Lucía Colmenarejo Pérez
Miguel García Moya

lucia.colmenarejo@estudiante.uam.es
miguel.garciamoya@estudiante.uam.es

DISEÑO (ORIENTADO A OBJETOS)

Aplicación: PAD-E

(Plataforma de Aprendizaje Didáctico y Educativo)

FECHA: 08-03-2017

1.DIAGRAMA DE CLASES

En el diagrama de clases lo que se busca es representar cómo va a ser la estructura de nuestra aplicación, mostrando sus clases, sus atributos, operaciones(o métodos) y las relaciones entre los objetos. Como se puede apreciar en la imagen inferior hemos requerido de un elevado número de clases (o instancias) que pasaremos a ir describiendo una por una.

Uno de los puntos desde el cual podríamos iniciar la explicación de nuestro diagrama de clases es desde la clase de *Usuario.* Esta clase está formada por dos atributos: un string *Usuario* y un string *Contraseña*. Si nos fijamos esta clase va a ser una clase abstracta, de la cual van a heredar sus métodos tanto *Profesor* como *Alumno*, que son sus dos clases "hijas", además de tener los atributos que al ser comunes a ambas clases los podemos incluir en la clase *Usuario*. En la clase de *Usuario* sólo podemos registrarnos o salir de nuestra cuenta(*hacerLogin* y *hacerLogout* respectivamente). Una vez en la clase de *Profesor* , podemos ver que no tiene ningún atributo más que aquellos que le proporciona la clase de *Usuario*. En cambio, es una de las clases que más métodos contiene, como: *aceptarPeticionAdmision* (recibe una petición como parámetro y procede a su aceptación), *denegarPeticionAdmision* (recibe una petición de admisión y la rechaza), *accederAsignatura* (el profesor tendrá una forma distinta de acceder a asignatura que el alumno) y *accederTema* (misma situación que el el caso de acceder asignatura). A continuación, al lado derecho de la clase de *Profesor,* se encuentra la clase *Sistema*, la cual va a estar formada por un profesor (ya que en nuestra aplicación los profesores van a tener el mismo usuario y contraseña todos, por lo que los consideramos como un único usuario *Admin*). En esta clase vemos como no hay ningún atributo y simplemente aparecen los métodos que necesitan del sistema para su ejecución. Estos métodos son: *crearAsignatura* (el sistema es el que se encarga de crear un "espacio" para dicha asignatura) y *mandarNotificacion* (el sistema manda una notificación cuando el profesor acepte o deniegue una petición de admisión o cuando un ejercicio ha iniciado, es decir, ha llegado a su fecha de inicio). En el diagrama vemos que el sistema está compuesto por un número n de asignaturas, pero antes de ver esta clase vamos a proceder a la explicación de la clase de *Estudiante.*

La clase de *Estudiante* tiene como atributos, además de los explicados anteriormente en la clase de *Usuario*, el nombre, los apellidos, el número de identificación y el correo. Estudiante va a ser una de las clases que va a ser una pieza clave debido a los métodos que contiene. Esta clase tiene ocho métodos: *responderEjercicio* (se le pasa un ejercicio el cual va a tener que contestar el estudiante), *verCalificacion* (para que el estudiante pueda acceder a la calificación de la asignatura en la cual se encuentra), *enviarPeticionAdmision* (dicha función lo que va a permitir es que el estudiante cuando esta en una asignatura en la cual no está matriculado, pueda enviarle al profesor una solicitud de admisión en la asignatura actual), *accederAsignatura* (la utilizaremos siempre que el alumno quiera acceder a cualquiera de las asignaturas), *verApuntes* (le permite al estudiante acceder a los archivos que haya en la aplicación), *verRespuestas*(concede al estudiante el poder acceder a las respuestas del ejercicio que ya ha respondido previamente) y, por último, *accederTema* (función similar a la de accederAsignatura, pero para el caso en el que quiera acceder a los distintos temas que constituyen la asignatura). También cabe destacar que vamos a tener una

clase llamada *CalificacionAsignatura* que nos va a devolver la nota de cada una de las asignaturas(clase relacionada con *Asignatura* también).

Un estudiante tiene n asignaturas y una asignatura tiene n estudiantes (a la hora de la implementación lo adecuaremos a los datos especificados por el cliente). En la clase de *Asignatura,* podemos ver que tenemos dos atributos: nombre y número de estudiantes que cursan dicha asignatura. Por otra parte, podemos ver cuáles son los métodos que va a haber en esta clase: *darDeAlta* y *darDeBaja* (las cuales van a dar de alta o de baja a un estudiante de manera limitada hasta una fecha límite), *añadirTema* (para añadir un tema a la asignatura), *añadirEjercicio* (para introducir un ejercicio en la asignatura, a dicha función le pasamos las preguntas que va a llevar el ejercicio a añadir), *eliminarTema* (para destruir un tema), *modificarContenido* (le pasamos como argumento un tipo contenido, que posteriormente veremos que se va a tratar de los temas, los apuntes y los ejercicios, y permite modificarlo) y *mostrarCalificacionesAsigntura* (mostrará las calificaciones obtenidas en esa asignatura de cada uno de los ejercicios). Podemos observar que estudiante o va a tener acceso a todas las funciones de esta clase, ya que salvo la última función descrita en la anterior enumeración, el resto son requisitos que ha de cumplir el usuario profesor y no el estudiante. Cabe destacar una pequeña diferencia en la última función y es que no se va a mostrar igual las calificaciones para el profesor que para el estudiante (para el primero aparecerá un listado con todos los alumnos y sus calificaciones en cada ejercicio; mientras que para el estudiante aparecerá un listado con todos sus ejercicios y sus notas con peso y porcentajes).

