

Uno

Modelo de una universidad con MagicDraw, Papyrus y USE

1. Descripción del modelo

1.1 Clases

El modelo contiene las siguientes clases:

- Universidad: dicha clase contiene como atributo a *nombre (String)*.
- Titulacion: es una clase abstracta, esta contiene como atributo *nombre (String)*.
- Posgrado: es una subclase de *Titulacion*, es uno de los distintos tipos de la misma.
- Grado: al igual que *Posgrado*, es una subclase de *Titulacion*.
- Asignatura: dicha clase contiene como atributos *nombre (String)* y *creditos (Integer)*.
- Curso: esta contiene como atributo a *cursoAcademico (Integer)*.
- Persona: clase abstracta que nos servirá para definir los distintos roles.
- Profesor: es una de las subclases de *Persona*.
- Alumno: es una subclase de *Persona*.
- Titulo: representa la obtención de un título universitario.

1.2 Relaciones

El modelo contempla las siguiente relaciones:

- Oferta: relación de composición entre *Universidad* (1) y *Titulacion* (*). Describe todas las titulaciones que una universidad oferta.
- Otorgado: relación de asociación entre *Universidad* (1) y *Titulo* (*). Indica todos los títulos otorgados por una universidad.
- Obtiene: relación de asociación entre *Alumno* (1) y *Titulo* (*). Indica que alumno ha recibido cierto título.
- Contiene: relación de agregación entre *Titulacion* (1..*) y *Asignatura* (1..*). Describe las asignaturas que pertenece a una titulación.
- Pertenece: relación de asociación entre *Titulacion* (1) y *Titulo* (*). Indica a qué titulación pertenece cierto título.
- Obtenido: relación de asociación entre *Curso* (1) y *Titulación* (*). Indica en qué curso se obtuvo cierto título.
- Matricular: clase asociativa entre *Alumno* (*) y *Asignatura* (1..*). Contiene los atributos *calificación (Integer)* y *aprobado (boolean)*. Éste último es un atributo derivado que si la calificación es mayor o igual a 5 tiene valor true.
- Impartir: relación asociativa entre *Profesor* (1) y *Asignatura* (*). Asignaturas que imparte un profesor.
- Impartido: relación asociativa entre *Curso* (1) e *Impartir* (*). Curso en la que se imparte una asignatura por parte de un profesor.
- Matriculado: relación asociativa entre *Curso* (1) e *Matricular* (*). Curso en el que un alumno se matricula en una asignatura.
- MatriculadoTitulacion: relación asociativa entre *Alumno* (*) y *Titulacion* (1..*). Indica qué títulos tienen ciertos alumnos.
- Relaciones de herencia desde *Posgrado* y *Grado* hacia la clase abstracta *Titulacion*.

- Relaciones de herencia desde *Alumno* y *Profesor* hacia *Persona*. Se utiliza una división por roles para definir las distintas acciones que pueden ser tomadas por cada rol de persona.

1.3 Restricciones

El modelo cumple las siguientes restricciones:

- Clase *Matricular*:
 - *MatriculaSiNoAprobada*: Esta restricción comprueba si existe una matrícula cuya calificación esté aprobada, que para un mismo alumno en una misma asignatura todo curso en el que se haya matriculado es anterior o igual, porque debemos tener en cuenta que al obtener todas estas instancias también conseguiremos la misma instancia “self”, al que se ha aprobado. De esta manera conseguimos que el aprobado sea único y además sea el último curso en el que se ha matriculado.

$$\begin{aligned} &\text{"context Matricular inv MatriculaSiNoAprobada:} \\ &\text{self.aprobado implies Matricular.allInstances()->forAll(m |} \\ &\text{m.asignaturaMatricular = self.asignaturaMatricular and m.alumno = self.alumno} \\ &\text{implies m.curso.cursoAcademico <= self.curso.cursoAcademico)"} \end{aligned}$$
 - *CalificaciónCorrecta*: Comprobamos que la nota es o un valor número comprendido entre 0 y 10, ambos inclusive, o bien aún no está definido, porque aún no se ha evaluado.

$$\begin{aligned} &\text{"context Matricular inv CalificacionCorrecta:} \\ &\text{(self.calificacion >= 0 and self.calificacion <=10) or} \\ &\text{self.calificacion.isUndefined()"} \end{aligned}$$
- Clase *Alumno*:
 - *CreditosMinimosParaTitulo*: Comprueba que si un alumno tiene un título debe tener el mínimo número de créditos aprobados en asignaturas de dicha titulación, para ello comprobaremos que si existe una instancia de título donde el alumno sea igual a “self”, debemos comprobar primero si es una instancia de tipo Grado o Posgrado, esto se debe a que tienen distinto número de créditos mínimos. Una vez obtenido el tipo de titulación, conseguiremos todas las asignaturas donde nos hemos matriculado de esta titulación y posteriormente seleccionaremos solo las aprobadas, para luego sumar los créditos de las mismas y comprobar que el valor total sumado es mayor o igual que el mínimo, 60 para posgrado y 240 para grado.

$$\begin{aligned} &\text{"context Alumno inv CreditosMinimosParaTitulo:} \\ &\text{self.titulo->forAll(t | if(t.titulacion.ocIsKindOf(Grado)) then} \\ &\text{(self.asignaturaMatricular->select(a | a.titulacion->exists(aTit | aTit = t.titulacion)} \\ &\text{and a.matricular->exists(m | m.aprobado)).creditos->sum()>=240) else} \\ &\text{(self.asignaturaMatricular->select(a | a.titulacion->exists(aTit | aTit = t.titulacion)} \\ &\text{and a.matricular->exists(m | m.aprobado)).creditos->sum()>=60) endif)"} \end{aligned}$$
 - *TituloDeGradoParaMatricularseEnPosgrado*: Comprueba si para una instancia de alumno existe una titulación donde se ha matriculado que sea de

tipo posgrado, esto implicaría que existiese una instancia de título relacionada con el alumno cuya titulación asociada sea de tipo grado.

“context Alumno inv TituloDeGradoParaMatricularseEnPosgrado:
self.titulacion->exists(t | t.ocIsKindOf(Posgrado)) implies self.titulo->exists(t | t.titulacion.ocIsKindOf(Grado))”

- **AsignaturaPerteneceATitulo:** Dadas todas las asignaturas donde se ha matriculado un alumno comprobaremos que para cada una de ellas existirá una titulación que contenga dicha asignatura y en la que esté matriculado el alumno.

“context Alumno inv AsignaturaPerteneceATitulacion:
self.asignaturaMatricular->forAll(a | a.titulacion->exists(t | self.titulacion->exists(titMat | titMat.nombre = t.nombre)))”

- Clase *Persona*:
 - **ProfesorNoPuedeMatricularseEnAsignaturaQuelImparte:** Esta restricción comprueba que para todas las asignaturas que ha impartido un profesor una relación matricular para esa misma asignatura impartida en el mismo curso.

