2015/2016

ESPECIFICACIÓN DE REQUISITOS SOFTWARE

Eduardo Vela Galindo

Fernando Rivilla Bravo

Iván María Paredes

Rodrigo de Miguel González

Rubén Barrado González

Tomás Muñoz Testón

Universidad Complutense

Facultad de Informática

ÍNDICE

1. INTRODUCCIÓN.

1.1. Propósito

EL propósito de este documento es hacer la especificación de requisitos (SRS) siguiendo el estándar IEEE Std. 830-1998 para que nuestro sistema permita el correcto funcionamiento del gestor de un centro educativo.

1.2. Alcance.

Nuestro sistema se llama "SMS – School Management Software" y su principal objetivo es organizar un centro educativo de forma automática y cómoda para el cliente, mediante una BBDD que contenga la información de los alumnos, sus matrículas y sus cursos; y, por otra parte, la información de los profesores, sus turnos y las materias que imparten.

El programa podrá ser utilizado por el personal administrativo, que podrá gestionar las entidades anteriormente mencionadas.

1.3. Definiciones, Acrónimos y Abreviaturas.

SMS – School Management Software

SW: Software

HW: Hardware

BBDD: Base de datos

GUI: Graphical User Interface (Interfaz Grafica de Usuario)

IEEE: Institute of Electrical And Electronics Engineers (Instituto De ingenieros eléctricos y electrónicos).

JPA: Java Persistence Api

1.4. Referencias

Nuestra referencia es el Standard de IEEE 830-1998, 20 Oct. 1998 de la IEEE Computer Society.

1.5. Resumen.

Inicialmente se determina el ámbito de la aplicación, especificando como se integra en el sistema. En secciones posteriores se identifican funciones principales del sistema de forma muy general así como restricciones y los usuarios a los que va destinada.

Finalmente se hace un análisis exhaustivo de los requisitos haciendo una partición del sistema en las funciones identificadas, así como una visión de cómo podrían ser las interfaces externas de la misma.

2. DESCRIPCIÓN GENERAL

El software permitirá llevar a cabo la gestión de un centro educativo de forma cómoda y sencilla para el usuario.

2.1. Perspectiva del Producto

El sistema que vamos a desarrollar es para un proyecto universitario de la asignatura “Modelado de Software” por lo que es independiente ya que no lo relacionamos con otros productos.

2.2. Funciones del Producto

Versión DAO

2.2.1. Gestión de alumnos

Se encargará de manejar toda la información de los alumnos (creación, modificación y baja) en el sistema y listarlos. Nunca se podrá dar de baja a un alumno con una matrícula del curso actual y si se da de baja un alumno sus matrículas quedarán en el sistema.

2.2.2. Gestión de matriculas

El usuario de la aplicación podrá crear matrículas, modificarlas, borrarlas y listarlas en cualquier momento. También se podrán listar los alumnos que tengan más de N matrículas en los ultimos M años. Nunca se podrá borrar una matrícula una vez que el curso esté activo.

2.2.3. Gestión de cursos

Este apartado podrá manejar toda la información de los cursos (creación, modificación y eliminar) en el sistema y listarlos. Nunca se podrá eliminar un curso si tiene alumnos inscritos.

Versión JPA

2.2.4. Gestión de turnos

Cada profesor pertenecerá a un turno en el cual se podrá saber la hora de inicio y de fin así como si su horario es de mañana o de tarde. Tambien se podrá calcular el sueldo total de un turno en función del los profesore de dicho turno.

2.2.5. Gestión de profesores

Se podrán insertar nuevos profesores en el centro, modificarlos, borrarlos. A la hora de mostrarlos y listarlos se calculará también su sueldo en función del turno de trabajo que tengan.

2.2.6. Gestión de materias

El cliente final podrá crear reservas, modificarlas, borrarlas y listarlas en cualquier momento. Además, existe una opción de actualización de reservas que se encarga de asignar las plazas correspondientes a las reservas del día en el que se encuentra el sistema.

2.3. Características de los Usuarios

El producto estará destinado al personal administrativo de un centro educativo que deberá tener unos conocimientos básicos de informática, a nivel de usuario, para el manejo de la herramienta.

La aplicación se verá sencilla e intuitiva para que no exista ningún problema a la hora de manejarla.

2.4. Restricciones

La implementación del programa se hará en Java utilizando el software IBM RSA proporcionado por el profesor como herramienta de modelado y desarrollo. Para el almacenamiento de datos se usará una base de datos relacional.

Las fechas de las distintas entregas de partes del proyecto son una restricción temporal, por lo que se deben tener en cuenta a la hora de la planificación del mismo.

2.5. Suposiciones y Dependencias

Se presupone que el sistema correrá sobre una plataforma Windows, si no es así afectará a ciertos detalles técnicos y se tendrían que revisar y quizás cambiar los requisitos.

El sistema depende de una BBDD relacional (MySQL 5.x) para el almacenamiento de los datos de los alumnos, sus matrículas y cursos, y para el incremento de la aplicación se utilizará JPA para gestionar la persistencia de los profesores, sus turnos y materias que imparten.

