesores WHERE DNI = v_DNI);
  RETURN (v_Departamento);
END;
/
```



```
--RF-044
CREATE OR REPLACE FUNCTION FN_DedicacionTotal_Profesor (v_DNI IN Profesores.DNI%TYPE)
RETURN NUMBER
IS
    v_DedicacionTotal ProfesoresImpartenAsignaturas.Dedicacion%TYPE;
BEGIN
    SELECT SUM(Dedicacion) INTO v_DedicacionTotal FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI;
    RETURN (v_DedicacionTotal);
END;
/

--RF-045
CREATE OR REPLACE PROCEDURE PR_Asignaturas_Profesor (v_DNI IN Profesores.DNI%TYPE)
IS
    CURSOR C IS
        SELECT Nombre, Acronimo, Creditos, Periodo, Tipo FROM Asignaturas NATURAL JOIN ProfesoresImpartenAsignaturas WHERE DNI = v_DNI;
        v_Asignaturas C%ROWTYPE;
BEGIN
    OPEN C;
    FETCH C INTO v_Asignaturas;
    DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Periodo:', 25) || RPAD('Tipo:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 200, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acronimo, 25) || RPAD(v_Asignaturas.Creditos, 25) || RPAD(v_Asignaturas.Periodo, 25) ||
        RPAD(v_Asignaturas.Tipo, 25));
        FETCH C INTO v_Asignaturas;
    END LOOP;
    CLOSE C;
END;
/

--RF-046
CREATE OR REPLACE PROCEDURE PR_Dedicacion_Profesor (v_DNI IN Profesores.DNI%TYPE)
IS
    CURSOR C IS
        SELECT Nombre, Dedicacion FROM ProfesoresImpartenAsignaturas NATURAL JOIN Asignaturas WHERE DNI = v_DNI;
        v_Asignaturas C%ROWTYPE;
BEGIN
    OPEN C;
    FETCH C INTO v_Asignaturas;
    DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Dedicación', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 110, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Dedicacion, 25));
        FETCH C INTO v_Asignaturas;
    END LOOP;
    CLOSE C;
END;
/

--RF-047
CREATE OR REPLACE PROCEDURE PR_Categoría_Profesor (v_DNI IN Profesores.DNI%TYPE, v_Categoría IN Profesores.Categoría%TYPE)
IS
BEGIN
    UPDATE Profesores SET Categoría = v_Categoría WHERE DNI = v_DNI;
    COMMIT;
END;
/
```



```
--RF-048
CREATE OR REPLACE PROCEDURE PR_Adicion_Tutoria (v_Fecha IN Tutorias.Fecha%TYPE, v_Hora_Comienzo IN Tutorias.Hora_Comienzo%TYPE, v_Duracion IN Tutorias.Duracion%TYPE,
v_DNI_Prof IN Profesores.DNI%TYPE, v_DNI_Alum IN Alumnos.DNI%TYPE)
IS
BEGIN
    INSERT INTO Tutorias (Fecha, Hora_Comienzo, Duracion, DNI_Alum, DNI_Prof) VALUES (v_Fecha, v_Hora_Comienzo, v_Duracion, v_DNI_Alum, v_DNI_Prof);
    COMMIT;
END;
/

--RF-049
CREATE OR REPLACE PROCEDURE PR_Eliminar_Tutoria (v_OID_T IN Tutorias.OID_T%TYPE)
IS
BEGIN
    DELETE FROM Tutorias WHERE OID_T = v_OID_T;
    COMMIT;
END;
/

--RF-050
CREATE OR REPLACE PROCEDURE PR_Asignar_Profesor_Asignatura (v_DNI IN Profesores.DNI%TYPE, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, v_Dedicacion IN
ProfesoresImpartenAsignaturas.Dedicacion%TYPE)
IS
BEGIN
    INSERT INTO ProfesoresImpartenAsignaturas (DNI, Codigo_Asig, Dedicacion) VALUES (v_DNI, v_Codigo_Asig, v_Dedicacion);
    COMMIT;
END;
/

--RF-051
CREATE OR REPLACE PROCEDURE PR_Eliminar_Profesor_Asig (v_DNI IN Profesores.DNI%TYPE, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE)
IS
BEGIN
    DELETE FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI AND Codigo_Asig = v_Codigo_Asig;
    COMMIT;
END;
/

--RF-052
CREATE OR REPLACE PROCEDURE PR_Asignaturas_Aula (v_Aula IN Aulas.Nombre%TYPE, v_Fecha IN AsignaturasImpartidasEnAulas.Fecha%TYPE)
IS
    CURSOR C IS
        SELECT Nombre, Acronimo, Creditos, Periodo, Tipo FROM Asignaturas NATURAL JOIN AsignaturasImpartidasEnAulas WHERE OID_A IN (SELECT OID_A FROM Aulas WHERE Nombre =
v_Aula) AND TRUNC(Fecha) = v_Fecha;
        v_Asignaturas C%ROWTYPE;
BEGIN
    OPEN C;
    FETCH C INTO v_Asignaturas;
    DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 100) || RPAD('Acrónimo:', 25) || RPAD('Créditos:', 25) || RPAD('Periodo:', 25) || RPAD('Tipo:', 25));
    DBMS_OUTPUT.PUT_LINE(IPAD('-', 195, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Nombre, 100) || RPAD(v_Asignaturas.Acronimo, 25) || RPAD(v_Asignaturas.Creditos, 25) || RPAD(v_Asignaturas.Periodo, 25) ||
RPAD(v_Asignaturas.Tipo, 25));
        FETCH C INTO v_Asignaturas;
    END LOOP;
    CLOSE C;
END;
/
```



Triggers asociados a las reglas de negocio

```
--Triggers asociados a las reglas de negocio
--RN-001
CREATE OR REPLACE TRIGGER TR_Email_Alum
AFTER INSERT OR UPDATE OF Email ON Alumnos
FOR EACH ROW
DECLARE
    v_Email VARCHAR2(40) := :NEW.Email;
BEGIN
    IF v_Email NOT LIKE '%@alum.es' THEN
        RAISE_APPLICATION_ERROR(-20001, 'Un alumno matriculado debe tener un email acabado en @alum.es.');
    END IF;
END;
/

--RN-002
CREATE OR REPLACE TRIGGER TR_Creditos_Asig
AFTER INSERT OR UPDATE OF Creditos ON Asignaturas
FOR EACH ROW
DECLARE
    v_Creditos NUMBER := :NEW.Creditos;
BEGIN
    IF v_Creditos <= 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Los créditos de una asignatura deben ser un Número positivo distinto de 0.');
    END IF;
END;
/
```



```
--RN-003
CREATE OR REPLACE TRIGGER TR_Duracion_Beca
AFTER INSERT OR UPDATE OF Fecha_Comienzo, Fecha_Fin ON Becarios
FOR EACH ROW
DECLARE
    v_Fecha_Comienzo DATE := :NEW.Fecha_Comienzo;
    v_Fecha_Fin DATE := :NEW.Fecha_Fin;
    v_Duracion NUMBER;
BEGIN
    SELECT MONTHS_BETWEEN(TO_DATE(v_Fecha_Fin, 'DD/MM/YYYY'), TO_DATE(v_Fecha_Comienzo, 'DD/MM/YYYY')) INTO
    v_Duracion FROM DUAL;
    IF v_Duracion < 1 THEN
        RAISE_APPLICATION_ERROR(-20003, 'La duración de una beca debe ser como mínimo de 1 mes.');
    END IF;
END;
/
```

```
--RN-004
CREATE OR REPLACE TRIGGER TR_Minimo_Cuantia_Fija
AFTER INSERT OR UPDATE OF Cuantia_Fija ON Becas
FOR EACH ROW
DECLARE
    v_Cuantia_Fija NUMBER := :NEW.Cuantia_Fija;
BEGIN
    IF v_Cuantia_Fija < 1500 THEN
        RAISE_APPLICATION_ERROR(-20004, 'La cuantía total fija de una beca debe ser de al menos 1500.');
    END IF;
END;
/
```



```
--RN-005
CREATE OR REPLACE TRIGGER TR_Capacidad_Despachos
AFTER INSERT OR UPDATE OF Capacidad ON Despachos
FOR EACH ROW
DECLARE
    v_Capacidad NUMBER := :NEW.Capacidad;
BEGIN
    IF v_Capacidad > 3 THEN
        RAISE_APPLICATION_ERROR(-20005, 'La capacidad máxima de profesores en un despacho debe ser 3.');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER TR_Capacidad_Despachos_Prof
AFTER INSERT OR UPDATE OF OID_D ON Profesores
FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    v_OID_D_Nuevo NUMBER := :NEW.OID_D;
    v_OID_D_Antiguo NUMBER := :OLD.OID_D;
    v_NumProfesores NUMBER;
    v_Capacidad NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_NumProfesores FROM Profesores WHERE OID_D = v_OID_D_Nuevo;
    SELECT Capacidad INTO v_Capacidad FROM Despachos WHERE OID_D = v_OID_D_Nuevo;
    IF UPDATING ('OID_D') AND v_OID_D_Antiguo = v_OID_D_Nuevo THEN
        v_NumProfesores := v_NumProfesores;
    ELSE
        v_NumProfesores := v_NumProfesores + 1;
    END IF;
    IF v_NumProfesores > v_Capacidad THEN
        RAISE_APPLICATION_ERROR(-20006, 'La capacidad máxima de profesores de este despacho es de ' || v_Capacidad);
    END IF;
END;
/
```



```
--RN-006
CREATE OR REPLACE TRIGGER TR_Capacidad_Aulas
AFTER INSERT OR UPDATE OF Capacidad ON Aulas
FOR EACH ROW
DECLARE
    v_Capacidad NUMBER := :NEW.Capacidad;
    v_Tipo VARCHAR2(20) := :NEW.TIPO;
BEGIN
    IF v_Tipo LIKE 'Teoría' AND v_Capacidad > 70 THEN
        RAISE_APPLICATION_ERROR(-20007, 'La capacidad máxima de alumnos en un aula de tipo teoría debe ser 70.');
    END IF;
    IF v_Tipo LIKE 'Laboratorio' AND v_Capacidad > 30 THEN
        RAISE_APPLICATION_ERROR(-20008, 'La capacidad máxima de alumnos en un aula de tipo laboratorio debe ser 30.');
    END IF;
    IF v_Tipo LIKE 'Examen' AND v_Capacidad > 200 THEN
        RAISE_APPLICATION_ERROR(-20009, 'La capacidad máxima de alumnos en un aula de tipo examen debe ser 200.');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER TR_Capacidad_Aulas_Alum
BEFORE INSERT ON GruposAsistenAClaseEnAulas
FOR EACH ROW
DECLARE
    v_OID_Grup NUMBER := :NEW.OID_Grup;
    v_OID_A NUMBER := :NEW.OID_A;
    v_NumAlumnos NUMBER;
    v_Capacidad NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_NumAlumnos FROM AlumnosPertenecenAGrupos WHERE OID_Grup = v_OID_Grup;
    SELECT Capacidad INTO v_Capacidad FROM Aulas WHERE OID_A = v_OID_A;
    IF v_NumAlumnos > v_Capacidad THEN
        RAISE_APPLICATION_ERROR(-20010, 'La capacidad máxima de alumnos de este aula es de ' || v_Capacidad);
    END IF;
END;
/
```



```
--RN-007
CREATE OR REPLACE TRIGGER TR_Calificacion_Expediente
BEFORE INSERT ON AsignaturasPuntuadasConNotas
FOR EACH ROW
DECLARE
    CURSOR C IS
        SELECTCodigo_Asig, OID_N FROM AsignaturasPuntuadasConNotas;
    v_CodAsignaturaNueva NUMBER := :NEW.Codigo_Asig;
    v_NotaNueva NUMBER := :NEW.OID_N;
    v_Convocatoria VARCHAR(20);
    v_Curso NUMBER;
    v_Expediente NUMBER;
    v_ConvocatoriaAux VARCHAR(20);
    v_CursoAux NUMBER;
    v_ExpedienteAux NUMBER;
BEGIN
    SELECT Convocatoria INTO v_Convocatoria FROM Notas WHERE OID_N = v_NotaNueva;
    SELECT Curso INTO v_Curso FROM Notas WHERE OID_N = v_NotaNueva;
    SELECT Codigo_Exp INTO v_Expediente FROM Notas WHERE OID_N = v_NotaNueva;
    FOR Evaluacion IN C LOOP
        IF Evaluacion.Codigo_Asig = v_CodAsignaturaNueva THEN
            SELECT Convocatoria INTO v_ConvocatoriaAux FROM Notas WHERE OID_N = Evaluacion.OID_N;
            SELECT Curso INTO v_CursoAux FROM Notas WHERE OID_N = Evaluacion.OID_N;
            SELECT Codigo_Exp INTO v_ExpedienteAux FROM Notas WHERE OID_N = Evaluacion.OID_N;
            IF v_Convocatoria = v_ConvocatoriaAux AND v_Curso = v_CursoAux AND v_Expediente = v_ExpedienteAux THEN
                RAISE_APPLICATION_ERROR(-20011, 'El expediente de un alumno no puede contener calificaciones para
más de 1 convocatoria de una misma asignatura y curso.');
            END IF;
        END IF;
    END LOOP;
END;
/
```



```
--RN-008
CREATE OR REPLACE TRIGGER TR_AsigOpt_MismoNum_Creditos
AFTER INSERT OR UPDATE OF Creditos, Tipo ON Asignaturas
FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    CURSOR C IS
        SELECT Creditos FROM Asignaturas WHERE Tipo = 'Optativo';
    v_Creditos NUMBER := :NEW.Creditos;
    v_Tipo VARCHAR2(20) := :NEW.Tipo;
BEGIN
    FOR Asignatura IN C LOOP
        IF Asignatura.Creditos != v_Creditos AND v_Tipo LIKE 'Optativo' THEN
            RAISE_APPLICATION_ERROR(-20012, 'Todas las asignaturas optativas de un grado deben tener el mismo número
de créditos.');
        END IF;
    END LOOP;
END;
/

--RN-009
CREATE OR REPLACE TRIGGER TR_AsigOpt_Trimestrales
AFTER INSERT OR UPDATE OF Periodo, Tipo ON Asignaturas
FOR EACH ROW
DECLARE
    v_Periodo VARCHAR2(20) := :NEW.Periodo;
    v_Tipo VARCHAR2(20) := :NEW.Tipo;
BEGIN
    IF v_Tipo = 'Optativo' AND v_Periodo = 'Anual' THEN
        RAISE_APPLICATION_ERROR(-20013, 'Todas las asignaturas optativas de un grado deben ser cuatrimestrales.');
    END IF;
END;
/
```



```
--RN-010
CREATE OR REPLACE TRIGGER TR_CantCredOpt_A_Cursar
BEFORE INSERT ON MatriculasContienenAsignaturas
FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    CURSOR C IS
        SELECT OID_G FROM MatriculasPertenecenAGrados WHERE Código_Matric = :NEW.Código_Matric;
    v_Código_Asig NUMBER := :NEW.Código_Asig;
    v_Código_Matric NUMBER := :NEW.Código_Matric;
    v_CreditosOptMatriculados NUMBER;
    v_CreditosNuevaAsig NUMBER := 0;
    v_TotalCreditos NUMBER;
    v_Tipo VARCHAR2(20);
BEGIN
    FOR Grado IN C LOOP
        SELECT SUM(Creditos) INTO v_CreditosOptMatriculados FROM Asignaturas WHERE Tipo = 'Optativo' AND Código_Asig
        IN (SELECT DISTINCT Código_Asig FROM MatriculasContienenAsignaturas NATURAL JOIN Matriculas NATURAL JOIN
        MatriculasPertenecenAGrados WHERE OID_G = Grado.OID_G AND Código_Matric = v_Código_Matric);
        SELECT Tipo INTO v_Tipo FROM Asignaturas WHERE Código_Asig = v_Código_Asig;
        IF v_Tipo LIKE 'Optativo' THEN
            SELECT Creditos INTO v_CreditosNuevaAsig FROM Asignaturas WHERE Código_Asig = v_Código_Asig;
            v_TotalCreditos := v_CreditosNuevaAsig + v_CreditosOptMatriculados;
            IF v_TotalCreditos > 30 THEN
                RAISE_APPLICATION_ERROR(-20014, 'El número mínimo de créditos de asignaturas optativas que debe
                cursar un alumno debe estar comprendido entre 0 y 30.');
            END IF;
        END IF;
    END LOOP;
END;
/
```



```
--RN-011
CREATE OR REPLACE TRIGGER TR_Valor_Nota
AFTER INSERT OR UPDATE OF Valor ON Notas
FOR EACH ROW
DECLARE
    v_Valor NUMBER := :NEW.Valor;
BEGIN
    IF v_Valor < 0 OR v_Valor > 10 THEN
        RAISE_APPLICATION_ERROR(-20015, 'El valor de una nota a la hora de evaluar un alumno debe estar comprendida
entre 0 y 10.');
    END IF;
END;
/

--RN-012
CREATE OR REPLACE TRIGGER TR_Calificacion_Nota
AFTER INSERT OR UPDATE OF Valor, Calificacion ON Notas
FOR EACH ROW
DECLARE
    v_Valor NUMBER := :NEW.Valor;
    v_Calificacion VARCHAR2(20) := :NEW.Calificacion;
    v_EsCorrecto BOOLEAN := TRUE;
BEGIN
    IF v_Valor < 5 AND v_Calificacion != 'Suspens' THEN
        v_EsCorrecto := FALSE;
    END IF;
    IF v_Valor >= 5 AND v_Valor < 7 AND v_Calificacion != 'Aprobado' THEN
        v_EsCorrecto := FALSE;
    END IF;
    IF v_Valor >= 7 AND v_Valor < 9 AND v_Calificacion != 'Notable' THEN
        v_EsCorrecto := FALSE;
    END IF;
    IF v_Valor >= 9 AND v_Valor < 10 AND v_Calificacion != 'Sobresaliente' THEN
        v_EsCorrecto := FALSE;
    END IF;
    IF v_Valor = 10 AND v_Calificacion != 'Matrícula de honor' THEN
        v_EsCorrecto := FALSE;
    END IF;
    IF v_EsCorrecto = FALSE THEN
        RAISE_APPLICATION_ERROR(-20016, 'La calificación de una evaluación debe tener su correspondiente valor de
nota.');
    END IF;
END;
```



```
END IF;
END ;
/

--RN-013
CREATE OR REPLACE TRIGGER TR_AsigImpartidas_unico_Dept
AFTER INSERT OR UPDATE OF DNI, Código_Asig ON ProfesoresImpartenAsignaturas
FOR EACH ROW
DECLARE
    v_DNI VARCHAR2(10) := :NEW.DNI;
    v_Código_Asig NUMBER := :NEW.Código_Asig;
    v_OID_Dep_Prof NUMBER;
    v_OID_Dep_Asig NUMBER;
BEGIN
    SELECT OID_Dep INTO v_OID_Dep_Prof FROM Profesores WHERE DNI = v_DNI;
    SELECT OID_Dep INTO v_OID_Dep_Asig FROM Asignaturas WHERE Código_Asig = v_Código_Asig;
    IF v_OID_Dep_Prof != v_OID_Dep_Asig THEN
        RAISE_APPLICATION_ERROR(-20017, 'Un profesor no puede impartir una asignatura de otro departamento que no corresponda al suyo.');
    END IF;
END ;
/

