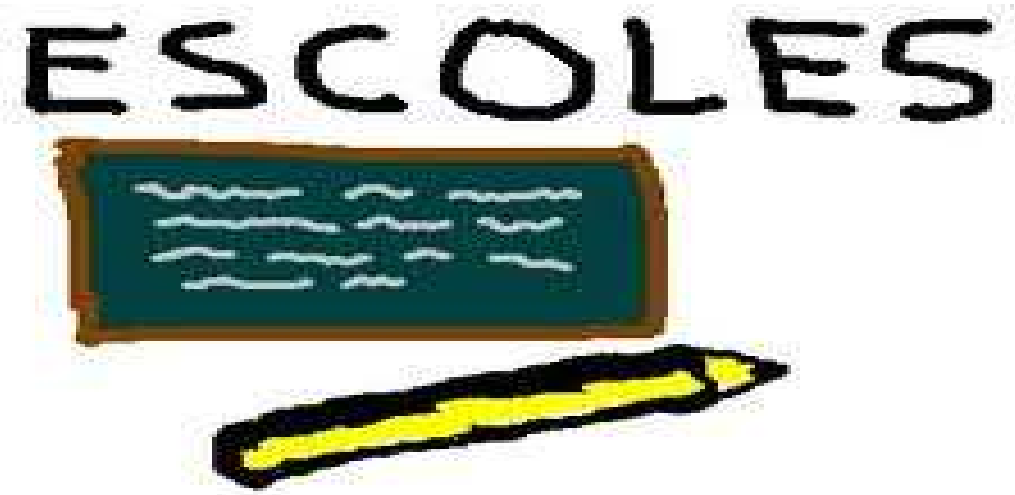
Enginyeria Tècnica en Informàtica de Gestió



DISSENY I IMPLEMENTACIÓ D'UN: SISTEMA DE GESTIÓ D'AMONESTACIONS I SANCIONS EN CENTRES EDUCATIUS

Treball Fi de Carrera : Base de dades Relacionals
MEMORIA

Autor : Óscar Beltrán Campamelos
Consultor : Manel Rella Ruiz

Blanes, a 12 de Junio de 2011

1. Dedicatoria y agradecimientos

Como dicen que de señores es ser corteses, me gustaria agradecer este trabajo y la finalizacion de esta carrera a toda mi familia, desde mi mujer pasando por mis 3 hijos y acabando por mis padres. Por lo que han tenido que sufrir para que yo pueda finalizarla, lo que hace unos años se me puso entre ceja y ceja que no es otra cosa que poder ser ingeniero técnico en informática de gestión.

No es una tarea facil la de estar estudiando, trabajando y ser padre más las labores de casa pero con tesón y sacrificio hemos podido llegar a un final feliz.

2. Resumen

La Generalitat de Catalunya por parte de la Conselleria d' Ensanyament nos ha encargado un proyecto por el cual debemos gestionar un aplicativo para intervenir en el sistema de gestión de las amonestaciones y posteriores sanciones en centros educativos.

Para poder ejecutar este proyecto debemos tener tablas maestras para tener los datos de los posibles alumnos, cursos, profesores, asignaturas, calendarios, visitas, matriculaciones, composiciones de cursos etc. Dado que no tenemos estos datos en tablas debemos implementarlas nosotros para poder solucionar nuestro aplicativo.

En este proyecto pondremos a disposición todos nuestros conocimientos adquiridos durante toda la carrera. Iremos dirigiendo el proyecto por todas las etapas para un desenlace óptimo, esto quiere decir una buena planificación, un buen ajuste de los requisitos funcionales, un buen análisis del diseño y una implementación acorde a lo solicitado.

El producto que utilizaremos es Oracle y el lenguaje para los procedimientos de almacenamiento es PL/SQL. En el crearemos el esqueleto de las tablas para después con los procedimientos y las pertinentes pruebas, verificar su buen funcionamiento con la tabla de log podremos chequear todos las incidencias. Por último insertaremos una batería de datos para poder verificar las estadísticas.

Index

3. Introducción

3.1 Justificación y aportación

El objetivo del trabajo final de carrera es aplicar todos los conocimientos aprendidos a lo largo de la carrera. Después de realizar tres asignaturas dentro del campo de base de datos como son base de datos I, base de datos II y sistema de gestión de base de datos he creído oportuno realizar un TFC relacionado con ellas. Por eso he escogido el tema de base de datos relacionales.

También justificaremos esta elección por que es un campo que me apasiona dentro de la informática. Particularmente interpreto la base de datos como el primer paso y el más importante para un desarrollo idóneo de cualquier aplicativo.

Por otro lado aportare un poco mi experiencia profesional dentro de este mundo de la informática y en concreto sobre la gestión del software.

3.1.1 Objetivos

En primer lugar la intención del TFC es poder desarrollar un proyecto como si estuviésemos trabajando con un cliente, esto quiere decir ejecutar dicho proyecto con aplicaciones de mercado (oracle, etc.), con un caso real como es gestionar para la Conselleria de Educación de la Generalitat de Catalunya un aplicativo para gestionar las amonestaciones y sanciones en centros educativos.

Entregaremos un diagrama entidad/relación con sus atributos y todas las restricciones de integridad. Por otro lado mediante scripts debemos crear las tablas, índices, triggers etc. Por ultimo con los procedimientos almacenados debemos implementar todos los requisitos funcionales del sistema. No olvidaremos que para cerciorarnos de que todo funcione correctamente ejecutaremos la aplicación con un juego de pruebas, la cual nos validará todos los requisitos expuestos.

Para realizar todo esto el plan de estudio nos presenta 3 entregas en la cual gradualmente haremos entrega del material especificado para cada una de ellas. Estas entregas son:

- PAC1: plan de trabajo, aquí desarrollaremos toda la planificación de las tareas que debemos implementar durante todo el proyecto.
- PAC2: diseño y análisis de toda la base de datos.
- PAC3: implementación y pruebas de toda la base de datos.
- Entrega Final: esta ya es la entrega de todo el TFC, esta integrada por memoria, presentación y producto.

3.2 Enfoque y método a seguir

Dentro del plan de trabajo que hemos desarrollado podemos observar que las fases están muy marcadas, esto es debido por ser un proyecto con carácter educativo. Estamos ante la obligación de un proyecto muy estándar y marcada por las directrices del trabajo académico.

Por lo tanto realizaremos un desarrollo en cascada esto quiere decir que dividimos el proyecto en diferentes etapas y hasta que no acaba una no podemos empezar otra. En un proyecto en la vida real esto es casi inviable dado la necesidad de ir haciendo diferentes etapas a la vez.

Pero dado que en este proyecto trabajamos solos nos marcamos las pautas y es difícil por no decir imposible ir implementando etapas sin acabar la anterior. Debemos llevar una planificación muy metódica. Por eso la importancia del tema.

Este tipo de desarrollo en cascada se identifica por una metodología como la siguiente:

1. Análisis de requisitos
2. Diseño del Sistema
3. Diseño del Programa
4. Codificación
5. Pruebas
6. Implantación
7. Mantenimiento

Dentro de estas fases nosotros obviaremos el diseño el mantenimiento dado el valor académico del proyecto.

3.3 Planificación

3.3.1 Fechas

Tareas	Fecha inicio	Fecha fin	Descripción
PAC1	3/03/2011	20/03/2011	Plan de trabajo y planificación del proyecto
PAC2	21/03/2011	17/04/2011	Diseño análisis BD
PAC3	18/04/2011	15/05/2011	Implementación y pruebas BD
ENTREGA FINAL	16/05/2011	12/06/2011	Entrega documentación final

3.4.2 Estructura de cada tarea

Explicaremos un poco más las actividades de cada tarea.

3.4.2.1 Plan de trabajo

En el explicaremos el objetivo del proyecto. Planificación del proyecto fechas claves y documentación a entregar.

3.4.2.2 Análisis y diseño base de datos

Análisis de todos los requerimientos y diseño del modelo conceptual (diagrama E/R), el diseño lógico y el diseño físico de la base de datos. Esta fase engloba la construcción de los diferentes scripts para poder crear la base de datos. En esta fase debemos tener en cuenta todas las reglas de negocio impuestas por el cliente.

Debemos tener en cuenta las tablas que necesitamos para poder ejecutar todas las implementaciones pedidas anteriormente. Con un buen diagrama y un buen análisis del diseño lógico y físico no debemos tener problema para poder implementar una adecuada base de datos.

En esta fase relacionaremos las diferentes tablas para poder tener una base de datos relacional en la cual tengamos todo en una relación. Es decir todas las tablas las tenemos relacionadas con las claves primarias y foráneas para crear un mapa relacional óptimo.

Debemos tener en cuenta las diferentes secuencia y triggers que podamos crear para una optimización adecuada al proyecto.

3.4.2.3 Implementación y pruebas base de datos

En esta fase crearemos todos los procedimientos con los requisitos establecidos por el cliente, estos procedimientos irán ejecutados sobre las tablas creadas en la fase anterior. Inserción de datos y diferentes pruebas para verificar la integridad de la base de datos.

Dentro de esta fase la implementación debe seguir una pauta para la creación de los procedimientos, estos se deben crear en una secuencia lógica. Dicha secuencia sigue la pauta de la creación de las tablas.

Por lo tanto la implementación tendrá este orden:

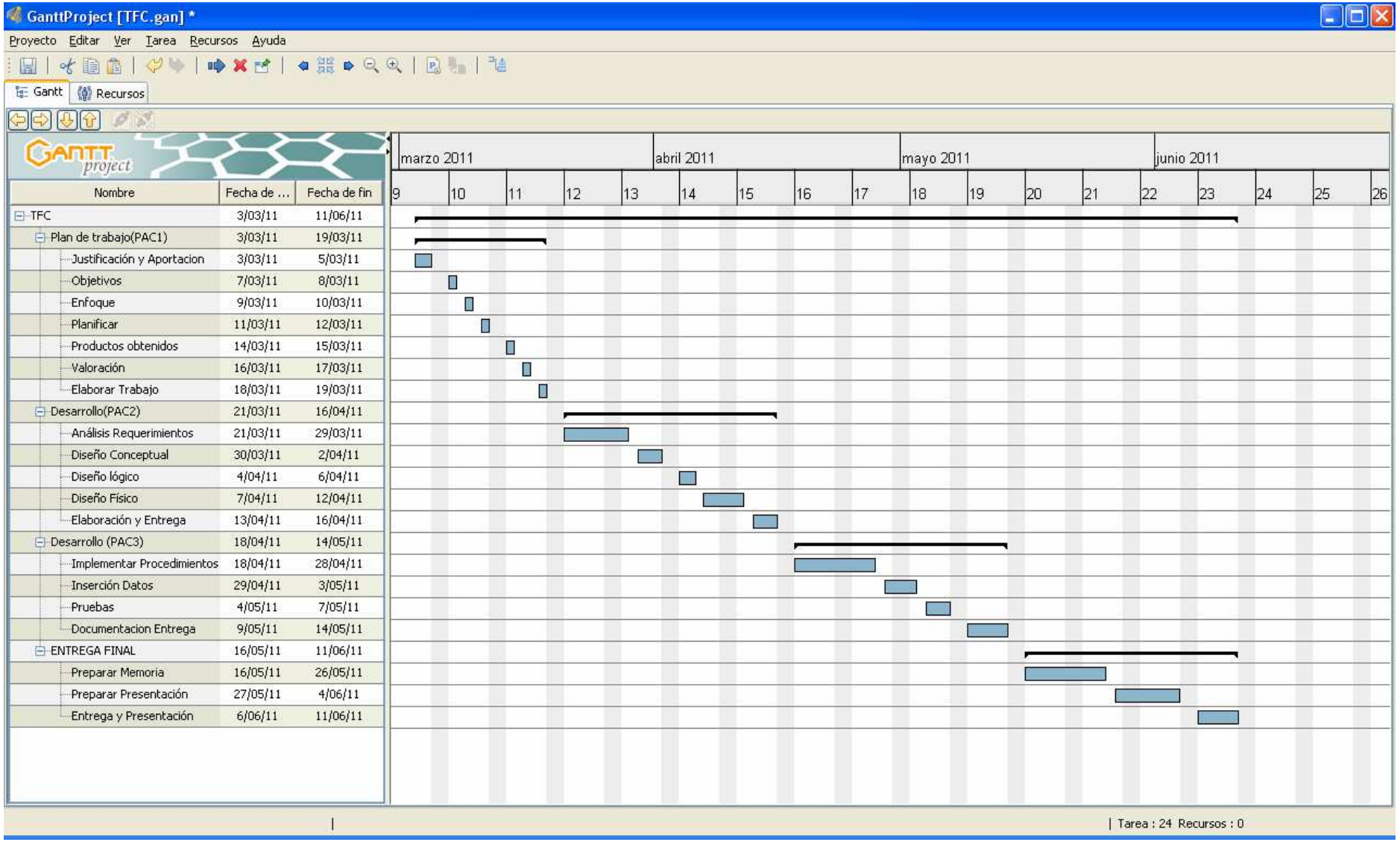
1. Gestión alumnos
2. Gestión profesores
3. Gestión asignaturas
4. Gestión cursos
5. Gestión tipo de amonestación
6. Gestión tipo de sanción
7. Gestión matriculan (relación alumnos y cursos)

8. Gestión composición(relación cursos, asignaturas y profesores)
9. Gestión calendario
10. Gestión visitas
11. Gestión regla activación
12. Gestión amonestación
13. Gestión sanción
14. Gestión estadísticos

3.4.2.4 Entrega documentación final

En esta fase prepararemos toda la documentación para entregar como la memoria final y la presentación final del proyecto.

3.4.3 Planificación Detallada


GanttProject [TFC.gan] *
Proyecto Editar Ver Tarea Recursos Ayuda
Gantt Recursos
GANTT project
Nombre
Fecha de ...
Fecha de fin
TFC 3/03/11 11/06/11
Plan de trabajo(PAC1) 3/03/11 19/03/11
Justificación y Aportacion 3/03/11 5/03/11
Objetivos 7/03/11 8/03/11
Enfoque 9/03/11 10/03/11
Planificar 11/03/11 12/03/11
Productos obtenidos 14/03/11 15/03/11
Valoración 16/03/11 17/03/11
Elaborar Trabajo 18/03/11 19/03/11
Desarrollo(PAC2) 21/03/11 16/04/11
Análisis Requerimientos 21/03/11 29/03/11
Diseño Conceptual 30/03/11 2/04/11
Diseño lógico 4/04/11 6/04/11
Diseño Físico 7/04/11 12/04/11
Elaboración y Entrega 13/04/11 16/04/11
Desarrollo (PAC3) 18/04/11 14/05/11
Implementar Procedimientos 18/04/11 28/04/11
Inserción Datos 29/04/11 3/05/11
Pruebas 4/05/11 7/05/11
Documentacion Entrega 9/05/11 14/05/11
ENTREGA FINAL 16/05/11 11/06/11
Preparar Memoria 16/05/11 26/05/11
Preparar Presentación 27/05/11 4/06/11
Entrega y Presentación 6/06/11 11/06/11
marzo 2011
abril 2011
mayo 2011
junio 2011
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
Tarea : 24 Recursos : 0

3.5 Productos obtenidos

Dentro de este proyecto una vez finalizado los productos obtenidos son

- Memoria TFC
- Presentación Power point
- Configuración de la herramienta para la base de datos ORACLE Database Express 10g Reléase 2
- Preparación entorno de trabajo SQL DEVELOPER 1.0.0.15
- Los diferentes scripts para la creación de la base de datos.
- Procedimientos de almacenamiento para los requisitos de la base de datos.
- Introducción de datos a la base de datos
- Scripts para las pruebas del control de la base de datos

3.6 Breve descripción de los otros capítulos

Tal y como hemos mostrado dentro de la planificación los siguientes capítulos desarrollaremos básicamente la base de datos.

Para ello deberemos seguir la pauta marcada.

Análisis de los requerimientos en este capítulo después de estudiar los requisitos que nos imponen, montaremos las directrices para un óptimo desarrollo del aplicativo.

Diseño en este capitulo se elabora todos los diseños par la elaboración de la base de datos que serán conceptual, lógico y físico.

Implementación en este capitulo después de tener toda la base de datos creada, procederemos a crear todos los procedimientos para controlar la base de datos con sus requerimientos.

Insertar datos y pruebas este capitulo después de insertar los datos verificaremos que los procedimientos realizan correctamente sus objetivos.

3.7 Valoración Económica i recursos necesarios

Para valorar este proyecto debemos gestionar dos diferentes tipos de perfiles informáticos para implementar todo el proyecto, estos son analista que en este caso sería también el jefe del proyecto y el programador.

Tenemos 100 días para realizar el proyecto lo que equivale a 14 semanas, si dejamos 2 días de descanso y aplicamos 2 horas cada día nos salen 140 horas de proyecto.

Valoramos que el analista trabaja 100 horas y el programador 40. El precio hora del analista es de 36 € y del programador 30.

Por lo tanto tenemos que:

Analista 100*36 = 3600 €
Programador 40*30 = 1200 €

Las licencias que tendríamos que comprar de momento son gratuitas, por lo tanto no deberíamos contar gastos de este tipo.

Total del proyecto 4800 €

4. Análisis de Requerimientos

4.1 Descripción

Crearemos una base de datos para la gestión de amonestaciones y sanciones en centros educativos de la Generalitat de Catalunya.

En dicha base de datos tendremos por una parte las tablas maestras para después con ellas gestionar las diferentes relaciones, para poder llevar a cabo el proyecto. Como tablas maestras están alumnos, profesores, cursos, asignaturas, tipos de amonestaciones y tipos de sanciones.

Las relaciones de estas entre ellas nos proporcionaran las diferentes tablas para acabar de relacionar el aplicativo. Por lo tanto un alumno se matricula en un curso. El curso lo componen unas asignaturas de las que da las clases un profesor con un calendario y un horario de visita para el centro.

Los alumnos matriculados tendrán su número de expediente y sus datos académicos y personales. Los cursos las asignaturas que tienen y el tutor o tutores si tienen. Los profesores las asignaturas que imparten, sus datos personales y horas de visita.

Por un lado tendremos un calendario escolar donde informaremos de los horarios de todas las asignaturas en cada curso además contemplará los días festivos, implementados por el calendario laboral.

Por otro lado debemos implementar el modelo de tipos de amonestaciones y tipos de sanciones para los alumnos, este modelo vendrá preestablecido y podremos definir nuevas.

Las amonestaciones las impone el profesor a los alumnos que componen los cursos. Según el tipo de amonestación impuesta a la amonestación y la gravedad de la misma preguntara a las reglas de activación si es mayor la gravedad puesta o acumulada que la establecida en caso afirmativo activa un tipo de sanción y por lo tanto una sanción. Dentro de la sanción siempre marcamos que amonestación la activo en caso de tener que borrarla poder hacerlo. La sanción puede hacerse manual por que dentro de la regla de activación tenemos puesto el código manual.