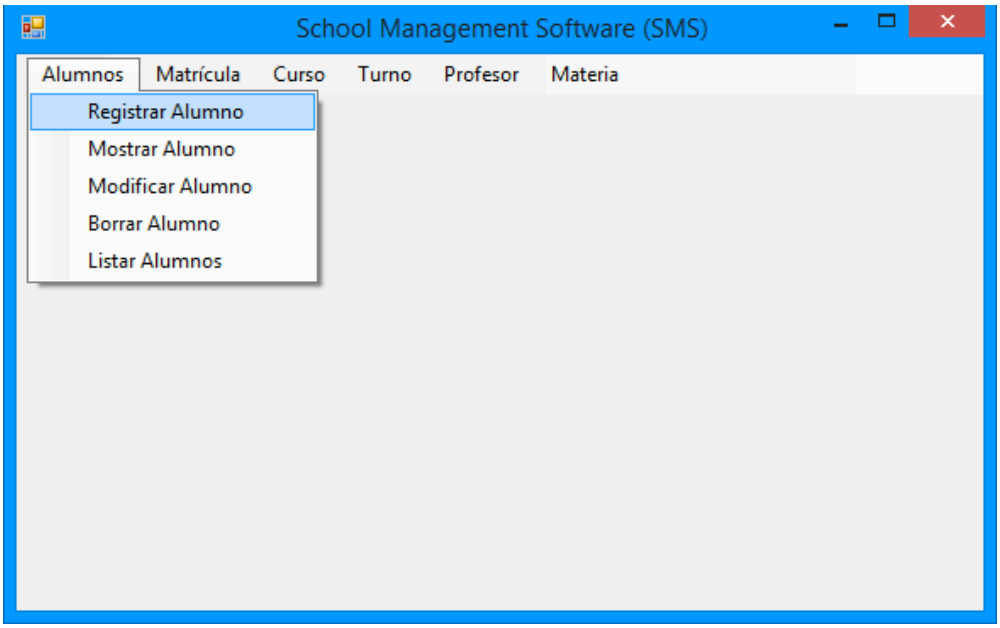
2.6. Requisitos Futuros

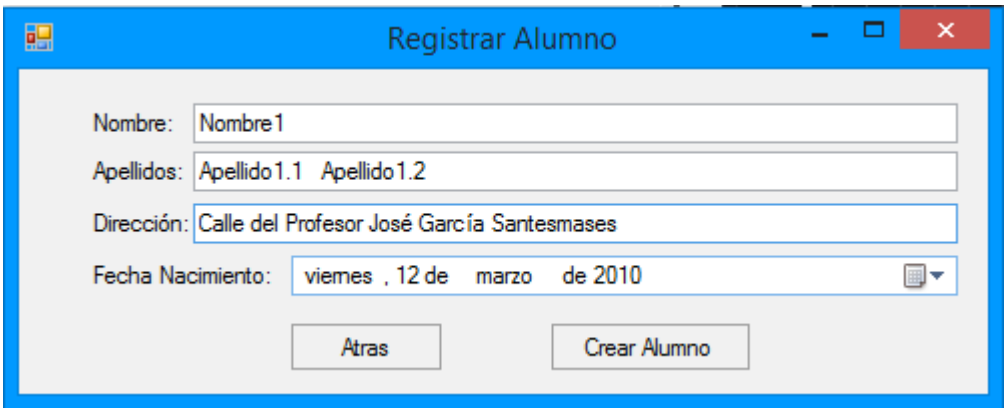
Ha fecha de la redacción de ésta SRS del proyecto “School Management Software” no se contemplan futuros requisitos.

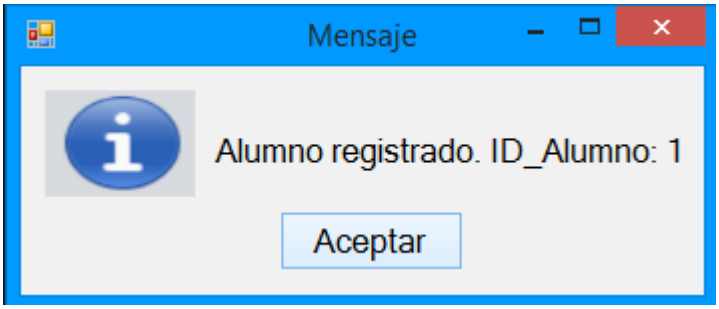
3. REQUISITOS ESPECÍFICOS

3.1. Interfaces externas

El usuario podrá acceder a las principales funcionalidades de nuestra aplicación mediante el menú principal de una forma sencilla.

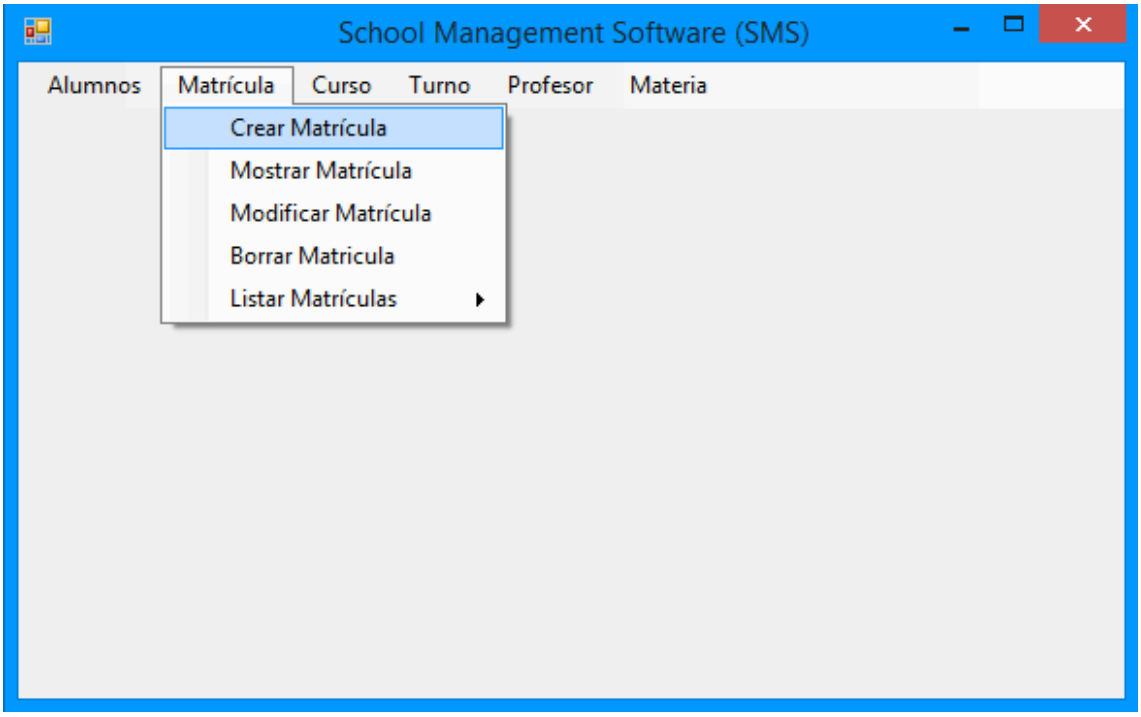


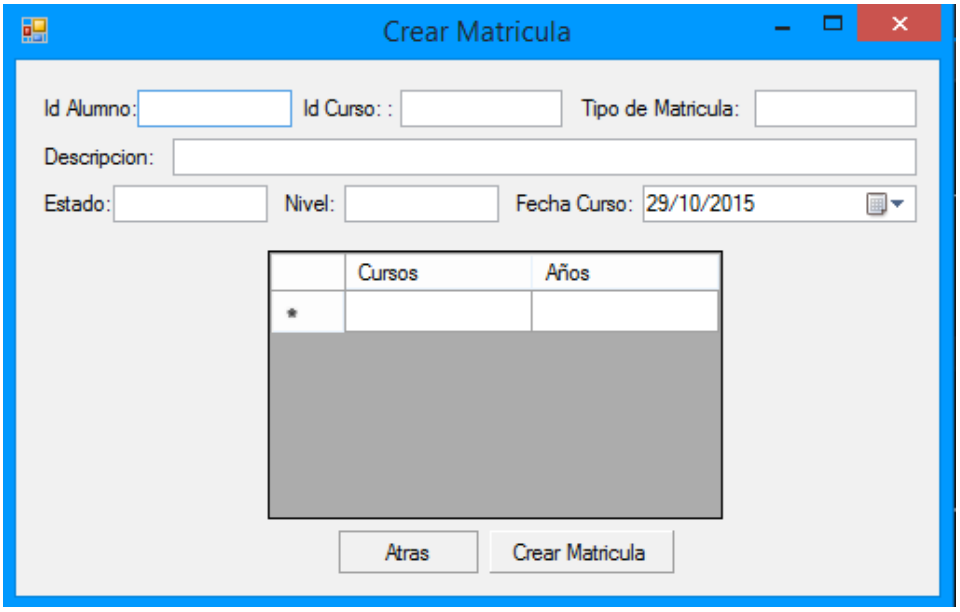


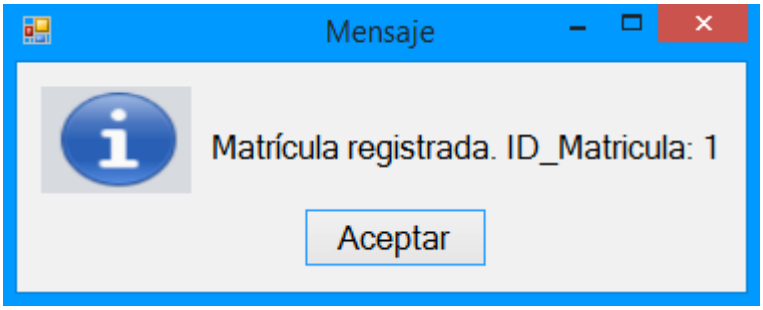


En el caso de querer registrar un alumno en el sistema, la aplicación solicitara la introducción de los datos tales como: Nombre; Apellidos; Dirección y Fecha de