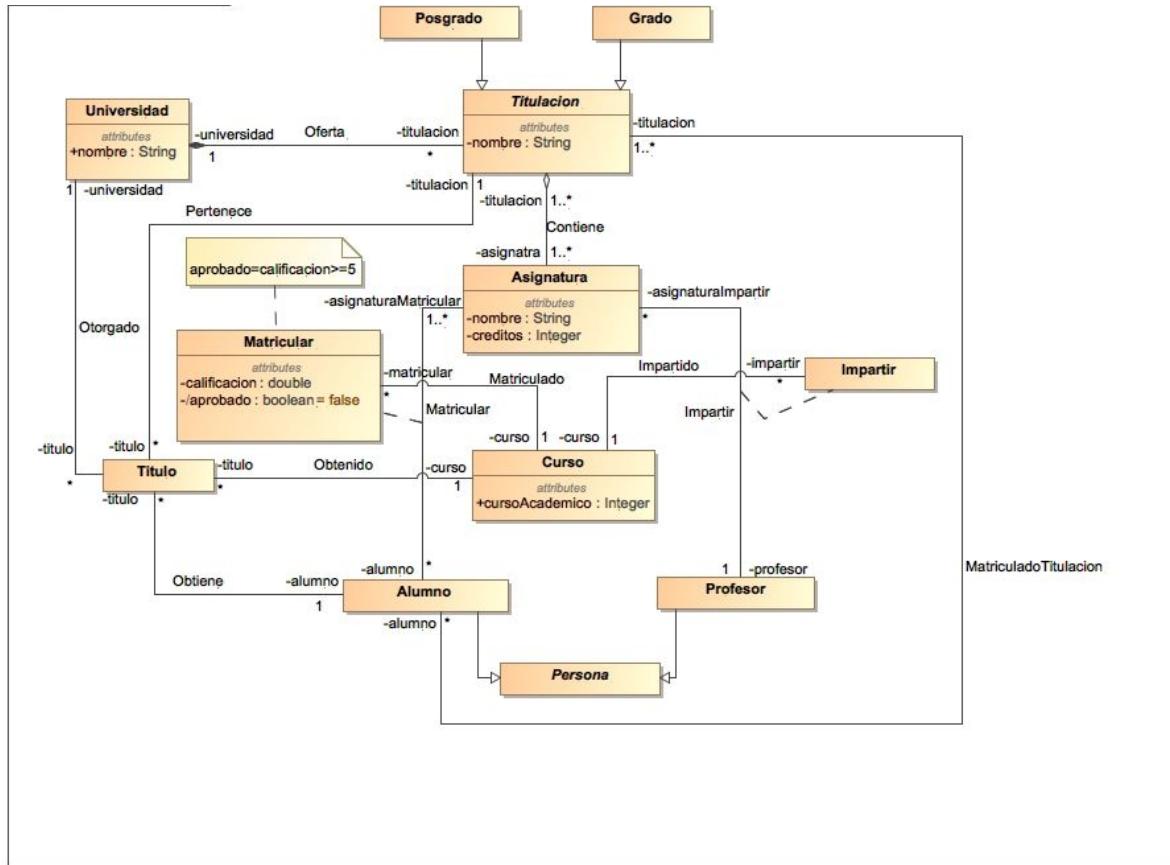
“context Persona inv ProfesorNoPuedeMatricularseEnAsignaturaQuelImparte:
self.ocIsKindOf(Profesor) and self.ocIsKindOf(Alumno) implies self.ocIAsType(Profesor).asignaturaImpartir->forAll(iMat | not self.ocIAsType(Alumno).asignaturaMatricular->exists(aMat | aMat = iMat and aMat.matricular.curso.cursoAcademico = iMat.matricular.curso.cursoAcademico))”

- Clase *Titulo*:
 - **CursoTitulo:** La restricción comprueba que el título del curso es el máximo curso en el que se ha matriculado de alguna asignatura en la titulación del mismo. El curso del título será igual a obtener todas las titulaciones, filtrar por aquella donde el nombre sea el mismo que la titulación del título conseguido y de entre todas las veces que nos hemos matriculado en él, obtener el máximo valor para curso.

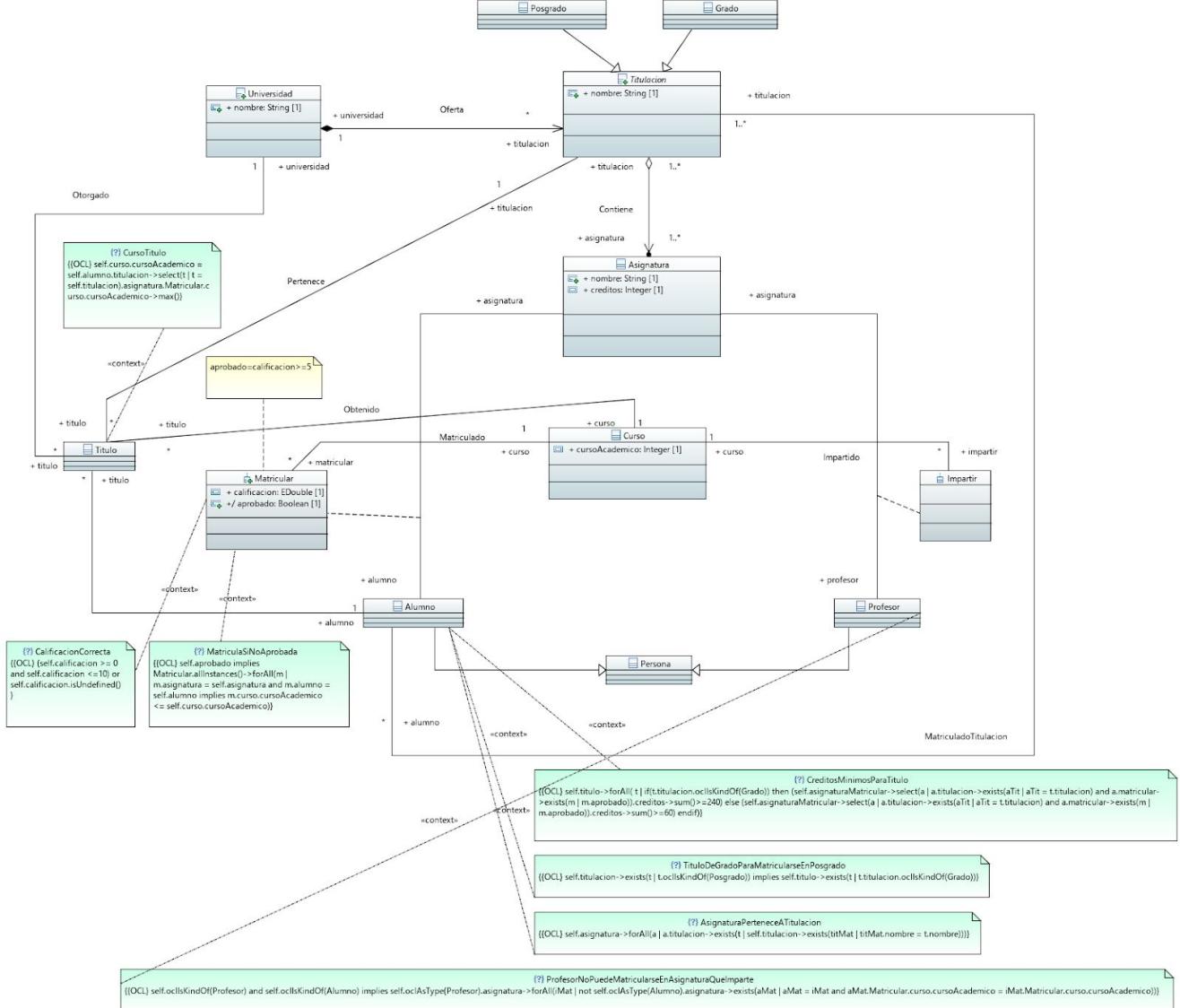
“context Titulo inv CursoTitulo:
self.curso.cursoAcademico = self.alumno.titulacion->select(t | t = self.titulacion).asignatura.matricular.curso.cursoAcademico->max”