El diseño nos permitirá las siguientes funcionalidades:

- Gestión alumnos: alta, baja y modificación.
- Gestión profesores: alta, baja y modificación.
- Gestión cursos: alta, baja y modificación.
- Gestión asignaturas: alta, baja y modificación.
- Gestión matriculas: alta, baja y modificación.

- Gestión componen: alta y baja.
- Gestión calendario escolar: alta, baja y modificación.
- Gestión de visitas escolares: alta, baja y modificación.
- Gestión tipo de amonestaciones: alta, baja y modificación.
- Gestión tipo de sanciones: alta, baja y modificación.
- Gestión de Reglas de activación: alta, baja y modificación.
- Gestión de amonestación: alta, baja y modificación.
- Diferentes consultas:

    - las amonestaciones impuestas con su información básica.

    - los alumnos de un curso con toda la información.

    - tipos de amonestaciones y sus tipos de sanciones

    - las amonestaciones y sanciones de los alumnos.

- Modulo estadístico: deberemos implementar diferentes tablas para este modulo, una vez que ejecutemos una amonestación este procedimiento activará otros procedimientos para las estadísticas de amonestaciones y si procede para una sanción este activará los procedimientos estadísticas para sanciones. Posteriormente con una consulta simple es decir un select para esas tablas poder extraer una serie de datos como son:

    - número de amonestaciones por alumno y alumnos sin amonestación.

    - profesor más amonestador por curso y media de amonestaciones profesor año.

    - número de sanciones por alumno y año, por curso año, el nombre del alumno más sancionado y media de sanciones alumnos curso.

4.2 Requisitos funcionales

Identificador	R1
Descripción	Crear y mantener datos de alumnos.

Identificador	R1 bis
Descripción	Crear y mantener datos de profesores.

Identificador	R2
Descripción	Crear y mantener datos de los cursos.

Identificador	R2 bis
Descripción	Crear y mantener datos de las asignaturas.

Identificador	R3
Descripción	Crear calendario para cada curso definido, con sus horarios para las diferentes asignaturas.

Identificador	R4
Descripción	Definir amonestaciones con los datos básicos para diferenciarlas.

Identificador	R5
Descripción	Existen tipos de amonestaciones predefinidas.

Identificador	R6
Descripción	Definir nuevas tipos de amonestaciones.

Identificador	R7
Descripción	Introducir sanciones según el tipo de amonestación o acumulación de estas.

Identificador	R8
Descripción	Cada sanción va unida a un tipo de amonestación.

Identificador	R9
Descripción	Tenemos sanciones predefinidas por los profesores.

Identificador	R10
Descripción	Para activar sanciones automáticamente tendremos la siguiente forma sintáctica: Si ([Amonestación] [<|>|=] [Valor numérico definido]) => "descripción de sanción"

Identificador	R11
Descripción	Tenemos una tabla con las matriculaciones, de los alumnos al curso. Los profesores responsables de los cursos y las asignaturas que imparten.

Identificador	R12

Descripción	Debemos informar del día y hora de atención a los padres por parte de los profesores y de consulta por parte de los alumnos.

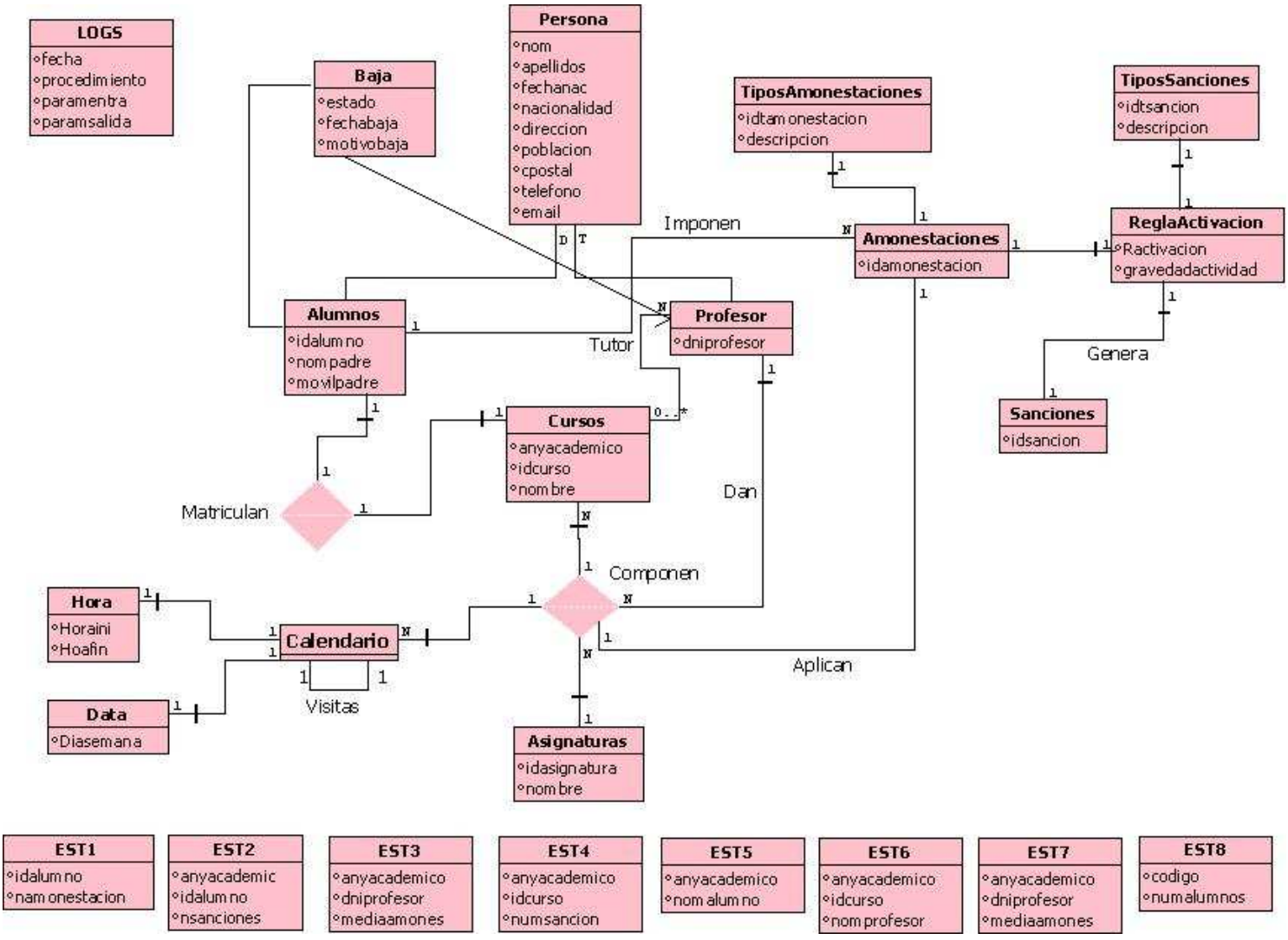
Identificador	R13
Descripción	Sabremos en cualquier momento las amonestaciones y sanciones de cualquier alumno.

Identificador	R14
Descripción	Aparte de todas las tablas maestras para poder desarrollara la aplicación. Tendremos unos listados para saber datos referentes a estas tablas e incluso interrelacionadas entre ellas.

Identificador	R15
Descripción	Podremos mostrar unos datos estadísticos, referentes a toda la aplicación. Estos datos serán integrados en tablas para poder trabajar en forma Bussines Intelligence.

5. Diseño Conceptual

5.1 Diagrama E/R



5.2 Entidades

Las entidades que tenemos según el diseño son:

- Alumnos
- Profesores
- Cursos (Está en relación con profesor, pero puede tener o no)
- Asignaturas
- Tipos de amonestaciones
- Tipos de sanciones

5.3 Entidades Relación

A continuación detallaremos las relaciones que hay entre las entidades y sus relaciones como quedan.

Entidad A	Relación	Entidad B
Curso	Componen	Asignaturas

Un curso siempre lo componen asignaturas, hay que hacer constar que el curso puede tener tutor, tutores o no tener. De curso solo hay uno y de asignaturas lo componen varias, por lo tanto tenemos una relación 1:N. Las dos entidades que componen esta relación son obligatorias, ya que sin una no existiría la relación.

Entidad A	Relación	Entidad B
Curso	Matriculan	Alumnos

En curso se matriculan alumnos. Curso para matricularse solo hay uno según la edad del alumno y alumnos hay N, capacidad que decide el centro, tenemos una relación 1:N. Las dos entidades son obligatorias ya que sin ninguna de las dos no tendría sentido la otra.

Entidad A	Relación	Entidad B
Data, Hora	Calendario	Componen

La data y las hora junto con la entidad componen hacen el calendario. En el calendario nos dirá día y hora de la semana en el cual se celebra una asignatura en un curso y por un profesor. Las dos entidades son obligatorias, y la relación es 1:1.

Entidad A	Relación	Entidad B
Data, Hora	Visitas	Componen

Esta es una entidad con el mismo rol que la de calendario pero esto es para visitas. Las condiciones de la anterior valen para esta.

Entidad A	Relación	Entidad B	Entidad C
Alumnos	Amonestaciones	Componen	TipoDeAmonestaciones

Esta es una relación de 3 tablas, el alumno que se le amonesta, el curso asignatura y el profesor responsable. Estas dos relaciones son obligatorias y tienen relación de un alumno y N componen (se le puede sancionar en diferentes asignaturas). El Tipo de amonestación es obligatoria y también será N, quedando una relación 1:N:N. Debemos hace constar que cuando amonestamos marcamos una gravedad (que puede ser acumulable o total) esta se valida con la gravedad de la tabla regla de activación y si es mayor se activan el proceso sancionador. Que pasaremos a explicar en el punto siguiente.

Entidad A	Relación	Entidad B
TipoDeAmonestaciones	Reglaactivacion	TipoDeSancion

La regla de activación viene en relación con un tipo de amonestación e impone un tipo de sanción, todo ello bajo el campo gravedad de actividad que marca cuando se lanza un sanción automáticamente. Tenemos la obligatoriedad de tener una relación con la tabla tipo de amonestación y la de tipo de sanción estamos ante 1:1. Por petición del cliente se nos pide poder imponer una sanción manual es decir sin pasar por la amonestación, para lo cual nuestra base de datos obliga a crear un código en la tabla regla de activación en la cual nos deja esa sanción.

Entidad A	Relación	Entidad B	Entidad C	Entidad D
Alumno	Sancion	Curso	ReglaDeActivacion	TipoDeSanciones

Para generar la sanción debemos tener el alumno en relación con el curso lo activa una regla de activación en concordancia con un tipo de sanción. Para ello podemos generarla automáticamente será lo más normal o manual. La tabla regla de activación debe tener todos los códigos, y con el tipo de sanción ya establecida nos dará una sanción. Por lo tanto tenemos obligatorio un alumno en un curso con una regla de activación y un tipo de sanción. Estamos ante 1:1:1:1, debemos tener un código de cada tabla.

5.4 Atributos Entidades

Enumeraremos los atributos de cada una de las entidades.

- **Alumno**: (Datos personales del alumno)
  Idalumno: char, (clave primaria)
  Nombre: char, nombre del alumno
  Apellidos: char, apellidos del alumno
  Fechanacimiento: date, fecha de nacimiento del alumno
  Nacionalidad: char, nacionalidad del alumno
  Dirección: char, dirección del alumno
  Población: char, población del alumno
  Ccpostal: integer, codigo postal de la dirección del alumno
  Telefono: integer, teléfono del alumno
  Nompadre: char, nombre del padre
  Movilpadre: integer, movil del padre
  Emailpadre: char, email del padre
  Estado: booleano, estado del registro, ` `=activo - 'N'=baja
  Fechadebaja: date, fecha del registro de baja
  Motivobaja: char, motivo de la baja

- **Profesor:** (Datos personales del profesor)
  Dniprofesor: char, (clave primaria)
  Nombre: char, nombre del profesor
  Apellidos: char, apellidos del profesor
  Fechanacimiento: date, fecha nacimiento del profesor
  Direccion: char, dirección del profesor

Población: char, población del profesor
Pais: char, país del profesor
Cpostal: integer, código postal del domicilio del profesor
Telefono: integer, teléfono del profesor
Movil: integer, móvil del profesor
Email: char, email del profesor
Estado: booleano, estado del registro, ' '=activo - 'N'=baja
Fechadebaja: date, fecha del registro de baja
Motivobaja: char, motivo de la baja

- **Curso:** (Cursos disponibles en el centro)
  Anyacademico: char, (clave primaria)
  Idcurso: char, (clave primaria)
  Nombre: char, descripción del curso
  Dniprofesor: (clave foránea de profesor)

- **Asignatura:** (Asignaturas disponibles en el centro)
  Idasignatura: char, (clave primaria)
  Nombre: char, descripción de la asignatura

- **Tipos de amonestaciones:** (Tipos de amonestaciones)
  Idtamonestacion: char, (clave primaria)
  Descripción: char, descripción del tipo de amonestación

- **Tipo de sanción:** (Tipo de sanciones)
  Idtsancion: char, (clave primaria)
  Descripción: char, descripción del tipo de sanción

5.5 Diseño lógico

Nuestro siguiente paso es transformar las entidades en relaciones dentro de la base de datos, para ello interpretaremos sus relaciones y según las reglas nos dará un tipo de relación.

5.5.1 Transformación y resultado

Las entidades propias quedan como se han explicado en el tema conceptual dentro de la especificación de atributos.

Para las interrelaciones queda claro que serán las claves foráneas de las entidades propias para relacionarse entre ellas, a continuación pasamos a explicar como quedarán las relaciones.

- **Componen:** (Composición de los cursos asignaturas y profesor que imparte la asignatura)
  Anyacademico: (clave primaria, foránea de curso)
  Idcurso: (clave primaria, foránea de curso)

Idasignatura: (clave primaria, foránea de asignatura)
Dniprofesor: (clave primaria, foránea de profesor)

- **Matriculan:** (Alumnos que se matriculan en los cursos)
  NumExpediente: char, (clave primaria)
  Idalumno: clave foránea de alumno
  Anyacademico: clave foránea de curso
  Idcurso: clave foránea de curso

- **Calendario:** (Calendario de las asignaturas y cursos en el centro)
  Anyacademico: (clave primaria, foránea de componen)
  Idcurso: (clave primaria, foránea de componen)
  Idasignatura: (clave primaria, foránea de componen)
  Dniprofesor: (clave primaria, foránea de componen)
  Diasemana: char, (clave primaria), día de clase
  Horainici: integer, hora inicio asignatura del curso
  Horafin: integer, hora fin asignatura del curso

- **Visita:** (Visitas para los padre y alumnos a los profesores)
  Anyacademico: (clave primaria, foránea de componen)
  Idcurso: (clave primaria, foránea de componen)
  Idasignatura: (clave primaria, foránea de componen)
  Dniprofesor: (clave primaria, foránea de componen)
  Diasemana: char, (clave primaria), día de clase
  Horainici: integer, hora inicio asignatura del curso
  Horafin: integer, hora fin asignatura del curso

- **Amonestación:** (Amonestación que imponemos a los alumnos)
  Idamonestacion: char, (clave primaria)
  Idtamonestacion: (clave primaria, foránea de tipo de amonestación)
  Idalumno: (clave primaria, foránea de alumno)
  Anyacademico: (clave primaria, foránea de calendario)
  Idcurso: (clave primaria, foránea de calendario)
  Idasignatura: (clave primaria, foránea de calendario)
  Dniprofesor: (clave primaria, foránea de calendario)
  Diasemana: (clave primaria, foránea de calendario)
  Hora: integer, hora de la amonestación
  Gravedad: integer, gravedad de la amonestación acumulable
  Comunicación: booleano, informar a los padres, 'S' – 'N', defecto N

- **Regla de activacion:** (Reglas para activar sanciones)
  Ractivacion: char, (clave primaria)
  Idtamonestacion: (clave foránea de tipo de amonestación)
  Gradoactividad: integer, número por el que se activa las reglas
  Idtsancion: (clave foránea de tipo de sanción)

- **Sancion:** (Sanciones impuestas a los alumnos)

Idsancion:	char, (clave primaria)
Idalumno:	(clave primaria, foránea de alumno)
Anyacademico:	(clave foránea de curso)
Idcurso:	(clave foránea de curso)
Ractivacion:	(clave foránea regla de activación)
Idtsancion:	(clave foránea tipo de sanción)
Descripción:	char, descripción de la sanción
Fechasancion:	date, fecha impuesta sanción
Idamonestacion:	char, referencia de amonestación

5.6 Diseño físico

En este apartado buscaremos toda la creación de la base de datos. Esto incluye la instalación del oracle y su configuración. Implementando posteriormente toda la creación de las tablas.

5.6.1 Configuración oracle

Partimos de la base que ya tenemos la instalación realizada. Por lo tanto configuraremos la creación del tablespace para albergar nuestra base de datos, para ello seguiremos los siguientes pasos:

```
CREATE TABLESPACE TFC
DATAFILE 'C:\oraclexe\oradata\xe\TFC.DBF' SIZE 40 M
```

Posteriormente crearemos un usuario para gestionar toda la creación y funcionalidades de las tablas, mediante el mandato siguiente lo crearemos:

```
CREATE USER TFC
IDENTIFIED BY tfc
DEFAULT TABLESPACE tfc
QUOTA UNLIMITED ON tfc
TEMPORARY TABLESPACE temp;
```

Ahora le otorgaremos todos los permisos para poder gestionar la base de datos creada y poder crear todas las funcionalidades necesarias. El mandato siguiente se lo otorga:

```
GRANT CREATE SESSION,
CREATE TABLE,
CREATE VIEW,
CREATE TRIGGER,
CREATE SEQUENCE,
CREATE PROCEDURE
TO TFC;
```

5.6.2 Creación de tablas

Con todos los datos del diseño lógico procederemos a crear las tablas en oracle mediante las sentencias necesarias para ello.

El tipo y el tamaño de campo esta escogido para poder acoger todos los datos correctamente. Los datos obligatorios los marcamos como NOT NULL y para chequear algunos datos usamos la cláusula CHECK.

En el anexo y en los scripts del producto podrán ver la estructura de las tablas.

5.6.3 Índices

Oracle ya nos crea los índices asociados a la clave primaria, por lo tanto de momento no crearemos ningún índice para poder acceder a cualquier tabla más rápidamente.

5.6.4 Secuencia y Trigger Sancion

Explicamos que hemos creado una secuencia para el índice de la tabla y un trigger para tanto alta como modificación lo tenga en cuenta.

5.7 Estadísticas

Para poder trabajar con un modulo estadístico, crearemos las tablas relacionadas para poder trabajar en ellas, tanto para insertar los datos a partir de las tablas originales como para poder visualizar los datos con un simple Select.

Por lo tanto explicaremos las funcionalidades que debe tener el modulo estadístico y posteriormente el diseño lógico de cada una de las tablas.