--RN-014
CREATE OR REPLACE TRIGGER TR_Dedicacion_Prof_Asig
AFTER INSERT OR UPDATE OF Código_Asig, Dedicación ON ProfesoresImpartenAsignaturas
FOR EACH ROW
DECLARE
    v_Código_Asig NUMBER := :NEW.Código_Asig;
    v_Dedicación NUMBER := :NEW.Dedicación;
    v_Creditos NUMBER;
BEGIN
    SELECT Creditos INTO v_Creditos FROM Asignaturas WHERE Código_Asig = v_Código_Asig;
    IF v_Dedicación <= 0 OR v_Dedicación > v_Creditos THEN
        RAISE_APPLICATION_ERROR(-20018, 'El número de créditos dedicados a una asignatura por un profesor debe ser mayor que 0 y menor o igual que el número de créditos de la asignatura.');
    END IF;
END ;
/
```



```
--RN-015
CREATE OR REPLACE TRIGGER TR_Dedicacion_Maxima_Prof
AFTER INSERT OR UPDATE OF DNI, Dedicacion ON ProfesoresImpartenAsignaturas
FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    v_DNI VARCHAR2(10) := :NEW.DNI;
    v_NuevaDedicacion NUMBER := :NEW.Dedicacion;
    v_AntiguaDedicacion NUMBER := :OLD.Dedicacion;
    v_DedicacionActual NUMBER;
    v_TotalCreditos NUMBER;
    v_CategoríaProfesor VARCHAR2(20);
BEGIN
    SELECT Categoría INTO v_CategoríaProfesor FROM Profesores WHERE DNI = v_DNI;
    SELECT SUM(Dedicacion) INTO v_DedicacionActual FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI;
    IF UPDATING ('Dedicacion') THEN
        v_TotalCreditos := v_DedicacionActual - v_AntiguaDedicacion + v_NuevaDedicacion;
    ELSE
        v_TotalCreditos := v_DedicacionActual + v_NuevaDedicacion;
    END IF;
    IF (v_TotalCreditos > 32) AND (v_CategoríaProfesor NOT LIKE 'Ayudante' OR v_CategoríaProfesor NOT LIKE 'Ayudante doctor') THEN
        RAISE_APPLICATION_ERROR(-20019, 'Un profesor no ayudante puede impartir un máximo de 32 créditos.');
    END IF;
    IF (v_TotalCreditos > 6) AND (v_CategoríaProfesor LIKE 'Ayudante' OR v_CategoríaProfesor LIKE 'Ayudante doctor')
    THEN
        RAISE_APPLICATION_ERROR(-20020, 'Un profesor ayudante puede impartir un máximo de 6 créditos.');
    END IF;
END;
/
```



```
--RN-016
CREATE OR REPLACE TRIGGER TR_Dias_Tutorias
AFTER INSERT OR UPDATE OF Fecha ON Tutorias
FOR EACH ROW
DECLARE
    v_Fecha DATE := :NEW.Fecha;
    v_Dia VARCHAR2(20);
BEGIN
    SELECT TO_CHAR(v_Fecha, 'D') INTO v_Dia FROM Dual;
    IF v_Dia IN (6, 7) THEN
        RAISE_APPLICATION_ERROR(-20021, 'Los días de la semana en donde se puede realizar una tutoría deben estar entre
lunes y viernes');
    END IF;
END;
/

--RN-017
CREATE OR REPLACE TRIGGER TR_Duracion_Tutorias
AFTER INSERT OR UPDATE OF Duracion ON Tutorias
FOR EACH ROW
DECLARE
    v_Duracion NUMBER := :NEW.Duracion;
BEGIN
    IF v_Duracion > 30 OR v_Duracion < 10 THEN
        RAISE_APPLICATION_ERROR(-20022, 'La duración de una tutoría debe ser de al menos 10 minutos y de máximo 30
minutos');
    END IF;
END;
/
```



Paquetes

```
--Función auxiliar
CREATE OR REPLACE FUNCTION ASSERT_EQUALS (v_Salida BOOLEAN, salidaEsperada BOOLEAN)
RETURN VARCHAR2
IS
BEGIN
    IF v_Salida = salidaEsperada THEN
        RETURN 'ÉXITO';
    ELSE
        RETURN 'FALLO';
    END IF;
END;
/

--Cabeceras de paquetes
--Tabla Alumnos
CREATE OR REPLACE PACKAGE PCK_Alumnos
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE, v_Nombre IN Alumnos.Nombre%TYPE, v_Apellidos IN Alumnos.Apellidos%TYPE, v_Fecha_Nacimiento IN
Alumnos.Fecha_Nacimiento%TYPE, v_Email IN Alumnos.Email%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE, v_Nombre IN Alumnos.Nombre%TYPE, v_Apellidos IN Alumnos.Apellidos%TYPE, v_Fecha_Nacimiento IN
Alumnos.Fecha_Nacimiento%TYPE, v_Email IN Alumnos.Email%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE, salidaEsperada BOOLEAN);
END;
/


--Tabla AlumnosPertenecenAGrupos
CREATE OR REPLACE PACKAGE PCK_AlumnosPertenecenAGrupos
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN AlumnosPertenecenAGrupos.DNI%TYPE, v_OID_Grup IN AlumnosPertenecenAGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN AlumnosPertenecenAGrupos.DNI%TYPE, v_OID_Grup IN AlumnosPertenecenAGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN);
END;
/


--Tabla Asignaturas
CREATE OR REPLACE PACKAGE PCK_Asignaturas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, v_Nombre IN Asignaturas.Nombre%TYPE, v_Acronimo IN Asignaturas.Acrônimo%TYPE,
v_Creditos IN Asignaturas.Creditos%TYPE, v_Periodo IN Asignaturas.Periodo%TYPE, v_Curso IN Asignaturas.Curso%TYPE, v_Tipo IN Asignaturas.Tipo%TYPE, v_OID_Dep IN
Asignaturas.OID_Dep%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, v_Nombre IN Asignaturas.Nombre%TYPE, v_Acronimo IN Asignaturas.Acrônimo%TYPE,
v_Creditos IN Asignaturas.Creditos%TYPE, v_Periodo IN Asignaturas.Periodo%TYPE, v_Curso IN Asignaturas.Curso%TYPE, v_Tipo IN Asignaturas.Tipo%TYPE, v_OID_Dep IN
Asignaturas.OID_Dep%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN);
END;
/
```



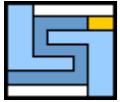
```
--Tabla AsignaturasPuntuadasConNotas
CREATE OR REPLACE PACKAGE PCK_AsigPuntuadasConNotas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPuntuadasConNotas.Codigo_Asig%TYPE, v_OID_N IN AsignaturasPuntuadasConNotas.OID_N%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPuntuadasConNotas.Codigo_Asig%TYPE, v_OID_N IN AsignaturasPuntuadasConNotas.OID_N%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla AsignaturasImpartidasEnAulas
CREATE OR REPLACE PACKAGE PCK_AsigImpartidasEnAulas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasImpartidasEnAulas.Codigo_Asig%TYPE, v_OID_A IN AsignaturasImpartidasEnAulas.OID_A%TYPE, v_Fecha IN AsignaturasImpartidasEnAulas.Fecha%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasImpartidasEnAulas.Codigo_Asig%TYPE, v_OID_A IN AsignaturasImpartidasEnAulas.OID_A%TYPE, v_Fecha IN AsignaturasImpartidasEnAulas.Fecha%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla AsignaturasPertenecenAGrados
CREATE OR REPLACE PACKAGE PCK_AsigPertenecenAGrados
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPertenecenAGrados.Codigo_Asig%TYPE, v_OID_G IN AsignaturasPertenecenAGrados.OID_G%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPertenecenAGrados.Codigo_Asig%TYPE, v_OID_G IN AsignaturasPertenecenAGrados.OID_G%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Aulas
CREATE OR REPLACE PACKAGE PCK_Aulas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Aulas.Nombre%TYPE, v_Capacidad IN Aulas.Capacidad%TYPE, v_Tipo IN Aulas.Tipo%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_A IN Aulas.OID_A%TYPE, v_Nombre IN Aulas.Nombre%TYPE, v_Capacidad IN Aulas.Capacidad%TYPE, v_Tipo IN Aulas.Tipo%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_A IN Aulas.OID_A%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Becarios
CREATE OR REPLACE PACKAGE PCK_Becarios
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Becarios.DNI%TYPE, v_Codigo_Beca IN Becarios.Codigo_Beca%TYPE, v_Fecha_Comienzo IN Becarios.Fecha_Comienzo%TYPE, v_Fecha_Fin IN Becarios.Fecha_Fin%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN Becarios.DNI%TYPE, v_Codigo_Beca IN Becarios.Codigo_Beca%TYPE, v_Fecha_Comienzo IN Becarios.Fecha_Comienzo%TYPE, v_Fecha_Fin IN Becarios.Fecha_Fin%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Becarios.DNI%TYPE, v_Codigo_Beca IN Becarios.Codigo_Beca%TYPE, salidaEsperada BOOLEAN);
END;
/
```



```
--Tabla Becas
CREATE OR REPLACE PACKAGE PCK_Becas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Beca IN Becas.Codigo_Beca%TYPE, v_Cuantia_Fija IN Becas.Cuantia_Fija%TYPE, v_Cuantia_Variable IN
Becas.Cuantia_Variable%TYPE, v_Tipo IN Becas.Tipo%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Beca IN Becas.Codigo_Beca%TYPE, v_Cuantia_Fija IN Becas.Cuantia_Fija%TYPE, v_Cuantia_Variable IN
Becas.Cuantia_Variable%TYPE, v_Tipo IN Becas.Tipo%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Beca IN Becas.Codigo_Beca%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Departamentos
CREATE OR REPLACE PACKAGE PCK_Departamentos
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Departamentos.Nombre%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Dep IN Departamentos.OID_Dep%TYPE, v_Nombre IN Departamentos.Nombre%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Dep IN Departamentos.OID_Dep%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Despachos
CREATE OR REPLACE PACKAGE PCK_Despachos
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Despachos.Nombre%TYPE, v_Capacidad IN Despachos.Capacidad%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_D IN Despachos.OID_D%TYPE, v_Nombre IN Despachos.Nombre%TYPE, v_Capacidad IN Despachos.Capacidad%TYPE, salidaEsperada
BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_D IN Despachos.OID_D%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Expedientes
CREATE OR REPLACE PACKAGE PCK_Expedientes
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Exp IN Expedientes.Codigo_Exp%TYPE, v_DNI IN Expedientes.DNI%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Exp IN Expedientes.Codigo_Exp%TYPE, v_DNI IN Expedientes.DNI%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Exp IN Expedientes.Codigo_Exp%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Grados
CREATE OR REPLACE PACKAGE PCK_Grados
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Grados.Nombre%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_G IN Grados.OID_G%TYPE, v_Nombre IN Grados.Nombre%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_G IN Grados.OID_G%TYPE, salidaEsperada BOOLEAN);
END;
/
```



```
--Tabla Grupos
CREATE OR REPLACE PACKAGE PCK_Grupos
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Grupos.Nombre%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Grup IN Grupos.OID_Grup%TYPE, v_Nombre IN Grupos.Nombre%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Grup IN Grupos.OID_Grup%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla GruposAsistenAClaseEnAulas
CREATE OR REPLACE PACKAGE PCK_GruposAsistenAClaseEnAulas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_OID_Grup IN GruposAsistenAClaseEnAulas.OID_Grup%TYPE, v_OID_A IN GruposAsistenAClaseEnAulas.OID_A%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Grup IN GruposAsistenAClaseEnAulas.OID_Grup%TYPE, v_OID_A IN GruposAsistenAClaseEnAulas.OID_A%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Matriculas
CREATE OR REPLACE PACKAGE PCK_Matriculas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Matric IN Matriculas.Codigo_Matric%TYPE, v_Fecha IN Matriculas.Fecha%TYPE, v_Curso IN Matriculas.Curso%TYPE, v_DNI IN Matriculas.DNI%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Matric IN Matriculas.Codigo_Matric%TYPE, v_Fecha IN Matriculas.Fecha%TYPE, v_Curso IN Matriculas.Curso%TYPE, v_DNI IN Matriculas.DNI%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Matric IN Matriculas.Codigo_Matric%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla MatriculasContienenAsignaturas
CREATE OR REPLACE PACKAGE PCK_MatriculasContienenAsig
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasContienenAsignaturas.Codigo_Matric%TYPE, v_Codigo_Asig IN MatriculasContienenAsignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasContienenAsignaturas.Codigo_Matric%TYPE, v_Codigo_Asig IN MatriculasContienenAsignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla MatriculasPertenecenAGrados
CREATE OR REPLACE PACKAGE PCK_MatricPertenecenAGrados
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasPertenecenAGrados.Codigo_Matric%TYPE, v_OID_G IN MatriculasPertenecenAGrados.OID_G%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasPertenecenAGrados.Codigo_Matric%TYPE, v_OID_G IN MatriculasPertenecenAGrados.OID_G%TYPE, salidaEsperada BOOLEAN);
END;
/
```



```
--Tabla Notas
CREATE OR REPLACE PACKAGE PCK_Notas
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Valor IN Notas.Valor%TYPE, v_Calificacion IN Notas.Calificacion%TYPE, v_Convocatoria IN Notas.Convocatoria%TYPE, v_Curso IN
Notas.Curso%TYPE, v_Fecha IN Notas.Fecha%TYPE, v_Codigo_Exp IN Notas.Codigo_Exp%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_N IN Notas OID_N%TYPE, v_Valor IN Notas.Valor%TYPE, v_Calificacion IN Notas.Calificacion%TYPE, v_Convocatoria IN
Notas.Convocatoria%TYPE, v_Curso IN Notas.Curso%TYPE, v_Fecha IN Notas.Fecha%TYPE, v_Codigo_Exp IN Notas.Codigo_Exp%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_N IN Notas OID_N%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla Profesores
CREATE OR REPLACE PACKAGE PCK_Profesores
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Profesores.DNI%TYPE, v_Nombre IN Profesores.Nombre%TYPE, v_Apellidos IN Profesores.Apellidos%TYPE, v_Fecha_Nacimiento IN
Profesores.Fecha_Nacimiento%TYPE, v_Email IN Profesores.Email%TYPE, v_Categoría IN Profesores.Categoría%TYPE, v_OID_D IN Profesores.OID_D%TYPE, v_OID_Dep IN
Profesores.OID_Dep%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN Profesores.DNI%TYPE, v_Nombre IN Profesores.Nombre%TYPE, v_Apellidos IN Profesores.Apellidos%TYPE, v_Fecha_Nacimiento IN
Profesores.Fecha_Nacimiento%TYPE, v_Email IN Profesores.Email%TYPE, v_Categoría IN Profesores.Categoría%TYPE, v_OID_D IN Profesores.OID_D%TYPE, v_OID_Dep IN
Profesores.OID_Dep%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Profesores.DNI%TYPE, salidaEsperada BOOLEAN);
END;
/

--Tabla ProfesoresImpartenAsignaturas
CREATE OR REPLACE PACKAGE PCK_ProfesoresImpartenAsig
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenAsignaturas.DNI%TYPE, v_Codigo_Asig IN ProfesoresImpartenAsignaturas.Codigo_Asig%TYPE, v_Dedicacion IN
ProfesoresImpartenAsignaturas.Dedicacion%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenAsignaturas.DNI%TYPE, v_Codigo_Asig IN ProfesoresImpartenAsignaturas.Codigo_Asig%TYPE, v_Dedicacion IN
ProfesoresImpartenAsignaturas.Dedicacion%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenAsignaturas.DNI%TYPE, v_Codigo_Asig IN ProfesoresImpartenAsignaturas.Codigo_Asig%TYPE, salidaEsperada
BOOLEAN);
END;
/

--Tabla ProfesoresImpartenEnGrupos
CREATE OR REPLACE PACKAGE PCK_ProfesoresImpartenEnGrupos
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenEnGrupos.DNI%TYPE, v_OID_Grup IN ProfesoresImpartenEnGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenEnGrupos.DNI%TYPE, v_OID_Grup IN ProfesoresImpartenEnGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN);
END;
/
```



```
--Tabla Tutorias
CREATE OR REPLACE PACKAGE PCK_Tutorias
IS
PROCEDURE Inicializar;
PROCEDURE Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Fecha IN Tutorias.Fecha%TYPE, v_Hora_Comienzo IN Tutorias.Hora_Comienzo%TYPE, v_Duracion IN Tutorias.Duracion%TYPE, v_DNI_Alum IN Tutorias.DNI_Alum%TYPE, v_DNI_Prof IN Tutorias.DNI_Prof%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_T IN Tutorias.OID_T%TYPE, v_Fecha IN Tutorias.Fecha%TYPE, v_Hora_Comienzo IN Tutorias.Hora_Comienzo%TYPE, v_Duracion IN Tutorias.Duracion%TYPE, v_DNI_Alum IN Tutorias.DNI_Alum%TYPE, v_DNI_Prof IN Tutorias.DNI_Prof%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_T IN Tutorias.OID_T%TYPE, salidaEsperada BOOLEAN);
END;
/

--Cuerpos de paquetes
--Tabla Alumnos
CREATE OR REPLACE PACKAGE BODY PCK_Alumnos
IS
CURSOR C IS
SELECT * FROM Alumnos;
v_Salida BOOLEAN := TRUE;
v_Alumnos Alumnos%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
DELETE FROM Alumnos;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
OPEN C;
FETCH C INTO v_Alumnos;
DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha_Nacimiento:', 25) || RPAD('Email:', 25));
DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
WHILE C%FOUND LOOP
DBMS_OUTPUT.PUT_LINE(RPAD(v_Alumnos.DNI, 25) || RPAD(v_Alumnos.Nombre, 25) || RPAD(v_Alumnos.Apellidos, 25) || RPAD(v_Alumnos.Fecha_Nacimiento, 25) ||
RPAD(v_Alumnos.Email, 25));
FETCH C INTO v_Alumnos;
END LOOP;
CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE, v_Nombre IN Alumnos.Nombre%TYPE, v_Apellidos IN Alumnos.Apellidos%TYPE, v_Fecha_Nacimiento IN Alumnos.Fecha_Nacimiento%TYPE, v_Email IN Alumnos.Email%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
INSERT INTO Alumnos (DNI, Nombre, Apellidos, Fecha_Nacimiento, Email) VALUES (v_DNI, v_Nombre, v_Apellidos, v_Fecha_Nacimiento, v_Email);
SELECT * INTO v_Alumnos FROM Alumnos WHERE DNI = v_DNI;
IF v_Alumnos.Nombre != v_Nombre AND v_Alumnos.Apellidos != v_Apellidos AND v_Alumnos.Fecha_Nacimiento != v_Fecha_Nacimiento AND v_Alumnos.Email != v_Email THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE, v_Nombre IN Alumnos.Nombre%TYPE, v_Apellidos IN Alumnos.Apellidos%TYPE, v_Fecha_Nacimiento IN Alumnos.Fecha_Nacimiento%TYPE, v_Email IN Alumnos.Email%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
```