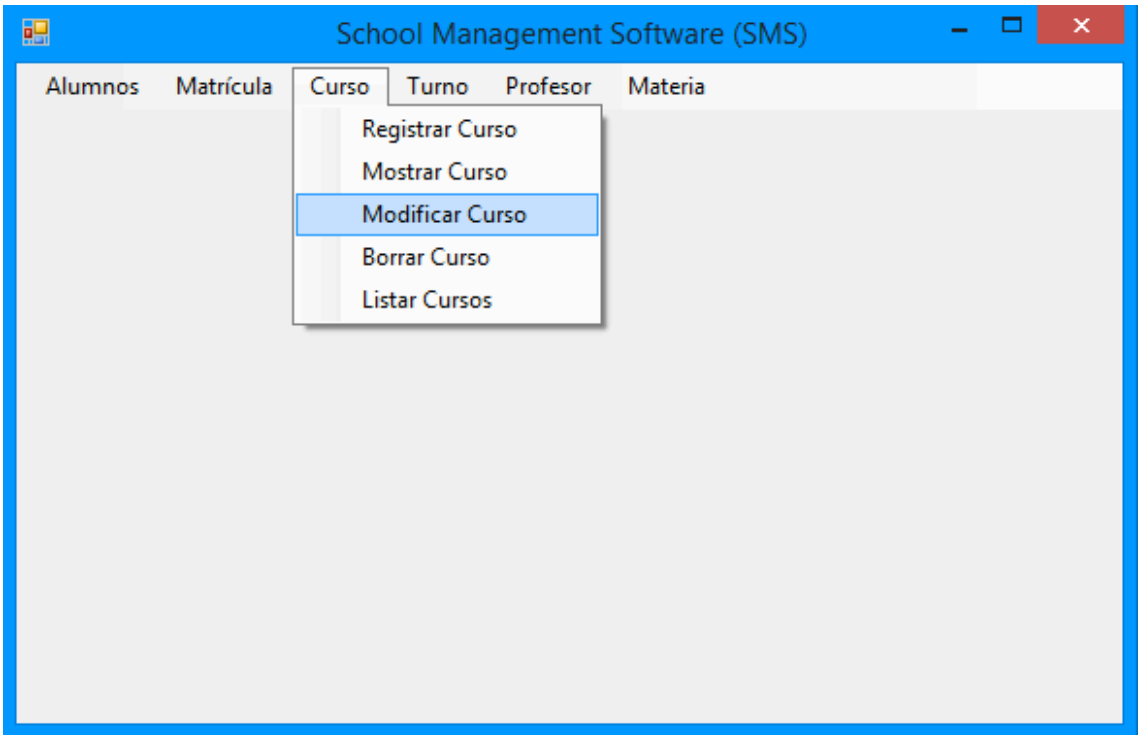
Nacimiento, al completar o no la operación la aplicación mostrará un mensaje informativo al usuario.

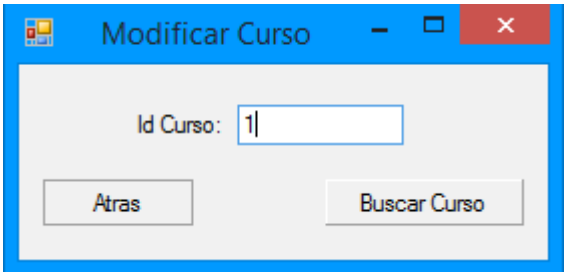


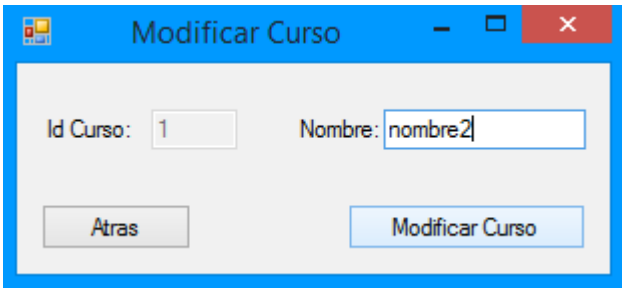


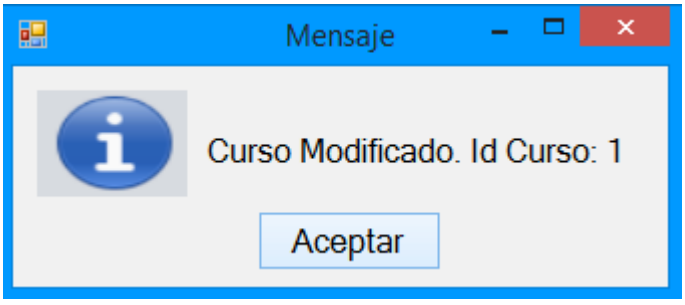


En el caso de querer registrar una matrícula en el sistema, la aplicación solicitara la introducción de los datos tales como: ID del alumno; ID del Curso; Tipo de Matricula; Descripción; Estado; Nivel y Fecha del Curso, al completar o no la operación la aplicación mostrará un mensaje informativo al usuario.