Funcionalidades:

- Número amonestaciones por alumno.
- Número de sanciones por alumno y año
- Media de amonestaciones por profesor y año
- Número de sanciones por curso y año
- El nombre del alumno mas sancionado en un año determinado
- El nombre del profesor más amonestador por curso
- Media de sanciones de los alumnos por curso
- Número de alumnos sin amonestaciones

Creación de las tablas a partir de las funcionalidades con sus atributos

Est1: (Número de amonestaciones por alumno)
Idalumno: char, (clave primaria foránea de alumno)

Numeroamonestaciones: integer, número de amonestaciones por alumno

Est2: (Número de sanciones por alumno y año)
Anyacademico: char, (clave primaria)
Idalumno: char, (clave primaria foránea de alumno)
Numerosanciones: integer, número de sanciones por alumno

Est3: (Media amonestaciones por profesor y año)
Anyacademico: char, (clave primaria)
Dniprofesor: char, (clave primaria foránea de profesor)
Mediaamonestaciones: number(5,2), media de amonestaciones por profesor y año

Est4: (Número de sanciones por curso y año)
Anyacademico: char, (clave primaria foránea de curso)
Idcurso: char, (clave primaria foránea de curso)
Numerosanciones: integer, número de sanciones por curso

Est5: (Nombre del alumno más sancionado en un año)
Anyacademico: char, (clave primaria)
NomAlumno: char, (clave primaria), nombre del alumno más sancionado

Est6: (Nombre del profesor más amonestador por curso)
Anyacademico: char, (clave primaria foránea de curso)
Idcurso: char, (clave primaria foránea de curso)
Nomprofesor: char, nombre del profesor más amonestador

Est7: (Media de sanciones que tienen los alumnos por curso)
Anyacademico: char, (clave primaria foránea de curso)
Idcurso: char, (clave primaria foránea de curso)
Mediasanciones: number(5,2), media de sanciones alumnos por curso

Est8: (Número de alumnos que no tienen amonestaciones)
Codigo: char, (clave primaria)
Numeroalumnos: integer, número de alumnos sin amonestaciones

5.8 Logs

Todas las llamadas a los procedimientos las debemos guardar en una tabla de logs, en ella guardaremos también los parámetros de entrada y los de salida.

La tabla de logs quedara con estos campos:

Logs
Fecha: timestamp, DIA y fecha que grabamos registro

Procedimiento:	char, procedimiento que almacenado
Paramentrada:	char, parámetros de entrada (separados con coma)
Paramsalida:	char, parámetros de salida (dirá si es ok o error)

6. Implementación

El paso siguiente después de haber diseñado las tablas es la implementación para el buen funcionamiento de dichas tablas.

Para tal efecto usaremos los procedimientos almacenados. Estos procedimientos tal y como se nos piden dentro de las especificaciones tendrán un parámetro de entrada y salida, por el cual sabremos si dicho procedimientos ha finalizado correctamente o no. Usaremos una secuencia y un trigger para la tabla (sanción).

A continuación detallaremos los procedimientos y sus parámetros de entrada:

• Alta o Modificar profesor	• Alta o Modificar alumno
o dniprofesor o nombre o apellidos o fechanacimiento o nacionalidad o direccion o población o cpostal o telefono o movil o email o estado o fechadebaja o motivobaja	o idalumno o nombre o apellidos o fechanacimiento o direccion o población o cpostal o telefono o nompadre o movilpadre o email o estado o fechadebaja o motivobaja
• Baja profesor	• Baja alumno
o dniprofesor o estado o fechadebaja o motivobaja	o idalumno o estado o fechadebaja o motivobaja
• Alta o Modificar curso	• Alta o Modificar asignaturas
o anyacademico o idcurso o nombre o dniprofesor	o idasignatura o nombre
• Baja curso	• Baja asignaturas
o anyacademico o idcurso	o idasignatura
• Alta o Modificar calendario	• Alta o Modifica visita
o anyacademico o idcurso o idasignatura o dniprofesor o diasemana o horainicio	o anyacademico o idcurso o idasignatura o dniprofesor o diasemana o horainicio

o horafin	o horafin
• Baja calendario	• Baja visita
o anyacademico o idcurso o idasignatura o dniprofesor o diasemana	o anyacademico o idcurso o idasignatura o dniprofesor o diasemana
• Alta o Modificar tipo de amonestaciones	• Alta o Modificar tipo de sancion
o idtamonestacion o descripcion	o idtsancion o descripcion
• Baja tipo de amonestaciones	• Baja tipo de sancion
o idtamonestacion	o idtsancion
• Alta componen	• Baja componen
o anyacademico o idcurso o idasignatura o dniprofesor	o anyacademico o idcurso o idasignatura o dniprofesor
• Alta o Modificar matriculan	• Baja matriculan
o numexpediente o idcurso o idalumno	o numexpediente
• Alta o Modificar amonestacion	• Baja amonestacion
o idamonestacion o idtamonestacion o idalumno o idcurso o idasignatura o dniprofesor o fechaamonestacion o gravedad o comunicacion	o idamonestacion o idtamonestacion o idalumno o idcurso o idasignatura o dniprofesor o fechaamonestacion
• Alta o Modificar regla activacion	• Baja regla activacion
o ractivacion o idtamonestacion o gravedadactividad o idtsancion	o ractivacion
• Alta o Baja estadisticas 1	• Alta o Baja estadisticas 2
o idalumno	o anyacademico o idalumno
• Alta estadisticas 3	• Alta o Baja estadisticas 4
o anyacademico o dniprofesor	o anyacademico o idcurso
• Alta estadisticas 5	• Alta estadisticas 6
o anyacademico	o anyacademico o idcurso
• Alta estadisticas 7	• Alta estadisticas 8

o anyacademico o idcurso	ninguno

A continuación explicaremos los procedimientos con más análisis:

6.1 Alta profesor

Crearemos un profesor en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla profesor.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla profesor pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_altaprofesor y los parámetros de entrada son:

p_altaProfesor (

dni	IN Varchar2,
nom	IN Varchar2,
ape	IN Varchar2,
fnac	IN Date,
nac	IN Varchar2,
dire	IN Varchar2,
pobla	IN Varchar2,
cpos	IN Varchar2,
tel	IN Varchar2,
mov	IN Varchar2,
ema	IN Varchar2,
est	IN char,
fba	IN Date,
mob	IN Varchar2,
salida	OUT Varchar2)

Donde cada valor es:

dni:	dni del profesor
nom:	nombre profesor
ape:	apellidos del profesor
fnac:	nacionalidad profesor
dire:	dirección profesor
pobla:	población profesor
cpos:	código postal profesor
tel:	teléfono profesor
mov:	móvil profesor
ema:	e-mail profesor

est: ‘ ’
fba: ’31/12/1999’
mob: ‘ ’

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.2 Baja profesor

Crearemos una baja lógica del profesor en la base de datos.
Esto significa que el campo estado pasa a ‘b’ se pone la fecha del día y el motivo de la baja.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un update del registro en la tabla profesor. Buscándolo con el parámetro de entrada.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro de la tabla profesor pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaprofesor y los parámetros de entrada son:

p_bajaProfesor (
dni IN Varchar2,
est IN char,
fba IN Date,
mob IN Varchar2,
salida OUT Varchar2

Donde cada valor es:
dni: dni del profesor
est: ‘b’
fba: sysdate
mob: motivo de la baja. Texto

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.3 Modificación profesor

Modificaremos un registro de la tabla profesor en la base de datos.
Esto significa que cualquier campo de esta tabla puede ser modificado, al tener bajas lógicas también podemos activar un registro esto quiere decir dejar el campo estado a ` `, fecha '31/12/1999' y el motivo en ` `.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un update del registro en la tabla profesor. Buscándolo con el parámetro de entrada.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro de la tabla profesor pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_modifprofesor y los parámetros de entrada son:

p_modifProfesor (

dni	IN Varchar2,
nom	IN Varchar2,
ape	IN Varchar2,
fnac	IN Date,
nac	IN Varchar2,
dire	IN Varchar2,
pobla	IN Varchar2,
cpos	IN Varchar2,
tel	IN Varchar2,
mov	IN Varchar2,
ema	IN Varchar2,
est	IN char,
fba	IN Date,
mob	IN Varchar2,
salida	OUT Varchar2

Donde cada valor es:

dni:	dni del profesor
nom:	nombre profesor
ape:	apellidos del profesor
fnac:	nacionalidad profesor
dire:	dirección profesor
pobla:	población profesor
cpos:	código postal profesor
tel:	teléfono profesor
mov:	móvil profesor
ema:	e-mail profesor
est:	estado
fec:	fecha de baja o '31/12/1999'
mov:	motivo de la baja o ` `.

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.4 Alta alumno

Crearemos un alumno en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla alumno.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla alumno pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_altaAlumno y los parámetros de entrada son:

p_altaAlumno (

ida	IN Varchar2,
nom	IN Varchar2,
ape	IN Varchar2,
fnac	IN Date,
nac	IN Varchar2,
dire	IN Varchar2,
pobla	IN Varchar2,
cpos	IN Varchar2,
tel	IN Varchar2,
nompa	IN Varchar
movpa	IN Varchar2,
ema	IN Varchar2,
est	IN char,
fba	IN Date,
mob	IN Varchar2,
salida	OUT Varchar2)

Donde cada valor es:

Ida :	codigo alumno
nom:	nombre alumno
ape:	apellidos del alumno
fnac:	nacionalidad alumno
dire:	dirección alumno
pobla:	población alumno
cpos:	código postal alumno

tel: teléfono alumno
nompa nombre padre del alumno
movpa: móvil padre del alumno
ema: e-mail padre alumno
est: ‘ ’
fba: ’31/12/1999’
mob: ‘ ‘

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.5 Baja alumno

Crearemos una baja lógica del alumno en la base de datos.
Esto significa que el campo estado pasa a ‘b’ se pone la fecha del día y el motivo de la baja.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un update del registro en la tabla alumno. Buscándolo con el parámetro de entrada.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro de la tabla alumno pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaAlumno y los parámetros de entrada son:
p_bajaAlumno (
ida IN Varchar2,
est IN char,
fba IN Date,
mob IN Varchar2,
salida OUT Varchar2
Donde cada valor es:
ida: codigo del alumno
est: ‘b’
fba: sysdate
mob: motivo de la baja. Texto

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.6 Modificación alumno

Modificaremos un registro de la tabla alumno en la base de datos.
Esto significa que cualquier campo de esta tabla puede ser modificado, al tener bajas lógicas también podemos activar un registro esto quiere decir dejar el campo estado a ` `, fecha '31/12/1999' y el motivo en ` `.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un update del registro en la tabla alumno. Buscándolo con el parámetro de entrada.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro de la tabla alumno pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_modifAlumno y los parámetros de entrada son:

```
p_modifAlumno (
        ida         IN Varchar2,
        nom         IN Varchar2,
        ape         IN Varchar2,
        fnac        IN Date,
        nac         IN Varchar2,
        dire        IN Varchar2,
        pobla       IN Varchar2,
        cpos        IN Varchar2,
        tel         IN Varchar2,
        nompa       IN Varchar
        movpa       IN Varchar2,
        ema         IN Varchar2,
        est         IN char,
        fba         IN Date,
        mob         IN Varchar2,
        salida      OUT Varchar2 )
```

Donde cada valor es:

Ida :	codigo alumno
nom:	nombre alumno
ape:	apellidos del alumno
fnac:	nacionalidad alumno
dire:	dirección alumno
pobla:	población alumno
cpos:	código postal alumno
tel:	teléfono alumno

nompa nombre padre del alumno
movpa: móvil padre del alumno
ema: e-mail padre alumno
est: ' '
fba: '31/12/1999'
mob: ' '

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.7 Alta asignatura

Crearemos una asignatura en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla asignatura.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla asignatura pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_altaAsignatura y los parámetros de entrada son:
p_altaAsignatura (
ida IN Varchar2,
nom IN Varchar2,
salida OUT Varchar2)
Donde cada valor es:
ida: codigo de la asignatura
nom: nombre de la asignatura
El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.8 Baja asignatura

Eliminaremos una asignatura en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla asignatura.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla asignatura pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaAsignatura y los parámetros de entrada son:

```
p_bajaAsignatura (
        ida         IN Varchar2,
        salida      OUT Varchar2 )
```

Donde cada valor es:

ida: codigo de la asignatura

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.9 Modificación asignatura

Modificaremos una asignatura en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla asignatura.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla asignatura pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_modifAsignatura y los parámetros de entrada son:

```
p_modifAsignatura (
        ida         IN Varchar2,
        nom         IN Varchar2,
        salida      OUT Varchar2 )
```

Donde cada valor es:

ida: codigo de la asignatura
nom: nombre de la asignatura

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.10 Alta Tipo de Sanción

Crearemos una tipo de sanción en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla tipodesancion.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla tipodesanción pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_altaTsancion y los parámetros de entrada son:

p_altaTsancion (
idt IN Varchar2,
des IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

idt: codigo del tipo de sanción
des: descripción del tipo de sanción

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.11 Baja Tipo de sanción

Eliminaremos un Tipo de sanción en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla Tipo de sanción.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla tipo de sancion pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaTsancion y los parámetros de entrada son:

p_bajaTsancion(

idt IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

idT: codigo del tipo de sancion

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.12 Modificación tipo de sanción

Modificaremos un tipo de sanción en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla tipo de sanción.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla tipo de sanción pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_modifTsancion y los parámetros de entrada son:

p_modifTsancion (
idt IN Varchar2,
des IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

idt: codigo del tipo de sanción
des: nombre del tipo de sanción

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.13 Alta Tipo de Amonestación

Crearemos una tipo de amonestación en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla tipodeamonestacion.

c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla tipodeamonestacion pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_altaTamonestacion y los parámetros de entrada son:

p_altaTamonestacion (
idta IN Varchar2,
des IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:
idta: codigo del tipo de amonestación
des: descripción del tipo de amonestación

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.14 Baja Tipo de amonestación

Eliminaremos un Tipo de amonestación en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla Tipo de amonestación.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla tipo de amonestación pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaTamonestacion y los parámetros de entrada son:

p_bajaTamonestacion(
idta IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:
idta: codigo del tipo de amonestacion

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.15 Modificación tipo de amonestación

Modificaremos un tipo de amonestación en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla tipo de amonestación.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla tipo de amonestación pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_modifTamonestacion y los parámetros de entrada son:

p_modifTamonestacion (
idta IN Varchar2,
des IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:
idta: codigo del tipo de amonestación
des: nombre del tipo de amonestación

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.16 Alta Curso

Crearemos un curso en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla curso.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla curso pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla profesor valida la existencia del dni del profesor.

El procedimiento se llama p_altaCurso y los parámetros de entrada son:

p_altaCurso (
ano IN Varchar2,
idc IN Varchar2,
nom IN Varchar2,
dni IN Varchar2,
salida OUT Varchar2)
Donde cada valor es:
ano: año academico
idc: codigo curso
nom: nombre del curso
dni: dni profesor tutor del curso
El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.17 Baja Curso

Eliminaremos un curso de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla curso.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla curso pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaCurso y los parámetros de entrada son:
p_bajaCurso(
ano IN Varchar2,
idc IN Varchar2,
salida OUT Varchar2)
Donde cada valor es:
ano: año academico
idc: codigo del curso
El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.18 Modificación curso

Modificaremos un curso de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla curso.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla curso pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla profesor valida la existencia del dni del profesor.

El procedimiento se llama p_modifCurso y los parámetros de entrada son:

```
p_modifCurso (
        ano         IN Varchar2,
        idc         IN Varchar2,
        nom         IN Varchar2,
        dni         IN Varchar2,
        salida      OUT Varchar2 )
```

Donde cada valor es:

ano: año academico
idc: codigo curso
nom: nombre del curso
dni: dni profesor tutor del curso

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.19 Alta Componen

Crearemos una composición en la base de datos. Esta composición no tendrá modificación dado que solo crearemos y borraremos al ser todo claves.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla composición.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla composición pero si crearemos un registro en la tabla log con todos los datos. La relación con

diferentes tablas curso, asignatura y profesor valida la existencia de las claves.

El procedimiento se llama p_altaComposicion y los parámetros de entrada son:

p_altaComposicion (
ano IN Varchar2,
idc IN Varchar2,
ida IN Varchar2,
dni IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:
ano: año academico
idc: codigo curso
ida: codigo asignatura
dni: dni profesor

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.20 Baja Componen

Eliminaremos una composición de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla componen.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla componen pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaComponen y los parámetros de entrada son:

p_bajaComponen(
ano IN Varchar2,
idc IN Varchar2,
ida IN Varchar2,
dni IN Varchar2
salida OUT Varchar2)

Donde cada valor es:
ano: año academico
idc: codigo del curso
ida: codigo asignatura
dni: dni profesor

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.21 Alta Calendario

Crearemos un calendario en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla calendario.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla calendario pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla componen valida la existencia de las claves.

El procedimiento se llama p_altaCalendario y los parámetros de entrada son:

p_altaCalendario (

ano	IN Varchar2,
idc	IN Varchar2,
ida	IN Varchar2,
dni	IN Varchar2,
dia	IN Varchar2,
hin	IN Integer,
hfi	IN Integer,
salida	OUT Varchar2)

Donde cada valor es:

ano:	año academico
idc:	codigo curso
ida:	codigo asignatura
dni:	dni profesor
dia:	dia de la semana
hin:	hora de inicio
hfi:	hora de fin

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.22 Baja Calendario

Eliminaremos un calendario de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla calendario.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla calendario pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaCalendario y los parámetros de entrada son:

p_bajaCalendario(
ano IN Varchar2,
idc IN Varchar2,
ida IN Varchar2,
dni IN Varchar2,
dia IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

ano: año academico
idc: codigo del curso
ida: codigo asignatura
dni: dni profesor
dia: dia de la semana

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.23 Modificación calendario

Modificaremos un calendario de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla calendario.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla curso pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla componen valida la existencia de las claves.

El procedimiento se llama p_modifCalendario y los parámetros de entrada son:

p_modifCalendario (

ano	IN Varchar2,
idc	IN Varchar2,
ida	IN Varchar2,
dni	IN Varchar2,
dia	IN Varchar2,
hin	IN Integer,
hfi	IN Integer,
salida	OUT Varchar2)

Donde cada valor es:

ano:	año academico
idc:	codigo curso
ida:	codigo asignatura
dni:	dni profesor tutor del curso
dia:	dia de la semana
hin:	hora de inicio
hfi:	hora de fin

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.24 Alta Visita

Crearemos una visita en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
a) Insertaremos un registro en la tabla visita.
b) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
c) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla visita pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla componen valida la existencia de las claves.

El procedimiento se llama p_altaVisita y los parámetros de entrada son:

p_altaVisita (

ano	IN Varchar2,
idc	IN Varchar2,
ida	IN Varchar2,
dni	IN Varchar2,
dia	IN Varchar2,

hin IN Integer,
hfi IN Integer,
salida OUT Varchar2)

Donde cada valor es:

ano: año academico
idc: codigo curso
ida: codigo asignatura
dni: dni profesor
dia: dia de la semana
hin: hora de inicio
hfi: hora de fin

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.25 Baja Visita

Eliminaremos una visita de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla visita.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla visita pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaVisita y los parámetros de entrada son:

p_bajaVisita(
ano IN Varchar2,
idc IN Varchar2,
ida IN Varchar2,
dni IN Varchar2,
dia IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

ano: año academico
idc: codigo curso
ida: codigo asignatura
dni: dni profesor
dia: dia de la semana

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.26 Modificación Visita

Modificaremos una visita de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla visita.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla visita pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla componen valida la existencia de las claves.