```
UPDATE Alumnos SET Nombre = v_Nombre, Apellidos = v_Apellidos, Fecha_Nacimiento = v_Fecha_Nacimiento, Email = v_Email WHERE DNI = v_DNI;
SELECT * INTO v_Alumnos FROM Alumnos WHERE DNI = v_DNI;
IF v_Alumnos.Nombre != v_Nombre AND v_Alumnos.Apellidos != v_Apellidos AND v_Alumnos.Fecha_Nacimiento != v_Fecha_Nacimiento AND v_Alumnos.Email != v_Email THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Alumnos.DNI%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumAlumnos NUMBER := 0;
BEGIN
    DELETE FROM Alumnos WHERE DNI = v_DNI;
    SELECT COUNT(*) INTO v_NumAlumnos FROM Alumnos WHERE DNI = v_DNI;
    IF v_NumAlumnos != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Eliminar;
END;
/
--Tabla AlumnosPertenecenAGrupos
CREATE OR REPLACE PACKAGE BODY PCK_AlumnosPertenecenAGrupos
IS
    CURSOR C IS
        SELECT * FROM AlumnosPertenecenAGrupos;
    v_Salida BOOLEAN := TRUE;
    v_AlumnosPertenecenAGrupos AlumnosPertenecenAGrupos%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM AlumnosPertenecenAGrupos;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_AlumnosPertenecenAGrupos;
    DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('OID_Grup:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_AlumnosPertenecenAGrupos.DNI, 25) || RPAD(v_AlumnosPertenecenAGrupos.OID_Grup, 25));
        FETCH C INTO v_AlumnosPertenecenAGrupos;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN AlumnosPertenecenAGrupos.DNI%TYPE, v_OID_Grup IN AlumnosPertenecenAGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO AlumnosPertenecenAGrupos (DNI, OID_Grup) VALUES (v_DNI, v_OID_grup);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));

```



```
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN AlumnosPertenecenAGrupos.DNI%TYPE, v_OID_Grup IN AlumnosPertenecenAGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN)
IS
  v_NumAlumnosPertenecenAGrupos NUMBER := 0;
BEGIN
  DELETE FROM AlumnosPertenecenAGrupos WHERE DNI = v_DNI AND OID_Grup = v_OID_Grup;
  SELECT COUNT(*) INTO v_NumAlumnosPertenecenAGrupos FROM AlumnosPertenecenAGrupos WHERE DNI = v_DNI AND OID_Grup = v_OID_Grup;
  IF v_NumAlumnosPertenecenAGrupos != 0 THEN
    v_Salida := FALSE;
  END IF;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Eliminar;
END;
/
--Tabla Asignaturas
CREATE OR REPLACE PACKAGE BODY PCK_Asignaturas
IS
  CURSOR C IS
    SELECT * FROM Asignaturas;
  v_Salida BOOLEAN := TRUE;
  v_Asignaturas Asignaturas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
  DELETE FROM Asignaturas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
  OPEN C;
  FETCH C INTO v_Asignaturas;
  DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Asig:', 25) || RPAD('Nombre:', 75) || RPAD('Acronimo:', 25) || RPAD('Creditos:', 25) || RPAD('Periodo:', 25) || RPAD('Curso:', 25)
  || RPAD('Tipo:', 25) || RPAD('OID_Dep:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 250, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Asignaturas.Codigo_Asig, 25) || RPAD(v_Asignaturas.Nombre, 75) || RPAD(v_Asignaturas.Acronimo, 25) || RPAD(v_Asignaturas.Creditos, 25)
  || RPAD(v_Asignaturas.Periodo, 25) || RPAD(v_Asignaturas.Curso, 25) || RPAD(v_Asignaturas.Tipo, 25) || RPAD(v_Asignaturas.OID_Dep, 25));
    FETCH C INTO v_Asignaturas;
  END LOOP;
  CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, v_Nombre IN Asignaturas.Nombre%TYPE, v_Acronimo IN Asignaturas.Acronimo%TYPE,
v_Creditos IN Asignaturas.Creditos%TYPE, v_Periodo IN Asignaturas.Periodo%TYPE, v_Curso IN Asignaturas.Curso%TYPE, v_Tipo IN Asignaturas.Tipo%TYPE, v_OID_Dep IN
Asignaturas.OID_Dep%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
  INSERT INTO Asignaturas (Codigo_Asig, Nombre, Acronimo, Creditos, Periodo, Curso, Tipo, OID_Dep) VALUES (v_Codigo_Asig, v_Nombre, v_Acronimo, v_Creditos, v_Periodo,
v_Curso, v_Tipo, v_OID_Dep);
  SELECT * INTO v_Asignaturas FROM Asignaturas WHERE Codigo_Asig = v_Codigo_Asig;
  IF (v_Asignaturas.Nombre != v_Nombre AND v_Asignaturas.Acronimo != v_Acronimo AND v_Asignaturas.Creditos != v_Creditos AND v_Asignaturas.Periodo != v_Periodo AND
v_Asignaturas.Curso != v_Curso AND v_Asignaturas.Tipo != v_Tipo AND v_Asignaturas.OID_Dep != v_OID_Dep) THEN
    v_Salida := FALSE;
  END IF;
END Insertar;
```



```
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, v_Nombre IN Asignaturas.Nombre%TYPE, v_Acronimo IN Asignaturas.Acronomo%TYPE,
v_Creditos IN Asignaturas.Creditos%TYPE, v_Periodo IN Asignaturas.Periodo%TYPE, v_Curso IN Asignaturas.Curso%TYPE, v_Tipo IN Asignaturas.Tipo%TYPE, v_OID_Dep IN
Asignaturas.OID_Dep%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
UPDATE Asignaturas SET Nombre = v_Nombre, Acronomo = v_Acronimo, Creditos = v_Creditos, Periodo = v_Periodo, Curso = v_Curso, Tipo = v_Tipo, OID_Dep = v_OID_Dep WHERE
Codigo_Asig = v_Codigo_asig;
SELECT * INTO v_Asignaturas FROM Asignaturas WHERE Codigo_Asig = v_Codigo_asig;
IF v_Asignaturas.Nombre != v_Nombre AND v_Asignaturas.Acronomo != v_Acronimo AND v_Asignaturas.Creditos != v_Creditos AND v_Asignaturas.Periodo != v_Periodo AND
v_Asignaturas.Curso = v_Curso AND v_Asignaturas.Tipo = v_Tipo AND v_Asignaturas.OID_Dep = v_OID_Dep THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN Asignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN)
IS
v_NumAsignaturas NUMBER := 0;
BEGIN
DELETE FROM Asignaturas WHERE Codigo_Asig = v_Codigo_Asig;
SELECT COUNT(*) INTO v_NumAsignaturas FROM Asignaturas WHERE Codigo_Asig = v_Codigo_Asig;
IF v_NumAsignaturas != 0 THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Eliminar;
END;
/
--Tabla AsignaturasPuntuadasConNotas
CREATE OR REPLACE PACKAGE BODY PCK_AsigPuntuadasConNotas
IS
CURSOR C IS
SELECT * FROM AsignaturasPuntuadasConNotas;
v_Salida BOOLEAN := TRUE;
v_AsignaturasPuntuadasConNotas AsignaturasPuntuadasConNotas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
DELETE FROM AsignaturasPuntuadasConNotas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
OPEN C;
```



```
FETCH C INTO v_AsignaturasPuntuadasConNotas;
DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Asig:', 25) || RPAD('OID_N:', 25));
DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_AsignaturasPuntuadasConNotas.Codigo_Asig, 25) || RPAD(v_AsignaturasPuntuadasConNotas.OID_N, 25));
    FETCH C INTO v_AsignaturasPuntuadasConNotas;
END LOOP;
CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPuntuadasConNotas.Codigo_Asig%TYPE, v_OID_N IN AsignaturasPuntuadasConNotas.OID_N%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO AsignaturasPuntuadasConNotas (Codigo_Asig, OID_N) VALUES (v_Codigo_Asig, v_OID_N);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPuntuadasConNotas.Codigo_Asig%TYPE, v_OID_N IN AsignaturasPuntuadasConNotas.OID_N%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumAsigPuntuadasConNotas NUMBER := 0;
BEGIN
    DELETE FROM AsignaturasPuntuadasConNotas WHERE Codigo_Asig = v_Codigo_Asig AND OID_N = v_OID_N;
    SELECT COUNT(*) INTO v_NumAsigPuntuadasConNotas FROM AsignaturasPuntuadasConNotas WHERE Codigo_Asig = v_Codigo_Asig AND OID_N = v_OID_N;
    IF v_NumAsigPuntuadasConNotas != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
/
--Tabla AsignaturasImpartidasEnAulas
CREATE OR REPLACE PACKAGE BODY PCK_AsigImpartidasEnAulas
IS
    CURSOR C IS
        SELECT * FROM AsignaturasImpartidasEnAulas;
    v_Salida BOOLEAN := TRUE;
    v_AsignaturasImpartidasEnAulas AsignaturasImpartidasEnAulas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM AsignaturasImpartidasEnAulas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_AsignaturasImpartidasEnAulas;
    DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Asig:', 25) || RPAD('OID_A:', 25) || RPAD('Fecha:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 75, '-'));
    WHILE C%FOUND LOOP
```



```
DBMS_OUTPUT.PUT_LINE(RPAD(v_AsignaturasImpartidasEnAulas.Codigo_Asig, 25) || RPAD(v_AsignaturasImpartidasEnAulas.OID_A, 25) ||  
RPAD(v_AsignaturasImpartidasEnAulas.Fecha, 25));  
    FETCH C INTO v_AsignaturasImpartidasEnAulas;  
    END LOOP;  
    CLOSE C;  
END Consultar;  
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasImpartidasEnAulas.Codigo_Asig%TYPE, v_OID_A IN AsignaturasImpartidasEnAulas.OID_A%TYPE, v_Fecha IN  
AsignaturasImpartidasEnAulas.Fecha%TYPE, salidaEsperada BOOLEAN)  
IS  
BEGIN  
    INSERT INTO AsignaturasImpartidasEnAulas (Codigo_Asig, OID_A, Fecha) VALUES (v_Codigo_Asig, v_OID_A, v_Fecha);  
    COMMIT;  
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));  
EXCEPTION  
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));  
        ROLLBACK;  
END Insertar;  
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasImpartidasEnAulas.Codigo_Asig%TYPE, v_OID_A IN AsignaturasImpartidasEnAulas.OID_A%TYPE, v_Fecha IN  
AsignaturasImpartidasEnAulas.Fecha%TYPE, salidaEsperada BOOLEAN)  
IS  
    v_NumAsigImpartidasEnAulas NUMBER := 0;  
BEGIN  
    DELETE FROM AsignaturasImpartidasEnAulas WHERE Codigo_Asig = v_Codigo_Asig AND OID_A = v_OID_A AND Fecha = v_Fecha;  
    SELECT COUNT(*) INTO v_NumAsigImpartidasEnAulas FROM AsignaturasImpartidasEnAulas WHERE Codigo_Asig = v_Codigo_Asig AND OID_A = v_OID_A AND Fecha = v_Fecha;  
    IF v_NumAsigImpartidasEnAulas != 0 THEN  
        v_Salida := FALSE;  
    END IF;  
    COMMIT;  
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));  
EXCEPTION  
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));  
        ROLLBACK;  
    END Eliminar;  
END;  
/  
  
--Tabla AsignaturasPertenecenAGrados  
CREATE OR REPLACE PACKAGE BODY PCK_AsigPertenecenAGrados  
IS  
    CURSOR C IS  
        SELECT * FROM AsignaturasPertenecenAGrados;  
    v_Salida BOOLEAN := TRUE;  
    v_AsigPertenecenAGrados AsignaturasPertenecenAGrados%ROWTYPE;  
PROCEDURE Inicializar  
IS  
BEGIN  
    DELETE FROM AsignaturasPertenecenAGrados;  
END Inicializar;  
PROCEDURE Consultar  
IS  
BEGIN  
    OPEN C;  
    FETCH C INTO v_AsigPertenecenAGrados;  
    DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Asig:', 25) || RPAD('OID_G', 25));  
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));  
    WHILE C%FOUND LOOP  
        DBMS_OUTPUT.PUT_LINE(RPAD(v_AsigPertenecenAGrados.Codigo_Asig, 25) || RPAD(v_AsigPertenecenAGrados.OID_G, 25));  
        FETCH C INTO v_AsigPertenecenAGrados;  
    END LOOP;  
    CLOSE C;
```



```
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPertenecenAGrados.Codigo_Asig%TYPE, v_OID_G IN AsignaturasPertenecenAGrados.OID_G%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO AsignaturasPertenecenAGrados (Codigo_Asig, OID_G) VALUES (v_Codigo_Asig, v_OID_G);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Asig IN AsignaturasPertenecenAGrados.Codigo_Asig%TYPE, v_OID_G IN AsignaturasPertenecenAGrados.OID_G%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumAsigPertenecenAGrados NUMBER := 0;
BEGIN
    DELETE FROM AsignaturasPertenecenAGrados WHERE Codigo_Asig = v_Codigo_Asig AND OID_G = v_OID_G;
    SELECT COUNT(*) INTO v_NumAsigPertenecenAGrados FROM AsignaturasPertenecenAGrados WHERE Codigo_Asig = v_Codigo_Asig AND OID_G = v_OID_G;
    IF v_NumAsigPertenecenAGrados != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
/

--Tabla Aulas
CREATE OR REPLACE PACKAGE BODY PCK_Aulas
IS
    CURSOR C IS
        SELECT * FROM Aulas;
    v_Salida BOOLEAN := TRUE;
    v_Aulas Aulas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM Aulas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_Aulas;
    DBMS_OUTPUT.PUT_LINE(RPAD('OID_A:', 25) || RPAD('Nombre:', 25) || RPAD('Capacidad:', 25) || RPAD('Tipo:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 100, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Aulas.OID_A, 25) || RPAD(v_Aulas.Nombre, 25) || RPAD(v_Aulas.Capacidad, 25) || RPAD(v_Aulas.Tipo, 25));
        FETCH C INTO v_Aulas;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Aulas.Nombre%TYPE, v_Capacidad IN Aulas.Capacidad%TYPE, v_Tipo IN Aulas.Tipo%TYPE, salidaEsperada BOOLEAN)
IS
    v_OID_A Aulas.OID_A%TYPE;
BEGIN
```



```
INSERT INTO Aulas (Nombre, Capacidad, Tipo) VALUES (v_Nombre, v_Capacidad, v_Tipo);
v_OID_A := SEC_Aulas.CURRVAL;
SELECT * INTO v_Aulas FROM Aulas WHERE OID_A = v_OID_A;
IF v_Aulas.Nombre != v_Nombre AND v_Aulas.Capacidad != v_Capacidad AND v_Aulas.Tipo != v_Tipo THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_A IN Aulas.OID_A%TYPE, v_Nombre IN Aulas.Nombre%TYPE, v_Capacidad IN Aulas.Capacidad%TYPE, v_Tipo IN Aulas.Tipo%TYPE,
salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Aulas SET Nombre = v_Nombre, Capacidad = v_Capacidad, Tipo = v_Tipo WHERE OID_A = v_OID_A;
    SELECT * INTO v_Aulas FROM Aulas WHERE OID_A = v_OID_A;
    IF v_Aulas.Nombre != v_Nombre AND v_Aulas.Capacidad != v_Capacidad AND v_Aulas.Tipo != v_Tipo THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_A IN Aulas.OID_A%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumAulas NUMBER := 0;
BEGIN
    DELETE FROM AULAS WHERE OID_A = v_OID_A;
    SELECT COUNT(*) INTO v_NumAulas FROM Aulas WHERE OID_A = v_OID_A;
    IF v_NumAulas != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Eliminar;
END;
/
--Tabla Becarios
CREATE OR REPLACE PACKAGE BODY PCK_Becarios
IS
    CURSOR C IS
        SELECT * FROM Becarios;
    v_Salida BOOLEAN := TRUE;
    v_Becarios Becarios%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM Becarios;
END Inicializar;
PROCEDURE Consultar
IS
```



```
BEGIN
    OPEN C;
    FETCH C INTO v_Becarios;
    DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Codigo_Beca:', 25) || RPAD('Fecha_Comienzo:', 25) || RPAD('Fecha_Fin:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 100, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Becarios.DNI, 25) || RPAD(v_Becarios.Codigo_Beca, 25) || RPAD(v_Becarios.Fecha_Comienzo, 25) || RPAD(v_Becarios.Fecha_Fin, 25));
        FETCH C INTO v_Becarios;
    END LOOP;
    CLOSE C;
END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Becarios.DNI%TYPE, v_Codigo_Beca IN Becarios.Codigo_Beca%TYPE, v_Fecha_Comienzo IN Becarios.Fecha_Comienzo%TYPE,
v_Fecha_Fin IN Becarios.Fecha_Fin%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO Becarios (DNI, Codigo_Beca, Fecha_Comienzo, Fecha_Fin) VALUES (v_DNI, v_Codigo_Beca, v_Fecha_Comienzo, v_Fecha_Fin);
    SELECT * INTO v_Becarios FROM Becarios WHERE DNI = v_DNI AND Codigo_Beca = v_Codigo_Beca;
    IF v_Becarios.Fecha_Comienzo != v_Fecha_Comienzo AND v_Becarios.Fecha_Fin != v_Fecha_Fin THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN Becarios.DNI%TYPE, v_Codigo_Beca IN Becarios.Codigo_Beca%TYPE, v_Fecha_Comienzo IN Becarios.Fecha_Comienzo%TYPE,
v_Fecha_Fin IN Becarios.Fecha_Fin%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Becarios SET Fecha_Comienzo=v_Fecha_Comienzo, Fecha_Fin=v_Fecha_Fin WHERE DNI = v_DNI AND Codigo_Beca = v_Codigo_Beca;
    SELECT * INTO v_Becarios FROM Becarios WHERE DNI = v_DNI AND Codigo_Beca = v_Codigo_Beca;
    IF v_Becarios.Fecha_Comienzo != v_Fecha_Comienzo AND v_Becarios.Fecha_Fin != v_Fecha_Fin THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Becarios.DNI%TYPE, v_Codigo_Beca IN Becarios.Codigo_Beca%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumBecarios NUMBER := 0;
BEGIN
    DELETE FROM Becarios WHERE DNI = v_DNI AND Codigo_Beca = v_Codigo_Beca;
    SELECT COUNT(*) INTO v_NumBecarios FROM Becarios WHERE DNI = v_DNI AND Codigo_Beca = v_Codigo_Beca;
    IF v_NumBecarios != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
```



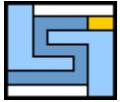
```
--Tabla Becas
CREATE OR REPLACE PACKAGE BODY PCK_Becas
IS
    CURSOR C IS
        SELECT * FROM Becas;
    v_Salida BOOLEAN := TRUE;
    v_Becas Becas%ROWTYPE;
    PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Becas;
    END Inicializar;
    PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Becas;
        DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Beca:', 25) || RPAD('Cuantia_Fija:', 25) || RPAD('Cuantia_Variable:', 25) || RPAD('Tipo:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 100, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Becas.Codigo_Beca, 25) || RPAD(v_Becas.Cuantia_Fija, 25) || RPAD(v_Becas.Cuantia_Variable, 25) || RPAD(v_Becas.Tipo, 25));
            FETCH C INTO v_Becas;
        END LOOP;
        CLOSE C;
    END Consultar;
    PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Beca IN Becas.Codigo_Beca%TYPE, v_Cuantia_Fija IN Becas.Cuantia_Fija%TYPE, v_Cuantia_Variable IN
Becas.Cuantia_Variable%TYPE, v_Tipo IN Becas.Tipo%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        INSERT INTO Becas (Codigo_Beca, Cuantia_Fija, Cuantia_Variable, Tipo) VALUES (v_Codigo_Beca, v_Cuantia_Fija, v_Cuantia_Variable, v_Tipo);
        SELECT * INTO v_Becas FROM Becas WHERE Codigo_Beca = v_Codigo_Beca;
        IF v_Becas.Cuantia_Fija != v_Cuantia_Fija AND v_Becas.Cuantia_Variable = v_Cuantia_Variable AND v_Becas.Tipo = v_Tipo THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Insertar;
    PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Beca IN Becas.Codigo_Beca%TYPE, v_Cuantia_Fija IN Becas.Cuantia_Fija%TYPE, v_Cuantia_Variable IN
Becas.Cuantia_Variable%TYPE, v_Tipo IN Becas.Tipo%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Becas SET Cuantia_Fija = v_Cuantia_Fija, Cuantia_Variable = v_Cuantia_Variable, Tipo = v_Tipo WHERE Codigo_Beca = v_Codigo_Beca;
        SELECT * INTO v_Becas FROM Becas WHERE Codigo_Beca = v_Codigo_Beca;
        IF v_Becas.Cuantia_Fija != v_Cuantia_Fija AND v_Becas.Cuantia_Variable = v_Cuantia_Variable AND v_Becas.Tipo = v_Tipo THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Actualizar;
    PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Beca IN Becas.Codigo_Beca%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumBecas NUMBER := 0;
    BEGIN
```