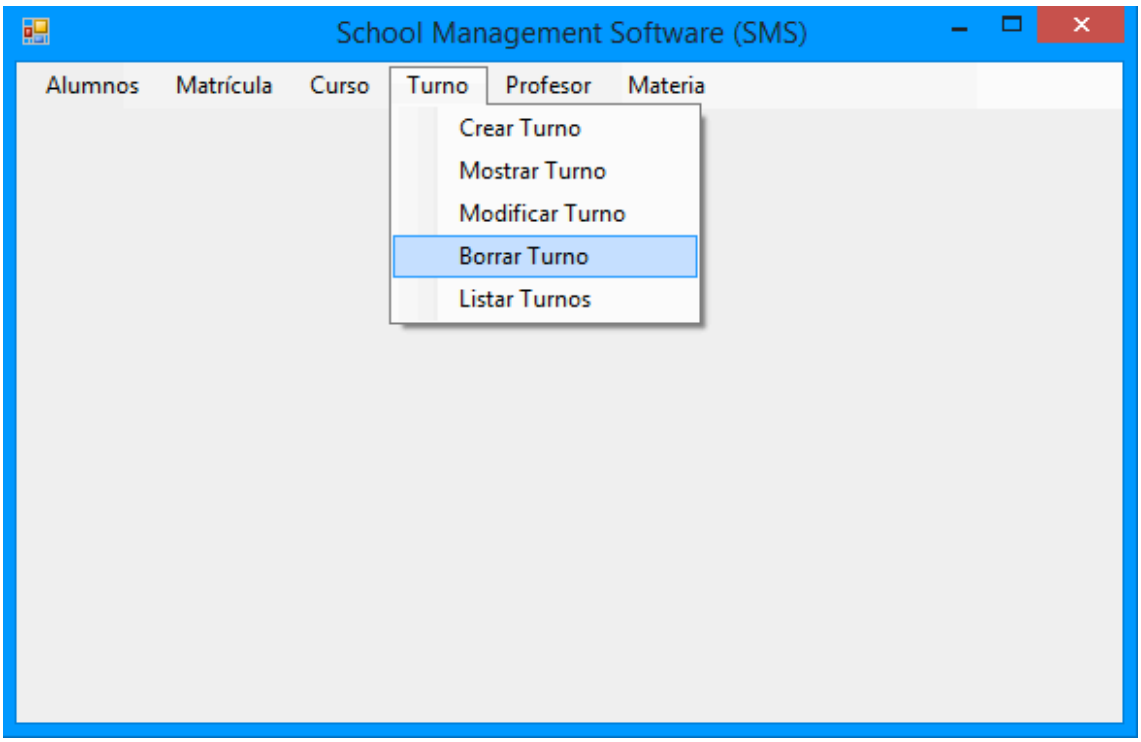


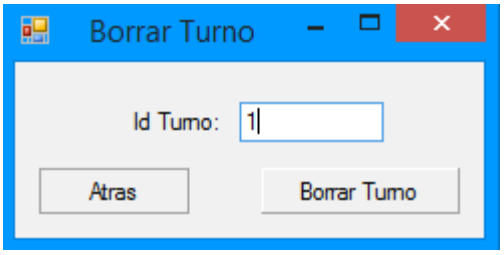


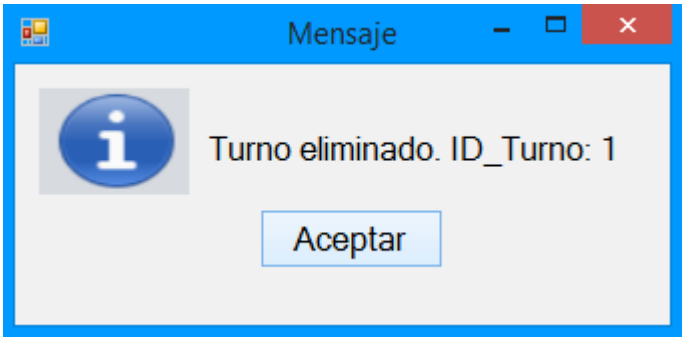




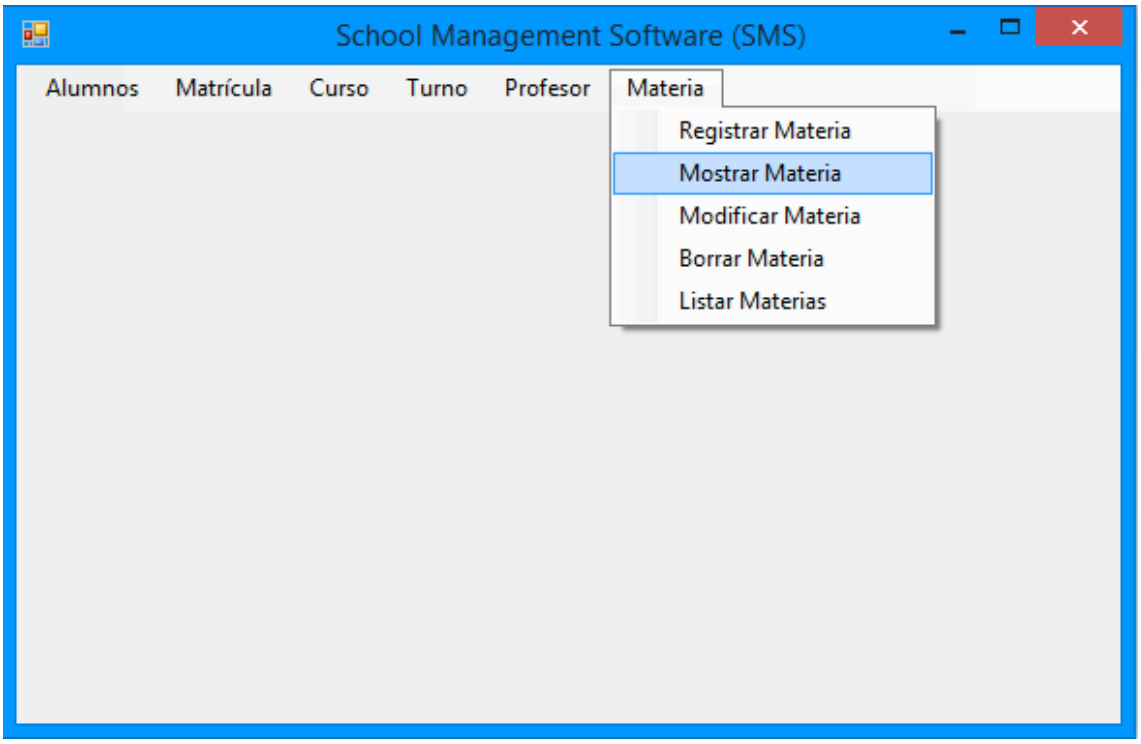
En el caso de querer modificar un curso en el sistema, la aplicación hará una búsqueda previa solicitando el ID del curso que se quiera modificar, una vez localizado el curso solicitara la introducción de los datos a modificar, en este caso, ID de curso se muestra pero no se puede modificar y si el resto de datos como es el nombre del curso, al completar o no la operación la aplicación mostrará un mensaje informativo al usuario.

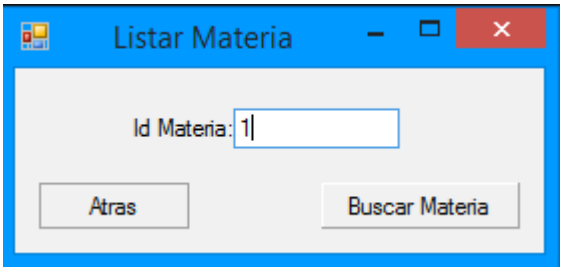




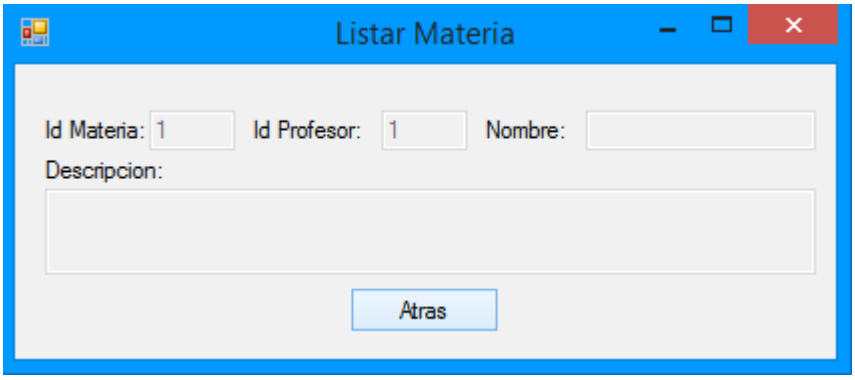


En el caso de querer dar de baja un turno en el sistema, la aplicación hará una búsqueda previa solicitando el ID del turno que se quiera dar de baja (baja lógica), al completar o no la operación la aplicación mostrará un mensaje informativo al usuario.