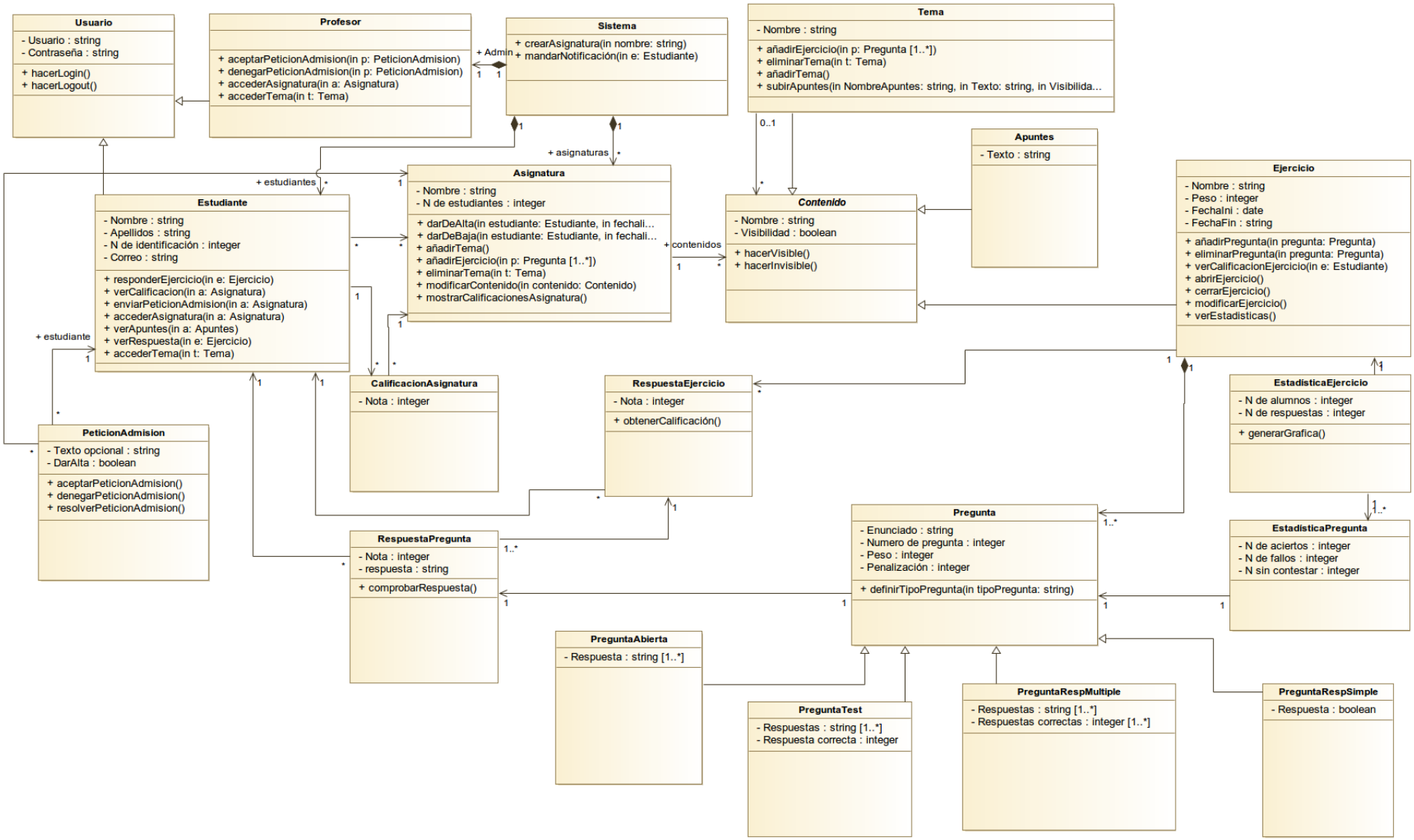
A continuación tenemos una asociación que nos informa de que la asignatura tiene n contenidos. Hemos definido una clase *Contenido* que va a ser una clase abstracta de la cual van a heredar atributos y métodos las siguientes tres clases: *Tema, Ejercicios* y *Apuntes*. En contenido tenemos un nombre y un boolean que nos informa de si ese contenido será o no será visible para el estudiante. En relación con este último atributo, tenemos dos métodos en *Contenido* que son *hacerVisible y hacerInvisible*. *Tema* es una de las clases que heredan de contenido, aparte tiene un nombre como atributo (nos referimos al nombre que puede tener la sección del tema). Dentro de esta clase tenemos las funciones de *añadirEjercicio, eliminarTema, añadirTema y subirApuntes.* Los dos métodos de a*ñadirEjercicio* y *eliminarTema* están en esta clase porque hay que percatarse de que un tema puede estar formado por subtemas que a su vez tengan su propio contenido, apuntes y ejercicios (de ahí la asociación entre *Tema* y *Contenido* de que un tema puede tener n contenidos). De la clase de *Apuntes* no hay nada relevante que destacar ya que únicamente tendrá un atributo que será su texto. En cambio, la clase de *Ejercicio* es una de las clases más complejas.