```
DELETE FROM Becas WHERE Código_Beca = v_Código_Beca;
SELECT COUNT(*) INTO v_NumBecas FROM Becas WHERE Código_Beca = v_Código_Beca;
IF v_NumBecas != 0 THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
/
--Tabla Departamentos
CREATE OR REPLACE PACKAGE BODY PCK_Departamentos
IS
    CURSOR C IS
        SELECT * FROM Departamentos;
    v_Salida BOOLEAN := TRUE;
    v_Departamentos Departamentos%ROWTYPE;
    PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Departamentos;
    END Inicializar;
    PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Departamentos;
        DBMS_OUTPUT.PUT_LINE(RPAD('OID_Dep:', 25) || RPAD('Nombre:', 50));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 75, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Departamentos.OID_Dep, 25) || RPAD(v_Departamentos.Nombre, 50));
            FETCH C INTO v_Departamentos;
        END LOOP;
        CLOSE C;
    END Consultar;
    PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Departamentos.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
        v_OID_Dep Departamentos.OID_Dep%TYPE;
    BEGIN
        INSERT INTO Departamentos (Nombre) VALUES (v_Nombre);
        v_OID_Dep := SEC_Departamentos.CURRVAL;
        SELECT * INTO v_Departamentos FROM Departamentos WHERE OID_Dep = v_OID_Dep;
        IF v_Departamentos.Nombre != v_Nombre THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
    PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Dep IN Departamentos.OID_Dep%TYPE, v_Nombre IN Departamentos.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Departamentos SET Nombre = v_Nombre WHERE OID_Dep = v_OID_Dep;
        SELECT * INTO v_Departamentos FROM Departamentos WHERE OID_Dep = v_OID_Dep;
```



```
IF v_Departamentos.Nombre != v_Nombre THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Dep IN Departamentos.OID_Dep%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumDepartamentos NUMBER := 0;
BEGIN
    DELETE FROM Departamentos WHERE OID_Dep = v_OID_Dep;
    SELECT COUNT(*) INTO v_NumDepartamentos FROM Departamentos WHERE OID_Dep = v_OID_Dep;
    IF v_NumDepartamentos != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Eliminar;
END;
/
--Tabla Despachos
CREATE OR REPLACE PACKAGE BODY PCK_Despachos
IS
    CURSOR C IS
        SELECT * FROM Despachos;
    v_Salida BOOLEAN := TRUE;
    v_Despachos Despachos%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM Despachos;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_Despachos;
    DBMS_OUTPUT.PUT_LINE(RPAD('OID_D:', 25) || RPAD('Nombre:', 25) || RPAD('Capacidad:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 75, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Despachos.OID_D, 25) || RPAD(v_Despachos.Nombre, 25) || RPAD(v_Despachos.Capacidad, 25));
        FETCH C INTO v_Despachos;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Despachos.Nombre%TYPE, v_Capacidad IN Despachos.Capacidad%TYPE, salidaEsperada BOOLEAN)
IS
    v_OID_D Despachos.OID_D%TYPE;
BEGIN
    INSERT INTO Despachos (Nombre, Capacidad) VALUES (v_Nombre, v_Capacidad);
    v_OID_D := SEC_Despachos.CURRVAL;
    SELECT * INTO v_Despachos FROM Despachos WHERE OID_D = v_OID_D;
    IF v_Despachos.Nombre != v_Nombre AND v_Despachos.Capacidad != v_Capacidad THEN
        v_Salida := FALSE;
    ELSE
        v_Salida := TRUE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Insertar;
END;
```



```
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_D IN Despachos.OID_D%TYPE, v_Nombre IN Despachos.Nombre%TYPE, v_Capacidad IN Despachos.Capacidad%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
UPDATE Despachos SET Nombre = v_Nombre, Capacidad = v_Capacidad WHERE OID_D = v_OID_D;
SELECT * INTO v_Despachos FROM Despachos WHERE OID_D = v_OID_D;
IF v_Despachos.Nombre != v_Nombre AND v_Despachos.Capacidad != v_Capacidad THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_D IN Despachos.OID_D%TYPE, salidaEsperada BOOLEAN)
IS
v_NumDespachos NUMBER := 0;
BEGIN
DELETE FROM Despachos WHERE OID_D = v_OID_D;
SELECT COUNT(*) INTO v_NumDespachos FROM Despachos WHERE OID_D = v_OID_D;
IF v_NumDespachos != 0 THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Eliminar;
END;
/
--Tabla Expedientes
CREATE OR REPLACE PACKAGE BODY PCK_Expedientes
IS
CURSOR C IS
SELECT * FROM Expedientes;
v_Salida BOOLEAN := TRUE;
v_Expedientes Expedientes%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
DELETE FROM Expedientes;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
OPEN C;
FETCH C INTO v_Expedientes;
DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Exp:', 25) || RPAD('DNI:', 25));

```



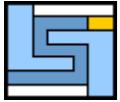
```
DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_Expedientes.Codigo_Exp, 25) || RPAD(v_Expedientes.DNI, 25));
    FETCH C INTO v_Expedientes;
END LOOP;
CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Exp IN Expedientes.Codigo_Exp%TYPE, v_DNI IN Expedientes.DNI%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO Expedientes (Codigo_Exp, DNI) VALUES (v_Codigo_Exp, v_DNI);
    SELECT * INTO v_Expedientes FROM Expedientes WHERE Codigo_Exp = v_Codigo_Exp;
    IF v_Expedientes.DNI != v_DNI THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Exp IN Expedientes.Codigo_Exp%TYPE, v_DNI IN Expedientes.DNI%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Expedientes SET DNI = v_DNI WHERE Codigo_Exp = v_Codigo_Exp;
    SELECT * INTO v_Expedientes FROM Expedientes WHERE Codigo_Exp = v_Codigo_Exp;
    IF v_Expedientes.DNI != v_DNI THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Exp IN Expedientes.Codigo_Exp%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumExpedientes NUMBER := 0;
BEGIN
    DELETE FROM Expedientes WHERE Codigo_Exp = v_Codigo_Exp;
    SELECT COUNT(*) INTO v_NumExpedientes FROM Expedientes WHERE Codigo_Exp = v_Codigo_Exp;
    IF v_NumExpedientes != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
/
```



```
--Tabla Grados
CREATE OR REPLACE PACKAGE BODY PCK_Grados
IS
    CURSOR C IS
        SELECT * FROM Grados;
    v_Salida BOOLEAN := TRUE;
    v_Grados Grados%ROWTYPE;
    PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Grados;
    END Inicializar;
    PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Grados;
        DBMS_OUTPUT.PUT_LINE(RPAD('OID_G:', 25) || RPAD('Nombre:', 55));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 80, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Grados.OID_G, 25) || RPAD(v_Grados.Nombre, 55));
            FETCH C INTO v_Grados;
        END LOOP;
        CLOSE C;
    END Consultar;
    PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Grados.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
        v_OID_G Grados.OID_G%TYPE;
    BEGIN
        INSERT INTO Grados (Nombre) VALUES (v_Nombre);
        v_OID_G := SEC_Grados.CURRVAL;
        SELECT * INTO v_Grados FROM Grados WHERE OID_G = v_OID_G;
        IF v_Grados.Nombre != v_Nombre THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Insertar;
    PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_G IN Grados.OID_G%TYPE, v_Nombre IN Grados.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Grados SET Nombre = v_Nombre WHERE OID_G = v_OID_G;
        SELECT * INTO v_Grados FROM Grados WHERE OID_G = v_OID_G;
        IF v_Grados.Nombre != v_Nombre THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
            ROLLBACK;
    END Actualizar;
    PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_G IN Grados.OID_G%TYPE, salidaEsperada BOOLEAN)
    IS
        v_NumGrados NUMBER := 0;
    BEGIN
```



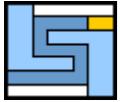
```
DELETE FROM Grados WHERE OID_G = v_OID_G;
SELECT COUNT(*) INTO v_NumGrados FROM Grados WHERE OID_G = v_OID_G;
IF v_NumGrados != 0 THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
/
--Tabla Grupos
CREATE OR REPLACE PACKAGE BODY PCK_Grupos
IS
    CURSOR C IS
        SELECT * FROM Grupos;
    v_Salida BOOLEAN := TRUE;
    v_Grupos Grupos%ROWTYPE;
    PROCEDURE Inicializar
    IS
    BEGIN
        DELETE FROM Grupos;
    END Inicializar;
    PROCEDURE Consultar
    IS
    BEGIN
        OPEN C;
        FETCH C INTO v_Grupos;
        DBMS_OUTPUT.PUT_LINE(RPAD('OID_Grup:', 25) || RPAD('Nombre:', 25));
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
        WHILE C%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(v_Grupos.OID_Grup, 25) || RPAD(v_Grupos.Nombre, 25));
            FETCH C INTO v_Grupos;
        END LOOP;
        CLOSE C;
    END Consultar;
    PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Nombre IN Grupos.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
        v_OID_Grup NUMBER;
    BEGIN
        INSERT INTO Grupos (Nombre) VALUES (v_Nombre);
        v_OID_Grup := SEC_Grupos.CURRVAL;
        SELECT * INTO v_Grupos FROM Grupos WHERE OID_Grup = v_OID_Grup;
        IF v_Grupos.Nombre != v_Nombre THEN
            v_Salida := FALSE;
        END IF;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
        EXCEPTION
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
                ROLLBACK;
    END Insertar;
    PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Grup IN Grupos.OID_Grup%TYPE, v_Nombre IN Grupos.Nombre%TYPE, salidaEsperada BOOLEAN)
    IS
    BEGIN
        UPDATE Grupos SET Nombre = v_Nombre WHERE OID_Grup = v_OID_Grup;
        SELECT * INTO v_Grupos FROM Grupos WHERE OID_Grup = v_OID_Grup;
```



```
IF v_Grupos.Nombre != v_Nombre THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Grup IN Grupos.OID_Grup%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumGrupos NUMBER := 0;
BEGIN
    DELETE FROM Grupos WHERE OID_Grup = v_OID_Grup;
    SELECT COUNT(*) INTO v_NumGrupos FROM Grupos WHERE OID_Grup = v_OID_Grup;
    IF v_NumGrupos != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Eliminar;
END;
/
--Tabla GruposAsistenAClaseEnAulas
CREATE OR REPLACE PACKAGE BODY PCK_GruposAsistenAClaseEnAulas
IS
    CURSOR C IS
        SELECT * FROM GruposAsistenAClaseEnAulas;
    v_Salida BOOLEAN := TRUE;
    v_GruposAsistenAClaseEnAulas GruposAsistenAClaseEnAulas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM GruposAsistenAClaseEnAulas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_GruposAsistenAClaseEnAulas;
    DBMS_OUTPUT.PUT_LINE(RPAD('OID_Grup:', 25) || RPAD('OID_A:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_GruposAsistenAClaseEnAulas.OID_Grup, 25) || RPAD(v_GruposAsistenAClaseEnAulas.OID_A, 25));
        FETCH C INTO v_GruposAsistenAClaseEnAulas;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_OID_Grup IN GruposAsistenAClaseEnAulas.OID_Grup%TYPE, v_OID_A IN GruposAsistenAClaseEnAulas.OID_A%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO GruposAsistenAClaseEnAulas (OID_Grup, OID_A) VALUES (v_OID_Grup, v_OID_A);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
```



```
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Grup IN GruposAsistenAClaseEnAulas.OID_Grup%TYPE, v_OID_A IN GruposAsistenAClaseEnAulas.OID_A%TYPE, salidaEsperada BOOLEAN)
IS
v_NumGrupAsistenAClaseEnAulas NUMBER := 0;
BEGIN
DELETE FROM GruposAsistenAClaseEnAulas WHERE OID_Grup = v_OID_Grup AND OID_A = v_OID_A;
SELECT COUNT(*) INTO v_NumGrupAsistenAClaseEnAulas FROM GruposAsistenAClaseEnAulas WHERE OID_Grup = v_OID_Grup AND OID_A = v_OID_A;
IF v_NumGrupAsistenAClaseEnAulas != 0 THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Eliminar;
END;
/
--Tabla Matriculas
CREATE OR REPLACE PACKAGE BODY PCK_Matriculas
IS
CURSOR C IS
SELECT * FROM Matriculas;
v_Salida BOOLEAN := TRUE;
v_Matriculas Matriculas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
DELETE FROM Matriculas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
OPEN C;
FETCH C INTO v_Matriculas;
DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Matric:', 25) || RPAD('Fecha:', 25)||RPAD('Curso:', 25)||RPAD('DNI:', 25));
DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
WHILE C%FOUND LOOP
DBMS_OUTPUT.PUT_LINE(RPAD(v_Matriculas.Codigo_Matric, 25) || RPAD(v_Matriculas.Fecha, 25)|| RPAD(v_Matriculas.Curso, 25)|| RPAD(v_Matriculas.DNI, 25));
FETCH C INTO v_Matriculas;
END LOOP;
CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Matric IN Matriculas.Codigo_Matric%TYPE, v_Fecha IN Matriculas.Fecha%TYPE, v_Curso IN Matriculas.Curso%TYPE, v_DNI IN Matriculas.DNI%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
INSERT INTO Matriculas (Codigo_Matric, Fecha, Curso, DNI) VALUES (v_Codigo_Matric, v_Fecha, v_Curso, v_DNI);
SELECT * INTO v_Matriculas FROM Matriculas WHERE Codigo_Matric = v_Codigo_Matric;
IF v_Matriculas.Fecha != v_Fecha AND v_Matriculas.Curso != v_Curso AND v_Matriculas.DNI != v_DNI THEN
v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
```



```
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_Codigo_Matric IN Matriculas.Codigo_Matric%TYPE, v_Fecha IN Matriculas.Fecha%TYPE, v_Curso IN Matriculas.Curso%TYPE, v_DNI IN Matriculas.DNI%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Matriculas SET Fecha = v_Fecha, Curso = v_Curso, DNI = v_DNI WHERE Codigo_Matric = v_Codigo_Matric;
    SELECT * INTO v_Matriculas FROM Matriculas WHERE Codigo_Matric = v_Codigo_Matric;
    IF v_Matriculas.Fecha != v_Fecha AND v_Matriculas.Curso != v_Curso AND v_Matriculas.DNI != v_DNI THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Matric IN Matriculas.Codigo_Matric%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumMatriculas NUMBER := 0;
BEGIN
    DELETE FROM Matriculas WHERE Codigo_Matric = v_Codigo_Matric;
    SELECT COUNT(*) INTO v_NumMatriculas FROM Matriculas WHERE Codigo_Matric = v_Codigo_Matric;
    IF v_NumMatriculas != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
/
--Tabla MatriculasContienenAsignaturas
CREATE OR REPLACE PACKAGE BODY PCK_MatriculasContienenAsig
IS
    CURSOR C IS
        SELECT * FROM MatriculasContienenAsignaturas;
    v_Salida BOOLEAN := TRUE;
    v_MatriculasContienenAsig MatriculasContienenAsignaturas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM MatriculasContienenAsignaturas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_MatriculasContienenAsig;
    DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Matric:', 25) || RPAD('Codigo_Asig:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_MatriculasContienenAsig.Codigo_Matric, 25) || RPAD(v_MatriculasContienenAsig.Codigo_Asig, 25));
        FETCH C INTO v_MatriculasContienenAsig;
    END LOOP;
    CLOSE C;
END Consultar;
```



```
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasContienenAsignaturas.Codigo_Matric%TYPE, v_Codigo_Asig IN MatriculasContienenAsignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO MatriculasContienenAsignaturas (Codigo_Matric, Codigo_Asig) VALUES (v_Codigo_Matric, v_Codigo_Asig);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasContienenAsignaturas.Codigo_Matric%TYPE, v_Codigo_Asig IN MatriculasContienenAsignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumMatriculasContienenAsig NUMBER := 0;
BEGIN
    DELETE FROM MatriculasContienenAsignaturas WHERE Codigo_Matric = v_Codigo_Matric AND Codigo_Asig = v_Codigo_Asig;
    SELECT COUNT(*) INTO v_NumMatriculasContienenAsig FROM MatriculasContienenAsignaturas WHERE Codigo_Matric = v_Codigo_Matric AND Codigo_Asig = v_Codigo_Asig;
    IF v_NumMatriculasContienenAsig != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
/
--Tabla MatriculasPertenecenAGrados
CREATE OR REPLACE PACKAGE BODY PCK_MatricPertenecenAGrados
IS
    CURSOR C IS
        SELECT * FROM MatriculasPertenecenAGrados;
    v_Salida BOOLEAN := TRUE;
    v_MatriculasContienenAsig MatriculasPertenecenAGrados%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM MatriculasPertenecenAGrados;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_MatriculasContienenAsig;
    DBMS_OUTPUT.PUT_LINE(RPAD('Codigo_Matric:', 25) || RPAD('OID_G:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_MatriculasContienenAsig.Codigo_Matric, 25) || RPAD(v_MatriculasContienenAsig.OID_G, 25));
        FETCH C INTO v_MatriculasContienenAsig;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasPertenecenAGrados.Codigo_Matric%TYPE, v_OID_G IN MatriculasPertenecenAGrados.OID_G%TYPE,
salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO MatriculasPertenecenAGrados (Codigo_Matric, OID_G) VALUES (v_Codigo_Matric, v_OID_G);
```



```
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_Codigo_Matric IN MatriculasPertenecenAGrados.Codigo_Matric%TYPE, v_OID_G IN MatriculasPertenecenAGrados.OID_G%TYPE,
salidaEsperada BOOLEAN)
IS
    v_NumMatricPertenecenAGrados NUMBER := 0;
BEGIN
    DELETE FROM MatriculasPertenecenAGrados WHERE Codigo_Matric = v_Codigo_Matric AND OID_G = v_OID_G;
    SELECT COUNT(*) INTO v_NumMatricPertenecenAGrados FROM MatriculasPertenecenAGrados WHERE Codigo_Matric = v_Codigo_Matric AND OID_G = v_OID_G;
    IF v_NumMatricPertenecenAGrados != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Eliminar;
END;
/