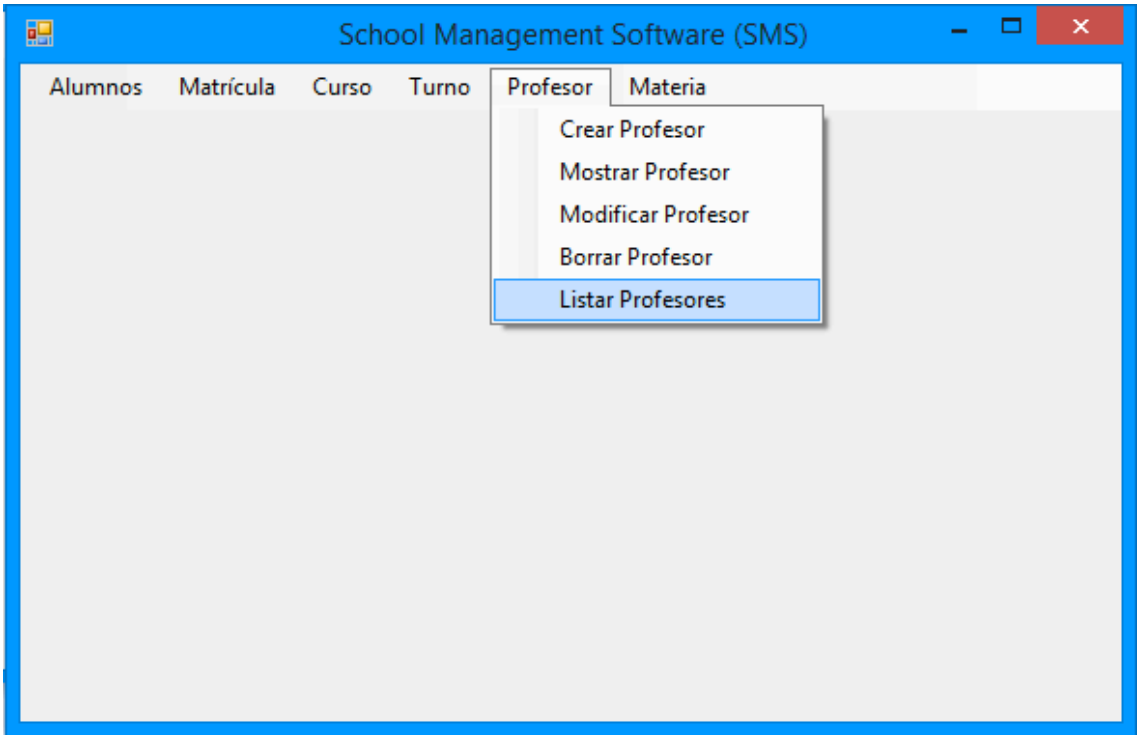


(mostrar Materia)



En el caso de querer mostrar o listar una materia en el sistema, la aplicación hará una búsqueda previa solicitando el ID de la materia en cuestión que se quiera mostrar y mostrara los datos por pantalla, al completar o no la operación la aplicación mostrará un mensaje informativo al usuario.

Como apunte añadir que en el caso de querer listar un conjunto de materias, alumnos, profesores…. Simplemente con seleccionar en el menú la opción de listar la aplicación mostraría el conjunto por pantalla


School Management Software (SMS)
Alumnos
Matrícula
Curso
Turno
Profesor
Materia
Crear Profesor
Mostrar Profesor
Modificar Profesor
Borrar Profesor
Listar Profesores

3.2. Funciones

Incremento DAO

- **Módulo Alumnos**

Función	**Crear Alumno *(ALU01)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Registra a un alumno en el sistema.
Entrada	Nombre, apellidos, fecha_nacimiento, dirección, teléfono
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	Nombre, apellidos, fecha_nacimiento, dirección, teléfono
Acción	Registra a un alumno en el sistema.
Precondición	○ El alumno no existe previamente. ○ Todos los campos, en GUI, están rellenos.
Postcondición	○ Alumno registrado en la base de datos
Efectos lat.	-

Función	**Muestra alumno *(ALU02)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra los datos de un alumno por pantalla.
Entrada	ID_Alumno

Salida	Muestra los datos por pantalla.
Origen	GUI
Destino	GUI
Necesita	DNI
Acción	Muestra los datos del alumno por pantalla.
Precondición	○ Tiene que existir el alumno en el sistema, es decir, tiene que haber un identificador previo.
Postcondición	-
Efectos lat.	-

Función	**Modificar Alumno *(ALU03)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica a un alumno ya registrado.
Entrada	ID_Alumno, Nombre, apellidos, fecha_nacimiento, dirección, teléfono
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	ID_Alumno
Acción	Modifica a un alumno registrado en el sistema.
Precondición	○ El alumno debe estar registrado previamente, es decir, tiene que existir el identificador del alumno previamente

Postcondición	○ Los datos del alumno se verán modificados en la base de datos
Efectos lat.	-

Función	**Borrar Alumno *(ALU04)***
Prioridad	Media
Estabilidad	Alta
Descripción	Da de baja a un alumno registrado (baja lógica).
Entrada	ID_Alumno
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	DNI
Acción	Da de baja a un Alumno del sistema (baja lógica).
Precondición	○ El alumno debe estar registrado en el sistema, es decir, debe existir el identificador del alumno que se desea borrar.
Postcondición	○ El alumno se da de baja del sistema (baja lógica).
Efectos lat.	-

Función	Listar alumnos *(ALU05)*
Prioridad	Alta
Estabilidad	Alta
Descripción	Lista los datos de los alumnos por pantalla.
Entrada	-
Salida	Muestra los datos por pantalla.
Origen	GUI
Destino	GUI
Necesita	-
Acción	Muestra los datos de los alumnos por pantalla.
Precondición	○ Tiene que existir al menos un alumno.
Postcondición	-
Efectos lat.	-

- **Módulo Matricula**

Función	Crear Matricula *(MTR01)*
Prioridad	Alta
Estabilidad	Media
Descripción	Crea una matrícula en el sistema.
Entrada	tipo_maricula, id_alumno, id_curso, descripción, estado, fecha_curso, Nivel
Salida	Mensaje de éxito e identificador.