2. Diagramas de clase

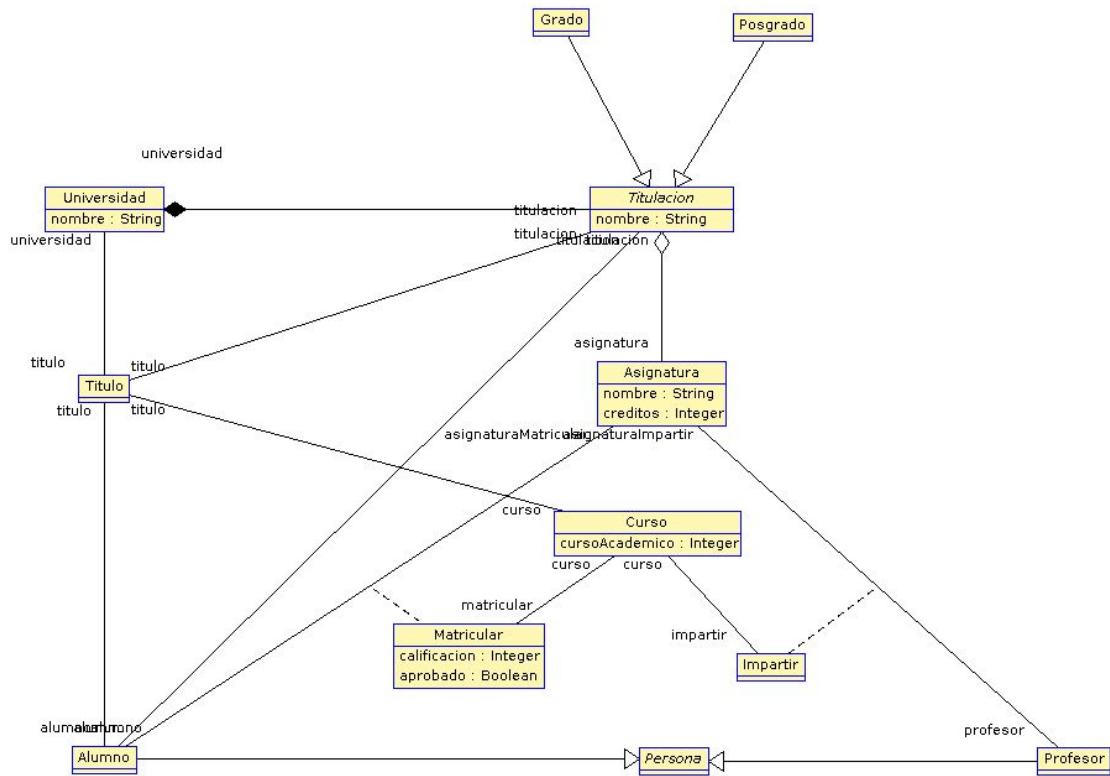
2.1 MagicDraw



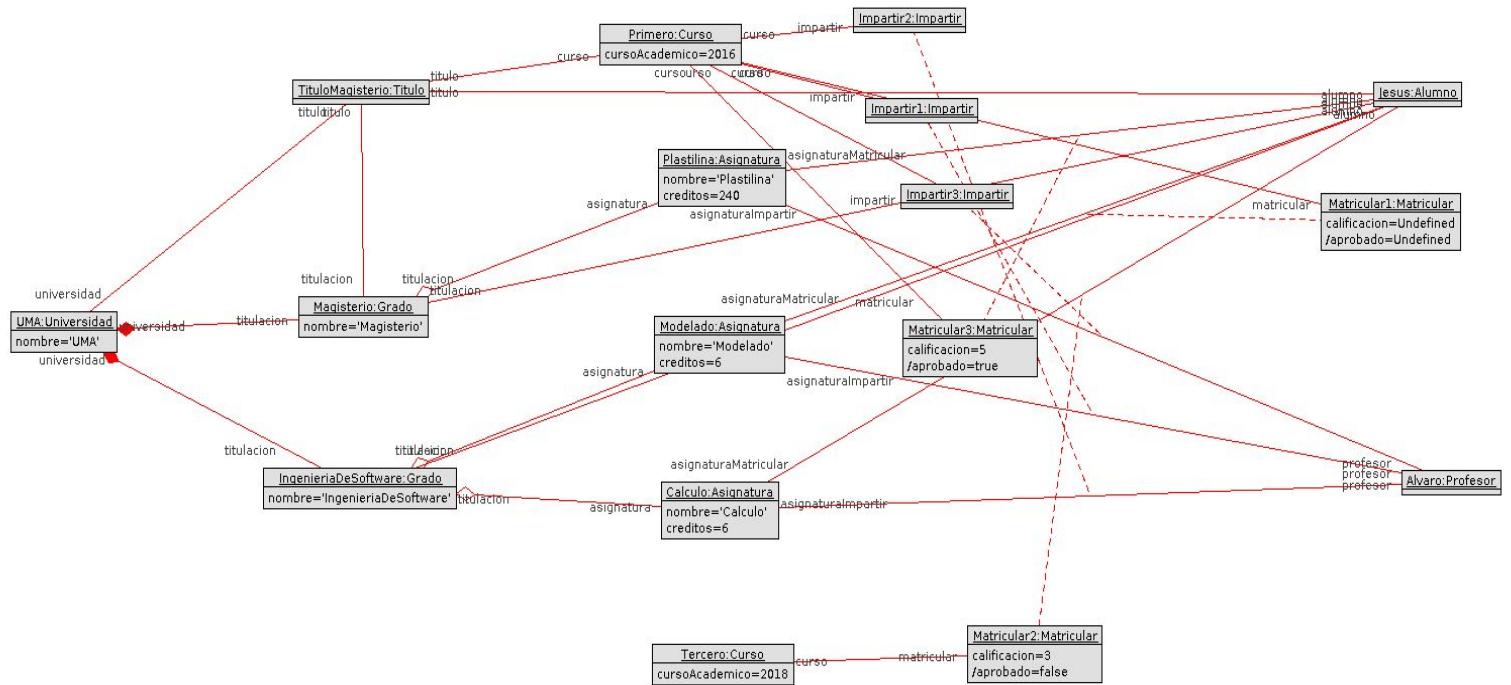
2.2 Papyrus



2.3 USE



2.4 Modelo de objetos en USE



Dos

Resumen del modelo UML

En este modelo UML de la Universidad hemos creado la clase Universidad que está relacionada mediante una agregación con la clase Titulación de la que heredan Grado y Postgrado, la cual se relaciona con la clase Asignatura mediante una agregación ya que una misma asignatura puede impartirse en distintas titulaciones y una asignatura sigue existiendo en el caso de que la titulación desaparezca, al igual que ocurre con las clases Universidad y Titulacion.

Tenemos la clase Persona de la que heredan Alumno y Profesor. Alumno está relacionado con asignatura mediante una relación de asociación llamada Matrícula, creando una clase de asociación que contiene el atributo nota, indicando que un alumno está matriculado en una asignatura. A su vez, Alumno está asociada con Titulación, creando una clase asociada llamada Título, dónde Título se refiere a la obtención del título. Profesor está relacionado con Asignatura mediante una relación de asociación, creando una clase de asociación llamada Imparte.