El procedimiento se llama p_modifVisita y los parámetros de entrada son:

p_modifVisita (

ano	IN Varchar2,
idc	IN Varchar2,
ida	IN Varchar2,
dni	IN Varchar2,
dia	IN Varchar2,
hin	IN Integer,
hfi	IN Integer,
salida	OUT Varchar2)

Donde cada valor es:

ano:	año academico
idc:	codigo curso
ida:	codigo asignatura
dni:	dni profesor
dia:	dia de la semana
hin:	hora de inicio
hfi:	hora de fin

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.27 Alta Matriculan

Crearemos una matriculación en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla matriculan.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla matriculan pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla alumno y curso valida la existencia de las claves.

El procedimiento se llama p_altaMatriculan y los parámetros de entrada son:

p_altaMatriculan (
nex IN Varchar2,
ida IN Varchar2,
ano IN Varchar2,
idc IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

nex: numero expediente
ida: codigo alumno
ano: año academico
idc: codigo curso

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.28 Baja Matriculan

Eliminaremos una matricula de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla matricula.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla matriculan pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaMatriculan y los parámetros de entrada son:

p_bajaMatriculan(
nex IN Varchar2,

salida OUT Varchar2)

Donde cada valor es:

nex: número expediente

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.29 Modificación Matriculan

Modificaremos una matricula de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla matriculan.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla matriculan pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla componen valida la existencia de las claves.

El procedimiento se llama p_modifMatriculan y los parámetros de entrada son:

p_modifMatriculan (

nex IN Varchar2,
ida IN Varchar2,
ano IN Varchar2,
idc IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

nex: numero expediente
ida: codigo alumno
ano: año academico
idc: codigo curso

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.30 Alta ReglaActivacion

Crearemos una regla de activación en la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla regla de activación.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla regla activación pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla tipodeamonestaciones y tipodesancion valida la existencia de las claves.

El procedimiento se llama p_altaRactivacion y los parámetros de entrada son:

p_altaRactivacion (
rac IN Varchar2,
idt IN Varchar2,
gra IN Integer,
ids IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:
rac: regla de activación
idt: tipo de amonestacion
gra: gravedad actividad
ids: tipo de sanción

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.30 Baja ReglaActivacion

Eliminaremos una regla activacion de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla reglaactivacion.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla reglaactivacion pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaRactivacion y los parámetros de entrada son:

p_bajaRactivacion(
rac IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

rac: regla activación

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.30 Modificar ReglaActivacion

Modificaremos una regla activación de la base de datos.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla regla activación.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla regla activacion pero si crearemos un registro en la tabla log con todos los datos. La relación con la tabla tipo de amonestación y tipo de sanción valida la existencia de las claves.

El procedimiento se llama p_modifRactivacion y los parámetros de entrada son:

p_modifRactivacion (

rac IN Varchar2,
idt IN Varchar2,
gra IN Integer,
ids IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

rac: regla de activación
idt: tipo de amonestacion
gra: gravedad actividad
ids: tipo de sanción

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.30 Alta Amonestación

Crearemos una amonestación en la base de datos. Este procedimiento es la pieza angular de todo el aplicativo, la creación de un registro en esta tabla dispara todas las creaciones en el resto de tablas de estadísticas referentes

amonestaciones. Pero también crea automáticamente una sanción siempre y cuando la regla de activación lo indique. La Regla de activación es la tabla con la cual valida la existencia del campo gravedad para poder activar esas sanciones y permite gestionar mejor el aplicativo para decidir cuando debe aplicarse una sanción según el tipo de amonestación y con el tipo de sanción según la gravedad y la amonestada. Hemos creado una secuencia para la creación del identificador de la sanción

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Insertaremos un registro en la tabla amonestación. Desde esta creación validaremos si la gravedad con la tabla regla de activacion es mayor o igual y entonces daremos de alta una sanción con las reglas estipuladas, la creación manual de un registro en la tabla sancion con el código del tipo de activacion manual.
   Desde aquí crearemos todas las tablas estadísticas relacionadas con las amonestaciones y sanciones. Estas tablas son las EST1..8.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no crearemos registro en la tabla amonestacion pero si crearemos un registro en la tabla log con todos los datos. La relación con las diferentes tablas valida la existencia de las claves.

El procedimiento se llama p_altaAmonestacion y los parámetros de entrada son:

p_altaAmonestacion (

idm	IN Varchar2,
idt	IN Varchar2,
ida	IN Varchar2,
ano	IN Varchar2,
idc	IN Varchar2,
ids	IN Varchar2,
dni	IN Varchar2,
dia	IN Varchar2,
hor	IN Integer,
gra	IN Integer,
com	IN Char,
des	IN Varchar2,
salida	OUT Varchar2

Donde cada valor es:

Idm:	codigo de amonestación
Idt:	codigo tipo de amonestación
Ida:	codigo alumno
ano:	año academico
idc:	codigo curso

ids: codigo asignatura
dni: dni profesor
dia: dia amonestado
hor: hora amonestado
gra: gravedad amonestación
com: comunicación a los padres si o no.
des: descripción de la amonestación si viene en blanco ponemos la del tipo de sancion.

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.31 Baja Amonestación

Eliminaremos una amonestación de la base de datos. La misma explicación del alta la aplicaremos a la baja. Indudablemente toda la creación al darse de baja debe echarse para atrás. Por eso debemos tener las mismas funcionalidades pero a la inversa debemos borrar la sanción acarreada por la amonestación esto lo logramos poniendo en la sanción que amonestación la origino y así la podemos eliminar. Todas las tablas estadísticas deben ser calculadas de nuevo dado que tenemos menos elementos. Por eso hay algunas tablas que deben restar sus números y otras la misma alta nos sirve por que calcula con los nuevos valores de amonestaciones o sanciones.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Realizaremos un delete del registro en la tabla amonestación. También borraremos el tipo de sanción impuesta por esta amonestación si la tuviese. Volvemos a calcular todas las tablas estadisticas.
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no borraremos el registro en la tabla amonestación pero si crearemos un registro en la tabla log con todos los datos.

El procedimiento se llama p_bajaAmonestacion y los parámetros de entrada son:

p_bajaAmonestacion(
idm IN Varchar2,
idt IN Varchar2,
ida IN Varchar2,
ano IN Varchar2,
idc IN Varchar2,
ids IN Varchar2,

dni IN Varchar2,
dia IN Varchar2,
salida OUT Varchar2)

Donde cada valor es:

Idm: codigo de amonestación
Idt: codigo tipo de amonestación
Ida: codigo alumno
ano: año academico
idc: codigo curso
ids: codigo asignatura
dni: dni profesor
dia: dia amonestado

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.32 Modificación Amonestación

Modificaremos una amonestación de la base de datos. Este proceso como no borra registros, solo modifica ciertos campos como son la hora o si ha sido comunicado a los padres o no, no afecta a la tabla sanción ni a las estadisticas.

Para dicho procedimiento siempre comprobaremos una serie de reglas para los parámetros de entrada.

a) La clave siempre debe estar informada e informados todos los datos, en caso contrario se lanza una excepción.
b) Updatearemos un registro en la tabla amonestación. Modificaremos la tabla sanción si esta amonestación hubiese acarreado una sanción
c) Si todo es correcto el parámetro de retorno es ok, e insertamos un registro en la tabla log.
d) En caso de excepciones el parámetro de retorno será el motivo de la excepción, no modificaremos el registro en la tabla amonestación pero si crearemos un registro en la tabla log con todos los datos. La relación con las diferentes tablas valida la existencia de las claves.

El procedimiento se llama p_modifAmonestacion y los parámetros de entrada son:

p_modifAmonestacion (
idm IN Varchar2,
idt IN Varchar2,
ida IN Varchar2,
ano IN Varchar2,
idc IN Varchar2,
ids IN Varchar2,
dni IN Varchar2,
dia IN Varchar2,

hor IN Integer,
gra IN Integer,
com IN Char,
salida OUT Varchar2)

Donde cada valor es:

Idm: codigo de amonestación
Idt: codigo tipo de amonestación
Ida: codigo alumno
ano: año academico
idc: codigo curso
ids: codigo asignatura
dni: dni profesor
dia: dia amonestado
hor: hora amonestado
gra: gravedad amonestación
com: comunicación a los padres si o no.

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

6.33 Estadísticas

Todas las estadísticas se crean al crearse o borrarse una amonestación o sanción. Dentro de su procedure activa cada uno de sus procesos para poder posteriormente con un simple select ver cada uno de sus valores.

Por lo tanto cada ESTX posee los campos necesarios para poder observar cada consulta.

Para crear la tabla EST1, este proceso se activa cuando se da de alta una amonestación, automáticamente se crea un registro indicándonos el número de amonestaciones que tiene cada alumno amonestado. Para la baja es lo mismo pero disminuiremos el número de amonestaciones en caso de quedarse a cero borraremos el registro. Para el alta y la baja los parámetros son los mismos.

AltaEST1 o bajaEST1 (
ida IN Varchar2,
salida2 OUT Varchar2)

Donde cada valor es:

ida: el código del alumno.

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST2, se activa al dar de alta una sanción y en ella tendremos las sanciones impuestas a un alumno en un año. Si lo damos de baja esa sanción disminuye el número de sanciones si llega a cero se borra.
Para el alta y la baja los parámetros son los mismos.

AltaEST2 o bajaEST2 (
ano IN Varchar2
ida IN Varchar2,
salida2 OUT Varchar2)

Donde cada valor es:
ano: año academico
ida: el código del alumno.

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST3, cuando creemos o borremos un registro en la tabla amonestación tendremos un proceso que recalcula las medias de amonestaciones de los profesores por año. En este caso solo tenemos un *altaEST3* (
ano IN Varchar2,
dni IN Varchar2,
salida2 OUT Varchar2).

Donde cada valor es:
ano: año academico
dni: dni de los profesores.

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST4, cuando insertemos o borremos un registro en sanción deberemos sumar o restar el numero de sanciones por curso y año académico.
altaEST4 o bajaEST4 (
ano IN Varchar2,
idci IN Varchar2,
salida2 OUT Varchar2).

Donde cada valor es:
ano: año academico
idc: codigo del curso.

El parámetro de salida será:
Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST5, al crear o borrar una sanción recalcularemos la tabla por lo tanto tenemos solo un proceso para alta y baja. En este proceso debemos buscar el nombre del alumno en la tabla alumno. En este caso solo tenemos un *altaEST5* (

ano IN Varchar2,
salida2 OUT Varchar2).

Donde cada valor es:

ano: año academico

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST6, deberemos recalcular cada vez que haya una amonestación por lo tanto es un proceso único para llenar la tabla. Cuando tengamos el dni debeos buscar el nombre del profesor en su tabla. En este caso solo tenemos un *altaEST6* (

ano IN Varchar2,
idc IN Varchar2,
salida2 OUT Varchar2).

Donde cada valor es:

ano: año academico
idc: código del curso

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST7, deberemos recalcular cada vez que haya una sanción por lo tanto es un proceso único para llenar la tabla. Debemos calcular el número total de sanciones y sacar ala media por la del curso. En este caso solo tenemos un *altaEST7* (

ano IN Varchar2,
idc IN Varchar2,
salida2 OUT Varchar2).

Donde cada valor es:

ano: año academico
idc: código del curso

El parámetro de salida será:

Ok: si todo ha sido correcto.
Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

Para crear la tabla EST8, se calcula cuando se crea una amonestación, y en este caso lo único que haces es comparar con el total de alumnos y nos dice el número de alumnos, que no tienen ninguna amonestación. . En este caso solo tenemos un *altaEST8* (

salida2 OUT Varchar2).

El parámetro de salida será:

Ok: si todo ha sido correcto.

Error: + Descripción del error, si tenemos alguna excepción.

En el anexo tendremos todo el código.

7. Pruebas

Para verificar el correcto funcionamiento de todo el aplicativo, hemos creado unos datos acorde con ello. Esto es un script en el cual ejecuta todos procedimientos con datos previamente preparados para poder verificar la mayor parte del sistema.

Esta carga es solo para verificar el buen funcionamiento del aplicativo, y poder implementarlo para que se vea el Testing con los mensajes mostrados y creados en nuestra tabla de Logs.

7.1 Secuencia de pruebas

Indudablemente las pruebas tienen que tener un orden para poder llenar todas las tablas con la relación que nuestra base de datos nos demanda. Por ello a continuación explicaremos un poco la filosofía que hemos seguido. Los datos indudablemente van dirigidos a un testing por lo tanto no son acciones y datos muy reales, la finalidad es ver que ante posibles errores el sistema esta preparado para mostrar sus excepciones.

Las pruebas que realizaremos serán casi idénticas para las diferentes tablas.
Esto es dar de alta, de baja y modificar un registro.

Pruebas registro:

- Daremos de alta un registro, verificamos su creación en la tabla y el contenido de la tabla logs. En procedimientos como el de amonestaciones verificamos también la creación de las tablas estadísticas y en caso de activarse las sanciones la creación de la tabla sanción. Las excepciones en este apartado son repetir la creación, dejar un campo clave en blanco o nulo y verificar en las tablas de relación que una clave foránea no exista.
- Modificar un registro, verificamos la modificación de algún campo en la tabla y la creación de la tabla logs. Las excepciones serán validar la existencia de la clave, que no vengan en blanco o nulo que las relaciones con claves foráneas no existan.
- Dar de baja un registro, verificamos que lo borre de la tabla o en las entidades alumnos y profesores simplemente updatea el registro poniendo el campo estado a baja e informado la fecha y el motivo de la baja, tiene que crear un registro en la tabla logs. En la tabla amonestación como ocurre en el alta verificaremos que borre o recalcule las estadísticas y en su caso borre las sanciones. Las excepciones serán verificar existencia de la clave para poder dar de baja, que no este la clave en blanco o nulo.

Para verificar los procedimientos tenemos la tabla logs que nos muestra en su paramsalida si todo ha ido bien con un ok y en caso de alguna excepción marca "Error + la descripción del error".

Después de estas pruebas, borraremos todos los datos e insertaremos otra batería de datos en un script para poder probar las estadística con muchos registros y poder verificar sus resultados.

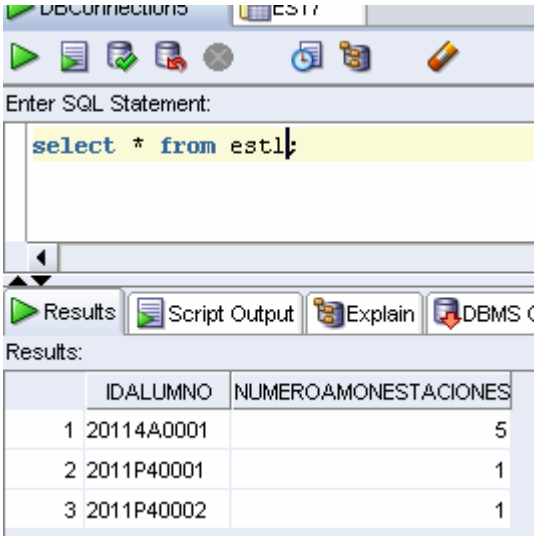
TABLA LOGS			
FECHA	PROCEDIMIENTO	PARAMETROS DE ENTRADA	PARAMETROS DE SALIDA
13/05/11 19:45:49	p_altaAlumno	20114A0001,Oscar,Beltran Pozas,13/11/01,Española,C/Vistamar 33, Blanes, 17300, 972337327,Oscar Beltran Campamelos, 647959859, oscar.beltran01@hotmail.es, , , ,	OK
13/05/11 19:46:39	p_altaAlumno	20114A0001,Oscar,Beltran Pozas,13/11/01,Española,C/Vistamar 33, Blanes, 17300, 972337327,Oscar Beltran Campamelos, 647959859, oscar.beltran01@hotmail.es, , , ,	ERROR: El alumno ya existe
13/05/11 19:47:15	p_altaAlumno	,Oscar,Beltran Pozas,13/11/01,Española,C/Vistamar 33, Blanes, 17300, 972337327,Oscar Beltran Campamelos, 647959859, oscar.beltran01@hotmail.es, , , ,	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:51:19	p_modifAlumno	20114A0001,Oscar,Beltran Pozas,13/11/01,Española,C/Mas Massonet 9 baix 1ª, Blanes, 17300, 972337327,Oscar Beltran Campamelos, 647959859, oscar.beltran01@hotmail.es, , , ,	OK
13/05/11 19:52:43	p_modifAlumno	99994A0001,Oscar,Beltran Pozas,13/11/01,Española,C/Mas Massonet 9 baix 1ª, Blanes, 17300, 972337327,Oscar Beltran Campamelos, 647959859, oscar.beltran01@hotmail.es, , , ,	ERROR: El alumno no existe
13/05/11 19:54:15	p_bajaAlumno	2011P40001, , , Le doy de baja por malo	OK
13/05/11 19:55:00	p_altaProfesor	17727433E,Oscar,Beltran Campamelos,19/06/68,Española,C/Vistamar 33,Blanes,17300,972337327,647959859,oscar.beltran77@hotmail.es, , , ,	OK
13/05/11 19:55:00	p_altaProfesor	17727433E,Oscar,Beltran Campamelos,19/06/68,Española,C/Vistamar 33,Blanes,17300,972337327,647959859,oscar.beltran77@hotmail.es, , , ,	ERROR: El profesor ya existe
13/05/11 19:55:00	p_altaProfesor	,Oscar,Beltran Campamelos,19/06/68,Española,C/Vistamar 33,Blanes,17300,972337327,647959859,oscar.beltran77@hotmail.es, , , ,	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:55:00	p_modifProfesor	17727433E, Oscar2, Beltran Campamelos, 19/06/1968, Española, C/Mas Masonnet 9 baix 1ª, Blanes, 17300, 972337327,647959859,oscar.beltran77@hotmail.es, , , ,	OK
13/05/11 19:55:00	p_bajaProfesor	17727433E, , , , Baja por enfermadad	OK
13/05/11 19:55:00	p_bajaProfesor	11111111E, , , , Baja por enfermadad	ERROR: No existe profesor
13/05/11 19:55:00	p_altaAsignatura	01, Matematicas	OK
13/05/11 19:55:00	p_altaAsignatura	01, Matematicas	ERROR: La asignatura ya existe
13/05/11 19:55:00	p_bajaAsignatura	01	OK
13/05/11 19:55:00	p_bajaAsignatura	66	ERROR: La asignatura no existe
13/05/11 19:55:00	p_modifAsignatura	01, MatematicasII	OK
13/05/11 19:55:00	p_modifAsignatura	, MatematicasII	ERROR: No se han rellenado los parámetros necesarios