Origen	GUI
Destino	GUI
Necesita	tipo_matricula, id_alumno, id_curso, descripción, estado, fecha_curso, Nivel
Acción	Registra una matrícula en el sistema.
Precondición	Todos los datos rellenos en GUI.
Postcondición	Plaza registrada en el sistema.
Efectos lat.	-

Función	**Mostrar Matricula *(MTR02)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra los datos de una Matrícula.
Entrada	Id_matricula
Salida	Muestra los datos de la Matricula.
Origen	GUI
Destino	GUI
Necesita	Id_matricula
Acción	Muestra información de la Matricula.
Precondición	○ Que la matricula exista en el sistema.
Postcondición	○ Datos mostrados por pantalla.
Efectos lat.	-

Función	**Modificar Matricula *(MTR03)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Se muestran los datos por pantalla, el usuario los modifica y el sistema registra los cambios.
Entrada	id_matricula.
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	id_matricula.
Acción	Modifica los datos de una matrícula ya registrada.
Precondición	Todos los datos están completos en GUI.
Postcondición	Datos modificados correctamente.
Efectos lat.	-

Función	**Borrar Matricula *(MTR04)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Elimina del sistema una matrícula registrada (baja lógica).
Entrada	ID_matricula
Salida	Mensaje de éxito e identificador.
Origen	GUI

Destino	GUI
Necesita	ID_matricula
Acción	Elimina del sistema una matrícula (baja lógica).
Precondición	ID_matricula válida. El curso no está activo.
Postcondición	Matricula dada de baja
Efectos lat.	-

Función	**Mostrar Todas las Matriculas *(MTR05)***
Prioridad	Alta
Estabilidad	Media
Descripción	Muestra todas las Matriculas.
Entrada	-
Salida	Listado de las matrículas.
Origen	GUI
Destino	GUI
Necesita	-
Acción	Lista por pantalla las matrículas.
Precondición	Existen matriculas registradas en el sistema, al menos una.
Postcondición	Listado de matrículas por pantalla.
Efectos lat.	-

Función	Mostrar las Matriculas en un periodo *(MTR06)*
Prioridad	Alta
Estabilidad	Media
Descripción	Muestra los datos de todas las Matriculas en un periodo de tiempo concreto.
Entrada	Fecha_curso_INI, fecha_curso_FIN
Salida	Listado de las matrículas en dicho periodo.
Origen	GUI
Destino	GUI
Necesita	-
Acción	Lista por pantalla las matrículas.
Precondición	Fecha de inicio anterior a la fecha de fin
Postcondición	Listado de matrículas por pantalla.

Función	Añadir Curso a Matricula*(MTR07)*
Prioridad	Alta
Estabilidad	Media
Descripción	Añade a la matricula un curso con un nivel.
Entrada	ID_Curso, ID_materia y Nivel
Salida	Mensaje de éxito.
Origen	GUI

Destino	GUI
Necesita	Listado de cursos.
Acción	Añade a la matricula un curso con un nivel concreto.
Precondición	ID_Curso válido. El curso existe en la Base de Datos y está activo.
Postcondición	Matricula actualizada con el curso y nivel añadidos.
Efectos lat.	-

Función	**Quitar Curso a Martricula *(MTR08)***
Prioridad	Alta
Estabilidad	Media
Descripción	Elimina de la matricula un curso.
Entrada	ID_Curso, ID_Materia y el nivel
Salida	Mensaje de éxito.
Origen	GUI
Destino	GUI
Necesita	Listado de cursos añadidos en la matricula
Acción	Elimina a la matricula un curso.
Precondición	ID_Curso válido. El curso está asignado a la matrícula.
Postcondición	Matricula actualizada sin el curso eliminados.
Efectos lat.	-

Función	**Mostrar alumnos con más de N matriculas en los últimos M años *(MTR09)***
Prioridad	Alta
Estabilidad	Media
Descripción	Lista por pantalla los alumnos cuya cantidad de matriculas es mayor que N en un periodo de M años.
Entrada	N_matriculas, M_años
Salida	Listado de alumnos con más de N matriculas en M años.
Origen	GUI
Destino	GUI
Necesita	N_matriculas, M_años
Acción	Lista por pantalla los alumnos cuya cantidad de matriculas es mayor que N en un periodo de M años.
Precondición	Tanto N_matriculas como M_años deben ser mayores que 0.
Postcondición	Listado de alumnos por pantalla.
Efectos lat.	-

- **Módulo Curso**

Función	**Registrar Curso (CUR01)**
Prioridad	Alta
Estabilidad	Alta
Descripción	Registra un curso nuevo en la BBDD.
Entrada	Nombre
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	Nombre
Acción	Registra un nuevo curso en el sistema.
Precondición	Todos los campos rellenos.
Postcondición	El curso queda registrado.
Efectos lat.	-

Función	**Mostrar Curso (CUR02)**
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra un curso concreto por pantalla.
Entrada	Id_Curso

Salida	Muestra un curso concreto por pantalla.
Origen	GUI
Destino	GUI
Necesita	ID_Curso
Acción	Muestra un curso concreto por pantalla.
Precondición	Existe el curso.
Postcondición	-
Efectos lat.	-

Función	**Modificar Curso (CUR03)**
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica un curso ya registrado.
Entrada	Id_Curso
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	ID_Curso.
Acción	Modifica un curso en el sistema.
Precondición	El Id_Curso existe.
Postcondición	El curso queda modificado en el sistema.
Efectos lat.	-

Función	Borrar Curso (CUR04)
Prioridad	Alta
Estabilidad	Alta
Descripción	Elimina un curso ya registrado.
Entrada	ID_curso.
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	ID_curso.
Acción	Elimina un curso del sistema.
Precondición	El ID de curso existe.
Postcondición	El curso queda eliminado del sistema (baja lógica)
Efectos lat.	-

Función	Mostrar Todos los cursos (CUR05)
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra todos los cursos registrados por pantalla.
Entrada	-
Salida	Muestra los cursos por pantalla.

Origen	GUI
Destino	GUI
Necesita	-
Acción	Muestra los cursos por pantalla.
Precondición	Hay al menos un curso registrado.
Postcondición	-
Efectos lat.	-

Incremento JPA

- **Módulo Turno**

Función	**Crear Turno *(TUR01)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Crea un turno en el sistema.
Entrada	hora_inicio, hora_fin, observaciones
Salida	Mensaje de éxito.
Origen	GUI
Destino	GUI
Necesita	hora_inicio, hora_fin, observaciones
Acción	Registra un turno en el sistema.
Precondición	Todos los datos rellenos.

Postcondición	Turno registrado en el sistema.
Efectos lat.	-

Función	**Mostrar Turno *(TUR02)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra los datos de los turnos.
Entrada	ID_turno
Salida	Muestra los datos de un turno.
Origen	GUI
Destino	GUI
Necesita	ID_turno
Acción	Muestra información de un turno
Precondición	El turno existe en el sistema.
Postcondición	Datos mostrados por pantalla.
Efectos lat.	-

Función	**Modificar Turno *(TUR03)***
Prioridad	Alta
Estabilidad	Media
Descripción	Se muestran los datos por pantalla, el usuario los modifica y el sistema registra los cambios.
Entrada	ID_turno, hora_inicio, hora_fin, observaciones.
Salida	Mensaje de éxito e identificador.
Origen	GUI
Destino	GUI
Necesita	ID_turno, hora_inicio, hora_fin, observaciones.
Acción	Modifica los datos de un turno ya registrado.
Precondición	Todos los datos están completos.
Postcondición	Datos modificados correctamente.
Efectos lat.	-

Función	**Borrar Turno *(TUR04)***
Prioridad	Alta
Estabilidad	Media
Descripción	Elimina del sistema un turno registrado.
Entrada	ID_turno

Salida	Mensaje de éxito e identificador
Origen	GUI
Destino	GUI
Necesita	ID_turno
Acción	Elimina del sistema un turno.
Precondición	ID_turno válido.
Postcondición	-
Efectos lat.	-