Esta clase Imparte está relacionada con la clase Curso, que contiene el atributo year, indicando el año de comienzo del curso, indicando que el profesor imparte una asignatura en un curso específico. La clase Curso está relacionada con la clase Título indicando en qué año se ha obtenido el título, así como con la clase Matrícula que informa a qué curso pertenece esa matrícula.

Restricciones

- **Un alumno solo puede aprobar una asignatura una vez, no pudiendo matricularse de asignaturas que ya ha aprobado en cursos anteriores.**
context Alumno inv NoMasUnaVezMatriculaAsignatura:
self.matricula->forAll(m:Matricula | (m.nota <> Undefined and m.nota >=5)
implies (self.matricula->excluding(m))->forAll(m2: Matricula | m.asignaturaAlumno =
m2.asignaturaAlumno
implies m2.cursoMatricula.year < m.cursoMatricula.year))

- **Cada curso los profesores evalúan a los alumnos matriculados en las asignaturas que imparten.**

context Profesor inv EvaluarAsignaturaQueImparte:
 self.asignaturaProfesor -> forAll(a:Asignatura | a.matricula.cursoMatricula.year ->
 includesAll(self.imparte.cursoImparte.year))

- **La nota será entre 0 y 10.**

context Matricula inv NotaEntre0y10:
 self.nota >= 0 and self.nota <= 10

- **Los profesores no pueden matricularse en aquellas asignaturas que imparten en ese curso, aunque sí en otras.**

context Matricula inv ProfesorNoAlumno:
 self.alumnoAsignatura->excludes(self.asignaturaAlumno.profesor)

- **Para obtener un título es preciso tener aprobados al menos 240 créditos de las asignaturas que componen una titulación de grado, o los 60 que componen una de posgrado.**

context Alumno inv CreditosObtenerGrado:
 self.titulacionAlumno->select(t:Titulacion|t.ocllsTypeOf(Grado))->forAll(g|
 g.alumnoAsignatura->select(a:Asignatura | a.matricula->exists(m |
 m.nota>=5)).creditos->sum()>=240 implies g.titulo->size()=1

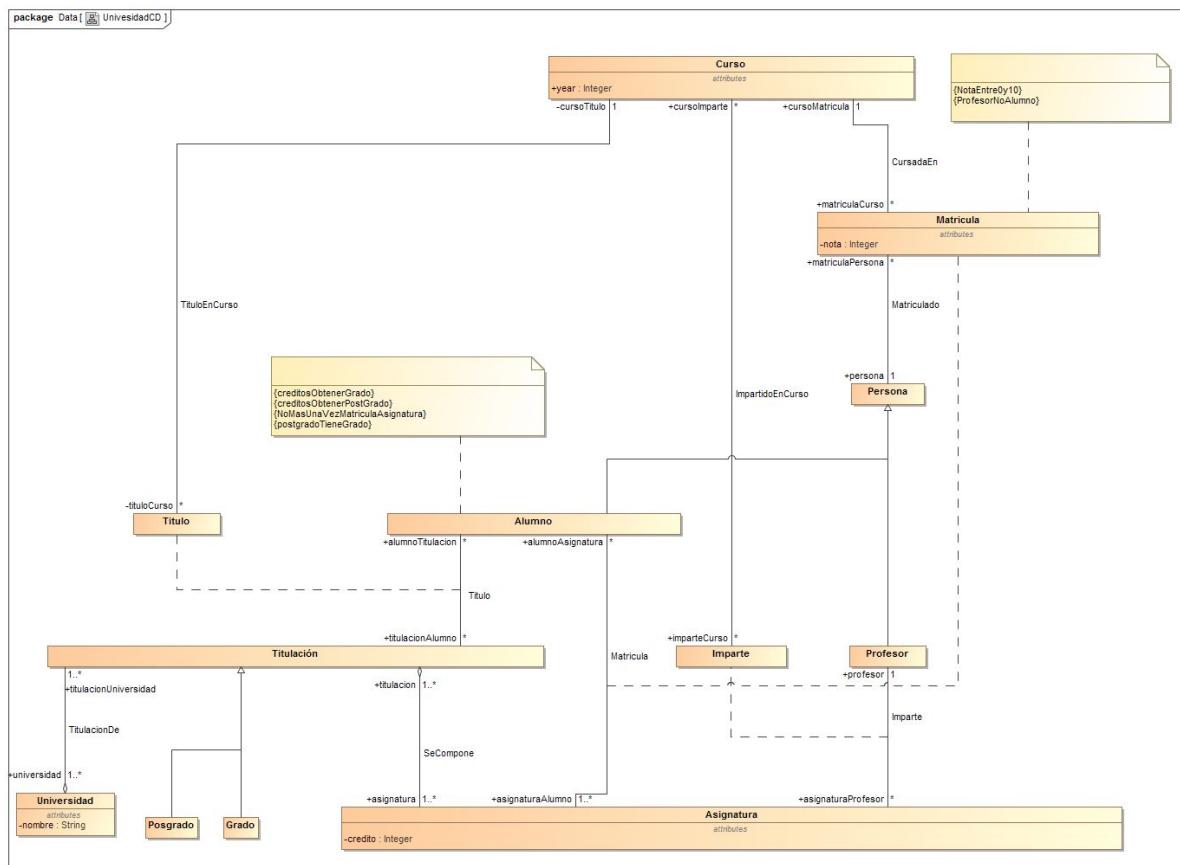
context Alumno inv CreditosObtenerPostGrado:
 self.titulacionAlumno->select(t:Titulacion|t.ocllsTypeOf(Posgrado))->forAll(g|
 g.alumnoAsignatura->select(a:Asignatura | a.matricula->exists(m |
 m.nota>=5)).creditos->sum()>=60 implies g.titulo->size()=1

- **Para poder matricularse de una asignatura de posgrado, el alumno ha de estar en posesión de un título de grado (de esa universidad o de otra).**

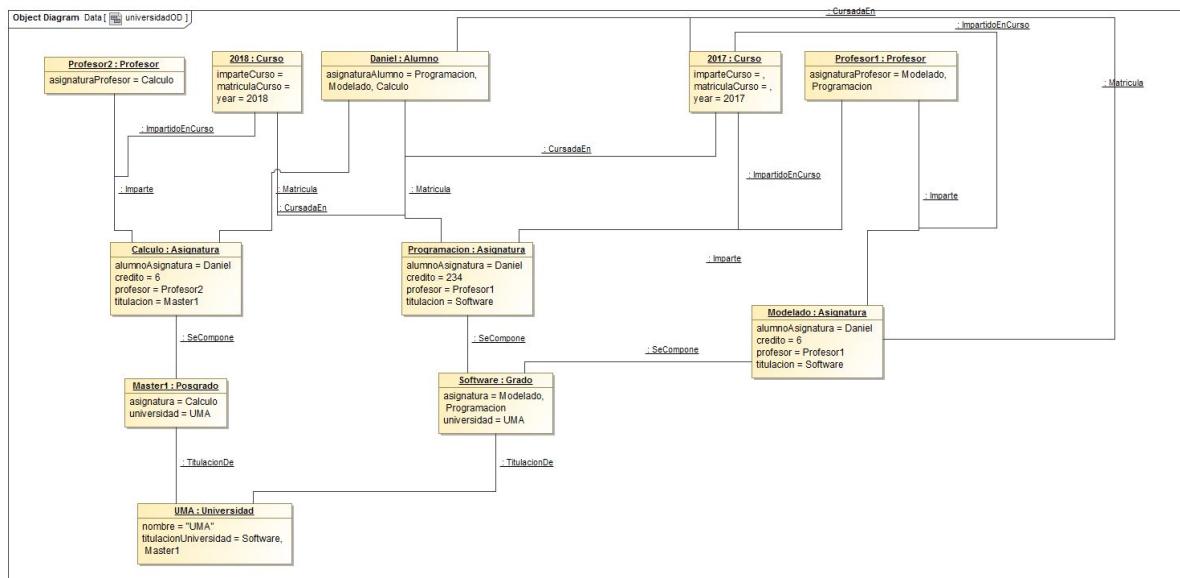
context Alumno inv PostgradoTieneGrado:
 self.asignaturaAlumno->select(as : Asignatura | as.titulacion->
 forAll(t:Titulacion | t.ocllsTypeOf(Posgrado)))->notEmpty() implies
 self.titulacionAlumno->
 select(t:Titulacion | t.ocllsTypeOf(Grado))->notEmpty()