13/05/11 19:55:00	p_altaCurso	2011, 4A, Cuarto A, 17727433E	OK
13/05/11 19:55:00	p_altaCurso	2011, 4A, Cuarto B, 17727433E	ERROR: El Curso ya existe
13/05/11 19:55:00	p_altaCurso	2011, , Cuarto A, 17727433E	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:55:00	p_bajaCurso	2011, 4A	OK
13/05/11 19:55:00	p_bajaCurso	2010, 4A	ERROR: No existe el Curso
13/05/11 19:55:00	p_modifCurso	2011, 4A, Cuarto A, 888888888E	ERROR: Error genérico al modificar un curso (no existe dni profesor en tabla profesor)
13/05/11 19:55:00	p_modifCurso	2011, 4A, Cuarto A, 999999999E	OK
13/05/11 19:55:00	p_altaTamonestacion	allegatard, Alumno llega tarde	OK
13/05/11 19:55:00	p_altaTamonestacion	allegatard, Alumno llega tarde	ERROR: El Tipo Amonestación ya existe
13/05/11 19:55:00	p_bajaTamonestacion	allegatard, Alumno llega tarde	OK
13/05/11 19:55:00	p_bajaTamonestacion	aaaaa	ERROR: No existe el Tipo de amonestación
13/05/11 19:55:00	p_modifTamonestacion	amalhabla, Alumno es mal hablado	OK
13/05/11 19:55:00	p_modifTamonestacion	, Alumno es mal hablado	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:55:00	p_altaTsancion	1hextraest,Quedarse una hora de estudio extra toda una semana	OK
13/05/11 19:55:00	p_altaTsancion	1hextraest,Quedarse una hora de estudio extra toda una semana	ERROR: El Tipo Sanción ya existe
13/05/11 19:55:00	p_altaTsancion	,Quedarse una hora de estudio extra toda una semana	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:55:00	p_bajaTsancion	1hextraes	OK
13/05/11 19:55:00	p_bajaTsancion	01	ERROR: No existe el Tipo de sanción
13/05/11 19:55:00	p_modifTsancion	1hextraest, Quedarse una hora de estudio extra toda la semana	OK
13/05/11	p_modifTsancion	98, Matematicas	ERROR: No existe el Tipo de

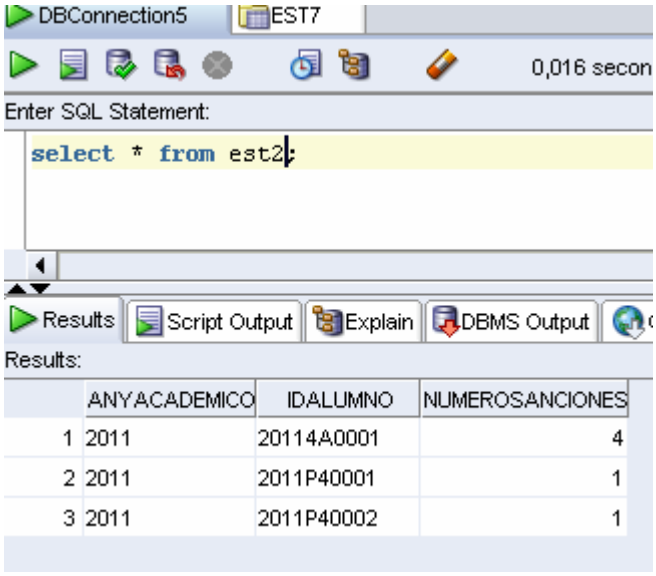
19:55:00			sanción
13/05/11 19:55:00	p_altaComponen	2011, 4A, 01, 17727433E	OK
13/05/11 19:55:00	p_altaComponen	2011, 4A, 01, 17727433E	ERROR: La Composición ya existe
13/05/11 19:55:00	p_bajaComponen	2011, 4A, 01, 17727433E	OK
13/05/11 19:55:00	p_bajaComponen	2099, 4A, 01, 17727433E	ERROR: No existe la Composición
13/05/11 19:55:00	p_altaMatriculan	00001,20114A0001,2011,4A	OK
13/05/11 19:55:00	p_altaMatriculan	00001,20114A0001,2011,4A	ERROR: La Matriculación ya existe
13/05/11 19:55:00	p_bajaMatriculan	00001,	OK
13/05/11 19:55:00	p_modifMatriculan	00001,20114A0001,2011,4B	OK
13/05/11 19:55:00	p_modifMatriculan	00099,20114A0001,2011,4B	ERROR: No existe la matricula
13/05/11 19:55:00	p_altaCalendario	2011, 4A, 01, 17727433E, Lunes, 9, 10	OK
13/05/11 19:55:00	p_altaCalendario	2011, 4A, 01, 17727433E, Lunes, 9, 10	ERROR: El Calendario ya existe
13/05/11 19:55:00	p_altaCalendario	2011, 99, 01, 17727433E, Lunes, 9, 10	ERROR: Error genérico al modificar un calendario(no existe el curso en tabla curso)
13/05/11 19:55:00	p_bajaCalendario	2011, 4A, 01, 17727433E, Lunes	OK
13/05/11 19:55:00	p_bajaCalendario	2011, 99, 01, 17727433E, Lunes	ERROR: No existe el Calendario
13/05/11 19:55:00	p_modifCalendario	2011, 4A, 01, 17727433E, Lunes, 9, 12	OK
13/05/11 19:55:00	p_modifCalendario	2011, , 01, 17727433E, Lunes, 9, 12	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:55:00	p_altaVisita	2011, 4A, 01, 17727433E, Lunes, 9, 10	OK

13/05/11 19:55:00	p_altaVisita	2011, 4A, 01, 17727433E, Lunes, 9, 10	ERROR: La visita ya existe
13/05/11 19:55:00	p_bajaVisita	2011, 4A, 01, 17727433E, Lunes	OK
13/05/11 19:55:00	p_modifVisita	2011, 4A, 01, 17727433E, Lunes, 9, 12	OK
13/05/11 19:55:00	p_modifVisita	2011, 99, 01, 17727433E, Lunes, 9, 12	ERROR: No existe la Visita
13/05/11 19:55:00	p_altaRactivacion	1,allegatard,3,1hextraest	OK
13/05/11 19:55:00	p_altaRactivacion	1,allegatard,3,1hextraest	ERROR: La regla Activación ya existe
13/05/11 19:55:00	p_bajaRactivacion	2, rst	OK
13/05/11 19:55:00	p_bajaRactivacion	9, rst	ERROR: No existe la regla de activación
13/05/11 19:55:00	p_modifRactivacion	2, aruido, 4, 1hextraest	OK
13/05/11 19:55:00	p_altaAmonestacion	00001, allegatard, 20114A0001, 2011, 4A, 01, 17727433E, Lunes, 9, 3, n, ,	OK
13/05/11 19:55:00	P_altaAmonestacion	00001, allegatard, 20114A0001, 2011, 4A, 01, 17727433E, Lunes, 9, 3, n, ,	ERROR: La amonestacion ya existe
13/05/11 19:55:00	P_altaAmonestacion	00001, , 20114A0001, 2011, 4A, 01, 17727433E, Lunes, 9, 3, n, ,	ERROR: No se han rellenado los parámetros necesarios
13/05/11 19:55:00	p_altaAmonestacion	00001, alalallala, 20114A0001, 2011, 4A, 01, 17727433E, Lunes, 9, 3, n, ,	ERROR: Error genérico al dar de alta una amonestacion(no existe el tipo amonestacion en tabla tipo amonestacion)
13/05/11 19:55:00	p_bajaAmonestacion	00001, allegatard, 20114A0001, 2011, 4A, 01, 17727433E, Lunes	OK
13/05/11 19:55:00	p_bajaAmonestacion	00001, allegatard, 20114A0001, 2011, 4A, 01, 17727433E, Lunes	ERROR: No existe la Amonestación
13/05/11 19:55:00	p_modifAmonestacion	00001, allegatard, 20114A0001, 2011, 4A, 01, 17727433E, Lunes, 9, 3, n, Se porta muy mal,	OK

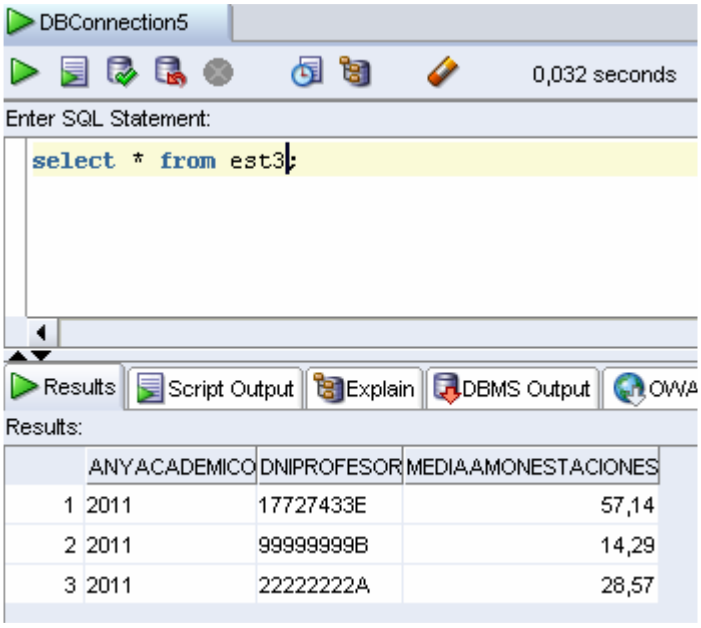
A continuación mostraré los select simples de las pruebas de estadísticas, desde un print de pantalla del developer



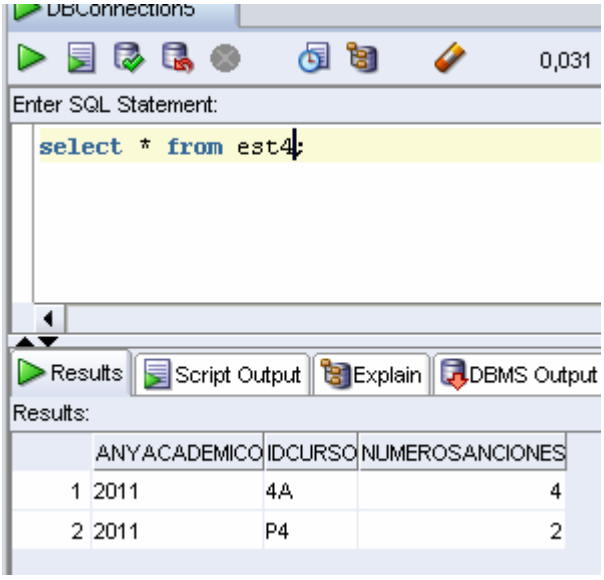
	IDALUMNO	NUMEROAMONESTACIONES
1	20114A0001	5
2	2011P40001	1
3	2011P40002	1



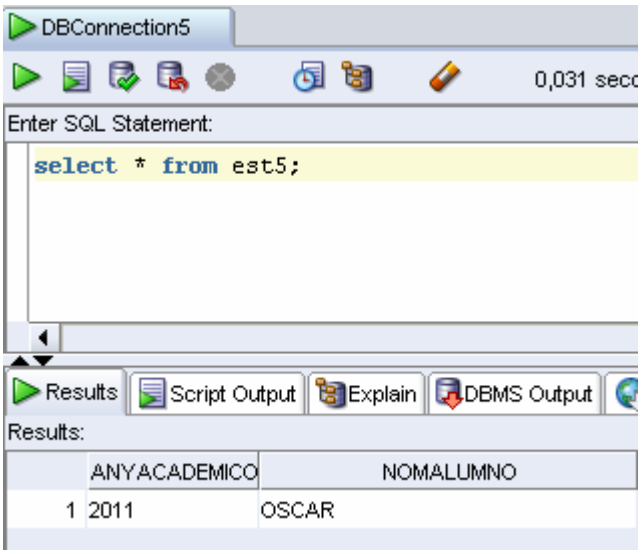
	ANYACADEMICO	IDALUMNO	NUMEROSANCIONES
1	2011	20114A0001	4
2	2011	2011P40001	1
3	2011	2011P40002	1

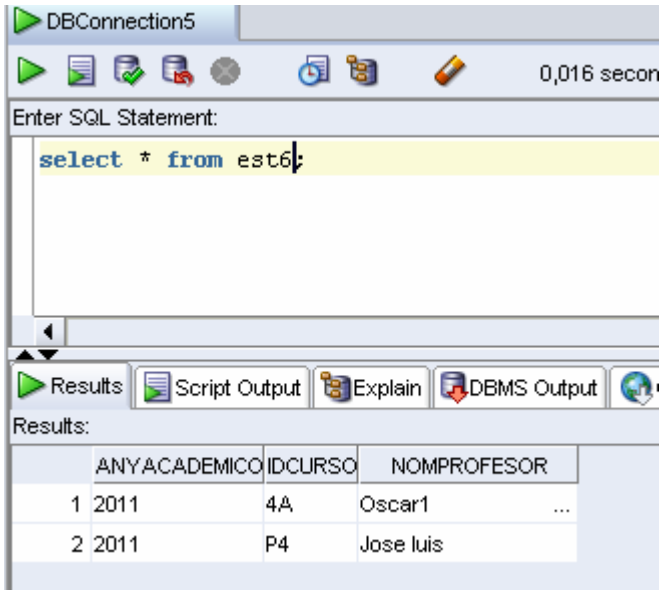


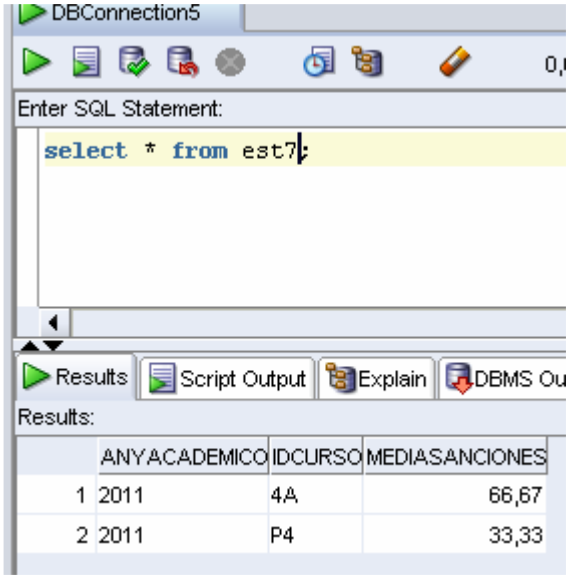
	ANYACADEMICO	DNIPROFESOR	MEDIAAMONESTACIONES
1	2011	17727433E	57,14
2	2011	99999999B	14,29
3	2011	22222222A	28,57

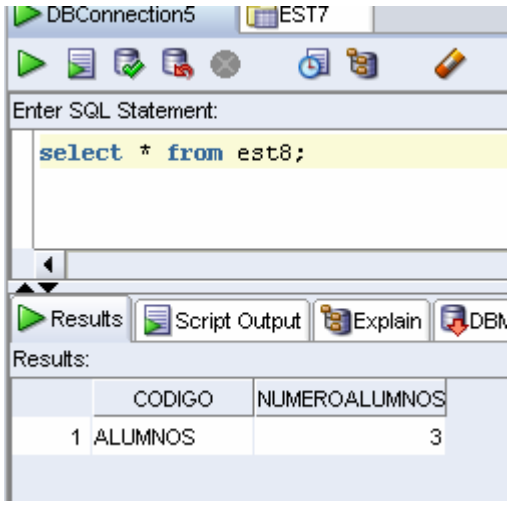


	ANYACADEMICO	IDCURSO	NUMEROSANCIONES
1	2011	4A	4
2	2011	P4	2


DBConnection5
0,031 seco
Enter SQL Statement:
select * from est5;
Results
Script Output
Explain
DBMS Output
Results:
ANYACADEMICO NOMALUMNO
1 2011 OSCAR


DBConnection5
0,016 secon
Enter SQL Statement:
select * from est6;
Results
Script Output
Explain
DBMS Output
Results:
ANYACADEMICO IDCURSO NOMPROFESOR
1 2011 4A Oscar1
2 2011 P4 Jose luis


DBConnection5
Enter SQL Statement:
select * from est7;
Results
Script Output
Explain
Results:
ANYACADEMICO IDCURSO MEDIASANCIONES
1 2011 4A 66,67
2 2011 P4 33,33


DBConnection5
EST7
Enter SQL Statement:
select * from est8;
Results
Script Output
Explain
Results:
CODIGO NUMEROALUMNOS
1 ALUMNOS 3

8. Conclusiones

Este proyecto me ha servido para poder ver como implementar un proyecto en situación casi real, indudablemente el tiempo a dedicar no es el mismo, no le podemos dedicar las mismas horas que aun proyecto real. Por ello la planificación realizada es uno de los aspectos más importantes para poder llevar a buen puerto el proyecto. Esta planificación se ha realizado en cascada para poder implementarlo mejor al ser un proyecto realizado por una persona es la mejor manera de llevarlo a cabo dado que todo pasa por uno mismo.

El proyecto se ha llevado a cabo desde sus inicios y se ha finalizado el aplicativo en perfecto funcionamiento para su uso por parte del cliente. Esto demuestra que hemos seguido todas las fases explicadas a lo largo de la carrera, para un óptimo desarrollo. Esto quiere decir todas las fases del análisis. A partir de los requerimientos del cliente hemos diseñado un marco conceptual paga poder conseguirlo. Esto significa todo el diseño conceptual, lógico y físico para después poder implementar todo y poder realizar unas pruebas unitarias primero y posteriormente globales y poder observar el comportamiento del aplicativo.

Todo el diseño de la base de datos ha sido realizado en Oracle es parecido a todas las bases de datos, el punto más complicado es realizar los procedimientos que se hacen en PL/SQL una manera de aprender más sobre este mundo de Oracle.

Hay que tener en cuenta la memoria una parte importante del proyecto por la cual se pueden tomar decisiones para cualquier proyecto, es seguramente el tema donde peor encajamos, por no ser un marco tan técnico y donde no nos encontramos tan a gusto. Pero indudablemente es un tema a tener en cuenta y donde debemos incidir en la buena línea a seguir.

9. Bibliografia

Diccionario

http://www.rae.es/rae.html

Código PL/SQL

http://www.oradev.com/oracle_string_functions.jsp

http://www.desarrolloweb.com/articulos/cursores-pl-sql-I.html

http://www.java2s.com/Tutorial/Oracle/0020__Introduction/Catalog0020__Introduction.htm

http://www.wikioracle.es/doku.php

Libros:

Materiales didácticos de las asignaturas de Base de datos I, Base de datos II, Sistemas de gestión de base de datos y Competencia comunicativa para profesionales de las TIC todos ellos impartidas por la Universidad Oberta de Catalunya.