--Tabla Notas
CREATE OR REPLACE PACKAGE BODY PCK_Notas
IS
    CURSOR C IS
        SELECT * FROM Notas;
    v_Salida BOOLEAN := TRUE;
    v_Notas Notas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM Notas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_Notas;
    DBMS_OUTPUT.PUT_LINE(RPAD('OID_N:', 25) || RPAD('Valor:', 25) || RPAD('Calificacion:', 25) || RPAD('Convocatoria:', 25) || RPAD('Curso:', 25) || RPAD('Fecha:', 25) ||
RPAD('Codigo_Exp:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 175, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Notas.OID_N, 25) || RPAD(v_Notas.Valor, 25) || RPAD(v_Notas.Calificacion, 25) || RPAD(v_Notas.Convocatoria, 25) || RPAD(v_Notas.Curso,
25) || RPAD(v_Notas.Fecha, 25) || RPAD(v_Notas.Codigo_Exp, 25));
        FETCH C INTO v_Notas;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Valor IN Notas.Valor%TYPE, v_Calificacion IN Notas.Calificacion%TYPE, v_Convocatoria IN Notas.Convocatoria%TYPE, v_Curso IN
Notas.Curso%TYPE, v_Fecha IN Notas.Fecha%TYPE, v_Codigo_Exp IN Notas.Codigo_Exp%TYPE, salidaEsperada BOOLEAN)
IS
    v_OID_N Notas.OID_N%TYPE;
BEGIN
    INSERT INTO Notas (Valor, Calificacion, Convocatoria, Curso, Fecha, Codigo_Exp) VALUES (v_Valor, v_Calificacion, v_Convocatoria, v_Curso, v_Fecha, v_Codigo_Exp);
    v_OID_N := SEC_Notas.CURRVAL;
    SELECT * INTO v_Notas FROM Notas WHERE OID_N = v_OID_N;
```



```
    IF v_Notas.Valor != v_Valor AND v_Notas.Calificacion != v_Calificacion AND v_Notas.Convocatoria != v_Convocatoria AND v_Curso != v_Curso AND v_Notas.Fecha != v_Fecha
AND v_Notas.Codigo_Exp != v_Codigo_Exp THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_N IN Notas.OID_N%TYPE, v_Valor IN Notas.Valor%TYPE, v_Calificacion IN Notas.Calificacion%TYPE, v_Convocatoria IN
Notas.Convocatoria%TYPE, v_Curso IN Notas.Curso%TYPE, v_Fecha IN Notas.Fecha%TYPE, v_Codigo_Exp IN Notas.Codigo_Exp%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Notas SET Valor = v_Valor, Calificacion = v_Calificacion, Convocatoria = v_Convocatoria, Curso = v_Curso, Fecha = v_Fecha, Codigo_Exp = v_Codigo_Exp WHERE
OID_N = v_OID_N;
    SELECT * INTO v_Notas FROM Notas WHERE OID_N = v_OID_N;
    IF v_Notas.Valor != v_Valor AND v_Notas.Calificacion != v_Calificacion AND v_Notas.Convocatoria != v_Convocatoria AND v_Curso != v_Curso AND v_Notas.Fecha != v_Fecha
AND v_Notas.Codigo_Exp != v_Codigo_Exp THEN
    v_Salida := FALSE;
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_N IN Notas.OID_N%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumNotas NUMBER := 0;
BEGIN
    DELETE FROM Notas WHERE OID_N = v_OID_N;
    SELECT COUNT(*) INTO v_NumNotas FROM Notas WHERE OID_N = v_OID_N;
    IF v_NumNotas != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Eliminar;
END;
/
--Tabla Profesores
CREATE OR REPLACE PACKAGE BODY PCK_Profesores
IS
    CURSOR C IS
        SELECT * FROM Profesores;
    v_Salida BOOLEAN := TRUE;
    v_Profesores Profesores%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM Profesores;
END Inicializar;
PROCEDURE Consultar
IS
```



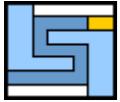
```
BEGIN
    OPEN C;
    FETCH C INTO v_Profesores;
    DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Nombre:', 25) || RPAD('Apellidos:', 25) || RPAD('Fecha_Nacimiento:', 25) || RPAD('Email:', 25) || RPAD('Categoria:', 25) || RPAD('OID_D:', 25) || RPAD('OID_Dep:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 200, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Profesores.DNI, 25) || RPAD(v_Profesores.Nombre, 25) || RPAD(v_Profesores.Apellidos, 25) || RPAD(v_Profesores.Fecha_Nacimiento, 25) || RPAD(v_Profesores.Email, 25) || RPAD(v_Profesores.Categoria, 25) || RPAD(v_Profesores.OID_D, 25) || RPAD(v_Profesores.OID_Dep, 25));
        FETCH C INTO v_Profesores;
    END LOOP;
    CLOSE C;
END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN Profesores.DNI%TYPE, v_Nombre IN Profesores.Nombre%TYPE, v_Apellidos IN Profesores.Apellidos%TYPE, v_Fecha_Nacimiento IN Profesores.Fecha_Nacimiento%TYPE, v_Email IN Profesores.Email%TYPE, v_Categoría IN Profesores.Categoria%TYPE, v_OID_D IN Profesores.OID_D%TYPE, v_OID_Dep IN Profesores.OID_Dep%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO Profesores (DNI, Nombre, Apellidos, Fecha_Nacimiento, Email, Categoria, OID_D, OID_Dep) VALUES (v_DNI, v_Nombre, v_Apellidos, v_Fecha_Nacimiento, v_Email, v_Categoría, v_OID_D, v_OID_Dep);
    SELECT * INTO v_Profesores FROM Profesores WHERE DNI = v_DNI;
    IF v_Profesores.Nombre != v_Nombre AND v_Profesores.Apellidos != v_Apellidos AND v_Profesores.Fecha_Nacimiento != v_Fecha_Nacimiento AND v_Profesores.Email != v_Email AND v_Profesores.Categoria != v_Categoría AND v_Profesores.OID_D != v_OID_D AND v_Profesores.OID_Dep != v_OID_Dep THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN Profesores.DNI%TYPE, v_Nombre IN Profesores.Nombre%TYPE, v_Apellidos IN Profesores.Apellidos%TYPE, v_Fecha_Nacimiento IN Profesores.Fecha_Nacimiento%TYPE, v_Email IN Profesores.Email%TYPE, v_Categoría IN Profesores.Categoria%TYPE, v_OID_D IN Profesores.OID_D%TYPE, v_OID_Dep IN Profesores.OID_Dep%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Profesores SET Nombre = v_Nombre, Apellidos = v_Apellidos, Fecha_Nacimiento = v_Fecha_Nacimiento, Email = v_Email, Categoria = v_Categoría, OID_D = v_OID_D, OID_Dep = v_OID_Dep WHERE DNI = v_DNI;
    SELECT * INTO v_Profesores FROM Profesores WHERE DNI = v_DNI;
    IF v_Profesores.Nombre != v_Nombre AND v_Profesores.Apellidos != v_Apellidos AND v_Profesores.Fecha_Nacimiento != v_Fecha_Nacimiento AND v_Profesores.Email != v_Email AND v_Profesores.Categoria != v_Categoría AND v_Profesores.OID_D != v_OID_D AND v_Profesores.OID_Dep != v_OID_Dep THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN Profesores.DNI%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumProfesores NUMBER := 0;
BEGIN
    DELETE FROM Profesores WHERE DNI = v_DNI;
    SELECT COUNT(*) INTO v_NumProfesores FROM Profesores WHERE DNI = v_DNI;
    IF v_NumProfesores != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));

```

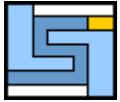


```
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Eliminar;
/
--Tabla ProfesoresImpartenAsignaturas
CREATE OR REPLACE PACKAGE BODY PCK_ProfesoresImpartenAsig
IS
  CURSOR C IS
    SELECT * FROM ProfesoresImpartenAsignaturas;
  v_Salida BOOLEAN := TRUE;
  v_ProfesoresAsig ProfesoresImpartenAsignaturas%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
  DELETE FROM ProfesoresImpartenAsignaturas;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
  OPEN C;
  FETCH C INTO v_ProfesoresAsig;
  DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('Codigo_Asig:', 25) || RPAD('Dedicacion:', 25));
  DBMS_OUTPUT.PUT_LINE(LPAD('-', 75, '-'));
  WHILE C%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(v_ProfesoresAsig.DNI, 25) || RPAD(v_ProfesoresAsig.Codigo_Asig, 25) || RPAD(v_ProfesoresAsig.Dedicacion, 25));
    FETCH C INTO v_ProfesoresAsig;
  END LOOP;
  CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenAsignaturas.DNI%TYPE, v_Codigo_Asig IN ProfesoresImpartenAsignaturas.Codigo_Asig%TYPE, v_Dedicacion IN ProfesoresImpartenAsignaturas.Dedicacion%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
  INSERT INTO ProfesoresImpartenAsignaturas (DNI, Código_Asig, Dedicacion) VALUES (v_DNI, v_Código_Asig, v_Dedicacion);
  SELECT * INTO v_ProfesoresAsig FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI AND Código_Asig = v_Código_Asig;
  IF v_ProfesoresAsig.Dedicacion != v_Dedicacion THEN
    v_Salida := FALSE;
  END IF;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
    ROLLBACK;
END Insertar;
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenAsignaturas.DNI%TYPE, v_Codigo_Asig IN ProfesoresImpartenAsignaturas.Codigo_Asig%TYPE, v_Dedicacion IN ProfesoresImpartenAsignaturas.Dedicacion%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
  UPDATE ProfesoresImpartenAsignaturas SET Dedicacion = v_Dedicacion WHERE DNI = v_DNI AND Código_Asig = v_Código_Asig;
  SELECT * INTO v_ProfesoresAsig FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI AND Código_Asig = v_Código_Asig;
  IF v_ProfesoresAsig.Dedicacion != v_Dedicacion THEN
    v_Salida := FALSE;
  END IF;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
  WHEN OTHERS THEN
```



```
DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenAsignaturas.DNI%TYPE, v_Codigo_Asig IN ProfesoresImpartenAsignaturas.Codigo_Asig%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumProfesoresImpartenAsig NUMBER := 0;
BEGIN
    DELETE FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI AND Codigo_Asig = v_Codigo_Asig;
    SELECT COUNT(*) INTO v_NumProfesoresImpartenAsig FROM ProfesoresImpartenAsignaturas WHERE DNI = v_DNI AND Codigo_Asig = v_Codigo_Asig;
    IF v_NumProfesoresImpartenAsig != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Eliminar;
END;
/

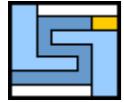
--Tabla ProfesoresImpartenEnGrupos
CREATE OR REPLACE PACKAGE BODY PCK_ProfesoresImpartenEnGrupos
IS
    CURSOR C IS
        SELECT * FROM ProfesoresImpartenEnGrupos;
    v_Salida BOOLEAN := TRUE;
    v_ProfesoresImpartenEnGrupos ProfesoresImpartenEnGrupos%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM ProfesoresImpartenEnGrupos;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_ProfesoresImpartenEnGrupos;
    DBMS_OUTPUT.PUT_LINE(RPAD('DNI:', 25) || RPAD('OID_Grup:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 50, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_ProfesoresImpartenEnGrupos.DNI, 25) || RPAD(v_ProfesoresImpartenEnGrupos.OID_Grup, 25));
        FETCH C INTO v_ProfesoresImpartenEnGrupos;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenEnGrupos.DNI%TYPE, v_OID_Grup IN ProfesoresImpartenEnGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    INSERT INTO ProfesoresImpartenEnGrupos (DNI, OID_Grup) VALUES (v_DNI, v_OID_Grup);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
    END Insertar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_DNI IN ProfesoresImpartenEnGrupos.DNI%TYPE, v_OID_Grup IN ProfesoresImpartenEnGrupos.OID_Grup%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumProfImpartenEnGrupos NUMBER := 0;
```



```
BEGIN
    DELETE FROM ProfesoresImpartenEnGrupos WHERE DNI = v_DNI AND OID_Grup = v_OID_Grup;
    SELECT COUNT(*) INTO v_NumProfImpartenEnGrupos FROM ProfesoresImpartenEnGrupos WHERE DNI = v_DNI AND OID_Grup = v_OID_Grup;
    IF v_NumProfImpartenEnGrupos != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
/
--Tabla Tutorias
CREATE OR REPLACE PACKAGE BODY PCK_Tutorias
IS
    CURSOR C IS
        SELECT * FROM Tutorias;
    v_Salida BOOLEAN := TRUE;
    v_Tutorias Tutorias%ROWTYPE;
PROCEDURE Inicializar
IS
BEGIN
    DELETE FROM Tutorias;
END Inicializar;
PROCEDURE Consultar
IS
BEGIN
    OPEN C;
    FETCH C INTO v_Tutorias;
    DBMS_OUTPUT.PUT_LINE(RPAD('OID_T:', 25) || RPAD('Fecha:', 25) || RPAD('Hora_Comienzo:', 25) || RPAD('Duracion:', 25) || RPAD('DNI_Alum:', 25) || RPAD('DNI_Prof:', 25));
    DBMS_OUTPUT.PUT_LINE(LPAD('-', 150, '-'));
    WHILE C%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_Tutorias.OID_T, 25) || RPAD(v_Tutorias.Fecha, 25) || RPAD(v_Tutorias.Hora_Comienzo, 25) || RPAD(v_Tutorias.Duracion, 25) ||
        RPAD(v_Tutorias.DNI_Alum, 25) || RPAD(v_Tutorias.DNI_Prof, 25));
        FETCH C INTO v_Tutorias;
    END LOOP;
    CLOSE C;
END Consultar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_Fecha IN Tutorias.Fecha%TYPE, v_Hora_Comienzo IN Tutorias.Hora_Comienzo%TYPE, v_Duracion IN Tutorias.Duracion%TYPE, v_DNI_Alum IN
Tutorias.DNI_Alum%TYPE, v_DNI_Prof IN Tutorias.DNI_Prof%TYPE, salidaEsperada BOOLEAN)
IS
    v_OID_T Tutorias.OID_T%TYPE;
BEGIN
    INSERT INTO Tutorias (Fecha, Hora_Comienzo, Duracion, DNI_Alum, DNI_Prof) VALUES (v_Fecha, v_Hora_Comienzo, v_Duracion, v_DNI_Alum, v_DNI_Prof);
    v_OID_T := SEC_Tutorias.CURRVAL;
    SELECT * INTO v_Tutorias FROM Tutorias WHERE OID_T = v_OID_T;
    IF v_Tutorias.Fecha != v_Fecha AND v_Tutorias.Hora_Comienzo != v_Hora_Comienzo AND v_Tutorias.Duracion != v_Duracion AND v_Tutorias.DNI_Alum != v_DNI_Alum AND
    v_Tutorias.DNI_Prof != v_DNI_Prof THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ':' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Insertar;
```



```
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_T IN Tutorias.OID_T%TYPE, v_Fecha IN Tutorias.Fecha%TYPE, v_Hora_Comienzo IN Tutorias.Hora_Comienzo%TYPE, v_Duracion IN Tutorias.Duracion%TYPE, v_DNI_Alum IN Tutorias.DNI_Alum%TYPE, v_DNI_Prof IN Tutorias.DNI_Prof%TYPE, salidaEsperada BOOLEAN)
IS
BEGIN
    UPDATE Tutorias SET Fecha = v_Fecha, Hora_Comienzo = v_Hora_Comienzo, Duracion = v_Duracion, DNI_Alum = v_DNI_Alum, DNI_Prof = v_DNI_Prof WHERE OID_T = v_OID_T;
    SELECT * INTO v_Tutorias FROM Tutorias WHERE OID_T = v_OID_T;
    IF v_Tutorias.Fecha != v_Fecha AND v_Tutorias.Hora_Comienzo != v_Hora_Comienzo AND v_Tutorias.Duracion != v_Duracion AND v_Tutorias.DNI_Alum != v_DNI_Alum AND v_Tutorias.DNI_Prof != v_DNI_Prof THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Actualizar;
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_T IN Tutorias.OID_T%TYPE, salidaEsperada BOOLEAN)
IS
    v_NumTutorias NUMBER := 0;
BEGIN
    DELETE FROM Tutorias WHERE OID_T = v_OID_T;
    SELECT COUNT(*) INTO v_NumTutorias FROM Tutorias WHERE OID_T = v_OID_T;
    IF v_NumTutorias != 0 THEN
        v_Salida := FALSE;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida, salidaEsperada));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE, salidaEsperada));
        ROLLBACK;
END Eliminar;
END;
/
```



Pruebas de paquetes



```
SET SERVEROUTPUT ON;

--Tabla Alumnos
BEGIN
PCK_Alumnos.Inicializar;

PCK_Alumnos.Insertar ('Insertar un alumno', '77843402V', 'Rubén', 'Bueno Menéndez', TO_DATE('25/08/1991', 'DD/MM/YYYY'), 'rubenbm@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '93839721C', 'Alfredo', 'Gomez Reyes', TO_DATE('18/03/1988', 'DD/MM/YYYY'), 'alfredogr@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '07842998K', 'lvaro', 'Alonso Agenjo', TO_DATE('17/10/1987', 'DD/MM/YYYY'), 'alvaroaa@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '00548678J', 'Diego', 'Barbero Cano', TO_DATE('02/11/1985', 'DD/MM/YYYY'), 'diegobc@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '35849097D', 'lvaro', 'Cano García', TO_DATE('30/07/1995', 'DD/MM/YYYY'), 'alvarocg@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '98523686N', 'Natalia', 'Cereza Arias', TO_DATE('17/06/1992', 'DD/MM/YYYY'), 'nataliaca@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '07804804F', 'Claudia', 'Frutos Vidal', TO_DATE('13/03/1993', 'DD/MM/YYYY'), 'claudiafv@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '26810393Y', 'María', 'Sanchez Muñoz', TO_DATE('10/02/1989', 'DD/MM/YYYY'), 'mariasm@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '64427553E', 'Alicia', 'Heredia Gomez', TO_DATE('15/10/1985', 'DD/MM/YYYY'), 'aliciahg@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '42312153L', 'Ainhoa', 'Santiago Cid', TO_DATE('23/05/1990', 'DD/MM/YYYY'), 'anhoasc@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '77008982Z', 'Naila', 'Muñoz Ruiz', TO_DATE('30/05/1992', 'DD/MM/YYYY'), 'nailamr@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '27576410P', 'Lucia', 'Perez Fuertes', TO_DATE('01/11/1992', 'DD/MM/YYYY'), 'luciapf@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '70964804Y', 'Natalia', 'Ruiz Camarena', TO_DATE('13/07/1993', 'DD/MM/YYYY'), 'nataliarc@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '67202652X', 'Roberto', 'Bravo Ballesteros', TO_DATE('23/03/1989', 'DD/MM/YYYY'), 'robertobb@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '63441519K', 'María', 'Reyes Méndez', TO_DATE('25/09/1987', 'DD/MM/YYYY'), 'mariaym@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '62968537B', 'Iván', 'Agustín Navarro', TO_DATE('05/02/1988', 'DD/MM/YYYY'), 'ivanan@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '23801240N', 'Paula', 'Baena Blázquez', TO_DATE('08/02/1985', 'DD/MM/YYYY'), 'paulabb@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '47966225R', 'Sergio', 'Caballero Cabeza', TO_DATE('28/10/1991', 'DD/MM/YYYY'), 'sergiocc@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '56336094D', 'Juan Antonio', 'Lago Medina', TO_DATE('03/02/1989', 'DD/MM/YYYY'), 'juanantoniolm@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno', '36081385C', 'Sebastián', 'Páez Vaquero', TO_DATE('22/01/1995', 'DD/MM/YYYY'), 'sebastianpv@alum.es', TRUE);
PCK_Alumnos.Insertar ('Insertar un alumno con DNI NULL', NULL, 'Juan Antonio', 'Giralda Torres', TO_DATE('04/09/1995', 'DD/MM/YYYY'), 'juanantoniogt@alum.es', FALSE);
PCK_Alumnos.Insertar ('Insertar un alumno violando PK', '77843402V', 'Juan Antonio', 'Giralda Torres', TO_DATE('04/09/1995', 'DD/MM/YYYY'), 'juanantoniogt@alum.es', FALSE);
PCK_Alumnos.Insertar ('Insertar un alumno violando CK_DNI_Alumnos', '563360949', 'Rubén', 'Bueno Menéndez', TO_DATE('25/08/1991', 'DD/MM/YYYY'), 'rubenbm@alum.es', FALSE);
PCK_Alumnos.Insertar ('Insertar un alumno incumpliendo la RN-001', '77843402V', 'Rubén', 'Bueno Menéndez', TO_DATE('25/08/1991', 'DD/MM/YYYY'), 'rubenbm@hotmail.es', FALSE);
PCK_Alumnos.Actualizar ('Actualizar un alumno', '77843402V', 'Rubén', 'Bueno Mendez', TO_DATE('25/08/1991', 'DD/MM/YYYY'), 'rubenbm@alum.es', TRUE);
PCK_Alumnos.Actualizar ('Actualizar un alumno incumpliendo la RN-001', '77843402V', 'Rubén', 'Bueno Menéndez', TO_DATE('25/08/1991', 'DD/MM/YYYY'), 'rubenbm@hotmail.es', FALSE);
PCK_Alumnos.Actualizar ('Actualizar un alumno inexistente', '99694117L', 'Susana', 'Giralda Torres', TO_DATE('04/09/1995', 'DD/MM/YYYY'), 'susanaagt@alum.es', FALSE);
PCK_Alumnos.Eliminar ('Eliminar un alumno', '77843402V', TRUE);
PCK_Alumnos.Consultar;
END;
/
```

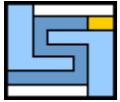