En la clase de *Ejercicio* tenemos como atributos su nombre, el peso que tendrá sobre la asignatura, su fecha de inicio y su fecha de fin (en estas dos hemos optado por elegir un tipo de dato tipo *Date* ya que nos va a comprobar implícitamente si una fecha va a ser válida o no). Los métodos de esta clase son: *añadirPregunta, eliminarPregunta, verCalificacionEjercicio, abrirEjercicio, cerrarEjercicio, modificarEjercicio* y *verEstadisticas*. Las funciones a las cuales tendrá acceso el profesor son *añadirPregunta, eliminarPregunta, verCalificacionEjercicio* y *verEstadisticas*; mientras que las funciones *abrirEjercicio* y *cerrarEjercicio* son llevadas a cabo por el sistema automáticamente al alcanzarse la fecha de inicio y la fecha de fin, respectivamente. Un ejercicio va a estar formado de preguntas, por lo que necesitaremos una clase *Pregunta*. Dado que el cliente especificó que quería tener en los ejercicios hasta cuatro tipos de preguntas, hemos

implementado cuatro clases que sean cada uno de los cuatro tipos de preguntas y que hereden de la clase *Pregunta*. Una pregunta va a tener un enunciado, un número de pregunta, un peso y una penalización (opcional) común a todos los tipos de preguntas. Cada una de los tipos de pregunta, definida por el método *definirTipodePregunta* (pregunta abierta, pregunta de respuesta múltiple, pregunta tipo test y preguntas de respuesta simple) van a tener como atributos un campo que sea la respuesta correcta y además aquellas que sean preguntas en las que haya que marcar una o varias opciones(pregunta tipo test y pregunta de respuesta múltiple) van a tener un dato tipo integer que nos diga el número de la respuesta correcta. Cada pregunta va a tener una respuesta dada por el estudiante, respuesta que se va a guardar y a recoger en la clase de *RespuestaPregunta*. Esta clase va a tener una función que va a consistir en comprobar que la solución dada por el estudiante es la correcta o no (este método se va a llamar *comprobarRespuesta*) . El conjunto de las notas obtenidas en esta clase debido a la comprobación de las respuestas, va a servir para que en la clase *RespuestaEjercicio* se obtenga la nota final del ejercicio.

Relacionado con la clase de *Ejercicio* también tenemos la clase de *EstadísticaEjercicio* que nos va a generar una gráfica (mediante el método *generarGráfica*) a partir del número de alumnos y el número de respuestas de cada ejercicio. Para hacer esta gráfica vamos a necesitar una clase que nos obtenga los datos necesarios de cada una de las preguntas del ejercicio, esta es la clase *EstádisticaPregunta*. Esta clase nos obtendrá el número de preguntas acertadas, erróneas y sin contestar por parte de los alumnos.

Por último, hemos de mencionar la clase de *PeticionAdmision*, la cual nos va a servir para determinar si un estudiante está dado o no de alta en una asignatura. Esta clase se compone de un string que es un texto opcional por si el alumno quiere enviar algún dato relevante al profesor; y un boolean que nos informará de la situación del estudiante es es asignatura. De esta forma, la clase tendrá tres métodos o funciones: *aceptarPeticionAdmision*, *denegarPeticionAdmision* y *resolverPeticionAdmision*. Esta instancia estará relacionada con *Estudiante* ya que un estudiante puede tener n peticiones de admisión; y de la misma manera una asignatura tendrá n peticiones de admisión (para que el profesor pueda decidir en un futuro a quién admitir o no en dicha asignatura):