MagicDraw

- Diagrama de Clases

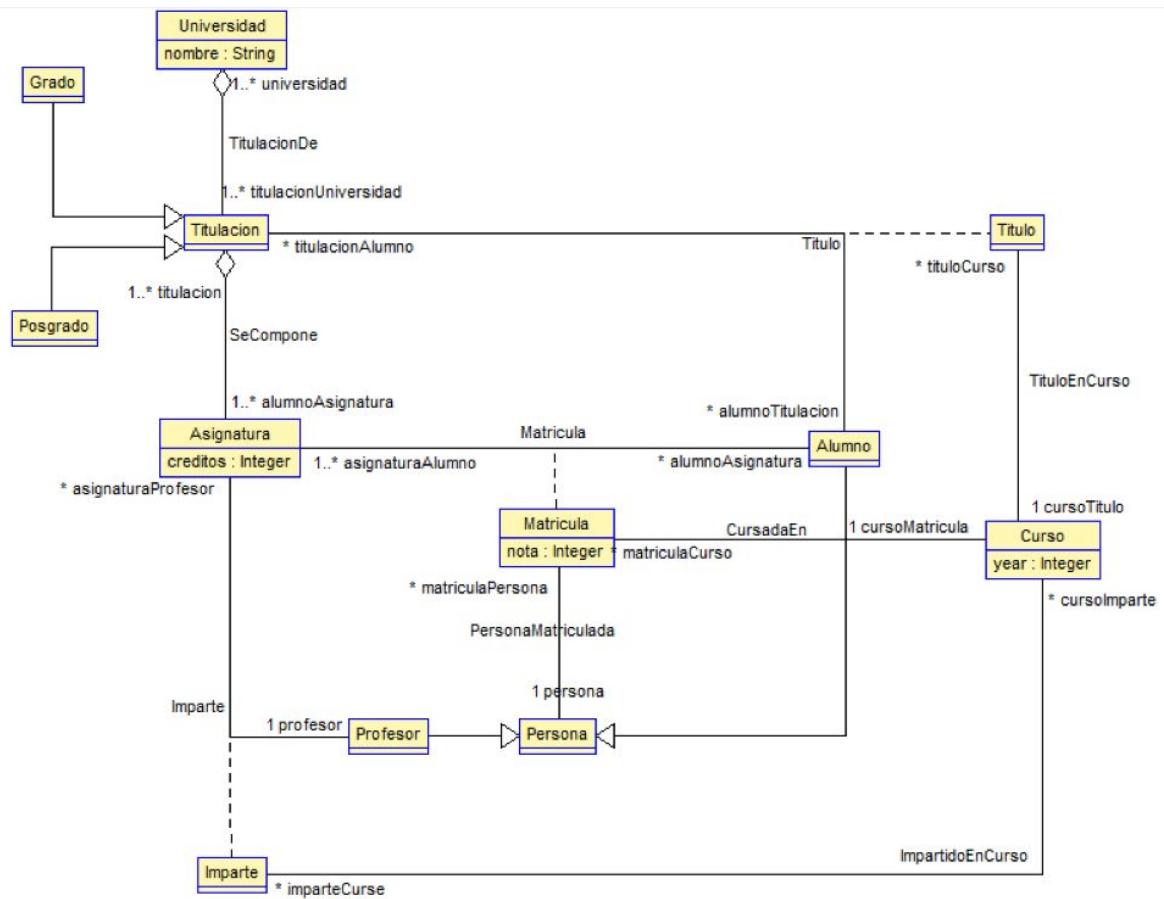


- Diagrama de Objetos

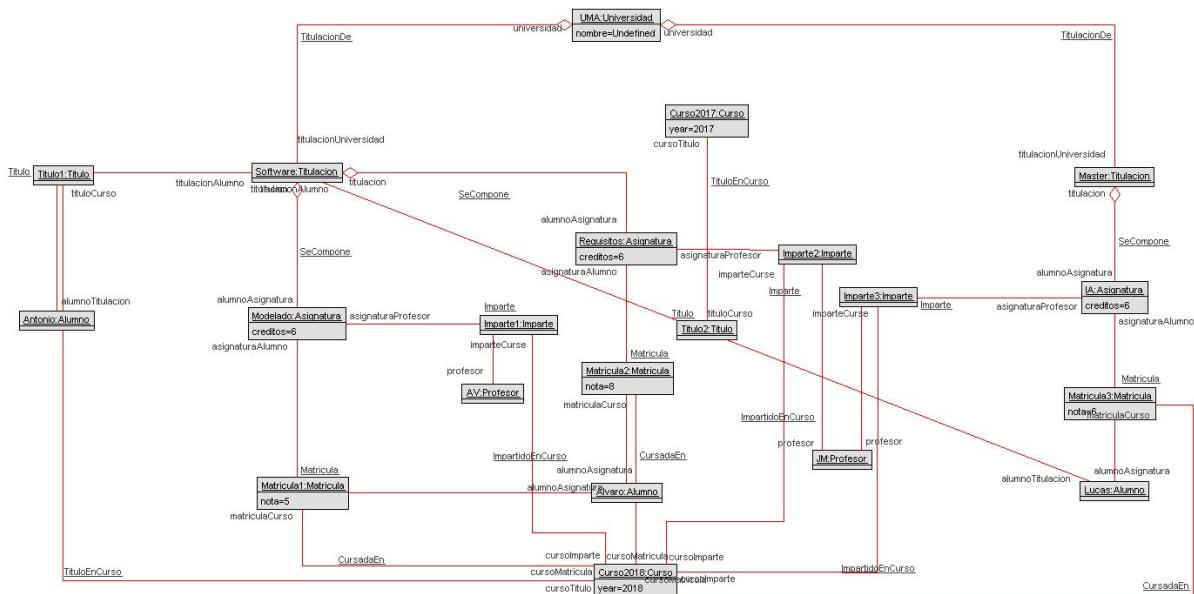


USE

- Diagrama de Clases

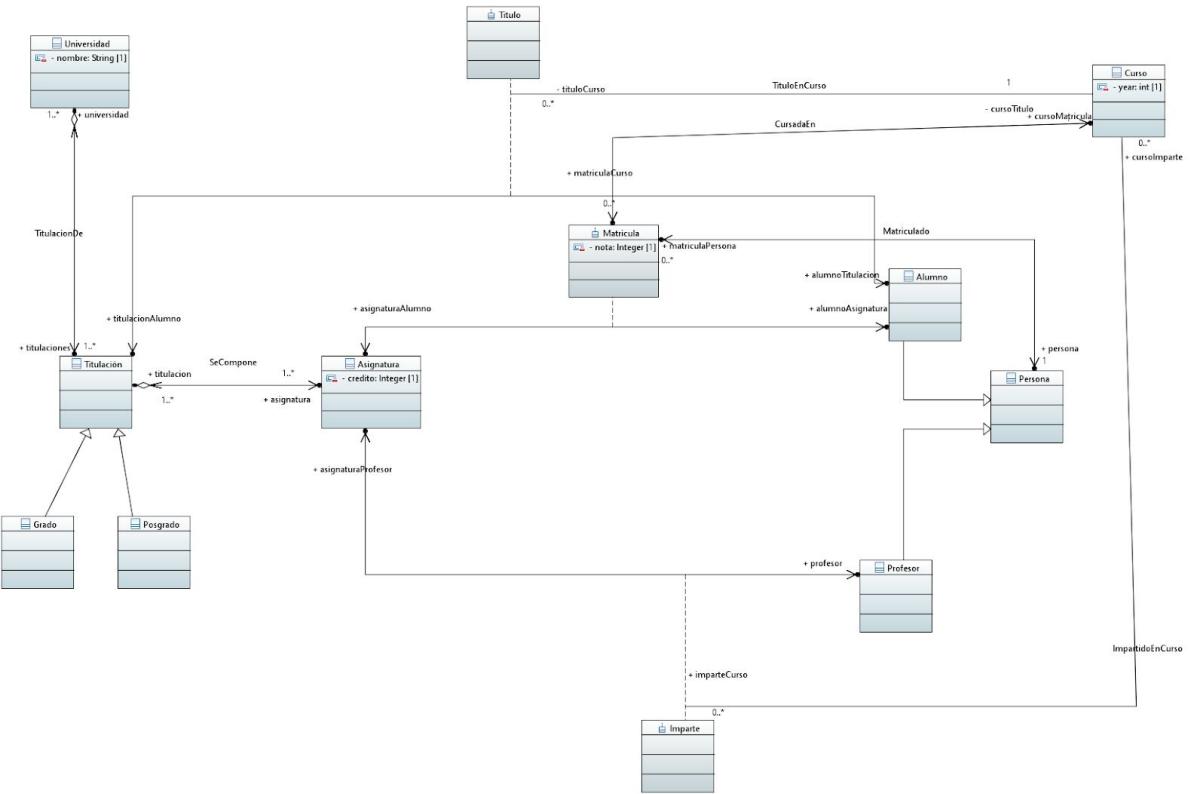