ANEXO – IMPLEMENTACION CODIGO

Tablas

```
CREATE TABLE alumno(
idalumno            Varchar2(10) NOT NULL,
nombre              Varchar2(30) NOT NULL,
apellidos           Varchar2(50) NOT NULL,
fechanacimiento     Date NOT NULL,
nacionalidad        Varchar2(30) NOT NULL,
direccion           Varchar2(50) NOT NULL,
poblacion           Varchar2(30) NOT NULL,
cpostal             Varchar2(10) NOT NULL,
telefono            Varchar2(10) NOT NULL,
nompadre            Varchar2(30),
movilpadre          Varchar2(10),
email               Varchar2(30),
estado              char(1),
fechadebaja         Date,
motivobaja          Varchar2(100),
PRIMARY KEY         (idalumno),
CHECK               (estado IN(' ','b'))
);

CREATE TABLE asignatura(
idasignatura        Varchar2(10) NOT NULL,
nombre              Varchar2(30) NOT NULL,
PRIMARY KEY         (idasignatura)
);

CREATE TABLE profesor(
dniprofesor         Varchar2(10) NOT NULL,
nombre              Varchar2(30)  NOT NULL,
apellidos           Varchar2(50) NOT NULL,
fechanacimiento     DATE NOT NULL,
nacionalidad        Varchar2(30) NOT NULL,
direccion           Varchar2(50) NOT NULL,
poblacion           Varchar2(30) NOT NULL,
cpostal             Varchar2(10) NOT NULL,
telefono            Varchar2(10),
movil               Varchar2(10),
email               Varchar2(30),
estado              Char(1),
fechadebaja         DATE,
motivobaja          Varchar2(100),
CHECK               (estado IN(' ','b')),
PRIMARY KEY         (dniprofesor)
);

CREATE TABLE curso(
anyacademico        Varchar2(4) NOT NULL,
idcurso             Varchar2(10) NOT NULL,
nombre              Varchar2(30) NOT NULL,
dniprofesor         Varchar2(10) NOT NULL,
PRIMARY KEY         (anyacademico, idcurso),
FOREIGN KEY         (dniprofesor) REFERENCES      profesor(dniprofesor)
);
```

```
CREATE TABLE tipodeamonestacion(
idtamonestacion          varchar2(10) NOT NULL,
descripcion              varchar2(80) NOT NULL,
PRIMARY KEY              (idtamonestacion)
) ;

CREATE TABLE tipodesancion(
idtsancion          varchar2(10) NOT NULL,
descripcion         varchar2(80) NOT NULL,
PRIMARY KEY         (idtsancion)
);

CREATE TABLE componen(
anyacademico        Varchar2(4) NOT NULL,
idcurso             Varchar2(10) NOT NULL,
idasignatura        Varchar2(10) NOT NULL,
dniprofesor         Varchar2(10) NOT NULL,
PRIMARY KEY         (anyacademico, idcurso, idasignatura,   dniprofesor),
FOREIGN KEY         (anyacademico, idcurso)         REFERENCES    curso(anyacademico, idcurso),
FOREIGN KEY         (idasignatura)                  REFERENCES    asignatura(idasignatura),
FOREIGN KEY         (dniprofesor)                   REFERENCES    profesor(dniprofesor)
);

CREATE TABLE matriculan(
numexpediente       Varchar2(10) NOT NULL,
idalumno            Varchar2(10) NOT NULL,
anyacademico        Varchar2(4) NOT NULL,
idcurso             Varchar2(10) NOT NULL,
PRIMARY KEY         (numexpediente),
FOREIGN KEY         (idalumno)              REFERENCES alumno(idalumno),
FOREIGN KEY         (anyacademico, idcurso)         REFERENCES    curso(anyacademico, idcurso)
);

CREATE TABLE calendario(
anyacademico        Varchar2(4) NOT NULL,
idcurso             Varchar2(10) NOT NULL,
idasignatura        Varchar2(10) NOT NULL,
dniprofesor         Varchar2(10) NOT NULL,
diasemana           Varchar2(10) NOT NULL,
horainicio          integer,
horafin             integer,
PRIMARY KEY         (anyacademico, idcurso, idasignatura, dniprofesor, diasemana),
FOREIGN KEY         (anyacademico, idcurso, idasignatura, dniprofesor)       REFERENCES
  componen(anyacademico,        idcurso,idasignatura,     dniprofesor)
);

CREATE TABLE visita(
anyacademico        Varchar2(4) NOT NULL,
idcurso             Varchar2(10) NOT NULL,
idasignatura        Varchar2(10) NOT NULL,
dniprofesor         Varchar2(10) NOT NULL,
diasemana           Varchar2(10) NOT NULL,
horainicio          integer,
horafin             integer,
PRIMARY KEY         (anyacademico, idcurso, idasignatura,   dniprofesor,      diasemana),
FOREIGN KEY         (anyacademico, idcurso, idasignatura,   dniprofesor)      REFERENCES
  componen(anyacademico,        idcurso,idasignatura,     dniprofesor)
```

```
);

CREATE TABLE amonestacion(
idamonestacion          Varchar2(10) NOT NULL,
idtamonestacion         Varchar2(10) NOT NULL,
idalumno                Varchar2(10) NOT NULL,
anyacademico            Varchar2(4) NOT NULL,
idcurso                 Varchar2(10) NOT NULL,
idasignatura            Varchar2(10) NOT NULL,
dniprofesor             Varchar2(10) NOT NULL,
diasemana               Varchar2(10) NOT NULL,
hora                    integer,
gravedad                integer,
comunicacion            char(1) default('n') check(comunicacion          IN('s','n')),
PRIMARY KEY             (idamonestacion, idtamonestacion, idalumno,  anyacademico,  idcurso,
                        idasignatura, dniprofesor, diasemana),
FOREIGN KEY             (idtamonestacion)       REFERENCES    tipodeamonestacion(idtamonestacion),
FOREIGN KEY             (idalumno)              REFERENCES alumno(idalumno),
FOREIGN KEY             (anyacademico, idcurso, idasignatura,   dniprofesor,diasemana)
        REFERENCES      calendario(anyacademico, idcurso,idasignatura, dniprofesor,     diasemana)
);

CREATE TABLE reglaactivacion(
ractivacion             Varchar2(10) NOT NULL,
idtamonestacion         Varchar2(10),
gravedadactividad       integer,
idtsancion              Varchar2(10),
PRIMARY KEY             (ractivacion),
FOREIGN KEY             (idtamonestacion)       REFERENCES    tipodeamonestacion(idtamonestacion),
FOREIGN KEY             (idtsancion)            REFERENCES    tipodesancion(idtsancion)
);

CREATE TABLE sancion(
idsancion               Varchar2(10) NOT NULL,
idalumno                Varchar2 (10) NOT NULL,
anyacademico            Varchar2(4) NOT NULL,
idcurso                 Varchar2 (10) NOT NULL,
ractivacion             Varchar2(10),
idtsancion              Varchar2(10),
descripcion             Varchar2(100),
fechasancion            Date,
idamonestacion          Varchar2(10),
PRIMARY KEY             (idsancion, idalumno),
FOREIGN KEY             (idalumno)              REFERENCES alumno(idalumno),
FOREIGN KEY             (anyacademico, idcurso) REFERENCES   curso(anyacademico, idcurso),
FOREIGN KEY             (ractivacion)           REFERENCES  reglaactivacion(ractivacion),
FOREIGN KEY             (idtsancion)            REFERENCES  tipodesancion(idtsancion)
);

CREATE TABLE Est1(
idalumno                Varchar2 (10) NOT NULL,
numeroamonestaciones    integer,
PRIMARY KEY             (idalumno),
FOREIGN KEY             (idalumno) REFERENCES alumno(idalumno)
);

CREATE TABLE Est2(
anyacademico            Varchar2(4) NOT NULL,
```

```
idalumno                Varchar2 (10) NOT NULL,
numerosanciones         integer,
PRIMARY KEY             (anyacademico, idalumno),
FOREIGN KEY             (idalumno) REFERENCES alumno(idalumno)
);

CREATE TABLE Est3(
anyacademico            Varchar2(4) NOT NULL,
dniprofesor             Varchar2 (10) NOT NULL,
mediaamonestaciones     number(5,2),
PRIMARY KEY             (anyacademico, dniprofesor),
FOREIGN KEY             (dniprofesor) REFERENCES      profesor(dniprofesor)
);

CREATE TABLE Est4(
anyacademico            Varchar2(4) NOT NULL,
idcurso                 Varchar2 (10) NOT NULL,
numerosanciones         integer,
PRIMARY KEY             (anyacademico,idcurso),
FOREIGN KEY             (anyacademico, idcurso) REFERENCES   curso(anyacademico, idcurso)
);

CREATE TABLE Est5(
anyacademico            Varchar2(4) NOT NULL,
nomalumno               Varchar2 (30) NOT NULL,
PRIMARY KEY             (anyacademico, nomalumno)
);

CREATE TABLE Est6(
anyacademico            Varchar2(4) NOT NULL,
idcurso                 Varchar2(10) NOT NULL,
nomprofesor             Varchar2 (30),
PRIMARY KEY             (anyacademico, idcurso),
FOREIGN KEY             (anyacademico, idcurso) REFERENCES   curso(anyacademico, idcurso)
);

CREATE TABLE Est7(
anyacademico            Varchar2(4) NOT NULL,
idcurso                 Varchar2(10) NOT NULL,
mediasanciones          Number(5,2),
PRIMARY KEY             (anyacademico, idcurso),
FOREIGN KEY             (anyacademico, idcurso) REFERENCES   curso(anyacademico, idcurso)
);

CREATE TABLE Est8(
codigo                  Varchar2(10) NOT NULL,
numeroalumnos           integer,
PRIMARY KEY             (codigo)
);

CREATE TABLE logs(
fecha                   timestamp NOT NULL,
procedimiento           Varchar2(30) NOT NULL,
paramsentrada           Varchar2(500) NOT NULL,
paramssalida            Varchar2(100) NOT NULL
);
```

Trigger y Secuencia

```
CREATE SEQUENCE seq_SANCION INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER inserir_id_SANCION
BEFORE INSERT ON SANCION
FOR EACH ROW
BEGIN
SELECT seq_SANCION.NEXTVAL INTO :NEW.idsancion
FROM DUAL;
END inserir_id_SANCION;
```