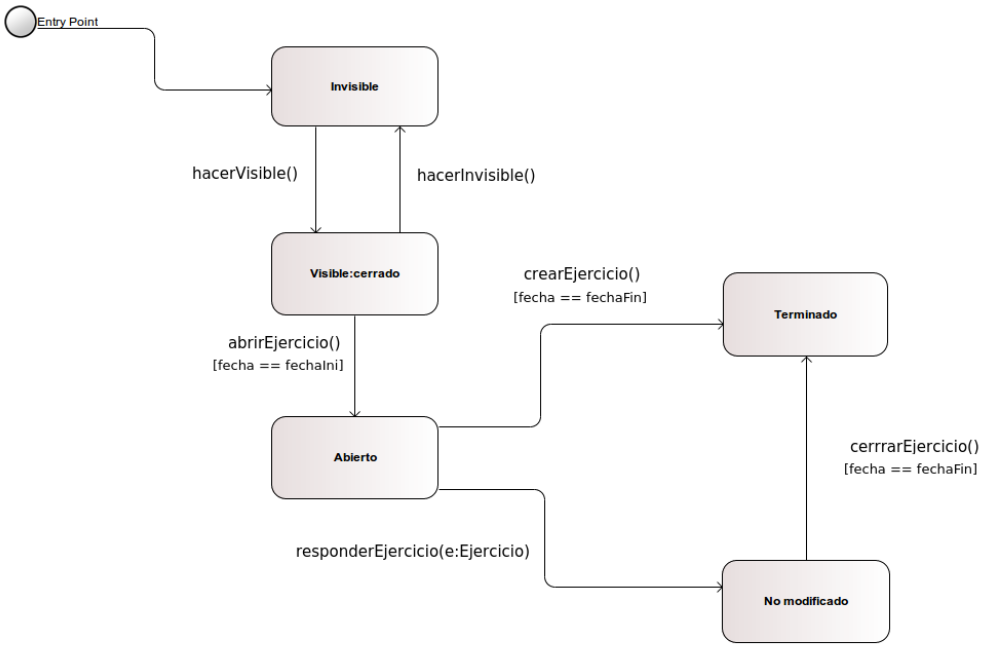

Usuario
- Usuario : string
- Contraseña : string
+ hacerLogin()
+ hacerLogout()
Profesor
+ aceptarPeticionAdmision(in p: PeticionAdmision)
+ denegarPeticionAdmision(in p: PeticionAdmision)
+ accederAsignatura(in a: Asignatura)
+ accederTema(in t: Tema)
+ Admin
Sistema
+ crearAsignatura(in nombre: string)
+ mandarNotificación(in e: Estudiante)
Tema
- Nombre : string
+ añadirEjercicio(in p: Pregunta [1..*])
+ eliminarTema(in t: Tema)
+ añadirTema()
+ subirApuntes(in NombreApuntes: string, in Texto: string, in Visibilida...
Apuntes
- Texto : string
+ asignaturas
+ estudiantes
Asignatura
- Nombre : string
- N de estudiantes : integer
+ darDeAlta(in estudiante: Estudiante, in fechali...
+ darDeBaja(in estudiante: Estudiante, in fechali...
+ añadirTema()
+ añadirEjercicio(in p: Pregunta [1..*])
+ eliminarTema(in t: Tema)
+ modificarContenido(in contenido: Contenido)
+ mostrarCalificacionesAsignatura()
+ contenidos
Contenido
- Nombre : string
- Visibilidad : boolean
+ hacerVisible()
+ hacerInvisible()
Ejercicio
- Nombre : string
- Peso : integer
- FechaIni : date
- FechaFin : string
+ añadirPregunta(in pregunta: Pregunta)
+ eliminarPregunta(in pregunta: Pregunta)
+ verCalificacionEjercicio(in e: Estudiante)
+ abrirEjercicio()
+ cerrarEjercicio()
+ modificarEjercicio()
+ verEstadisticas()
Estudiante
- Nombre : string
- Apellidos : string
- N de identificación : integer
- Correo : string
+ responderEjercicio(in e: Ejercicio)
+ verCalificacion(in a: Asignatura)
+ enviarPeticionAdmision(in a: Asignatura)
+ accederAsignatura(in a: Asignatura)
+ verApuntes(in a: Apuntes)
+ verRespuesta(in e: Ejercicio)
+ accederTema(in t: Tema)
+ estudiante
CalificacionAsignatura
- Nota : integer
RespuestaEjercicio
- Nota : integer
+ obtenerCalificación()
EstadisticaEjercicio
- N de alumnos : integer
- N de respuestas : integer
+ generarGrafica()
PeticionAdmision
- Texto opcional : string
- DarAlta : boolean
+ aceptarPeticionAdmision()
+ denegarPeticionAdmision()
+ resolverPeticionAdmision()
RespuestaPregunta
- Nota : integer
- respuesta : string
+ comprobarRespuesta()
Pregunta
- Enunciado : string
- Numero de pregunta : integer
- Peso : integer
- Penalización : integer
+ definirTipoPregunta(in tipoPregunta: string)
EstadisticaPregunta
- N de aciertos : integer
- N de fallos : integer
- N sin contestar : integer
PreguntaAbierta
- Respuesta : string [1..*]
PreguntaTest
- Respuestas : string [1..*]
- Respuesta correcta : integer
PreguntaRespMultiple
- Respuestas : string [1..*]
- Respuestas correctas : integer [1..*]
PreguntaRespSimple
- Respuesta : boolean

2.DIAGRAMAS DE TRANSICIÓN DE ESTADOS

Estos diagramas lo que se encargan es de darnos información de cómo va a evolucionar el estado de un objeto ante las invocaciones de los métodos.