```
--Tabla Aulas
BEGIN
PCK_Aulas.Inicializar;
PCK_Aulas.Insertar ('Insertar un aula', 'A0.01', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A0.02', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A0.03', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A0.04', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A1.01', 50, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A1.02', 50, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'H0.01', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'H0.02', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'H0.03', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'H0.04', 70, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'H1.01', 50, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'H1.02', 50, 'Teoría, TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'F1.01', 30, 'Laboratorio', TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'F1.02', 30, 'Laboratorio', TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'G1.01', 10, 'Laboratorio', TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A3.01', 150, 'Examen', TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A3.02', 150, 'Examen', TRUE);
PCK_Aulas.Insertar ('Insertar un aula', 'A4.01', 200, 'Examen', TRUE);
PCK_Aulas.Insertar ('Insertar un aula con Nombre NULL', NULL, 70, 'Teoría, FALSE);
PCK_Aulas.Insertar ('Insertar un aula violando AK', 'A0.01', 70, 'Teoría, FALSE);
PCK_Aulas.Insertar ('Insertar un aula violando CK_Tipo_Aulas', 'F2.01', 70, 'Oficina', FALSE);
PCK_Aulas.Insertar ('Insertar un aula incumpliendo la RN-006', 'A0.05', 130, 'Teoría, FALSE);
PCK_Aulas.Insertar ('Insertar un aula incumpliendo la RN-006', 'F0.06', 50, 'Laboratorio', FALSE);
PCK_Aulas.Insertar ('Insertar un aula incumpliendo la RN-006', 'A3.07', 250, 'Examen', FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula', 1, 'A0.07', 70, 'Teoría, TRUE);
PCK_Aulas.Actualizar ('Actualizar un aula con Nombre NULL', 1, NULL, 70, 'Teoría, FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula violando AK', 1, 'A0.03', 70, 'Teoría, FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula violando CK_Tipo_Aulas', 1, 'F2.01', 70, 'Oficina', FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula incumpliendo la RN-006', 1, 'A0.05', 130, 'Teoría, FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula incumpliendo la RN-006', 1, 'F0.06', 50, 'Laboratorio', FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula incumpliendo la RN-006', 1, 'A3.07', 250, 'Examen', FALSE);
PCK_Aulas.Actualizar ('Actualizar un aula inexistente', 666, 'A3.07', 150, 'Examen', FALSE);
PCK_Aulas.Eliminar ('Eliminar un aula', 2, TRUE);
PCK_Aulas.Consultar;
END;
/
```



```
--Tabla Becas
BEGIN
PCK_Becas.Inicializar;
PCK_Becas.Insertar ('Insertar una beca', '61700LIW', 2000, 80, 'Ordinaria', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '28194XAM', 1500, 810, 'Ordinaria', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '18537TZN', 2400, 785, 'Ordinaria', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '91440SRA', 1600, 630, 'Ordinaria', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '84676UFX', 3680, 720, 'Movilidad', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '93073MJH', 2400, 380, 'Movilidad', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '13617EVK', 1500, 670, 'Movilidad', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '13617BAU', 6400, 890, 'Empresa', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '07284VPY', 2000, 670, 'Empresa', TRUE);
PCK_Becas.Insertar ('Insertar una beca', '38720AJG', 2500, 540, 'Empresa', TRUE);
PCK_Becas.Insertar ('Insertar una beca con Codigo_Beca NULL', NULL, 1500, 500, 'Ordinaria', FALSE);
PCK_Becas.Insertar ('Insertar una beca violando PK', '61700LIW', 1500, 500, 'Ordinaria', FALSE);
PCK_Becas.Insertar ('Insertar una beca violando CK_Codigo_Becas', '6938532T', 1500, 500, 'Ordinaria', FALSE);
PCK_Becas.Insertar ('Insertar una beca violando CK_Tipo_Becas', '69385YOS', 1500, 500, 'Extra', FALSE);
PCK_Becas.Insertar ('Insertar una beca incumpliendo la RN-004', '69385YOS', 1000, 300, 'Ordinaria', FALSE);
PCK_Becas.Actualizar ('Actualizar una beca', '61700LIW', 2500, 400, 'Movilidad', TRUE);
PCK_Becas.Actualizar ('Actualizar una beca violando CK_Tipo_Becas', '61700LIW', 1500, 500, 'Extra', FALSE);
PCK_Becas.Actualizar ('Actualizar una beca incumpliendo la RN-004', '61700LIW', 1000, 300, 'Ordinaria', FALSE);
PCK_Becas.Actualizar ('Actualizar una beca inexistente', '95025YFA', 2500, 400, 'Movilidad', FALSE);
PCK_Becas.Eliminar ('Eliminar una beca', '28194XAM', TRUE);
PCK_Becas.Consultar;
END;
/

--Tabla Departamentos
BEGIN
PCK_Departamentos.Inicializar;
PCK_Departamentos.Insertar ('Insertar un departamento', 'Física Aplicada I', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Electrónica y Electromagnetismo', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Ciencias de la Comput. e Int. Artificial', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Arquitectura y Tecnolog. de Computadores', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Ingeniería Aeroespacial y Mecánica de Fluidos', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Ingeniería de Sistemas y Automática', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Lenguajes y Sistemas Informáticos', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Matemática Aplicada I', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento', 'Tecnología Electrónica', TRUE);
PCK_Departamentos.Insertar ('Insertar un departamento con Nombre NULL', NULL, FALSE);
PCK_Departamentos.Insertar ('Insertar un departamento violando AK', 'Lenguajes y Sistemas Informáticos', FALSE);
PCK_Departamentos.Actualizar ('Actualizar un departamento', 1, 'Biología Celular', TRUE);
PCK_Departamentos.Actualizar ('Actualizar un departamento con Nombre NULL', 1, NULL, FALSE);
PCK_Departamentos.Actualizar ('Actualizar un departamento violando AK', 1, 'Tecnología Electrónica', FALSE);
PCK_Departamentos.Actualizar ('Actualizar un departamento inexistente', 666, 'Biología Celular', FALSE);
PCK_Departamentos.Eliminar ('Eliminar un departamento', 20, TRUE);
PCK_Departamentos.Consultar;
END;
/
```



```
--Tabla Despachos
BEGIN
PCK_Despachos.Inicializar;
PCK_Despachos.Insertar ('Insertar un despacho', 'F0.20', 2, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F0.21', 2, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F0.22', 2, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F0.23', 2, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F1.20', 3, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F1.21', 3, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F1.22', 3, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'F1.23', 3, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'G0.20', 1, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'G0.21', 1, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho', 'G1.20', 2, TRUE);
PCK_Despachos.Insertar ('Insertar un despacho con Nombre NULL', NULL, 2, FALSE);
PCK_Despachos.Insertar ('Insertar un despacho violando AK', 'F0.20', 2, FALSE);
PCK_Despachos.Insertar ('Insertar un despacho incumpliendo la RN-005', 'F0.25', 5, FALSE);
PCK_Despachos.Actualizar ('Actualizar un despacho', 1, 'F0.25', 2, TRUE);
PCK_Despachos.Actualizar ('Actualizar un despacho con Nombre NULL', 1, NULL, 2, FALSE);
PCK_Despachos.Actualizar ('Actualizar un despacho violando AK', 'F1.21', NULL, 3, FALSE);
PCK_Despachos.Actualizar ('Actualizar un despacho incumpliendo la RN-005', 1, 'F0.30', 5, FALSE);
PCK_Despachos.Actualizar ('Actualizar un despacho inexistente', 666, 'F0.30', 2, FALSE);
PCK_Despachos.Eliminar ('Eliminar un despacho', 11, TRUE);
PCK_Despachos.Consultar;
END;
/
```

```
--Tabla Grados
BEGIN
PCK_Grados.Inicializar;
PCK_Grados.Insertar ('Insertar un grado', 'Ingeniería Informática - Ingeniería de Computadores', TRUE);
PCK_Grados.Insertar ('Insertar un grado', 'Ingeniería Informática - Ingeniería del Software', TRUE);
PCK_Grados.Insertar ('Insertar un grado', 'Ingeniería Informática - Tecnologías Informáticas', TRUE);
PCK_Grados.Insertar ('Insertar un grado', 'Ingeniería de la Salud', TRUE);
PCK_Grados.Insertar ('Insertar un grado con Nombre NULL', NULL, FALSE);
PCK_Grados.Insertar ('Insertar un grado violando AK', 'Ingeniería Informática - Ingeniería del Software', FALSE);
PCK_Grados.Actualizar ('Actualizar un grado', 1, 'Matemáticas', TRUE);
PCK_Grados.Actualizar ('Actualizar un grado con Nombre NULL', 1, NULL, FALSE);
PCK_Grados.Actualizar ('Actualizar un grado violando AK', 1, 'Ingeniería Informática - Tecnologías Informáticas', FALSE);
PCK_Grados.Actualizar ('Actualizar un grado inexistente', 666, 'Ingeniería Informática - Tecnologías Informáticas', FALSE);
PCK_Grados.Eliminar ('Eliminar un grado', 2, TRUE);
PCK_Grados.Consultar;
END;
/
```



```
--Tabla Grupos
BEGIN
PCK_Grupos.Inicializar;
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 1', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 2', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 3', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 4', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 5', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 6', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 7', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 8', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 9', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 10', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 11', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 12', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 13', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 14', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 15', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo', 'Grupo 16', TRUE);
PCK_Grupos.Insertar ('Insertar un grupo con Nombre NULL', NULL, FALSE);
PCK_Grupos.Insertar ('Insertar un grupo violando AK', 'Grupo 2', FALSE);
PCK_Grupos.Actualizar ('Actualizar un grupo', 1, 'Grupo 17', TRUE);
PCK_Grupos.Actualizar ('Actualizar un grupo con Nombre NULL', 1, NULL, FALSE);
PCK_Grupos.Actualizar ('Actualizar un grupo violando AK', 1, 'Grupo 2', FALSE);
PCK_Grupos.Actualizar ('Actualizar un grupo inexistente', 666, 'Grupo 2', FALSE);
PCK_Grupos.Eliminar ('Eliminar un grupo', 2, TRUE);
PCK_Grupos.Consultar;
END;
/
```

```
--Tabla AlumnosPertenecenAGrupos
BEGIN
PCK_AlumnosPertenecenAGrupos.Inicializar;
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '42312153L', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '42312153L', 4, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '93839721C', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '07842998K', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '07842998K', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '00548678J', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '35849097D', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '98523686N', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '98523686N', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '07804804F', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '26810393Y', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '26810393Y', 3, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '64427553E', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '36081385C', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '56336094D', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '47966225R', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '23801240N', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '63441519K', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '67202652X', 8, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '70964804Y', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '27576410P', 1, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '77008982Z', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '63441519K', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '93839721C', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '56336094D', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '64427553E', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '35849097D', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '77008982Z', 7, TRUE);
```



```
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '27576410P', 7, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '42312153L', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '47966225R', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '27576410P', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '36081385C', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '26810393Y', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo', '00548678J', 5, TRUE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo con DNI NULL', NULL, 6, FALSE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo con OID_Grup NULL', '67202652X', NULL, FALSE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo violando PK', '42312153L', 1, FALSE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo violando FK de alumnos', '12345678P', 1, FALSE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo violando FK de grupos', '42312153L', 5656, FALSE);
PCK_AlumnosPertenecenAGrupos.Insertar ('Insertar un alumno perteneciente a un grupo violando CK_DNI_AlumPertAGrup', '778434028', 1, FALSE);
PCK_AlumnosPertenecenAGrupos.Eliminar ('Eliminar un alumno perteneciente a un grupo', '42312153L', 4, TRUE);
PCK_AlumnosPertenecenAGrupos.Consultar;
END;
/

--Tabla Asignaturas
BEGIN
PCK_Asignaturas.Inicializar;
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050004, 'Fundamentos Físicos de la Informática', 'FFI', 6, 'Primer cuatrimestre', 1, 'Troncal', 1, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050040, 'Integración de Sistemas Físicos e Informáticos', 'ISFI', 6, 'Primer cuatrimestre', 4, 'Optativo', 1, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050029, 'Aplicaciones de Soft Computing', 'ASC', 6, 'Segundo cuatrimestre', 4, 'Optativo', 2, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050012, 'Lenguaje Informática', 'LI', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 3, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050024, 'Inteligencia Artificial', 'IA', 6, 'Primer cuatrimestre', 3, 'Obligatorio', 3, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050015, 'Arquitectura de Computadores', 'AC', 6, 'Segundo cuatrimestre', 2, 'Obligatorio', 4, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050001, 'Fundamentos de Programación', 'FP', 12, 'Anual', 1, 'Troncal', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050010, 'Análisis y Diseño de Datos y Algoritmos', 'ADDA', 12, 'Anual', 2, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050011, 'Introducción a la Ingeniería del Software y los Sistemas de Información', 'IISSI', 12, 'Anual', 2, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050014, 'Sistemas Operativos', 'SO', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050016, 'Arquitectura e Integración de Sistemas Software', 'AISS', 6, 'Segundo cuatrimestre', 2, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050018, 'Diseño Pruebas', 'DP', 12, 'Anual', 3, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050019, 'Proceso Software y Gestión', 'PSG', 12, 'Anual', 3, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050020, 'Ingeniería de Requisitos', 'IR', 6, 'Primer cuatrimestre', 3, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050026, 'Prácticas Externas', 'PE', 6, 'Segundo cuatrimestre', 4, 'Optativo', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050027, 'Acceso Inteligente a la Información', 'AII', 6, 'Primer cuatrimestre', 4, 'Optativo', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050032, 'Evolución Gestión de la Configuración', 'EGC', 6, 'Primer cuatrimestre', 4, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050035, 'Planificación Gestión de Proyectos Informáticos', 'PGPI', 6, 'Segundo cuatrimestre', 4, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050039, 'Ingeniería del Software y Práctica Profesional', 'ISPP', 6, 'Primer cuatrimestre', 4, 'Obligatorio', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050043, 'Seguridad en Sistemas Informáticos y en Internet', 'SSII', 6, 'Segundo cuatrimestre', 4, 'Optativo', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050007, 'Álgebra Lineal y Numérica', 'ALN', 6, 'Segundo cuatrimestre', 1, 'Troncal', 8, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050017, 'Matemática Discreta', 'MD', 6, 'Segundo cuatrimestre', 2, 'Obligatorio', 8, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050002, 'Cálculo Infinitesimal y Numérica', 'CIN', 6, 'Primer cuatrimestre', 1, 'Troncal', 8, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050003, 'Circuitos Electrónicos Digitales', 'CED', 6, 'Primer cuatrimestre', 1, 'Troncal', 9, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050009, 'Estructura de Computadores', 'EDC', 6, 'Segundo cuatrimestre', 1, 'Troncal', 9, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050013, 'Redes de Computadores', 'RC', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 9, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura', 2050041, 'Optimización de Sistemas', 'OS', 6, 'Primer cuatrimestre', 4, 'Optativo', 7, TRUE);
PCK_Asignaturas.Insertar ('Insertar una asignatura con Código_Asig NULL', NULL, 'Redes de Computadores', 'RC', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 9, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura con Nombre NULL', 2050045, NULL, 'RC', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 9, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura violando PK', 2050004, 'Teledeteción', 'T', 6, 'Primer cuatrimestre', 4, 'Optativo', 9, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura violando AK', 2050041, 'Fundamentos Físicos de la Informática', 'FFI', 6, 'Primer cuatrimestre', 1, 'Troncal', 1, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura violando FK de departamentos', 2050045, 'Trabajo Fin de Grado', 'TFG', 12, 'Anual', 4, 'Obligatorio', 33, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura violando CK_Cod_Asig_Asignaturas', 20588, 'Modelado y Simulación', 'MSN', 6, 'Primer cuatrimestre', 3, 'Obligatorio', 8, FALSE);
```



```
PCK_Asignaturas.Insertar ('Insertar una asignatura violando CK_Periodo_Asignaturas', 2050021, 'Modelado y Simulación numérica', 'MSN', 6, 'Indefinido', 3, 'Obligatorio', 8, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura violando CK_Curso_Asignaturas', 2050021, 'Modelado y Simulación numérica', 'MSN', 6, 'Primer cuatrimestre', 7, 'Obligatorio', 8, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura violando CK_Tipo_Asignaturas', 2050021, 'Modelado y Simulación numérica', 'MSN', 6, 'Primer cuatrimestre', 3, 'Opcional', 8, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura incumpliendo la RN-002', 2050021, 'Modelado y Simulación numérica', 'MSN', 0, 'Primer cuatrimestre', 3, 'Optativo', 8, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura incumpliendo la RN-008', 2050021, 'Modelado y Simulación numérica', 'MSN', 12, 'Primer cuatrimestre', 3, 'Optativo', 8, FALSE);
PCK_Asignaturas.Insertar ('Insertar una asignatura incumpliendo la RN-009', 2050021, 'Modelado y Simulación numérica', 'MSN', 6, 'Anual', 3, 'Optativo', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura', 2050013, 'Redes de Computación', 'RC', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 9, TRUE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura con Nombre NULL', 2050013, NULL, 'RC', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 9, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura violando AK', 2050013, 'Fundamentos Físicos de la Informática', 'FFI', 6, 'Primer cuatrimestre', 1, 'Troncal', 1, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura violando FK de departamentos', 2050013, 'Trabajo Fin de Grado', 'TFG', 12, 'Anual', 4, 'Obligatorio', 33, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura violando CK_Periodo_Asignaturas', 2050013, 'Modelado y Simulación numérica', 'MSN', 6, 'Indefinido', 3, 'Obligatorio', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura violando CK_Curso_Asignaturas', 2050013, 'Modelado y Simulación numérica', 'MSN', 6, 'Primer cuatrimestre', 7, 'Obligatorio', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura violando CK_Tipo_Asignaturas', 2050013, 'Modelado y Simulación numérica', 'MSN', 6, 'Primer cuatrimestre', 3, 'Opcional', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura incumpliendo la RN-002', 2050013, 'Modelado y Simulación numérica', 'MSN', 0, 'Primer cuatrimestre', 3, 'Optativo', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura incumpliendo la RN-008', 2050013, 'Modelado y Simulación numérica', 'MSN', 12, 'Primer cuatrimestre', 3, 'Optativo', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura incumpliendo la RN-009', 2050013, 'Modelado y Simulación numérica', 'MSN', 6, 'Anual', 3, 'Optativo', 8, FALSE);
PCK_Asignaturas.Actualizar ('Actualizar una asignatura inexistente', 2999999, 'Redes de Computación', 'RC', 6, 'Primer cuatrimestre', 2, 'Obligatorio', 9, FALSE);
PCK_Asignaturas.Eliminar ('Eliminar una asignatura', 2050013, TRUE);
PCK_Asignaturas.Consultar;
END;
/