- Diagrama de Objetos

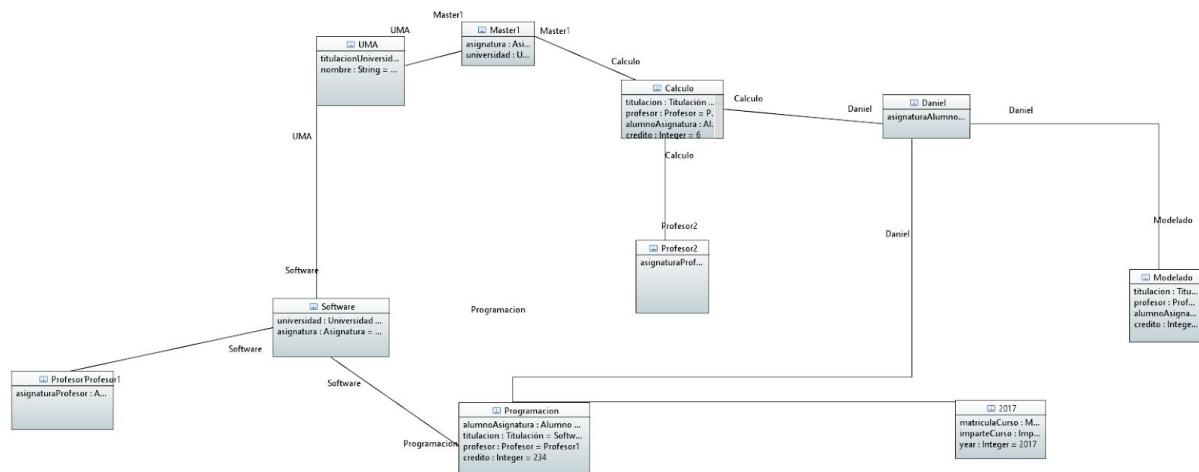


Papyrus

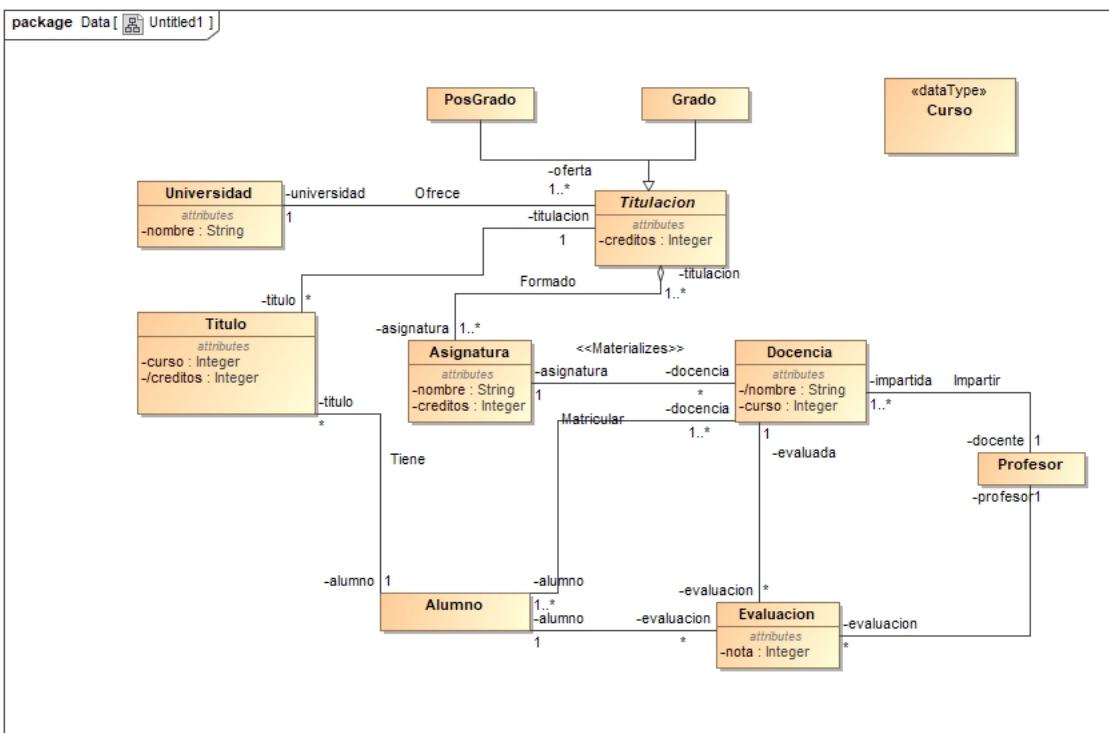
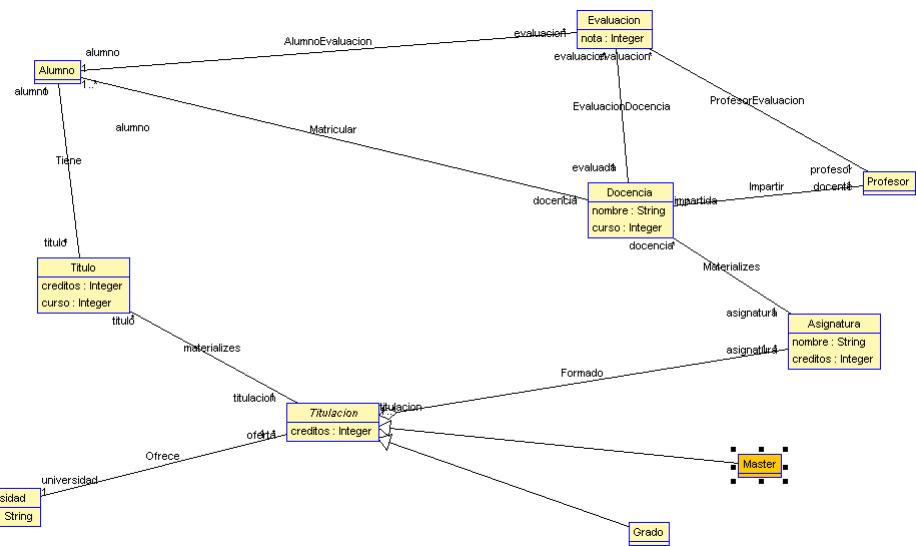
- Diagrama de Clases