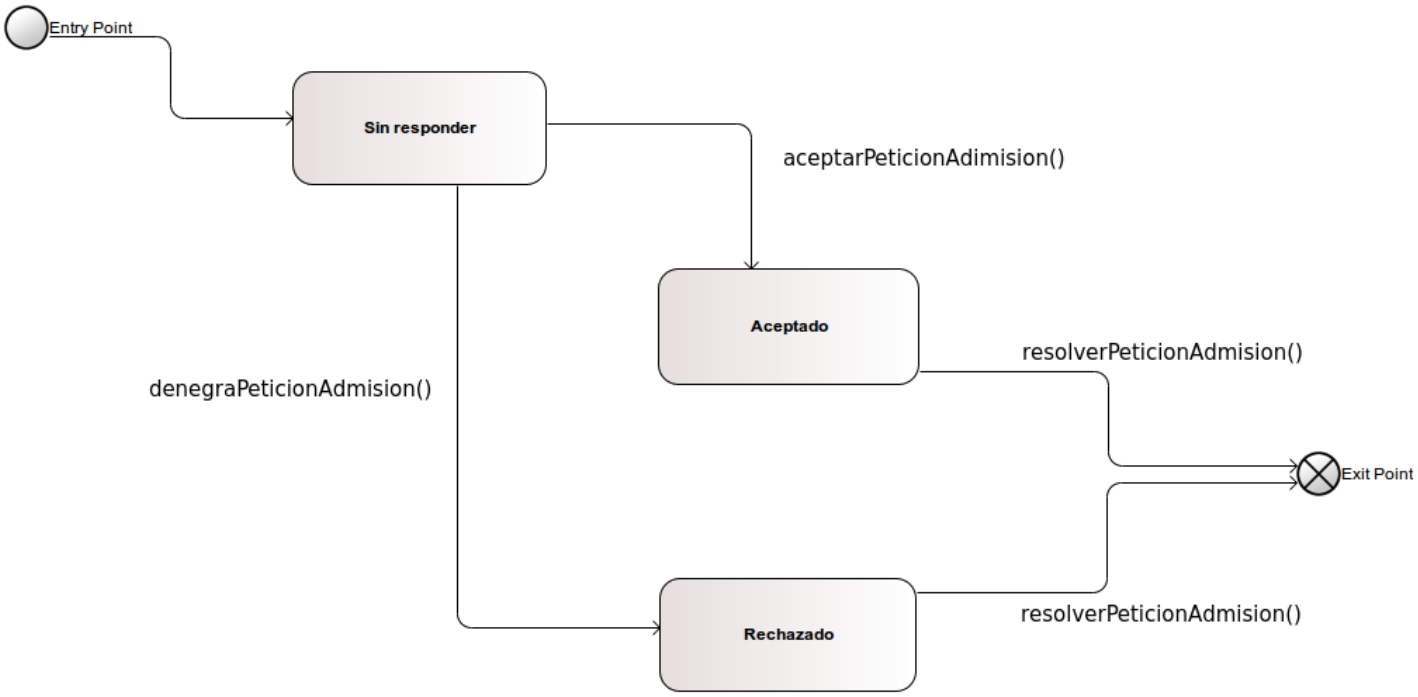
Crear ejercicio:

El primer ejemplo que hemos elegido de nuestro diagrama es la transición de estados que va a sufrir un ejercicio cuando es creado. Como vemos nuestro ejercicio una vez que lo crea el profesor este va a estar en un estado de no visible (*invisible*). El ejercicio cambiará de estado cuando sobre él se aplique la función de *hacerVisible* definida en la clase *Contenido* (función que puede heredar la clase de *Ejercicio*). Ahora nuestro ejercicio está en un estado de *visible* (cabe destacar que podemos volver al estado de *invisible* utilizando la función de *hacerInvisible*) y pasará a estar en el estado de *abierto* si se ejecuta la función de *abrirEjercicio* cuando la fecha es la misma que la fecha de inicio del ejercicio. Del estado de *abierto* puede pasar a dos estados: *no modificado* (si efectuamos la función de *responderEjercicio*) o *terminado* (si ejecutamos el método de *cerrarEjercicio* cuando la fecha sea igual a la fecha de fin del ejercicio). También podemos pasar de *no modificado* a *terminado* cuando se realice la función de *cerrarEjercicio* explicada anteriormente.



Petición de admisión:

En este segundo ejemplo hemos optado por realizar el diagrama de estados de una petición de admisión. El estado inicial de la petición es el de *sin responder*, dado que el alumno ha enviado la solicitud al profesor y del cual todavía no ha obtenido respuesta. Si el profesor realiza el método de *aceptarPeticionAdmision*, la petición pasará de estar en el estado de *sin resolver* al estado de *aceptada*. De manera similar pasa de *sin responder* a *rechazada*, pero empleando la función de *denegarPeticiónAdmisión*. Finalmente tanto el estado de aceptada como el de rechazada pueden llegar al estado final en el que sale la petición, si se ejecuta la función de *resolverPeticionAdmision*.