--Tabla Expedientes
BEGIN
PCK_Expedientes.Inicializar;
PCK_Expedientes.Insertar ('Insertar un expediente', 84656361, '93839721C', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 94814291, '07842998K', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 72715608, '27576410P', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 24992812, '00548678J', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 66217035, '56336094D', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 67367181, '64427553E', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 42079230, '62968537B', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 30004413, '42312153L', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 85838415, '67202652X', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 19955161, '77008982Z', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 95326106, '00548678J', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 64586570, '70964804Y', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 44748275, '63441519K', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 67032771, '70964804Y', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 78208070, '23801240N', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 53909587, '23801240N', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 12226352, '56336094D', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 78002608, '27576410P', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 17361731, '36081385C', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 61654321, '62968537B', TRUE);
```



```
PCK_Expedientes.Insertar ('Insertar un expediente', 83665705, '93839721C', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 32407767, '36081385C', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente', 42794228, '67202652X', TRUE);
PCK_Expedientes.Insertar ('Insertar un expediente con Codigo_Exp NULL', NULL, '47966225R', FALSE);
PCK_Expedientes.Insertar ('Insertar un expediente con DNI NULL', 14833197, NULL, FALSE);
PCK_Expedientes.Insertar ('Insertar un expediente violando PK', 84656361, '47966225R', FALSE);
PCK_Expedientes.Insertar ('Insertar un expediente violando FK de alumnos', 14833197, '77777797R', FALSE);
PCK_Expedientes.Insertar ('Insertar un expediente violando CK_Codigo_Expedientes', 7468, '47966225R', FALSE);
PCK_Expedientes.Insertar ('Insertar un expediente violando CK_DNI_Expedientes', 14833197, '4795TS625Y', FALSE);
PCK_Expedientes.Actualizar ('Actualizar un expediente', 84656361, '27576410P', TRUE);
PCK_Expedientes.Actualizar ('Actualizar un expediente con DNI NULL', 84656361, NULL, FALSE);
PCK_Expedientes.Actualizar ('Actualizar un expediente violando FK de alumnos', 84656361, '77777797R', FALSE);
PCK_Expedientes.Actualizar ('Actualizar un expediente violando CK_DNI_Expedientes', 84656361, '4795TS625Y', FALSE);
PCK_Expedientes.Actualizar ('Actualizar un expediente inexistente', 99999999, '4795TS625Y', FALSE);
PCK_Expedientes.Eliminar ('Eliminar un expediente', 94814291, TRUE);
PCK_Expedientes.Consultar;
END;
/

--Tabla Notas
BEGIN
PCK_Notas.Inicializar;
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('5,4'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('6,5'), 'Aprobado', 'Segunda', 2, TO_DATE('01/09/2017', 'DD/MM/YYYY'), 42794228, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('4,5'), 'Suspens', 'Tercera', 3, TO_DATE('03/12/2017', 'DD/MM/YYYY'), 32407767, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('8,2'), 'Notable', 'Segunda', 4, TO_DATE('01/09/2017', 'DD/MM/YYYY'), 83665705, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('8,9'), 'Notable', 'Primera', 3, TO_DATE('24/01/2017', 'DD/MM/YYYY'), 83665705, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('1,6'), 'Suspens', 'Tercera', 1, TO_DATE('02/12/2017', 'DD/MM/YYYY'), 61654321, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('1'), 'Suspens', 'Segunda', 1, TO_DATE('02/09/2017', 'DD/MM/YYYY'), 17361731, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('6,3'), 'Aprobado', 'Tercera', 1, TO_DATE('03/12/2017', 'DD/MM/YYYY'), 78002608, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('2'), 'Suspens', 'Primera', 2, TO_DATE('23/01/2017', 'DD/MM/YYYY'), 12226352, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('9,5'), 'Sobresaliente', 'Tercera', 3, TO_DATE('04/12/2017', 'DD/MM/YYYY'), 53909587, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('9'), 'Sobresaliente', 'Segunda', 1, TO_DATE('07/09/2017', 'DD/MM/YYYY'), 78208070, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('10'), 'Matrícula de honor', 'Segunda', 3, TO_DATE('07/09/2017', 'DD/MM/YYYY'), 44748275, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('6,2'), 'Aprobado', 'Primera', 2, TO_DATE('22/01/2017', 'DD/MM/YYYY'), 95326106, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('0'), 'Suspens', 'Segunda', 1, TO_DATE('05/09/2017', 'DD/MM/YYYY'), 64586570, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('5,5'), 'Aprobado', 'Tercera', 2, TO_DATE('05/12/2017', 'DD/MM/YYYY'), 64586570, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('3,5'), 'Suspens', 'Primera', 3, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 95326106, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('9,1'), 'Sobresaliente', 'Segunda', 1, TO_DATE('04/09/2017', 'DD/MM/YYYY'), 19955161, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('6,6'), 'Aprobado', 'Primera', 1, TO_DATE('28/01/2017', 'DD/MM/YYYY'), 42079230, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('8,2'), 'Notable', 'Segunda', 4, TO_DATE('06/09/2017', 'DD/MM/YYYY'), 85838415, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('10'), 'Matrícula de honor', 'Primera', 4, TO_DATE('29/01/2017', 'DD/MM/YYYY'), 66217035, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('9,8'), 'Sobresaliente', 'Tercera', 2, TO_DATE('06/12/2017', 'DD/MM/YYYY'), 66217035, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('2,5'), 'Suspens', 'Segunda', 3, TO_DATE('07/09/2017', 'DD/MM/YYYY'), 67032771, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('9,4'), 'Sobresaliente', 'Primera', 2, TO_DATE('30/01/2017', 'DD/MM/YYYY'), 72715608, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('5,9'), 'Aprobado', 'Segunda', 1, TO_DATE('01/09/2017', 'DD/MM/YYYY'), 84656361, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('9,2'), 'Sobresaliente', 'Tercera', 2, TO_DATE('06/12/2017', 'DD/MM/YYYY'), 61654321, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('7,4'), 'Notable', 'Segunda', 2, TO_DATE('07/09/2017', 'DD/MM/YYYY'), 12226352, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('10'), 'Matrícula de honor', 'Segunda', 1, TO_DATE('01/09/2017', 'DD/MM/YYYY'), 61654321, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('3'), 'Suspens', 'Segunda', 1, TO_DATE('06/09/2017', 'DD/MM/YYYY'), 61654321, TRUE);
PCK_Notas.Insertar ('Insertar una nota', TO_NUMBER('6,5'), 'Aprobado', 'Tercera', 2, TO_DATE('19/12/2017', 'DD/MM/YYYY'), 83665705, TRUE);
PCK_Notas.Insertar ('Insertar una nota con Codigo_Exp NULL', TO_NUMBER('5'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), NULL, FALSE);
PCK_Notas.Insertar ('Insertar una nota violando FK de expedientes', TO_NUMBER('5'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 99999999, FALSE);
PCK_Notas.Insertar ('Insertar una nota violando CK_Calificacion_Notas', TO_NUMBER('5'), 'Suficiente', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Insertar ('Insertar una nota violando CK_Convocatoria_Notas', TO_NUMBER('5'), 'Aprobado', 'Cuarto', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Insertar ('Insertar una nota violando CK_Curso_Notas', TO_NUMBER('5'), 'Aprobado', 'Primera', 7, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Insertar ('Insertar una nota incumpliendo la RN-011', TO_NUMBER('15'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Insertar ('Insertar una nota incumpliendo la RN-012', TO_NUMBER('5'), 'Sobresaliente', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota', 1, TO_NUMBER('6,5'), 'Aprobado', 'Tercera', 2, TO_DATE('19/12/2017', 'DD/MM/YYYY'), 83665705, TRUE);
PCK_Notas.Actualizar ('Actualizar una nota con Codigo_Exp NULL', 1, TO_NUMBER('5'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), NULL, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota violando FK de expedientes', 1, TO_NUMBER('5'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 99999999, FALSE);
```



```
PCK_Notas.Actualizar ('Actualizar una nota violando CK_Calificacion_Notas', 1, TO_NUMBER('5'), 'Suficiente', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota violando CK_Convocatoria_Notas', 1, TO_NUMBER('5'), 'Aprobado', 'Cuarta', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota violando CK_Curso_Notas', 1, TO_NUMBER('5'), 'Aprobado', 'Primera', 7, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota incumpliendo la RN-011', 1, TO_NUMBER('15'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota incumpliendo la RN-012', 1, TO_NUMBER('5'), 'Sobresaliente', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Actualizar ('Actualizar una nota inexistente', 666, TO_NUMBER('5'), 'Aprobado', 'Primera', 1, TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361, FALSE);
PCK_Notas.Eliminar ('Eliminar una nota', 2, TRUE);
PCK_Notas.Consultar;
END;
/

--Tabla AsignaturasPuntuadasConNotas
BEGIN
PCK_AsigPuntuadasConNotas.Inicializar;
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050004, 1, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050024, 4, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050018, 5, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050035, 14, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050043, 1, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050003, 16, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050014, 5, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050004, 6, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050010, 22, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050012, 21, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050001, 1, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050026, 6, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050004, 10, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050015, 5, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050027, 9, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050032, 7, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050039, 8, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050001, 7, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050035, 8, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050035, 20, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050002, 28, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050020, 6, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050043, 9, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota', 2050010, 27, TRUE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota con Codigo_Asig NULL', NULL, 9, FALSE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota con OID_N NULL', 2050043, NULL, FALSE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota violando PK', 2050004, 1, FALSE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota violando FK de asignaturas', 2999999, 1, FALSE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota violando FK de notas', 2050004, 666, FALSE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota violando CK_Cod_Asig_AsigPuntConNot', 20004, 4, FALSE);
PCK_AsigPuntuadasConNotas.Insertar ('Insertar una asignatura puntuada con una nota incumpliendo la RN-007', 2050010, 28, FALSE);
PCK_AsigPuntuadasConNotas.Eliminar ('Eliminar una asignatura puntuada con una nota', 2050004, 1, TRUE);
PCK_AsigPuntuadasConNotas.Consultar;
END;
/

--Tabla AsignaturasImpartidasEnAulas
BEGIN
PCK_AsigImpartidasEnAulas.Inicializar;
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050004, 1, TO_DATE('07/10/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050004, 7, TO_DATE('15/11/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050004, 3, TO_DATE('16/12/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050035, 5, TO_DATE('16/10/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050035, 6, TO_DATE('24/09/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050035, 2, TO_DATE('03/02/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050043, 1, TO_DATE('05/10/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050043, 9, TO_DATE('06/12/2016', 'DD/MM/YYYY'), TRUE);
```



```
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050043, 5, TO_DATE('07/05/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050015, 1, TO_DATE('07/04/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050015, 7, TO_DATE('20/01/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050002, 4, TO_DATE('26/11/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050002, 1, TO_DATE('30/03/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050001, 3, TO_DATE('23/02/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050001, 2, TO_DATE('07/10/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050012, 6, TO_DATE('02/04/2017', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula', 2050012, 1, TO_DATE('01/10/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula con Código_Asig NULL', NULL, 1, TO_DATE('01/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula con OID_A NULL', 2050012, NULL, TO_DATE('01/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula con Fecha NULL', 2050012, 1, NULL, FALSE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula violando PK', 2050004, 1, TO_DATE('07/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula violando FK de asignaturas', 2999999, 1, TO_DATE('07/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula violando FK de aulas', 2050004, 666, TO_DATE('07/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_AsigImpartidasEnAulas.Insertar ('Insertar una asignatura impartida en un aula violando CK_Cod_Asig_AsigImpartEnAulas', 2004, 1, TO_DATE('07/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_AsigImpartidasEnAulas.Eliminar ('Eliminar una asignatura impartida en un aula', 2050004, 7, TO_DATE('15/11/2016', 'DD/MM/YYYY'), TRUE);
PCK_AsigImpartidasEnAulas.Consultar;
END;
/
```

--Tabla AsignaturasPertenecenAGrados

```
BEGIN
PCK_AsigPertenecenAGrados.Inicializar;
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050004, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050004, 3, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050043, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050043, 4, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050035, 3, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050035, 4, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050012, 4, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050012, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050001, 3, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050001, 4, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050001, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050029, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050026, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050027, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050040, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado', 2050041, 1, TRUE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado con Código_Asig NULL', NULL, 1, FALSE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado con OID_G NULL', 2050004, NULL, FALSE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado violando PK', 2050004, 1, FALSE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado violando FK de asignaturas', 2999999, 1, FALSE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado violando FK de grados', 2050004, 666, FALSE);
PCK_AsigPertenecenAGrados.Insertar ('Insertar una asignatura perteneciente a un grado violando CK_Cod_Asig_AsigPertAGrad', 2004, 1, FALSE);
PCK_AsigPertenecenAGrados.Eliminar ('Eliminar una asignatura perteneciente a un grado', 2050004, 1, TRUE);
PCK_AsigPertenecenAGrados.Consultar;
END;
/
```

--Tabla Becarios

```
BEGIN
PCK_Becarios.Inicializar;
PCK_Becarios.Insertar ('Insertar un becario', '93839721C', '61700LIW', TO_DATE('07/10/2016', 'DD/MM/YYYY'), TO_DATE('23/05/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '07842998K', '18537TZN', TO_DATE('08/11/2016', 'DD/MM/YYYY'), TO_DATE('26/02/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '67202652X', '18537TZN', TO_DATE('15/09/2016', 'DD/MM/YYYY'), TO_DATE('05/03/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '27576410P', '93073MJH', TO_DATE('03/11/2016', 'DD/MM/YYYY'), TO_DATE('24/06/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '67202652X', '38720AJG', TO_DATE('13/12/2016', 'DD/MM/YYYY'), TO_DATE('27/01/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '70964804Y', '07284VPY', TO_DATE('17/10/2016', 'DD/MM/YYYY'), TO_DATE('28/02/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '47966225R', '18537TZN', TO_DATE('27/09/2016', 'DD/MM/YYYY'), TO_DATE('29/03/2017', 'DD/MM/YYYY'), TRUE);
```



```
PCK_Becarios.Insertar ('Insertar un becario', '00548678J', '13617BAU', TO_DATE('14/11/2016', 'DD/MM/YYYY'), TO_DATE('14/04/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '64427553E', '84676UFX', TO_DATE('23/12/2016', 'DD/MM/YYYY'), TO_DATE('05/05/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario', '00548678J', '84676UFX', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Insertar ('Insertar un becario con DNI NULL', NULL, '84676UFX', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario con Código_Beca NULL', '00548678J', NULL, TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario violando PK', '93839721C', '61700LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario violando FK de alumnos', '88888888C', '61700LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario violando FK de becas', '93839721C', '99990LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario violando CK_DNI_Becarios', '938C', '61700LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario violando CK_Código_Becas_Becarios', '93839721C', '610099W', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario violando CK_Fechas_Becarios', '93839721C', '61700LIW', TO_DATE('06/06/2017', 'DD/MM/YYYY'), TO_DATE('30/09/2016', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Insertar ('Insertar un becario incumpliendo la RN-003', '93839721C', '61700LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Actualizar ('Actualizar un becario', '93839721C', '61700LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), TRUE);
PCK_Becarios.Actualizar ('Actualizar un becario violando CK_Fechas_Becarios', '93839721C', '61700LIW', TO_DATE('06/06/2017', 'DD/MM/YYYY'), TO_DATE('30/09/2016', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Actualizar ('Actualizar un becario incumpliendo la RN-003', '93839721C', '61700LIW', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/10/2016', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Actualizar ('Actualizar un becario inexistente', '67202652Q', '12345TZN', TO_DATE('30/09/2016', 'DD/MM/YYYY'), TO_DATE('06/06/2017', 'DD/MM/YYYY'), FALSE);
PCK_Becarios.Eliminar ('Eliminar un becario', '07842998K', '18537TZN', TRUE);
PCK_Becarios.Consultar;
END;
/
```

--Tabla GruposAsistenAClaseEnAulas

```
BEGIN
PCK_GruposAsistenAClaseEnAulas.Inicializar;
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 1, 1, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 3, 4, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 4, 1, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 5, 4, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 6, 3, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 7, 3, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 8, 1, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 1, 7, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 4, 6, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 5, 5, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 7, 4, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 9, 1, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 10, 4, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula', 9, 4, TRUE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula con OID_Grup NULL', NULL, 2, FALSE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula con OID_A NULL', 1, NULL, FALSE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula violando FK de grupos', 666, 1, FALSE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula violando FK de aulas', 1, 666, FALSE);
PCK_GruposAsistenAClaseEnAulas.Insertar ('Insertar un grupo que asiste a clase en un aula incumpliendo la RN-006', 5, 15, FALSE);
PCK_GruposAsistenAClaseEnAulas.Eliminar ('Eliminar un grupo que asiste a clase en un aula', 6, 3, TRUE);
PCK_GruposAsistenAClaseEnAulas.Consultar;
END;
/
```