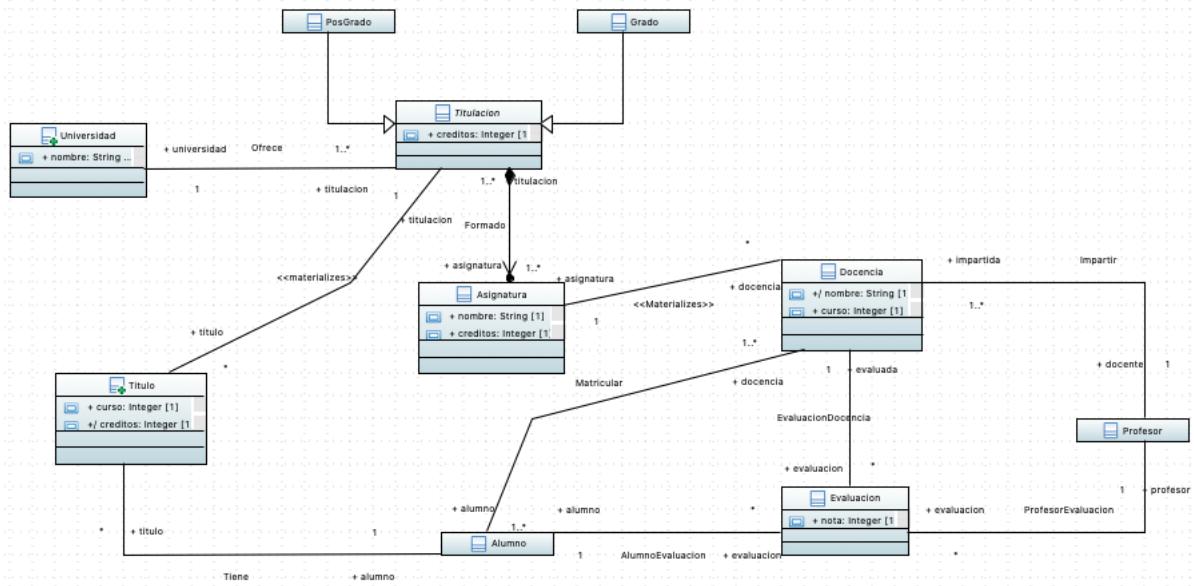


- Diagrama de Objetos



Tres





El presente modelo representa una solución a el enunciado de la práctica expresado mediante las entidades y relaciones que se encuentran en el mismo. A continuación se explican algunas de las entidades y relaciones que pueden dar lugar a dudas.

- Titulación es una entidad abstracta que representa al conjunto de tanto Posgrados como Grados que son impartidos en una universidad. La titulación está compuesta por las asignaturas que son requeridas para obtener su respectivo título.
- Docencia es la materialización de una asignatura siendo ésta la impartición de los conocimientos que son englobados en una asignatura por parte de un profesor hacia un conjunto de alumnos en un curso concreto. Para que esta exista es necesario que haya un profesor que la imparta y al menos un alumno al que sea impartida.
- Una evaluación resulta de la calificación que un profesor realiza a un alumno respecto a una docencia.
- Un título es la materialización de una titulación simbolizando éste los conocimientos adquiridos por un alumno en las asignaturas de su respectiva titulación. Dado que un título tiene asociada una titulación de esta se puede obtener la universidad a la que pertenece.

Se ha decidido no realizar una entidad persona de la que hereden alumno y profesor debido a que aunque un profesor pueda ser alumno, en UML no existe restricción respecto al asignar tipos múltiples y una instancia puede pertenecer a ambos tipos.

2. Restricciones de integridad

2.1. Restricciones de titulación

```
context Titulacion inv creditosTitulacionObligatoria:
not self.creditos.octIsUndefined()
```

La restricción superior exige que toda titulación tiene que tener definido un número de créditos.

```
context PosGrado inv creditosPostGrado60:  
    self.creditos = 60
```

La restricción superior exige que toda titulación de posgrado tiene que tener 60 como cantidad de créditos.

```
context Grado inv creditosGrado240:  
    self.creditos = 240
```

La restricción superior exige que toda titulación de grado tiene que tener 240 como cantidad de créditos.

2.2. Restricciones de Asignatura

```
context Asignatura inv creditosObligatorio:  
    not self.creditos.oclisUndefined()
```

La restricción superior exige que una asignatura no puede tener una carga de créditos no asignada.

```
context Asignatura inv nombreAsignaturaObligatoria:  
    not self.nombre.oclisUndefined()
```

La restricción superior exige que una asignatura debe tener un nombre asignado

```
context Asignatura inv creditosPositivo:  
    self.creditos > 0
```

La restricción superior exige que los créditos de una asignatura tienen que ser positivos.

2.3. Restricciones de Docencia

```
context Docencia inv mismoNombre:  
    self.nombre = self.asignatura.nombre
```

La restricción superior exige que una asignatura materializada en una Docencia debe tener el mismo nombre que su materialización.

```
context Docencia inv cursoObligatorio:  
    not self.curso.oclisUndefined()
```

La restricción superior exige que una asignatura materializada en una Docencia debe tener definido el campo *curso*.

2.4. Restricciones de Evaluación

```
context Evaluacion inv nota0Hasta10:  
    self.nota >= 0 and self.nota <= 10
```

La restricción superior exige que una evaluación debe estar calificada con una nota entre 0 y 10 ambos inclusive.

```
context Evaluacion inv evaluacionEvaluadaPorElMismoProfesorQueLaImparte:
    self.evaluada.docente = self.profesor
```

La restricción superior exige que un profesor sea quien evalúe a su grupo de docencia

2.5. Restricciones de Alumno

```
context Alumno inv alumnoMatriculadoEnPostgradoImplicaTituloGrado:
    self.docencia.asignatura.exists(asig | asig.titulacion.forAll(t |
        t.oclIsTypeOf(Master)))
        implies
    self.titulo.exists(t | t.titulacion.oclIsTypeOf(Master))
```

La restricción superior exige que para matricularse de una asignatura de posgrado el alumno tiene que tener al menos un título de grado

2.6. Restricciones de Profesor

```
context Profesor inv profesorNoImparteASiMismo:
    self.impartida.alumno->forAll(al | al <> self)
```

La restricción superior exige que un profesor no pueda impartir una asignatura donde haya un alumno que sea él mismo.