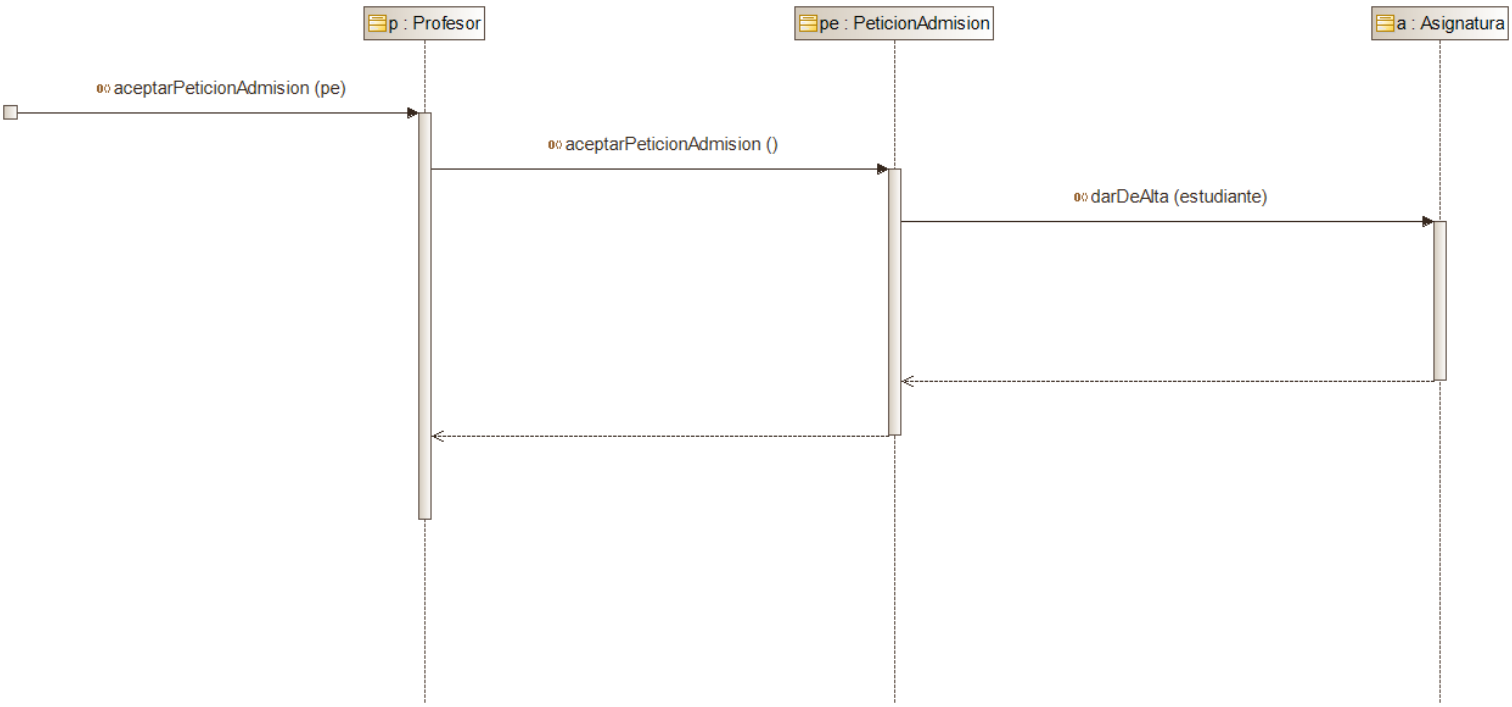


3.DIAGRAMAS DE SECUENCIA

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema. Cada uno de los objetos que definamos va a tener una línea de vida (representada mediante una caja en posición vertical) y el paso del tiempo se va a representar de arriba a abajo.

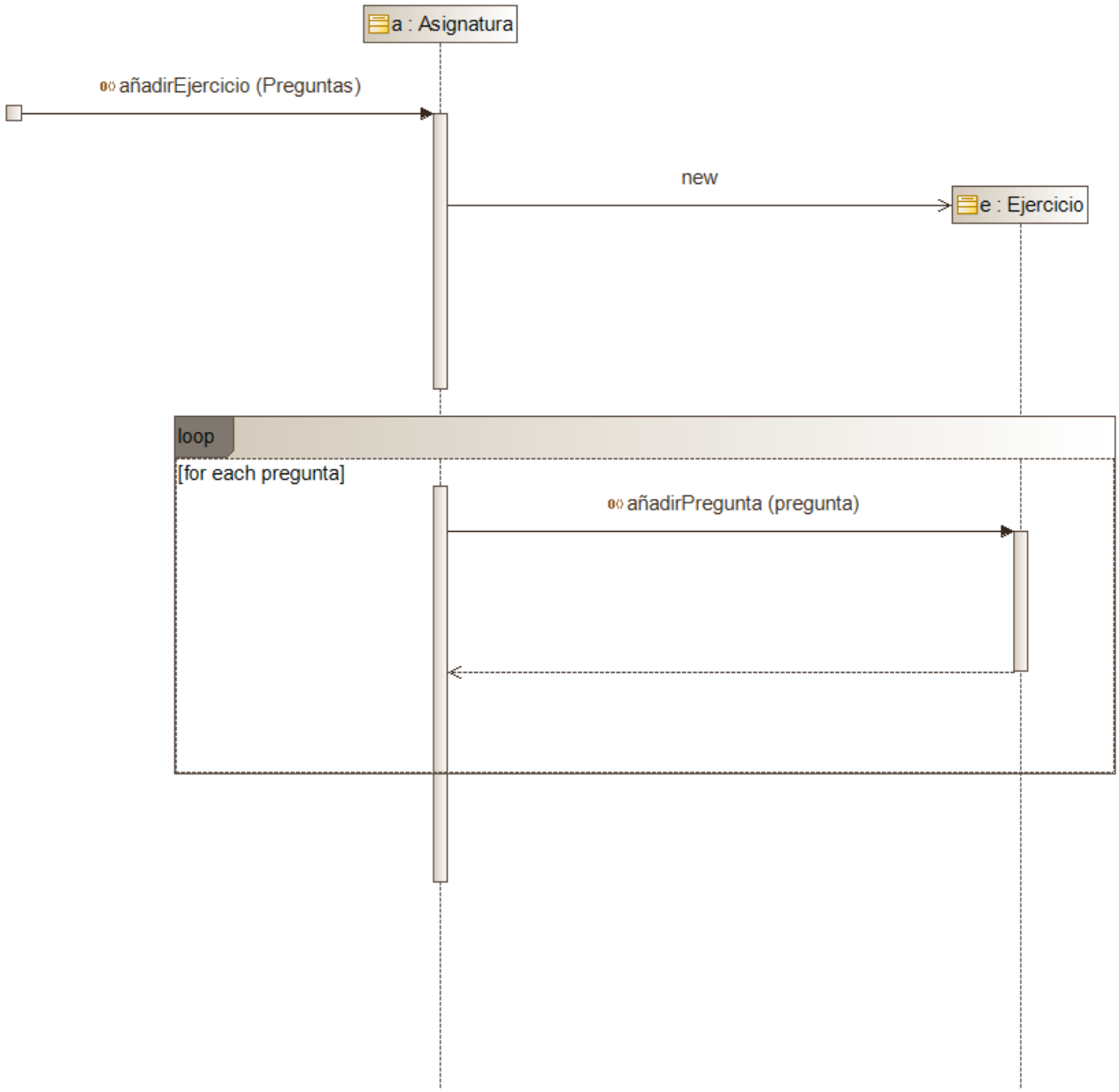
Aceptar petición de admisión:

En primer lugar vamos a realizar sobre el objeto de profesor la función de *aceptarPeticionAdmision*. Posteriormente sobren petición ejecutamos el método de *aceptarAdmisionPeticion*. La diferencia de una función a otra es que la primera está en la clase de *Profesor* y se le pasa la petición como argumento, y a segunda pertenece a la clase de *PeticionAdmision*. Por último sobre el objeto a (que es del tipo *Asignatura*, y corresponde a la asignatura de la petición de admisión), aplicamos la función de *darDeAlta* y le pasamos como parámetro el estudiante que nos había mandado en un primer momento la solicitud de admisión.



Añadir ejercicio:

Invocamos al método de *añadirEjercicio* en el objeto *a*, que es de tipo *Asignatura*. A este método le pasamos como argumento las preguntas que queremos introducir en el ejercicio a crear. Posteriormente, es necesario que creemos un ejercicio (*e*). Una vez creado el ejercicio, lo que vamos a hacer es un bucle for en el cual para cada pregunta que quiera añadir el profesor, se va a llamar al método de *añadirPregunta* (pasándole como parámetro la pregunta que se quiere añadir al ejercicio) y así tantas veces como el profesor quiera.



4.MATRIZ DE TRAZABILIDAD:

Como podemos comprobar todos los requisitos(columna de la izquierda, que es igual en todas las hojas) tienen su correspondiente representación en el dominio de la solución a implementar.

	hacerLogin()	aceptarPeticionAdmision	denegarPeticionAdimision	accederAsignatura (como profesor)	AccederTema (cpmp prof)	crearAsignatura	mandarNotificacion	añadirEjercicio	añadirTema
Hacer login (alum y prof)	X								
Entrar asignaturas				X					
Mirar apuntes				X	X				
Mirar ejercicios				X	X				
Ver respuestas				X					
Ver soluciones				X					
Revisar calificaciones									
Pedir admision									
Crear asignaturas				X		X			
Subir apuntes				X	X				
Crear ejercicios				X	X		X		
Definir tipo de ejercicio									
Decidir puntuacion en acierto									
Definir penalización en fallo									
Decidir peso									
Definir fecha de inicio							X		
Definir fecha de fin							X		

	hacerLogin()	aceptarPeticionAdmision	denegarPeticionAdimision	accederAsignatura (como profesor)	AccederTema (como prof)	crearAsignatura	mandarNotificacion	añadirEjercicio	añadirTema
Crear tema				X	X				X
Organizar contenido				X	X			X	
Dar de alta				X					
Dar de baja				X					
Aceptar petición de admisión		X		X					
Denegar petición de adimisión			X	X					
Expulsar temporalmente				X					
Readmitir temporalmente				X					
Ajustar visibilidad				X	X				
Ver calficación ejercicio				X	X				
Ver calificación total de asignatura				X					
Ver estadísticas				X	X				
Mandar notificación							X		
Modificar ejercicio				X	X				
Mandar aviso de cambio de fecha							X		
Hacer log									