Procedimientos Almacenados

```
CREATE OR REPLACE PROCEDURE p_altaAlumno (
ida       IN Varchar2,
nom       IN Varchar2,
ape       IN Varchar2,
fnac      IN Date,
nac       IN Varchar2,
dire      IN Varchar2,
pobla     IN Varchar2,
cpos      IN Varchar2,
tel       IN Varchar2,
nompa     IN Varchar2,
movpa     IN Varchar2,
ema       IN Varchar2,
est       IN char,
fba       IN Date,
mob       IN Varchar2,
salida    OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
  RAISE men_err;
END IF;
INSERT INTO ALUMNO (idalumno, nombre, apellidos, fechanacimiento, nacionalidad, direccion, poblacion, cpostal, telefono,
          nompadre, movilpadre, email, estado, fechadebaja, motivobaja)
          VALUES (ida, nom, ape, fnac, nac, dire, pobla, cpos, tel, nompa, movpa, ema, est, fba, mob);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_altaAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
          ',' || cpos || ',' || tel || ',' || nompa || ','|| movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
          salida := 'ERROR: El alumno ya existe';
          INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_altaAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
          ',' || cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN STORAGE_ERROR THEN
          salida := 'ERROR: No pudo insertarse';
          INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_altaAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
          ',' || cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN men_err THEN
          salida := 'ERROR: No se han rellenado los parámetros necesarios';
          INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_altaAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
          ',' || cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN OTHERS THEN
          salida := 'ERROR: Error genérico al dar de alta un alumno';
          INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
```

```
        VALUES (SYSDATE, 'p_altaAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
        ',' || cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
END;

CREATE OR REPLACE PROCEDURE p_bajaAlumno (
ida       IN Varchar2,
est       IN char,
fba       IN Date,
mob       IN Varchar2,
salida    OUT Varchar2
) IS
ida_co            INTEGER;
noExiste_ida      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
  RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO ida_co
    FROM ALUMNO
    WHERE idalumno = ida;
    IF (ida_co = 0) THEN
      RAISE noExiste_ida;
    END IF;
 UPDATE ALUMNO
        SET idalumno     = ida,
        estado           = 'b',
        fechadebaja      = SYSDATE,
        motivobaja       =mob
        WHERE idalumno =ida;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaAlumno', ida || ',' || est || ',' || fba || ',' || mob, salida);
EXCEPTION
WHEN noExiste_ida THEN
        salida := 'ERROR: No existe alumno';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaAlumno', ida || ',' || est || ',' || fba || ',' || mob, salida);
WHEN STORAGE_ERROR THEN
        salida := 'ERROR: No pudo insertarse';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaAlumno', ida || ',' || est || ',' || fba || ',' || mob, salida);
WHEN men_err THEN
        salida := 'ERROR: No se han rellenado los parámetros necesarios';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaAlumno', ida || ',' || est || ',' || fba || ',' || mob, salida);
WHEN OTHERS THEN
        salida := 'ERROR: Error genérico al dar de baja un alumno';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaAlumno', ida || ',' || est || ',' || fba || ',' || mob, salida);
END;

CREATE OR REPLACE PROCEDURE p_modifAlumno (
ida       IN Varchar2,
nom       IN Varchar2,
ape       IN Varchar2,
fnac      IN Date,
nac       IN Varchar2,
dire      IN Varchar2,
pobla     IN Varchar2,
cpos      IN Varchar2,
tel       IN Varchar2,
nompa     IN Varchar2,
```

```
movpa   IN Varchar2,
ema     IN Varchar2,
est     IN char,
fba     IN Date,
mob     IN Varchar2,
salida  OUT Varchar2
) IS
ida_co  INTEGER;
noExiste_ida      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
  RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO ida_co
    FROM ALUMNO
    WHERE idalumno = ida;
    IF (ida_co = 0) THEN
     RAISE noExiste_ida;
    END IF;
 UPDATE ALUMNO
          SET idalumno      = ida,
          nombre            =nom,
          apellidos         =ape,
          fechanacimiento   =fnac,
          nacionalidad      =nac,
          direccion         =dire,
          población         =pobla,
          cpostal           =cpos,
          telefono          =tel,
          nompadre          =nompa,
          movilpadre        =movpa,
          email             =ema,
          estado            = est,
          fechadebaja       = fba,
          motivobaja        =mob
          WHERE idalumno =ida;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
          ',' || cpos || ',' || tel || ',' || nompa || ','|| movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
EXCEPTION
WHEN noExiste_ida THEN
    salida := 'ERROR: El alumno no existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla || ',' ||
          cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla || ',' ||
          cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar un alumno';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAlumno', ida || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla || ',' ||
          cpos || ',' || tel || ',' || nompa || ',' || movpa || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
END;

CREATE OR REPLACE PROCEDURE p_altaProfesor (
dni     IN Varchar2,
nom     IN Varchar2,
ape     IN Varchar2,
```

```
fnac    IN Date,
nac     IN Varchar2,
dire    IN Varchar2,
pobla   IN Varchar2,
cpos    IN Varchar2,
tel     IN Varchar2,
mov     IN Varchar2,
ema     IN Varchar2,
est     IN char,
fba     IN Date,
mob     IN Varchar2,
salida  OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (dni IS NULL OR dni = '') THEN
    RAISE men_err;
END IF;
INSERT INTO PROFESOR (dniprofesor, nombre, apellidos, fechanacimiento, nacionalidad, direccion, poblacion, cpostal,
        telefono, movil, email, estado, fechadebaja, motivobaja)
        VALUES (dni, nom, ape, fnac, nac, dire, pobla, cpos, tel, mov, ema, est, fba, mob);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
        ',' || cpos || ',' || tel || ','|| mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
        salida := 'ERROR: La licencia ya existe';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
        ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN STORAGE_ERROR THEN
        salida := 'ERROR: No pudo insertarse';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
        ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN men_err THEN
        salida := 'ERROR: No se han rellenado los parámetros necesarios';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
        ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN OTHERS THEN
        salida := 'ERROR: Error genérico al dar de alta un profesor';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla ||
        ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
END;

CREATE OR REPLACE PROCEDURE p_bajaProfesor (
dni     IN Varchar2,
est     IN char,
fba     IN Date,
mob     IN Varchar2,
salida  OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (dni IS NULL OR dni = '') THEN
        RAISE men_err;
END IF;
UPDATE PROFESOR
        SET dniprofesor     =dni,
        estado              ='b',
        fechadebaja         =SYSDATE,
```

```
        motivobaja          =mob
        WHERE dniprofesor=dni;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaProfesor', dni || ',' || est || ',' || fba || ',' || mob, salida);
EXCEPTION
WHEN NO_DATA_FOUND THEN
        salida := 'ERROR: No existe profesor';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaProfesor', dni || ',' || est || ',' || fba || ',' || mob, salida);
WHEN STORAGE_ERROR THEN
        salida := 'ERROR: No pudo insertarse';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaProfesor', dni || ',' || est || ',' || fba || ',' || mob, salida);
WHEN men_err THEN
        salida := 'ERROR: No se han rellenado los parámetros necesarios';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaProfesor', dni || ',' || est || ',' || fba || ',' || mob, salida);
WHEN OTHERS THEN
        salida := 'ERROR: Error genérico al dar de baja un profesor';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaProfesor', dni || ',' || est || ',' || fba || ',' || mob, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifProfesor (
dni     IN Varchar2,
nom     IN Varchar2,
ape     IN Varchar2,
fnac    IN Date,
nac     IN Varchar2,
dire    IN Varchar2,
pobla   IN Varchar2,
cpos    IN Varchar2,
tel     IN Varchar2,
mov     IN Varchar2,
ema     IN Varchar2,
est     IN char,
fba     IN Date,
mob     IN Varchar2,
salida  OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (dni IS NULL OR dni = '') THEN
   RAISE men_err;
END IF;
UPDATE PROFESOR
        SET dniprofesor   =dni,
        nombre            =nom,
        apellidos         =ape,
        fechanacimiento   =fnac,
        nacionalidad      =nac,
        direccion         =dire,
        población         =pobla,
        cpostal           =cpos,
        telefono          =tel,
        movil             =mov,
        email             =ema,
        estado            =est,
        fechadebaja       =fba,
        motivobaja        =mob
WHERE dniprofesor         =dni;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
```

```
        VALUES (SYSDATE, 'p_modifProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla
        || ',' || cpos || ',' || tel || ','|| mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
EXCEPTION
WHEN NO_DATA_FOUND THEN
        salida := 'ERROR: Profesor no encontrado';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla
        || ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN men_err THEN
        salida := 'ERROR: No se han rellenado los parámetros necesarios';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla
        || ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
WHEN OTHERS THEN
        salida := 'ERROR: Error genérico al modificar un profesor';
        INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifProfesor', dni || ',' || nom || ',' || ape || ',' || fnac || ',' || nac || ',' || dire || ',' || pobla
        || ',' || cpos || ',' || tel || ',' || mov || ',' || ema || ',' || est || ',' || fba || ',' || mob, salida);
END;

CREATE OR REPLACE PROCEDURE p_altaAsignatura (
ida     IN Varchar2,
nom     IN Varchar2,
salida  OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
  RAISE men_err;
END IF;
INSERT INTO ASIGNATURA (idasignatura, nombre)
VALUES (ida, nom);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
VALUES (SYSDATE, 'p_altaAsignatura', ida || ',' || nom, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: La asignatura ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaAsignatura', ida || ',' || nom, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaAsignatura', ida || ',' || nom, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaAsignatura', ida || ',' || nom, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta una asignatura';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaAsignatura', ida || ',' || nom, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaAsignatura (
ida     IN Varchar2,
salida  OUT Varchar2
) IS
ida_co          INTEGER;
noExiste_ida    EXCEPTION;
men_err         EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
    RAISE men_err;
```

```
END IF;
SELECT COUNT (*)
    INTO ida_co
    FROM ASIGNATURA
    WHERE idasignatura = ida;
  IF (ida_co = 0) THEN
     RAISE noExiste_ida;
  END IF;

   DELETE FROM ASIGNATURA WHERE idasignatura=ida;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'p_bajaAsignatura', ida, salida);
EXCEPTION
WHEN noExiste_ida THEN
    salida := 'ERROR: La asignatura no existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_bajaAsignatura', ida, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_bajaAsignatura', ida, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja una asignatura';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_bajaAsignatura', ida, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifAsignatura (
ida       IN Varchar2,
nom       IN Varchar2,
salida    OUT Varchar2
) IS
ida_co              INTEGER;
noExiste_ida        EXCEPTION;
men_err             EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)  INTO ida_co
    FROM ASIGNATURA
    WHERE idasignatura = ida;
  IF (ida_co = 0) THEN
     RAISE noExiste_ida;
  END IF;

UPDATE ASIGNATURA
          SET nombre           =nom
          WHERE idasignatura =ida;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'p_modifAsignatura', ida || ',' || nom, salida);
EXCEPTION
WHEN noExiste_ida THEN
    salida := 'ERROR: La asignatura no existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAsignatura', ida || ',' || nom, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAsignatura', ida || ',' || nom, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar una asignatura';
```

```
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAsignatura', ida || ',' || nom, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaCurso (
ano       IN Varchar2,
idc       IN Varchar2,
nom       IN Varchar2,
dni       IN Varchar2,
salida    OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
          RAISE men_err;
END IF;
INSERT INTO CURSO (anyacademico, idcurso, nombre, dniprofesor)
          VALUES (ano, idc, nom, dni);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_altaCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: El Curso ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta un Curso';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaCurso (
ano       IN Varchar2,
idc       IN Varchar2,
salida    OUT Varchar2
) IS
curso_co          INTEGER;
noExiste_cur      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO curso_co
    FROM CURSO
    WHERE anyacademico = ano AND idcurso=idc;
  IF (curso_co = 0) THEN
     RAISE noExiste_cur;
  END IF;

DELETE FROM CURSO WHERE anyacademico = ano AND idcurso=idc;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaCurso', ano || ',' || idc, salida);
```

```
EXCEPTION
WHEN noExiste_cur THEN
    salida := 'ERROR: No existe el Curso';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaCurso', ano || ',' || idc, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaCurso', ano || ',' || idc, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja un curso';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaCurso', ano || ',' || idc, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifCurso (
ano     IN Varchar2,
idc     IN Varchar2,
nom     IN Varchar2,
dni     IN Varchar2,
salida  OUT Varchar2
) IS
curso_co        INTEGER;
noExiste_cur    EXCEPTION;
men_err         EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
        RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO curso_co
    FROM CURSO
    WHERE anyacademico = ano AND idcurso=idc;
  IF (curso_co = 0) THEN
    RAISE noExiste_cur;
  END IF;

UPDATE CURSO
        SET nombre      =nom,
        dniprofesor     =dni
        WHERE anyacademico = ano AND idcurso=idc;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
EXCEPTION
WHEN noExiste_cur THEN
    salida := 'ERROR: No existe el Curso';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar un curso';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifCurso', ano || ',' || idc || ',' || nom || ',' || dni, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaComponen (
ano     IN Varchar2,
idc     IN Varchar2,
ida     IN Varchar2,
dni     IN Varchar2,
```

```
salida     OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' ) THEN
          RAISE men_err;
END IF;
INSERT INTO COMPONEN (anyacademico, idcurso, idasignatura, dniprofesor)
          VALUES (ano, idc, ida, dni);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_altaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: La Composición ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCalendario', ano || ',' || idc || ',' || ida || ',' || dni, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta la Composición';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaComponen (
ano       IN Varchar2,
idc       IN Varchar2,
ida       IN Varchar2,
dni       IN Varchar2,
salida    OUT Varchar2
) IS
compo_co          INTEGER;
noExiste_com      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' ) THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO compo_co
    FROM COMPONEN
    WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni;
  IF (compo_co = 0) THEN
     RAISE noExiste_com;
  END IF;

DELETE FROM COMPONEN WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
EXCEPTION
WHEN noExiste_com THEN
    salida := 'ERROR: No existe la Composición';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
```

```
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja una composición';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaComponen', ano || ',' || idc || ',' || ida || ',' || dni, salida);
END;

CREATE OR REPLACE PROCEDURE p_altaCalendario (
ano     IN Varchar2,
idc     IN Varchar2,
ida     IN Varchar2,
dni     IN Varchar2,
dia     IN Varchar2,
hin     IN Integer,
hfi     IN Integer,
salida  OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
        RAISE men_err;
END IF;
INSERT INTO CALENDARIO (anyacademico, idcurso, idasignatura, dniprofesor, diasemana, horainicio, horafin)
        VALUES (ano, idc, ida, dni, dia, hin, hfi);
salida := 'OK';
  INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
  VALUES (SYSDATE, 'p_altaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: El Calendario ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta el Calendario';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaCalendario (
ano     IN Varchar2,
idc     IN Varchar2,
ida     IN Varchar2,
dni     IN Varchar2,
dia     IN Varchar2,
salida  OUT Varchar2
) IS
calen_co        INTEGER;
noExiste_cal    EXCEPTION;
men_err         EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
```

```
        RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO calen_co
    FROM CALENDARIO
    WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND diasemana=dia;
  IF (calen_co = 0) THEN
     RAISE noExiste_cal;
  END IF;

DELETE FROM CALENDARIO WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni
AND diasemana=dia;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
EXCEPTION
WHEN noExiste_cal THEN
    salida := 'ERROR: No existe el Calendario';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_bajaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_bajaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja un calendario';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_bajaCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifCalendario (
ano       IN Varchar2,
idc       IN Varchar2,
ida       IN Varchar2,
dni       IN Varchar2,
dia       IN Varchar2,
hin       IN Integer,
hfi       IN Integer,
salida    OUT Varchar2
) IS
calen_co          INTEGER;
noExiste_cal      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
        RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO calen_co
    FROM CALENDARIO
    WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND diasemana=dia;
  IF (calen_co = 0) THEN
     RAISE noExiste_cal;
  END IF;

UPDATE CALENDARIO
        SET horainicio    =hin,
         horafin          =hfi
        WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND diasemana=dia;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
      VALUES (SYSDATE, 'p_modifCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
EXCEPTION
```

```
WHEN noExiste_cal THEN
    salida := 'ERROR: No existe el Calendario';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar un calendario';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifCalendario', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaVisita (
ano       IN Varchar2,
idc       IN Varchar2,
ida       IN Varchar2,
dni       IN Varchar2,
dia       IN Varchar2,
hin       IN Integer,
hfi       IN Integer,
salida    OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
          RAISE men_err;
END IF;
INSERT INTO VISITA (anyacademico, idcurso, idasignatura, dniprofesor, diasemana, horainicio, horafin)
          VALUES (ano, idc, ida, dni, dia, hin, hfi);
salida := 'OK';
  INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
       VALUES (SYSDATE, 'p_altaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: La visita ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
      VALUES (SYSDATE, 'p_altaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta la Visita';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_altaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaVisita (
ano       IN Varchar2,
idc       IN Varchar2,
ida       IN Varchar2,
dni       IN Varchar2,
dia       IN Varchar2,
salida    OUT Varchar2
) IS
calen_co          INTEGER;
noExiste_cal      EXCEPTION;
```

```
men_err          EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
         RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO calen_co
    FROM VISITA
    WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND diasemana=dia;
  IF (calen_co = 0) THEN
     RAISE noExiste_cal;
  END IF;

DELETE FROM VISITA WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND
diasemana=dia;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
         VALUES (SYSDATE, 'p_bajaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
EXCEPTION
WHEN noExiste_cal THEN
    salida := 'ERROR: No existe la Visita';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
         VALUES (SYSDATE, 'p_bajaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
         VALUES (SYSDATE, 'p_bajaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja una visita';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
         VALUES (SYSDATE, 'p_bajaVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifVisita (
ano      IN Varchar2,
idc      IN Varchar2,
ida      IN Varchar2,
dni      IN Varchar2,
dia      IN Varchar2,
hin      IN Integer,
hfi      IN Integer,
salida   OUT Varchar2
) IS
calen_co         INTEGER;
noExiste_cal     EXCEPTION;
men_err          EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '' OR ida IS NULL OR ida = '' OR
   dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
         RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO calen_co
    FROM VISITA
    WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND diasemana=dia;
  IF (calen_co = 0) THEN
     RAISE noExiste_cal;
  END IF;

UPDATE VISITA
         SET horainicio    =hin,
         horafin           =hfi
         WHERE anyacademico = ano AND idcurso=idc AND idasignatura=ida AND dniprofesor=dni AND diasemana=dia;
```

```
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'p_modifVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
EXCEPTION
WHEN noExiste_cal THEN
    salida := 'ERROR: No existe la Visita';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar una visita';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifVisita', ano || ',' || idc || ',' || ida || ',' || dni || ',' || dia || ',' || hin || ',' || hfi, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaMatriculan (
nex       IN Varchar2,
ida       IN Varchar2,
ano       IN Varchar2,
idc       IN Varchar2,
salida    OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (nex IS NULL OR nex = '') THEN
        RAISE men_err;
END IF;
INSERT INTO MATRICULAN (numexpediente, idalumno, anyacademico, idcurso)
        VALUES (nex, ida, ano, idc);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaMatriculan', nex || ',' || ida || ',' || ano || ',' || idc, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: La Matriculación ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaMatriculan', nex || ',' || ida || ',' || ano || ',' || idc, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaMatriculan', nex || ',' || ida || ',' || ano || ',' || idc, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaMatriculan', nex || ',' || ida || ',' || ano || ',' || idc, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta una Matriculación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
         VALUES (SYSDATE, 'p_altaMatriculan', nex || ',' || ida || ',' || ano || ',' || idc, salida);
END;

CREATE OR REPLACE PROCEDURE p_bajaMatriculan (
nex       IN Varchar2,
salida    OUT Varchar2
) IS
matri_co          INTEGER;
noExiste_mat      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (nex IS NULL OR nex = '' ) THEN
        RAISE men_err;
```

```
END IF;
SELECT COUNT (*)
    INTO matri_co
    FROM MATRICULAN
    WHERE numexpediente = nex;
  IF (matri_co = 0) THEN
     RAISE noExiste_mat;
  END IF;

DELETE FROM MATRICULAN WHERE numexpediente = nex;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaMatriculan', nex, salida);
EXCEPTION
WHEN noExiste_mat THEN
    salida := 'ERROR: No existe la matricula';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaMatriculan', nex, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaMatriculan', nex, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja una matricula';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaMatriculan', nex, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifMatriculan (
nex       IN Varchar2,
ida       IN Varchar2,
ano       IN Varchar2,
idc       IN Varchar2,
salida    OUT Varchar2
) IS
matri_co          INTEGER;
noExiste_mat      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (nex IS NULL OR nex = '' ) THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO matri_co
    FROM MATRICULAN
    WHERE numexpediente = nex;
  IF (matri_co = 0) THEN
     RAISE noExiste_mat;
  END IF;

UPDATE MATRICULAN
          SET idalumno     =ida,
          anyacademico     =ano,
          incurso          =idc
          WHERE numexpediente = nex;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'p_modifMatriculan', nex || ',' || ida || ',' || ano || ',' || idc, salida);
EXCEPTION
WHEN noExiste_mat THEN
    salida := 'ERROR: No existe la matricula';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifMatricula', nex || ',' || ida || ',' || ano || ',' || idc, salida);
WHEN men_err THEN
```

```