Función	**Listar Turnos (TUR05)**
Prioridad	Alta
Estabilidad	Media
Descripción	Muestra los datos de todos los turnos.
Entrada	-
Salida	Listado de los turnos.
Origen	GUI
Destino	GUI
Necesita	-
Acción	Lista por pantalla todos los turnos.
Precondición	Existen al menos un turno.
Postcondición	Listado de turnos por pantalla.
Efectos lat.	-

Función	Calcular nómina de un turno (TUR06)
Prioridad	Alta
Estabilidad	Alta
Descripción	Se calcula la nómina de un turno sumando los sueldos de todos los profesores activos en ese turno.
Entrada	ID_turno
Salida	Nómina calculada del turno.
Origen	GUI
Destino	GUI
Necesita	ID_turno como clave foránea.
Acción	Calcula la nómina del turno.
Precondición	Que el turno exista
Postcondición	Nomina del curso calculada
Efectos lat.	-

- **Módulo Profesor**

Función	**Crear Profesor (PRF01)**
Prioridad	Alta
Estabilidad	Alta
Descripción	Crea un profesor en el sistema.
Entrada	ID_Turno, nombre, apellidos, teléfono, email
Salida	Mensaje de éxito e identificador del profesor creado
Origen	GUI
Destino	GUI
Necesita	ID_turno como clave foránea
Acción	Registra un profesor en el sistema.
Precondición	No exista el mismo profesor e ID_Turno sea correcto
Postcondición	Profesor registrado en el sistema.
Efectos lat.	-

Función	**Mostrar Profesor *(PRF02)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra los datos de un profesor.
Entrada	Id_Profesor
Salida	Los datos de un profesor.
Origen	GUI

Destino	GUI
Necesita	Id_profesor
Acción	Muestra información de un profesor
Precondición	El profesor existe en el sistema.
Postcondición	-
Efectos lat.	-

Función	**Modificar Profesor *(PRF03)***
Prioridad	Alta
Estabilidad	Media
Descripción	Se muestran los datos por pantalla, el usuario los modifica y el sistema registra los cambios.
Entrada	Id_prof y posteriormente ID_Turno, nombre, apellidos telefono, email
Salida	Mensaje de éxito con el ID del profesor.
Origen	GUI
Destino	GUI
Necesita	ID_turno como clave foránea
Acción	Modifica los datos de un profesor ya registrado.
Precondición	El profesor debe existir en la BBDD.
Postcondición	Datos modificados correctamente.
Efectos lat.	-

Función	**Borrar Profesor *(PRF04)***
Prioridad	Alta
Estabilidad	Media
Descripción	Se da de baja (lógica) un profesor del sistema.
Entrada	Id_profesor
Salida	Mensaje de éxito e identificador del profesor.
Origen	GUI
Destino	GUI
Necesita	ID_turno como clave foránea
Acción	Modifica los datos de un profesor ya registrado.
Precondición	El profesor debe existir en la BBDD.
Postcondición	Profesor dado de baja.
Efectos lat.	-

Función	**Listar Profesores *(PRF05)***
Prioridad	Alta
Estabilidad	Alta
Descripción	Muestra los datos de todos los profesores.
Entrada	-
Salida	Listado de los profesores.
Origen	GUI

Destino	GUI
Necesita	-
Acción	Lista por pantalla los datos de todos los profesores.
Precondición	Existen profesores registrados en el sistema.
Postcondición	-
Efectos lat.	No muestra nada al estar la lista vacía o error en la BBDD.

Función	**Añadir materia a Profesor *(PRF07)***
Prioridad	Alta
Estabilidad	Media
Descripción	Añade a un profesor una materia.
Entrada	ID_Materia
Salida	Mensaje de éxito.
Origen	GUI
Destino	GUI
Necesita	Listado de materias.
Acción	Añade a un profesor una materia.
Precondición	ID_Materia válido. La materia existe en la Base de Datos y está activo.
Postcondición	Profesor actualizado con la materia añadida.
Efectos lat.	-

Función	Quitar materia a Profesor *(MTR08)*
Prioridad	Alta
Estabilidad	Media
Descripción	Quita una materia asignada aun profesor.
Entrada	ID_Materia
Salida	Mensaje de éxito.
Origen	GUI
Destino	GUI
Necesita	Listado de las materias del pofesor
Acción	Quita una materia asignada aun profesor
Precondición	ID_Materia válido. La materia está asignada al profesor.
Postcondición	Profesor actualizado sin la materia.
Efectos lat.	-

- **Módulo Materia**

Función	Registrar materia (MAT01)
Prioridad	Alta
Estabilidad	Alta
Descripción	Añade una materia al sistema.
Entrada	ID_profesor, nombre, descripción.
Salida	Confirmación de la operación e identificador.

Origen	GUI
Destino	GUI
Necesita	Datos nueva materia, listado de plazas.
Acción	Añade una nueva materia a la BBDD.
Precondición	Se han rellenado todos los datos y no exista una materia con el mismo nombre.
Postcondición	-
Efectos lat.	-

Función	**Mostrar materia (MAT02)**
Prioridad	Alta
Estabilidad	Alta
Descripción	Lista los datos de una materia por pantalla.
Entrada	ID_materia
Salida	Muestra los datos por pantalla.
Origen	GUI
Destino	GUI
Necesita	ID_materia
Acción	Muestra los datos de la materia por pantalla.
Precondición	Tiene que existir la materia.
Postcondición	-
Efectos lat.	-

Función	Modificar materia (MAT03)
Prioridad	Alta
Estabilidad	Alta
Descripción	Modifica a una materia existente.
Entrada	ID_materia, ID_profesor, nombre, descripción.
Salida	Confirmación de la operación.
Origen	GUI
Destino	GUI
Necesita	ID_materia, ID_profesor, nombre, descripción.
Acción	Modifica a una materia registrado en la BBDD.
Precondición	Tiene que existir la materia.
Postcondición	-
Efectos lat.	-

Función	Borrar materia (MAT04)
Prioridad	Alta
Estabilidad	Alta
Descripción	Elimina a una materia existente.
Entrada	ID_materia
Salida	Confirmación de la operación e identificador.
Origen	GUI

Destino	GUI
Necesita	ID_materia
Acción	Baja lógica de una materia en el sistema.
Precondición	Tiene que existir la materia.
Postcondición	-
Efectos lat.	-

Función	**Listar materias (MAT05)**
Prioridad	Alta
Estabilidad	Media
Descripción	Lista los datos de las materias por pantalla.
Entrada	-
Salida	Muestra los datos por pantalla.
Origen	GUI
Destino	GUI
Necesita	-
Acción	Muestra los datos de las materias por pantalla.
Precondición	Tiene que existir al menos una materia.
Postcondición	-
Efectos lat.	-

3.3. Requisitos de rendimiento

Aunque la aplicación será de escritorio se contempla la posibilidad de que tenga varios usuarios simultáneamente, usando para ello una política de concurrencia pesimista en el incremento DAO de la aplicación y una gestión de la persistencia con la API Java JPA para el segundo incremento de la aplicación.

La BBDD se modificará cada vez que se produzca una variación en la información almacenada en dicha BBDD.