	subirApuntes()	ResponderEjercicio() (incluye mirar el ejer)	verCalificacion()	enviarPeticionAdimision	accederAsignatura (como alumno)	accederTema (como alumno)	verApuntes	verRespuesta	darDeAlta
Hacer login (alum y prof)									
Entrar asignaturas					X				
Mirar apuntes					X	X	X		
Mirar ejercicios		X			X	X			
Ver respuestas					X			X	
Ver soluciones					X	X		X	
Revisar calificaciones			X						
Pedir admision				X	X				
Crear asignaturas									
Subir apuntes	X								
Crear ejercicios									
Definir tipo de ejercicio									
Decidir puntuacion en acierto									
Definir penalización en fallo									
Decidir peso									
Definir fecha de inicio									
Definir fecha de fin									

	subirApuntes()	ResponderEjercicio() (incluye mirar el ejer)	verCalificacion()	enviarPeticionAdimision	accederAsignatura (como alumno)	accederTema (como alumno)	verApuntes	verRespuesta	darDeAlta
Crear tema									
Organizar contenido									
Dar de alta									X
Dar de baja									
Aceptar petición de admisión									
Denegar petición de adimisión									
Expulsar temporalmente									
Readmitir temporalmente									X
Ajustar visibilidad									
Ver calficación ejercicio									
Ver calificación total de asignatura					X				
Ver estadísticas									
Mandar notificación									
Modificar ejercicio									
Mandar aviso de cambio de fecha									
Hacer log									

	darDeBaja()	añadirTema()	añadirEjercicio()	eliminarTema()	modificarContenido()	mostrarCalificacionesAsignatura()	hacerVisible()	hacerInvisible()	añadirPregunta()
Hacer login (alum y prof)									
Entrar asignaturas									
Mirar apuntes									
Mirar ejercicios									
Ver respuestas									
Ver soluciones									
Revisar calificaciones									
Pedir admision									
Crear asignaturas									
Subir apuntes									
Crear ejercicios			X						X
Definir tipo de ejercicio									X
Decidir puntuacion en acierto									X
Definir penalización en fallo									X
Decidir peso									X
Definir fecha de inicio									X
Definir fecha de fin									

	darDeBaja()	añadirTema()	añadirEjercicio()	eliminarTema()	modificarContenido()	mostrarCalificacionesAsignatura()	hacerVisible()	hacerInvisible()	añadirPregunta()
Crear tema		X							
Organizar contenido				X	X				
Dar de alta							X	X	
Dar de baja	X								
Aceptar petición de admisión									
Denegar petición de adimisión									
Expulsar temporalmente	X								
Readmitir temporalmente									
Ajustar visibilidad					X		X	X	
Ver calficación ejercicio									
Ver calificación total de asignatura						X			
Ver estadísticas									
Mandar notificación							X		
Modificar ejercicio									
Mandar aviso de cambio de fecha									
Hacer log									

	eliminarPregunta()	verCalificacionEjercicio()	abrirEjercicio()	cerrarEjercicio()	modificarEjercicio()	verEStadisticas()	aceptarPeticionAdmision()	denegarPeticionAdmision()	resolverPeticionAdmision()
Hacer login (alum y prof)									
Entrar asignaturas									
Mirar apuntes									
Mirar ejercicios									
Ver respuestas									
Ver soluciones									
Revisar calificaciones		X							
Pedir admision									
Crear asignaturas									
Subir apuntes									
Crear ejercicios	X								
Definir tipo de ejercicio									
Decidir puntuacion en acierto									
Definir penalización en fallo									
Decidir peso									
Definir fecha de inicio									
Definir fecha de fin									

	eliminarPregunta()	verCalificacionEjercicio()	abrirEjercicio()	cerrarEjercicio()	modificarEjercicio()	verEStadisticas()	aceptarPeticionAdmision()	denegarPeticionAdmision()	resolverPeticionAdmision()
Crear tema									
Organizar contenido									
Dar de alta									
Dar de baja									
Aceptar petición de admisión							X		
Denegar petición de adimisión								X	
Expulsar temporalmente									
Readmitir temporalmente									
Ajustar visibilidad									
Ver calficación ejercicio			X						
Ver calificación total de asignatura									
Ver estadísticas						X			
Mandar notificación									
Modificar ejercicio					X				
Mandar aviso de cambio de fecha									
Hacer log									

	obtenerCalificacion()	generarGrafica()	comprobarRespuesta()	definirTipoPregunta()	hacerLogout()
Hacer login (alum y prof)					
Entrar asignaturas					
Mirar apuntes					
Mirar ejercicios					
Ver respuestas					
Ver soluciones					
Revisar calificaciones	X				
Pedir admision					
Crear asignaturas					
Subir apuntes					
Crear ejercicios					
Definir tipo de ejercicio				X	
Decidir puntuacion en acierto					
Definir penalización en fallo					
Decidir peso					
Definir fecha de inicio					
Definir fecha de fin					

	obtenerCalificacion()	generarGrafica()	comprobarRespuesta()	definirTipoPregunta()	hacerLogout()
Crear tema					
Organizar contenido					
Dar de alta					
Dar de baja					
Aceptar petición de admisión					
Denegar petición de adimisión					
Expulsar temporalmente					
Readmitir temporalmente					
Ajustar visibilidad					
Ver calficación ejercicio	X				
Ver calificación total de asignatura					
Ver estadísticas		X			
Mandar notificación					
Modificar ejercicio					
Mandar aviso de cambio de fecha					
Hacer log					X