```
--Tabla Matriculas
BEGIN
PCK_Matriculas.Inicializar;
PCK_Matriculas.Insertar ('Insertar una matrícula', 3447488, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 1, '93839721C', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 2459519, TO_DATE('03/09/2016', 'DD/MM/YYYY'), 2, '07842998K', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 3171154, TO_DATE('05/09/2016', 'DD/MM/YYYY'), 1, '00548678J', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 1190202, TO_DATE('10/09/2016', 'DD/MM/YYYY'), 3, '35849097D', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 3771130, TO_DATE('03/09/2016', 'DD/MM/YYYY'), 1, '98523686N', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 9333670, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 4, '93839721C', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 3204793, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 2, '07804804F', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 6521080, TO_DATE('06/09/2016', 'DD/MM/YYYY'), 2, '26810393Y', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 1367797, TO_DATE('03/09/2016', 'DD/MM/YYYY'), 3, '42312153L', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 6851033, TO_DATE('08/09/2016', 'DD/MM/YYYY'), 1, '42312153L', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 2623254, TO_DATE('02/09/2016', 'DD/MM/YYYY'), 2, '27576410P', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 7314633, TO_DATE('11/09/2016', 'DD/MM/YYYY'), 4, '64427553E', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 2826843, TO_DATE('05/09/2016', 'DD/MM/YYYY'), 1, '07804804F', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 1326913, TO_DATE('08/09/2016', 'DD/MM/YYYY'), 3, '77008982Z', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 7805950, TO_DATE('04/09/2016', 'DD/MM/YYYY'), 1, '62968537B', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 1065287, TO_DATE('08/09/2016', 'DD/MM/YYYY'), 3, '47966225R', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 1535713, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 2, '23801240N', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 3988080, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 3, '36081385C', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula', 6255933, TO_DATE('02/09/2016', 'DD/MM/YYYY'), 1, '67202652X', TRUE);
PCK_Matriculas.Insertar ('Insertar una matrícula con Código_Matric NULL', NULL, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 1, '67202652X', FALSE);
PCK_Matriculas.Insertar ('Insertar una matrícula con DNI NULL', 1571054, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 1, NULL, FALSE);
PCK_Matriculas.Insertar ('Insertar una matrícula violando PK', 3447488, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 2, '67202652X', FALSE);
PCK_Matriculas.Insertar ('Insertar una matrícula violando FK de alumnos', 1571054, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 2, '99999999X', FALSE);
PCK_Matriculas.Insertar ('Insertar una matrícula violando CK_Cod_Matric_Matriculas', 6636, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 1, '67202652X', FALSE);
PCK_Matriculas.Insertar ('Insertar una matrícula violando CK_Curso_Matriculas', 1571054, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 5, '67202652X', FALSE);
PCK_Matriculas.Actualizar ('Actualizar una matrícula', 3447488, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 1, '67202652X', TRUE);
PCK_Matriculas.Actualizar ('Actualizar una matrícula con DNI NULL', 3447488, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 1, NULL, FALSE);
PCK_Matriculas.Actualizar ('Actualizar una matrícula violando FK de alumnos', 3447488, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 2, '99999999X', FALSE);
PCK_Matriculas.Actualizar ('Actualizar una matrícula violando CK_Curso_Matriculas', 3447488, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 5, '67202652X', FALSE);
PCK_Matriculas.Actualizar ('Actualizar una matrícula inexistente', 3499988, TO_DATE('01/09/2016', 'DD/MM/YYYY'), 2, '67202652X', FALSE);
PCK_Matriculas.Eliminar ('Actualizar una matrícula inexistente', 2459519, TRUE);
PCK_Matriculas.Consultar;
END;
/

--Tabla MatriculasPertenecenAGrados
BEGIN
PCK_MatricPertenecenAGrados.Inicializar;
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 3447488, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 1190202, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 6851033, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 6255933, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 7805950, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 3447488, 4, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 1190202, 4, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 6255933, 4, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 2826843, 4, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 7805950, 4, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 9333670, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 3204793, 3, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 2623254, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 6521080, 4, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 6851033, 3, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado', 2826843, 1, TRUE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado con Código_Matric NULL', NULL, 1, FALSE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado con OID_G NULL', 2826843, NULL, FALSE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado violando PK', 3447488, 1, FALSE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado violando FK de matrículas', 9999999, 1, FALSE);
```



```
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado violando FK de grados', 3447488, 666, FALSE);
PCK_MatricPertenecenAGrados.Insertar ('Insertar una matrícula que pertenece a un grado violando CK_Cod_Matric_MatricPertAGrad', 7575, 1, FALSE);
PCK_MatricPertenecenAGrados.Eliminar ('Eliminar una matrícula que pertenece a un grado', 6255933, 1, TRUE);
PCK_MatricPertenecenAGrados.Consultar;
END;
/

--Tabla MatriculasContienenAsignaturas
BEGIN
PCK_MatriculasContienenAsig.Inicializar;
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 3447488, 2050004, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 3447488, 2050040, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 3447488, 2050029, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 3447488, 2050015, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 3447488, 2050019, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 3447488, 2050026, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 1190202, 2050007, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 1190202, 2050004, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 1190202, 2050035, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 1190202, 2050027, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 1190202, 2050011, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 1190202, 2050043, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 6851033, 2050040, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 6851033, 2050035, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 6851033, 2050002, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 6851033, 2050001, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 6851033, 2050010, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 2826843, 2050017, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 2826843, 2050004, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 2826843, 2050035, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 2826843, 2050014, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 2826843, 2050020, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050026, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050020, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050011, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050024, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050043, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050027, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050029, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura', 7805950, 2050040, TRUE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura con Código_Matric NULL', NULL, 2050043, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura con Código_Asig NULL', 2826843, NULL, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura violando PK', 7805950, 2050043, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura violando FK de matrículas', 9999999, 2050043, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura violando FK de asignaturas', 7805950, 9999999, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura violando CK_Cod_Matric_MatricContAsig', 7850, 2050043, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura violando CK_Cod_Asig_MatricContAsig', 7805950, 4633, FALSE);
PCK_MatriculasContienenAsig.Insertar ('Insertar una matrícula que contiene una asignatura incumpliendo la RN-010', 7805950, 2050041, FALSE);
PCK_MatriculasContienenAsig.Eliminar ('Eliminar una matrícula que contiene una asignatura', 3447488, 2050040, TRUE);
PCK_MatriculasContienenAsig.Consultar;
END;
/
```



```
--Tabla Profesores
BEGIN
PCK_Profesores.Inicializar;
PCK_Profesores.Insertar ('Insertar un profesor', '53712051J', 'Rosa María', 'Acien Zuruta', TO_DATE('12/02/1988', 'DD/MM/YYYY'), 'rosamariaaz@gmail.com', 'Catedrático', 1, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '75098488S', 'Daniel', 'Amate Garrido', TO_DATE('22/03/1976', 'DD/MM/YYYY'), 'danielag@gmail.com', 'Titular', 2, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '45295530C', 'José', 'Benayas Perez', TO_DATE('10/07/1967', 'DD/MM/YYYY'), 'josebp@gmail.com', 'Contratado doctor', 3, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '78035832Y', 'Susana', 'Bernabe Casanova', TO_DATE('04/07/1968', 'DD/MM/YYYY'), 'susancb@gmail.com', 'Colaborador', 7, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '78035130V', 'Irene', 'Bernal Ruiz', TO_DATE('04/09/1980', 'DD/MM/YYYY'), 'irenebr@gmail.com', 'Ayudante', 9, 9, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '75238658T', 'Magdalena', 'Carrera Benitez', TO_DATE('06/03/1974', 'DD/MM/YYYY'), 'magdalenacb@gmail.com', 'Ayudante doctor', 1, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '75258403B', 'Natalia', 'Casas García', TO_DATE('17/08/1962', 'DD/MM/YYYY'), 'nataliacg@gmail.com', 'Interino', 5, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '75243008A', 'Encarnación', 'Cortes Ibáñez', TO_DATE('18/05/1964', 'DD/MM/YYYY'), 'encarnacionci@gmail.com', 'Contratado doctor', 5, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor', '75257344X', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 5, 7, TRUE);
PCK_Profesores.Insertar ('Insertar un profesor con DNI NULL', NULL, 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 7, FALSE);
PCK_Profesores.Insertar ('Insertar un profesor violando PK', '53712051J', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 7, FALSE);
PCK_Profesores.Insertar ('Insertar un profesor violando FK de despachos', '67842443X', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 666, 7, FALSE);
PCK_Profesores.Insertar ('Insertar un profesor violando FK de departamentos', '67842443X', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 666, FALSE);
PCK_Profesores.Insertar ('Insertar un profesor violando CK_DNI_Profesores', '6784fd', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 7, FALSE);
PCK_Profesores.Insertar ('Insertar un profesor violando CK_Categoría_Profesores', '67842443X', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Temporal', 8, 7, FALSE);
PCK_Profesores.Insertar ('Insertar un profesor incumpliendo la RN-005', '67842443X', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 1, 7, FALSE);
PCK_Profesores.Actualizar ('Actualizar un profesor', '53712051J', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 1, 7, TRUE);
PCK_Profesores.Actualizar ('Actualizar un profesor con DNI NULL', NULL, 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 7, FALSE);
PCK_Profesores.Actualizar ('Actualizar un profesor violando FK de despachos', '53712051J', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 666, 7, FALSE);
PCK_Profesores.Actualizar ('Actualizar un profesor violando FK de departamentos', '53712051J', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 666, FALSE);
PCK_Profesores.Actualizar ('Actualizar un profesor violando CK_Categoría_Profesores', '53712051J', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Temporal', 8, 7, FALSE);
PCK_Profesores.Actualizar ('Actualizar un profesor incumpliendo la RN-005', '53712051J', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 5, 7, FALSE);
PCK_Profesores.Actualizar ('Actualizar un profesor inexistente', '13067547M', 'Mónica', 'Gonzalez Navas', TO_DATE('10/01/1980', 'DD/MM/YYYY'), 'monicagn@gmail.com', 'Titular', 8, 7, FALSE);
PCK_Profesores.Eliminar ('Eliminar un profesor', '75098488S', TRUE);
PCK_Profesores.Consultar;
END;
/
```

```
--Tabla ProfesoresImpartenAsignaturas
BEGIN
PCK_ProfesoresImpartenAsig.Inicializar;
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '53712051J', 2050001, 12, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '75257344X', 2050010, 12, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '45295530C', 2050011, 12, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '78035832Y', 2050018, 12, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '75238658T', 2050026, 6, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '75258403B', 2050019, 6, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '78035130V', 2050003, 6, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '53712051J', 2050043, 6, TRUE);
```



```
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '75257344X', 2050011, 12, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '45295530C', 2050010, 12, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '78035832Y', 2050014, 6, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '53712051J', 2050039, 6, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura', '53712051J', 2050041, 6, TRUE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura con DNI NULL', NULL, 2050039, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura con Código_Asig NULL', '53712051J', NULL, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura violando PK', '53712051J', 2050001, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura violando FK de profesores', '999999999J', 2050001, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura violando FK de asignaturas', '53712051J', 6770001, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura violando CK_DNI_ProfImpAsig', '5371GF51J', 2050001, 12, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura violando CK_Cod_Asig_ProfImpAsig', '75257344X', 2010, 12, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura incumpliendo la RN-013', '75257344X', 2050015, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura incumpliendo la RN-014', '78035832Y', 2050026, 12, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura incumpliendo la RN-015', '78035130V', 2050009, 6, FALSE);
PCK_ProfesoresImpartenAsig.Insertar ('Insertar un profesor que imparte una asignatura incumpliendo la RN-015', '53712051J', 2050010, 6, FALSE);
PCK_ProfesoresImpartenAsig.Eliminar ('Eliminar un profesor que imparte una asignatura', '75257344X', 2050010, TRUE);
PCK_ProfesoresImpartenAsig.Consultar;
END;
/

--Tabla ProfesoresImpartenEnGrupos
BEGIN
PCK_ProfesoresImpartenEnGrupos.Inicializar;
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '53712051J', 3, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '53712051J', 4, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '53712051J', 6, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '53712051J', 8, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '78035832Y', 7, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '78035832Y', 10, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '78035832Y', 14, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '78035832Y', 3, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '45295530C', 6, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '45295530C', 5, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '45295530C', 8, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '75258403B', 3, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo', '75258403B', 1, TRUE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo con DNI NULL', NULL, 1, FALSE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo con OID_Grup NULL', '75258403B', NULL, FALSE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo violando PK', '53712051J', 3, FALSE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo violando FK de profesores', '88888584J', 3, FALSE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo violando FK de grupos', '53712051J', 666, FALSE);
PCK_ProfesoresImpartenEnGrupos.Insertar ('Insertar un profesor que imparte en un grupo violando CK_DNI_ProfImpEnGrup', '537ds1J', 6, FALSE);
PCK_ProfesoresImpartenEnGrupos.Eliminar ('Eliminar un profesor que imparte en un grupo', '53712051J', 3, TRUE);
PCK_ProfesoresImpartenEnGrupos.Consultar;
END;
/
```



```
--Tabla Tutorias
BEGIN
PCK_Tutorias.Inicializar;
PCK_Tutorias.Insertar ('Insertar una tutoría, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '78035832Y', TRUE);
PCK_Tutorias.Insertar ('Insertar una tutoría, TO_DATE('17/11/2016', 'DD/MM/YYYY'), '18:30', 25, '07842998K', '75258403B', TRUE);
PCK_Tutorias.Insertar ('Insertar una tutoría, TO_DATE('19/12/2016', 'DD/MM/YYYY'), '20:00', 20, '27576410P', '78035832Y', TRUE);
PCK_Tutorias.Insertar ('Insertar una tutoría, TO_DATE('21/02/2017', 'DD/MM/YYYY'), '09:30', 10, '07842998K', '75258403B', TRUE);
PCK_Tutorias.Insertar ('Insertar una tutoría, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '16:00', 30, '62968537B', '53712051J', TRUE);
PCK_Tutorias.Insertar ('Insertar una tutoría, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '12:50', 25, '62968537B', '53712051J', TRUE);
PCK_Tutorias.Insertar ('Insertar una tutoríaDNI_Alum NULL', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, NULL, '78035832Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaDNI_Prof NULL', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', NULL, FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaviolando FK de alumnos', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '55533672C', '78035832Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaviolando FK de profesores', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '77537276Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaviolando CK_Hora_Comienzo_Tutorias', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '35:80', 20, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaviolando CK_DNI_Alum_Tutorias', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '938s1C', '78035832Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaviolando CK_DNI_Prof_Tutorias', TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '780gd832Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaincumpliendo la RN-016', TO_DATE('08/01/2017', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Insertar ('Insertar una tutoríaincumpliendo la RN-017', TO_DATE('12/01/2017', 'DD/MM/YYYY'), '17:00', 50, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoría, 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '12:50', 25, '62968537B', '53712051J', TRUE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaDNI_Alum NULL', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, NULL, '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaDNI_Prof NULL', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', NULL, FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaviolando FK de alumnos', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '55533672C', '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaviolando FK de profesores', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '77537276Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaviolando CK_Hora_Comienzo_Tutorias', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '35:80', 20, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaviolando CK_DNI_Alum_Tutorias', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '938s1C', '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaviolando CK_DNI_Prof_Tutorias', 1, TO_DATE('12/10/2016', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '780gd832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaincumpliendo la RN-016', 1, TO_DATE('08/01/2017', 'DD/MM/YYYY'), '17:00', 20, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríaincumpliendo la RN-017', 1, TO_DATE('12/01/2017', 'DD/MM/YYYY'), '17:00', 50, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Actualizar ('Actualizar una tutoríainexistente', 666, TO_DATE('12/01/2017', 'DD/MM/YYYY'), '17:00', 50, '93839721C', '78035832Y', FALSE);
PCK_Tutorias.Eliminar ('Eliminar una tutoría, 2, TRUE);
PCK_Tutorias.Consultar;
END;
/
```



Pruebas de procedimientos y funciones asociados a RF



```
SET SERVEROUTPUT ON;

--RF-001
EXEC PR_Matricular_Asig(2623254, 2050032);

--RF-002
EXEC PR_Anular_Asig(2623254, 2050032);

--RF-003
EXEC PR_Evaluuar_Alum(2050004, 1);

--RF-004
EXEC PR_Alum_Matriculado_En_Asig('35849097D', 2050004);

--RF-005
EXEC PR_CalfMed_Agrup_Por_Asig('56336094D');

--RF-006
EXEC PR_Asig_Orden_Por_Curso('07804804F');

--RF-007
EXEC PR_Expediente_Alumno('93839721C');

--RF-008
EXEC PR_Asociar_Asignatura_Dep(2050029, 'Biología Celular');

--RF-009
EXEC PR_Asociar_Beca('91440SRA', '93839721C', TO_DATE('07/10/2016',
'DD/MM/YYYY'), TO_DATE('23/05/2017', 'DD/MM/YYYY'));

--RF-010
EXEC PR_Asociar_Duracion_Beca('61700LIW', '93839721C', TO_DATE('23/05/2017',
'DD/MM/YYYY'));

--RF-011
EXEC PR_Asociar_Cuantia_Fija_Beca('61700LIW', 3000);

--RF-012
EXEC PR_Asociar_Cuantia_Var_Beca('61700LIW', 100);

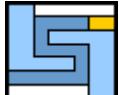
--RF-013
EXEC PR_Becas_Alumno('93839721C');

--RF-014
EXEC PR_Alumnos_Matriculados;

--RF-015
EXEC PR_Profesores_Impartiendo;

--RF-016
EXEC PR_Espacios_AgrupPor_Tipo;

--RF-017
EXEC PR_Profesores_Departamento('Lenguajes y Sistemas Informáticos');
```



```
--RF-018
EXEC PR_Asignaturas_Departamento('Lenguajes y Sistemas Informáticos');

--RF-019
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_Grupo_Mas_Alumnos);
END;
/

--RF-020
EXEC PR_Eliminar_Asignatura_Dep(2050012)

--RF-021
EXEC PR_Asociar_Profesor_Dep('53712051J', 'Lenguajes y Sistemas Informáticos');

--RF-022
EXEC PR_Eliminar_Profesor_Dep('53712051J');

--RF-023
EXEC PR_Prof_Imparte_Asig(2050029);

--RF-024
EXEC PR_Prof_Imparte_Asigs_Dep('Lenguajes y Sistemas Informáticos');

--RF-025
EXEC PR_NumProf_AgrupadosPor_Asig;

--RF-026
EXEC PR_NumTut_AgrupadosPor_Prof;

--RF-027
EXEC PR_Profesores_Despacho('F0.25');

--RF-028
EXEC PR_Notas_Expediente(78002608);

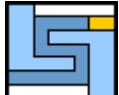
--RF-029
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_NotaMedia_Expediente(61654321));
END;
/

--RF-030
EXEC PR_Evaluar_Examen(TO_NUMBER('5,4'), 'Aprobado', 'Primera', 1,
TO_DATE('25/01/2017', 'DD/MM/YYYY'), 84656361);

--RF-031
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_MejorNota_Expediente(61654321));
END;
/

--RF-032
EXEC PR_AsignaturasOblig_Grado('Ingeniería Informática - Tecnologías
Informáticas');

--RF-033
EXEC PR_AsignaturasOpt_Grado('Matemáticas');
```



```
--RF-034
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_CreditosOptativos_Grado('Ingeniería de la Salud'));
END;
/

--RF-035
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_CreditosObligatorios_Grado('Matemáticas'));
END;
/

--RF-036
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_CreditosTroncales_Grado('Ingeniería de la Salud'));
END;
/

--RF-037
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_CreditosTotales_Grado('Ingeniería de la Salud'));
END;
/

--RF-038
EXEC PR_Departamentos_Grado('Ingeniería de la Salud');

--RF-039
EXEC PR_Profesores_Grado('Ingeniería de la Salud');

--RF-040
EXEC PR_Asignaturas_Grado_Curso('Ingeniería de la Salud', 2);

--RF-041
EXEC PR_AsignaturasAnuales_Grado('Ingeniería de la Salud');

--RF-042
EXEC PR_Tutorias_Profesor('53712051J');

--RF-043
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_Departamento_Profesor('75238658T'));
END;
/

--RF-044
BEGIN
    DBMS_OUTPUT.PUT_LINE(FN_DedicacionTotal_Profesor('53712051J'));
END;
/

--RF-045
EXEC PR_Asignaturas_Profesor('53712051J');

--RF-046
EXEC PR_Dedicacion_Profesor('53712051J');

--RF-047
EXEC PR_Categoría_Profesor('53712051J', 'Interino');
```



```
--RF-048
EXEC PR_Adicion_Tutoria(TO_DATE('25/01/2017', 'DD/MM/YYYY'), '17:00', 20,
'93839721C', '78035832Y');

--RF-049
EXEC PR_Eliminar_Tutoria(1);

--RF-050
EXEC PR_Asignar_Profesor_Asignatura('75258403B', 2050011, 6);

--RF-051
EXEC PR_Eliminar_Profesor_Asig('53712051J', 2050043);

--RF-052
EXEC PR_Asignaturas_Aula('A0.07', TO_DATE('01/10/2016', 'DD/MM/YYYY'));
```