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifMatricula', nex || ',' || ida || ',' || ano || ',' || idc, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar una matricula';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_modifMatricula', nex || ',' || ida || ',' || ano || ',' || idc, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaTamonestacion (
idta      IN Varchar2,
des       IN Varchar2,
salida    OUT Varchar2
) IS
men_err EXCEPTION;
BEGIN
IF (idta IS NULL OR idta = '') THEN
        RAISE men_err;
END IF;
INSERT INTO TIPODEAMONESTACION (idtamonestacion, descripcion)
        VALUES (idta, des);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTamonestacion', idta || ',' || des, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: El Tipo Amonestación ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTamonestacion', idta || ',' || des, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTamonestacion', idta || ',' || des, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTamonestacion', idta || ',' || des, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta una Tipo de Amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTamonestacion', idta || ',' || des, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaTamonestacion (
idta      IN Varchar2,
salida    OUT Varchar2
) IS
idta_co             INTEGER;
noExiste_idta       EXCEPTION;
men_err             EXCEPTION;
BEGIN
IF (idta IS NULL OR idta = '') THEN
        RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO idta_co
    FROM TIPODEAMONESTACION
    WHERE idtamonestacion = idta;
  IF (idta_co = 0) THEN
     RAISE noExiste_idta;
  END IF;

DELETE FROM TIPODEAMONESTACION WHERE idtamonestacion=idta;
   salida := 'OK';
```

```
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaTamonestacion', idta, salida);
EXCEPTION
WHEN noExiste_idta THEN
    salida := 'ERROR: No existe el Tipo de amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaTamonestacion', idta, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaTamonestacion', idta, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja una asignatura';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaTamonestacion', idta, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifTamonestacion (
idta      IN Varchar2,
des       IN Varchar2,
salida    OUT Varchar2
) IS
idta_co           INTEGER;
noExiste_idta     EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (idta IS NULL OR idta = '') THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO idta_co
    FROM TIPODEAMONESTACION
    WHERE idtamonestacion = idta;
  IF (idta_co = 0) THEN
     RAISE noExiste_idta;
  END IF;

UPDATE TIPODEAMONESTACION
          SET descripción           =des
          WHERE idtamonestacion     =idta;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'p_modifTamonestacion', idta || ',' || des, salida);
EXCEPTION
WHEN noExiste_idta THEN
    salida := 'ERROR: No existe el Tipo de amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifTamonestacion', idta || ',' || des, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifTamonestacion', idta || ',' || des, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja un tipo de amonestacion';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifTamonestacion', idta || ',' || des, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaTsancion (
idt       IN Varchar2,
des       IN Varchar2,
salida    OUT Varchar2
) IS
men_err EXCEPTION;
```

```
BEGIN
IF (idt IS NULL OR idt = '') THEN
        RAISE men_err;
END IF;
INSERT INTO TIPODESANCION (idtsancion, descripcion)
        VALUES (idt, des);
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTsancion', idt || ',' || des, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: El Tipo Sanción ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTsancion', idt || ',' || des, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTsancion', idt || ',' || des, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTsancion', idt || ',' || des, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta una Tipo de Sanción';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaTsancion', idt || ',' || des, salida);
END;

CREATE OR REPLACE PROCEDURE p_bajaTsancion (
idt         IN Varchar2,
salida      OUT Varchar2
) IS
idt_co              INTEGER;
noExiste_idt        EXCEPTION;
men_err             EXCEPTION;
BEGIN
IF (idt IS NULL OR idt = '') THEN
        RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO idt_co
    FROM TIPODESANCION
    WHERE idtsancion = idt;
  IF (idt_co = 0) THEN
    RAISE noExiste_idt;
  END IF;

DELETE FROM TIPODESANCION WHERE idtsancion=idt;
   salida := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaTsancion', idt, salida);
EXCEPTION
WHEN noExiste_idt THEN
    salida := 'ERROR: No existe el Tipo de sanción';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaTsancion', idt, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaTsancion', idt, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja un tipo de sanción';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_bajaTsancion', idt, salida);
```

```
END;

CREATE OR REPLACE  PROCEDURE p_modifTsancion (
idt         IN Varchar2,
des         IN Varchar2,
salida      OUT Varchar2
) IS
idt_co              INTEGER;
noExiste_idt        EXCEPTION;
men_err             EXCEPTION;
BEGIN
IF (idt IS NULL OR idt = '') THEN
            RAISE men_err;
END IF;
SELECT COUNT (*)
     INTO idt_co
     FROM TIPODESANCION
     WHERE idtsancion = idt;
   IF (idt_co = 0) THEN
      RAISE noExiste_idt;
   END IF;

UPDATE TIPODESANCION
            SET descripción    =des
            WHERE idtsancion=idt;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifTsancion', idt || ',' || des, salida);
EXCEPTION
WHEN noExiste_idt THEN
     salida := 'ERROR: No existe el Tipo de sanción';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
             VALUES (SYSDATE, 'p_modifTsancion', idt || ',' || des, salida);
WHEN men_err THEN
     salida := 'ERROR: No se han rellenado los parámetros necesarios';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'p_modifTsancion', idt || ',' || des, salida);
WHEN OTHERS THEN
     salida := 'ERROR: Error genérico al dar de baja un tipo de sanción';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'p_modifTsancion', idt || ',' || des, salida);
END;

CREATE OR REPLACE  PROCEDURE p_altaAmonestacion (
idm         IN Varchar2,
idt         IN Varchar2,
ida         IN Varchar2,
ano         IN Varchar2,
idc         IN Varchar2,
ids         IN Varchar2,
dni         IN Varchar2,
dia         IN Varchar2,
hor         IN Integer,
gra         IN Integer,
com         IN Char,
des         IN Varchar2,
salida      OUT Varchar2
) IS
rac         char(10);
nm          Integer;
na          char(80);
grv         Integer;
isa         char(10);
men_err EXCEPTION;
```

```
BEGIN
IF (idm IS NULL OR idm = '' OR idt IS NULL OR idt = '' OR ida IS NULL OR ida = '' OR ano IS NULL OR ano = '' OR
   idc IS NULL OR idc = '' OR ids IS NULL OR ids = '' OR dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
        RAISE men_err;
END IF;

INSERT INTO AMONESTACION (idamonestacion, idtamonestacion, idalumno, anyacademico, idcurso, idasignatura,
        dniprofesor, diasemana, hora, gravedad, comunicacion)
        VALUES (idm, idt, ida, ano, idc, ids, dni, dia, hor, gra, com);
altaEST1(ida, salida);
altaEST3(ano, dni, salida);
altaEST6(ano, idc, salida);
altaEST8(salida);
SELECT ractivacion, gravedadactividad, idtsancion  INTO rac, grv, isa FROM REGLAACTIVACION
  WHERE idtamonestacion=RTRIM(idt);
  IF (des IS NULL OR des=' ') THEN
    SELECT descripcion INTO na FROM TIPODESANCION WHERE idtsancion=RTRIM(isa);
  ELSE
    na:=des;
  END IF;
  IF (gra >= grv) THEN
   INSERT INTO SANCION (idsancion, idalumno, anyacademico, idcurso, ractivacion, idtsancion, descripcion, fechasancion,
        idamonestacion)
        VALUES('dummy', ida, ano, RTRIM(idc), RTRIM(rac), RTRIM(isa), na, SYSDATE, RTRIM(idm));
   altaEST2(ano, ida, salida);
   altaEST4(ano, idc, salida);
   altaEST5(ano, salida);
   altaEST7(ano, idc, salida);
  END IF;
  salida := 'OK';
  INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
        ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    salida := 'ERROR: La amonestacion ya existe';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
        ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
WHEN STORAGE_ERROR THEN
    salida := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
        ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
        ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de alta una Amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'p_altaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
        ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
END;

CREATE OR REPLACE  PROCEDURE p_bajaAmonestacion (
idm      IN Varchar2,
idt      IN Varchar2,
ida      IN Varchar2,
ano      IN Varchar2,
idc      IN Varchar2,
ids      IN Varchar2,
dni      IN Varchar2,
```

```
dia       IN Varchar2,
salida    OUT Varchar2
) IS
amone_co          INTEGER;
noExiste_amo      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (idm IS NULL OR idm = '' OR idt IS NULL OR idt = '' OR ida IS NULL OR ida = '' OR ano IS NULL OR ano = '' OR
   idc IS NULL OR idc = '' OR ids IS NULL OR ids = '' OR dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO amone_co
    FROM AMONESTACION
    WHERE idamonestacion=idm AND idtamonestacion=idt AND idalumno=ida AND anyacademico=ano AND
    idcurso=idc AND idasignatura=ids AND dniprofesor=dni AND diasemana=dia;
  IF (amone_co = 0) THEN
     RAISE noExiste_amo;
  END IF;
  DELETE FROM AMONESTACION WHERE idamonestacion=idm AND idtamonestacion=idt AND idalumno=ida AND
          anyacademico=ano AND idcurso=idc AND idasignatura=ids AND dniprofesor=dni AND diasemana=dia;
    bajaEST1(ida, salida);
    altaEST3(ano, dni, salida);
    altaEST6(ano, idc, salida);
    altaEST8(salida);
    DELETE FROM SANCION WHERE idamonestacion=idm;
    bajaEST2(ano, ida, salida);
    bajaEST4(ano, idc, salida);
    altaEST5(ano, salida);
    altaEST7(ano, idc, salida);
    salida := 'OK';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia, salida);
EXCEPTION
WHEN noExiste_amo THEN
    salida := 'ERROR: No existe la Amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al dar de baja una Amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_bajaAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia, salida);
END;

CREATE OR REPLACE  PROCEDURE p_modifAmonestacion (
idm       IN Varchar2,
idt       IN Varchar2,
ida       IN Varchar2,
ano       IN Varchar2,
idc       IN Varchar2,
ids       IN Varchar2,
dni       IN Varchar2,
dia       IN Varchar2,
hor       IN Integer,
gra       IN Integer,
com       IN Char,
```

```
salida     OUT Varchar2
) IS
amone_co          INTEGER;
noExiste_amo      EXCEPTION;
men_err           EXCEPTION;
BEGIN
IF (idm IS NULL OR idm = '' OR idt IS NULL OR idt = '' OR ida IS NULL OR ida = '' OR ano IS NULL OR ano = '' OR
   idc IS NULL OR idc = '' OR ids IS NULL OR ids = '' OR dni IS NULL OR dni = '' OR dia IS NULL OR dia = '') THEN
          RAISE men_err;
END IF;
SELECT COUNT (*)
    INTO amone_co
    FROM AMONESTACION
    WHERE idamonestacion=idm AND idtamonestacion=idt AND idalumno=ida AND anyacademico=ano AND
    idcurso=idc AND idasignatura=ids AND dniprofesor=dni AND diasemana=dia;
  IF (amone_co = 0) THEN
     RAISE noExiste_amo;
  END IF;
UPDATE AMONESTACION
          SET hora          =hor,
          gravedad          =gra,
          comunicación      =com
          WHERE idamonestacion=idm AND idtamonestacion=idt AND idalumno=ida AND anyacademico=ano AND
          idcurso=idc AND idasignatura=ids AND dniprofesor=dni AND diasemana=dia;
salida := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'p_modifAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni || ',' ||
          dia || ',' || hor || ',' || gra || ',' || com, salida);
EXCEPTION
WHEN noExiste_amo THEN
    salida := 'ERROR: No existe la Amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
WHEN men_err THEN
    salida := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
WHEN OTHERS THEN
    salida := 'ERROR: Error genérico al modificar una Amonestación';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'p_modifAmonestacion', idm || ',' || idt || ',' || ida || ',' || ano || ',' || idc || ',' || ids || ',' || dni ||
          ',' || dia || ',' || hor || ',' || gra || ',' || com, salida);
END;

CREATE OR REPLACE  PROCEDURE altaEST1 (
ida         IN Varchar2,
salida2     OUT Varchar2
) IS
nm          Integer;
men_err EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
          RAISE men_err;
END IF;
nm:=1;
INSERT INTO EST1 (idalumno, numeroamonestaciones) VALUES(ida, nm);
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST1', ida || ',' || nm, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    SELECT numeroamonestaciones INTO nm FROM EST1 WHERE idalumno=ida;
```

```
    nm:=nm+1;
    UPDATE EST1 SET NUMEROAMONESTACIONES=nm WHERE idalumno=ida;
    salida2 := 'OK';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'altaEST1', ida || ',' || nm, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'altaEST1', ida || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'altaEST1', ida || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST2';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'altaEST1', ida || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE bajaEST1 (
ida        IN Varchar2,
salida2   OUT Varchar2
) IS
nm         Integer;
men_err EXCEPTION;
BEGIN
IF (ida IS NULL OR ida = '') THEN
        RAISE men_err;
END IF;
SELECT numeroamonestaciones INTO nm FROM EST1 WHERE idalumno=ida;
nm:=nm-1;
IF (nm=0) THEN
    DELETE FROM EST1 WHERE idalumno=ida;
ELSE
    UPDATE EST1 SET NUMEROAMONESTACIONES=nm WHERE idalumno=ida;
END IF;
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'bajaEST1', ida || ',' || nm, salida2);
EXCEPTION
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo darse de baja';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'bajaEST1', ida || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'bajaEST1', ida || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de baja EST1';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'bajaEST1', ida || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE altaEST2 (
ano        IN Varchar2,
ida        IN Varchar2,
salida2   OUT Varchar2
) IS
nm         Integer;
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR ida IS NULL OR ida = '') THEN
        RAISE men_err;
```

```
END IF;
nm:=1;
INSERT INTO EST2 (anyacademico, idalumno, numerosanciones) VALUES(ano, ida, nm);
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
  VALUES (SYSDATE, 'altaEST2', ano || ',' || ida || ',' || nm, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    SELECT numerosanciones INTO nm FROM EST2 WHERE anyacademico=ano AND idalumno=ida;
    nm:=nm+1;
    UPDATE EST2 SET NUMEROSANCIONES=nm WHERE anyacademico=ano AND idalumno=ida;
    salida2 := 'OK';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'altaEST2', ano || ',' || ida || ',' || nm, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST2', ano || ',' || ida || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST2', ano || ',' || ida || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST2';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST2', ano || ',' || ida || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE bajaEST2 (
ano       IN Varchar2,
ida       IN Varchar2,
salida2   OUT Varchar2
) IS
nm        Integer;
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR ida IS NULL OR ida = '') THEN
          RAISE men_err;
END IF;
SELECT numerosanciones INTO nm FROM EST2 WHERE anyacademico=ano AND idalumno=ida;
nm:=nm-1;
IF (nm=0) THEN
     DELETE FROM EST2 WHERE anyacademico=ano AND idalumno=ida;
ELSE
     UPDATE EST2 SET NUMEROSANCIONES=nm WHERE anyacademico=ano AND idalumno=ida;
END IF;
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'bajaEST2', ano || ',' || ida || ',' || nm, salida2);
EXCEPTION
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'bajaEST2', ano || ',' || ida || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'bajaEST2', ano || ',' || ida || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de baja EST2';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'bajaEST2', ano || ',' || ida || ',' || nm, salida2);
END;
```

```
CREATE OR REPLACE  PROCEDURE altaEST3 (
ano       IN Varchar2,
dni       IN Varchar2,
salida2   OUT Varchar2
) IS
tot        Integer;
top        Integer;
nm         Number;
dni2       Varchar2(10);
dni3       Varchar2(10);
cursor cur IS SELECT dniprofesor FROM EST3;
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR dni IS NULL OR dni = '') THEN
          RAISE men_err;
END IF;
   nm:=0.00;
   INSERT INTO EST3 (anyacademico, dniprofesor, mediaamonestaciones) VALUES(ano, dni, nm);
   SELECT count(*) INTO tot FROM AMONESTACION WHERE anyacademico=ano;
   OPEN cur;
   LOOP
       FETCH cur INTO dni3;
       EXIT WHEN cur%NOTFOUND;
       SELECT count(*) INTO top FROM AMONESTACION WHERE anyacademico=ano AND dniprofesor=dni3;
       nm:=top/tot;
       nm:=nm*100;
       UPDATE EST3 SET mediaamonestaciones=nm WHERE anyacademico=ano AND dniprofesor=dni3;
   END LOOP;
   close cur;
   salida2 := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST3', ano || ',' || dni || ',' || nm, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   SELECT count(*) INTO tot FROM AMONESTACION WHERE anyacademico=ano;
   OPEN cur;
   LOOP
       FETCH cur INTO dni3;
       EXIT WHEN cur%NOTFOUND;
       SELECT count(*) INTO top FROM AMONESTACION WHERE anyacademico=ano AND dniprofesor=dni3;
       nm:=top/tot;
       nm:=nm*100;
       UPDATE EST3 SET mediaamonestaciones=nm WHERE anyacademico=ano AND dniprofesor=dni3;
   END LOOP;
   CLOSE cur;
   salida2 := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST3', ano || ',' || dni || ',' || nm, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST3', ano || ',' || dni || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST3', ano || ',' || dni || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST3';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST3', ano || ',' || dni || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE altaEST4 (
ano       IN Varchar2,
```

```
idc        IN Varchar2,
salida2   OUT Varchar2
) IS
nm         Integer;
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
          RAISE men_err;
END IF;
nm:=1;
INSERT INTO EST4 (anyacademico, idcurso, numerosanciones) VALUES(ano, idc, nm);
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
  VALUES (SYSDATE, 'altaEST4', ano || ',' || idc || ',' || nm, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    SELECT numerosanciones INTO nm FROM EST4 WHERE anyacademico=ano AND idcurso=idc;
    nm:=nm+1;
    UPDATE EST4 SET NUMEROSANCIONES=nm WHERE anyacademico=ano AND idcurso=idc;
    salida2 := 'OK';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
              VALUES (SYSDATE, 'altaEST4', ano || ',' || idc || ',' || nm, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST4', ano || ',' || idc || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST4', ano || ',' || idc || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST4';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST4', ano || ',' || idc || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE bajaEST4 (
ano       IN Varchar2,
idc       IN Varchar2,
salida2    OUT Varchar2
) IS
nm         Integer;
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
          RAISE men_err;
END IF;
SELECT numerosanciones INTO nm FROM EST4 WHERE anyacademico=ano AND idcurso=idc;
nm:=nm-1;
IF (nm=0) THEN
      DELETE FROM EST4 WHERE anyacademico=ano AND idcurso=idc;
ELSE
       UPDATE EST4 SET NUMEROSANCIONES=nm WHERE anyacademico=ano AND idcurso=idc;
END IF;
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
    VALUES (SYSDATE, 'bajaEST4', ano || ',' || idc || ',' || nm, salida2);
EXCEPTION
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
              VALUES (SYSDATE, 'bajaEST4', ano || ',' || idc || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
```

```
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'bajaEST4', ano || ',' || idc || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de baja EST4';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST4', ano || ',' || idc || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE altaEST5 (
ano       IN Varchar2,
salida2   OUT Varchar2
) IS
nm        Integer;
ida       char(10);
nom       char(30);
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '') THEN
  RAISE men_err;
END IF;
SELECT max(numerosanciones) INTO nm FROM EST2 WHERE anyacademico=ano;
SELECT idalumno INTO ida FROM EST2 WHERE anyacademico=ano and numerosanciones=nm;
SELECT nombre INTO nom FROM ALUMNO WHERE idalumno=ida;
INSERT INTO EST5 (anyacademico, nomalumno) VALUES(ano, nom);
salida2 := 'OK';
INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
   VALUES (SYSDATE, 'altaEST5', ano || ',' || nom, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    UPDATE EST5 SET nomalumno=nom WHERE anyacademico=ano;
    salida2 := 'OK';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST5', ano || ',' || nom, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST5', ano || ',' || nom, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST5', ano || ',' || nom, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST5';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST2', ano || ',' || nom, salida2);
END;

CREATE OR REPLACE  PROCEDURE altaEST6 (
ano       IN Varchar2,
idc       IN Varchar2,
salida2   OUT Varchar2
) IS
top       Integer;
cu        Integer;
top1      Integer;
ana       Varchar2(4);
icu       Varchar2(10);
dni       Varchar2(10);
pro       char(10);
nom       char(30);
cursor cur IS  select count(*), dniprofesor, anyacademico, idcurso from amonestacion
   group by anyacademico, idcurso,dniprofesor order by count(*) desc;
men_err EXCEPTION;
BEGIN
```

```
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
  RAISE men_err;
END IF;
    OPEN cur;
    LOOP
        FETCH cur INTO top, dni, ana, icu;
         EXIT WHEN cur%NOTFOUND;
         SELECT nombre INTO nom FROM PROFESOR WHERE dniprofesor=RTRIM(dni);
         SELECT count(*) INTO cu FROM EST6 WHERE anyacademico=ana AND idcurso=icu;
         IF cu > 0 THEN
            IF top > cu THEN
               UPDATE EST6 SET nomprofesor=nom WHERE anyacademico=ana AND idcurso=icu;
            END IF;
        ELSE
            INSERT INTO EST6 (anyacademico, idcurso, nomprofesor) VALUES(ana, icu, nom);
        END IF;
     END LOOP;
     CLOSE cur;
     salida2 := 'OK';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'altaEST6', ano || ',' || idc || ',' || nom, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
     OPEN cur;
     LOOP
         FETCH cur INTO top, dni, ana, icu;
         EXIT WHEN cur%NOTFOUND;
         SELECT nombre INTO nom FROM PROFESOR WHERE dniprofesor=RTRIM(dni);
         UPDATE EST6 SET nomprofesor=nom WHERE anyacademico=ana AND idcurso=icu;
     END LOOP;
    CLOSE cur;
     salida2 := 'OK';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
     VALUES (SYSDATE, 'altaEST6', ano || ',' || idc || ',' || nom, salida2);
WHEN STORAGE_ERROR THEN
     salida2 := 'ERROR: No pudo insertarse';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'altaEST6', ano || ',' || idc || ',' || nom, salida2);
WHEN men_err THEN
     salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'altaEST6', ano || ',' || idc || ',' || nom, salida2);
WHEN OTHERS THEN
     salida2 := 'ERROR: Error genérico al dar de alta EST6';
     INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
            VALUES (SYSDATE, 'altaEST6', ano || ',' || idc || ',' || nom, salida2);
END;

CREATE OR REPLACE  PROCEDURE altaEST7 (
ano       IN Varchar2,
idc       IN Varchar2,
salida2   OUT Varchar2
) IS
nm        Number;
tot       Integer;
top       Integer;
ana       Varchar2(4);
icu       Varchar2(10);
cursor cur IS SELECT count(*), anyacademico, idcurso FROM SANCION group by anyacademico, idcurso;
men_err EXCEPTION;
BEGIN
IF (ano IS NULL OR ano = '' OR idc IS NULL OR idc = '') THEN
  RAISE men_err;
END IF;
```

```
   nm:=0.00;
   INSERT INTO EST7 (anyacademico, idcurso, mediasanciones) VALUES(ano, idc, nm);
   SELECT count(*) INTO tot FROM SANCION;
   OPEN cur;
   LOOP
      FETCH cur INTO top, ana, icu;
      EXIT WHEN cur%NOTFOUND;
      nm:=top/tot;
      nm:=nm*100;
      UPDATE EST7 SET mediasanciones=nm WHERE anyacademico=ana AND idcurso=RTRIM(icu);
   END LOOP;
   CLOSE cur;
   salida2 := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST7', ano || ',' || idc || ',' || nm, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    SELECT count(*) INTO tot FROM SANCION;
    OPEN cur;
    LOOP
        FETCH cur INTO top, ana, icu;
        EXIT WHEN cur%NOTFOUND;
        nm:=top/tot;
        nm:=nm*100;
        UPDATE EST7 SET mediasanciones=nm WHERE anyacademico=ana AND idcurso=RTRIM(icu);
   END LOOP;
   CLOSE cur;
   salida2 := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST7', ano || ',' || idc || ',' || nm, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
           VALUES (SYSDATE, 'altaEST7', ano || ',' || idc || ',' || nm, salida2);
WHEN men_err THEN
    salida2 := 'ERROR: No se han rellenado los parámetros necesarios';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST7', ano || ',' || idc || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST7';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST7', ano || ',' || idc || ',' || nm, salida2);
END;

CREATE OR REPLACE  PROCEDURE altaEST8 (
salida2   OUT Varchar2
) IS
nm         Integer;
men_err EXCEPTION;
BEGIN
   select count(*) INTO nm from alumno where idalumno not IN(select idalumno from amonestacion);
   INSERT INTO EST8 (codigo, numeroalumnos) VALUES('ALUMNOS', nm);
   salida2 := 'OK';
   INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST8', 'ALUMNOS' || ',' || nm, salida2);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    UPDATE EST8 SET NUMEROALUMNOS=nm WHERE codigo='ALUMNOS';
    salida2 := 'OK';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
          VALUES (SYSDATE, 'altaEST8', 'ALUMNOS' || ',' || nm, salida2);
WHEN STORAGE_ERROR THEN
    salida2 := 'ERROR: No pudo insertarse';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
```

```
        VALUES (SYSDATE, 'altaEST8', 'ALUMNOS' || ',' || nm, salida2);
WHEN OTHERS THEN
    salida2 := 'ERROR: Error genérico al dar de alta EST8';
    INSERT INTO LOGS (fecha, procedimiento, paramsentrada, paramssalida)
        VALUES (SYSDATE, 'altaEST8', 'ALUMNOS' || ',' || nm, salida2);
END;
```