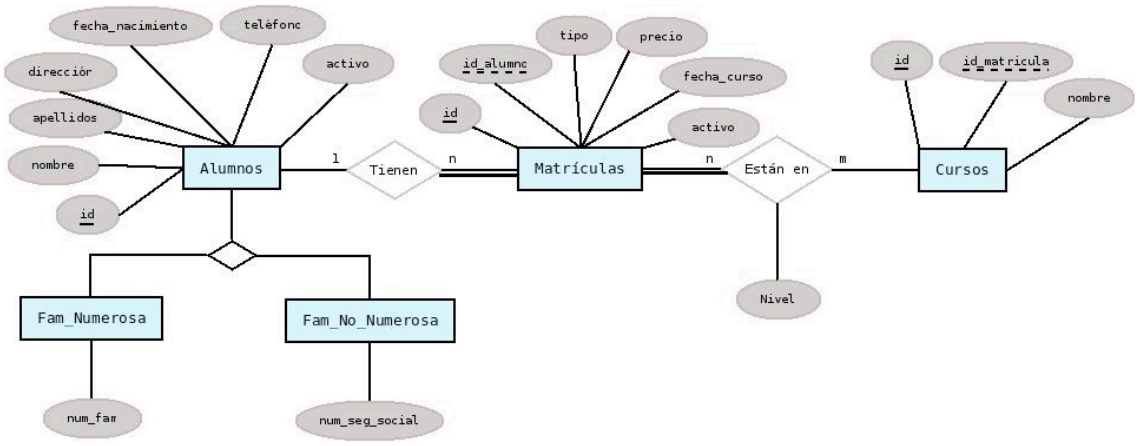
Los datos se almacenarán hasta el máximo que nos permita nuestra BBDD.

3.4. Requisitos lógicos de la BBDD.

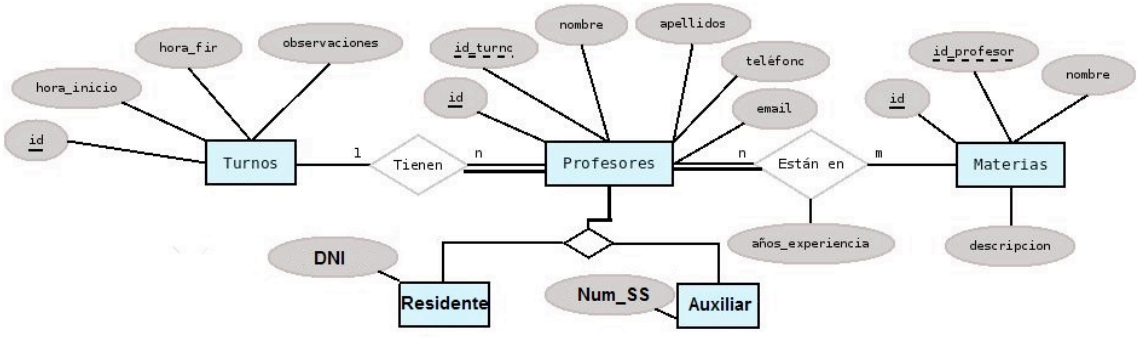
- **DIAGRAMA ENTIDAD RELACIÓN**

Este diagrama representa el funcionamiento de la base de dato relacional del sistema pudiéndose diferencias los distintos incrementos de la aplicación y sus módulos.

Incremento DAO



Incremento JPA



3.5. Restricciones de diseño.

Restricciones I

- Es obligatorio aplicar técnicas de IS.
- La aplicación no podrá ser ni una biblioteca, ni un videoclub.

Restricciones II

- La primera entrega del proyecto (SRS) será el día 30/10/2015. Consistirá en la especificación de requisitos software de la aplicación según formato IEEE 830-1998.
- La segunda entrega (parte DAO) del proyecto será el día 10/12/2015. Consistirá en la primera versión de la aplicación, incluyendo su modelo UML 2.x.
- La tercera entrega del proyecto (parte JPA) será el día 28/01/2016. Consistirá en la segunda versión de la aplicación, incluyendo su modelo UML 2.x.

Restricciones III

- El modelo UML 2.x de la aplicación, será en formato IBM Rational Software Architect 8.0.3.
- El lenguaje de implementación debe ser Java.
- La aplicación puede ser de escritorio o web.
- Se utilizará el sistema de control de versiones (SCV) de la Facultad de Informática para gestionar la documentación, el modelo y el código.
- La persistencia de los datos debe hacerse en formato relacional (se recomienda utilizar MySQL 5.x).
- La arquitectura de la aplicación será multicapa.
- Deben aplicarse los siguientes patrones obligatoriamente en la primera versión de la aplicación:
  - o Service to worker
  - o Transfer object
  - o Data Access Object
  - o Alguna query tal y como se vieron en clase.
- Deben aplicarse los siguientes patrones obligatoriamente en la segunda versión de la aplicación:
  - o Business Object
  - o Domain Store (implementado con JPA 2.x. Se recomienda usar EclipseLink 2.x).
- Así, la primera versión persistirá los datos directamente en una base de datos relacional, utilizando objetos transferencia en negocio. La segunda versión, modular, persistirá los datos utilizando JPA, utilizando objetos del negocio en negocio.
- Ambas versiones gestionarán transacciones y concurrencia.

Restricciones IV

- El proyecto debe realizarse en equipo.
- El número de miembros de cada equipo debe ser exactamente seis.
- En casos excepcionales, y previa consulta con el profesor de esta asignatura antes del día 30/10/2015, se podría permitir la existencia de algún equipo de otro tamaño.
- En caso de existir equipos de alumnos de tamaño distinto de seis, el profesor se reserva el derecho de modificar la composición de estos equipos.
- Lo antes posible, los equipos deberán enviar un e-mail al profesor (anavarro@fdi.ucm.es) con el nombre del proyecto, los nombres de los integrantes del equipo, y la dirección de correo electrónico del responsable del equipo para los repositorios del SCV de la facultad.
- Cada entrega es prerrequisito de la siguiente.
- No se admiten entregas después de acabar la sesión de la asignatura correspondiente al día de cada entrega.

Restricciones V

- El día de la entrega segunda y tercera se procederá a una ejecución del proyecto para comprobar la implementación de los requisitos.
- Todas las entregas deben hacerse en un CD-ROM etiquetado con el nombre del proyecto.
- En las entregas segunda y tercera debe incluirse una memoria.
- En las entregas segunda y tercera deben incluirse los archivos correspondientes al diseño en formato IBM Rational Software Architect 8.0.3.
- En las entregas segunda y tercera deben incluirse los archivos de cada aplicación, tanto los archivos fuente como los compilados (que en particular deben se ejecutables).
- La existencia de cualquier tipo de virus en cualquier soporte informático entregado al profesor invalidará la entrega.
- A pesar de no exigir el plan del proyecto, se exhorta y anima a los alumnos a su realización.
- Los alumnos son los únicos responsables del contenido almacenado en el SCV. La presencia de contenido inadecuado en el SCV conllevará su borrado y la invalidación del proyecto.
- Los repositorios de la facultad serán BORRADOS el día 01/10/2016.

Nota: El profesor de la asignatura se reserva la posibilidad de modificar este proyecto con el fin de poder corregir cualquier error que pudiera existir en su enunciado.

3.6. Atributos del sistema software.

Se diferencia entre alumnos con familia numerosa y no numerosa; y de turnos de mañana o tarde.