2.7. Título

```
context Titulo inv cursoTituloObligatorio:
    not self.curso.oclIsUndefined()
```

La restricción superior exige que el curso del título tiene que existir para toda instancia.

```
context Titulo inv mismoNumeroDeCreditosTotales:
    self.creditos = self.titulacion.creditos
```

La restricción superior exige que el número de créditos de título debe ser igual al de la titulación.

Archivos adjuntos

1. universidad.use
2. universidad.mdzip
3. universidad.di

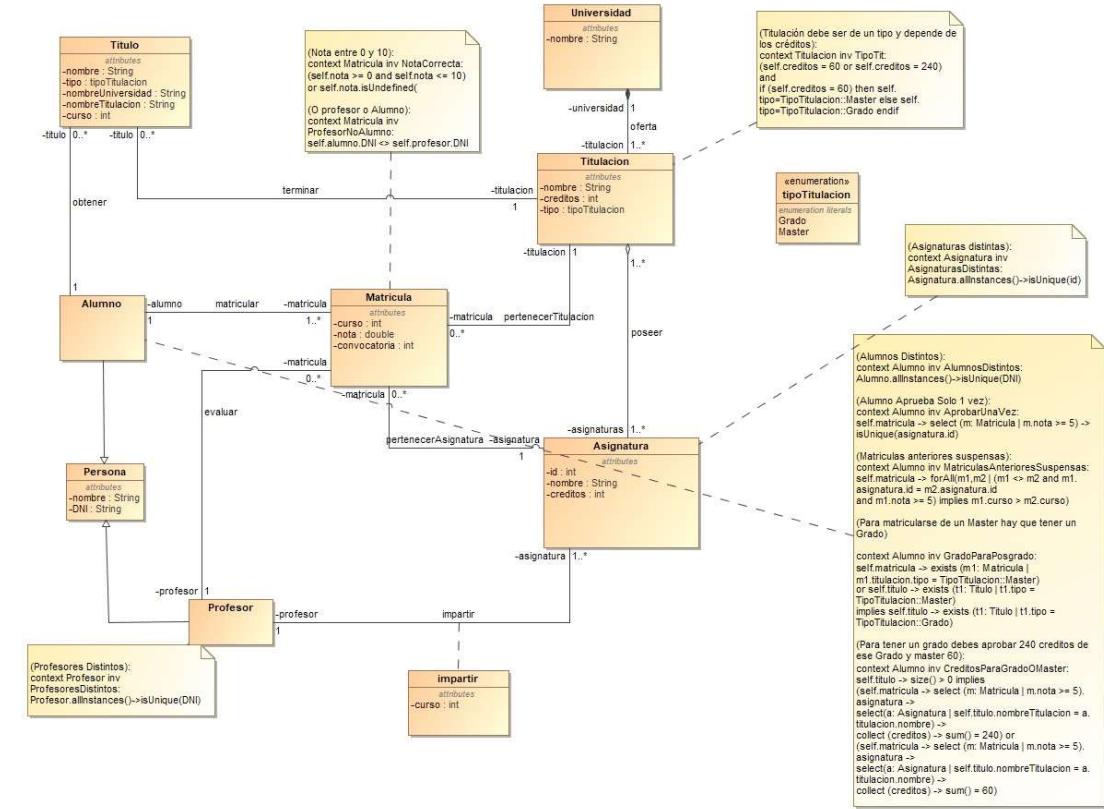
Cuatro

PRÁCTICA 1: MODELO UNIVERSITARIO

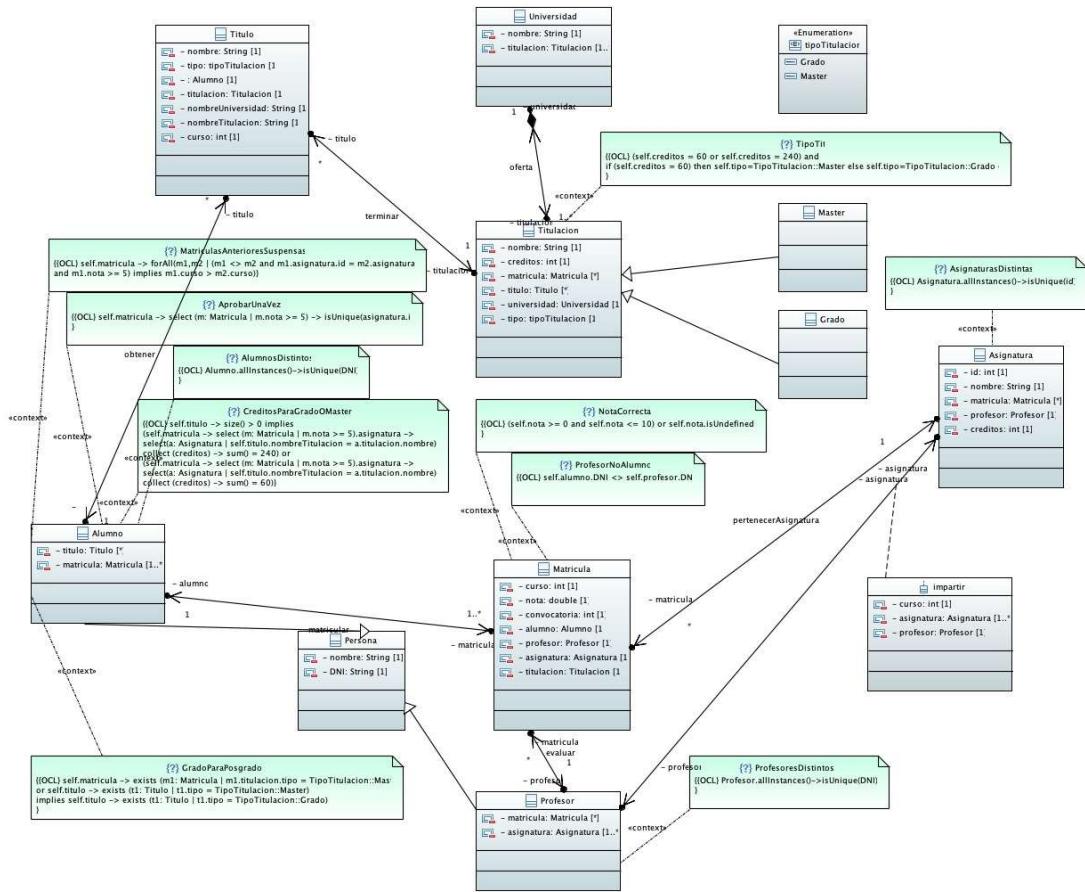
Introducción:

Este documento pretender explicar el modelo universitario, descrito con diagramas de clases realizados con las herramientas “MagicDraw”, “USE” y “Papyrus”, incluyendo la explicación de todas sus clases, relaciones y restricciones. Además, se realizarán diagramas de objetos que serán probados con “USE” para comprobar que cumplen las restricciones del modelo.

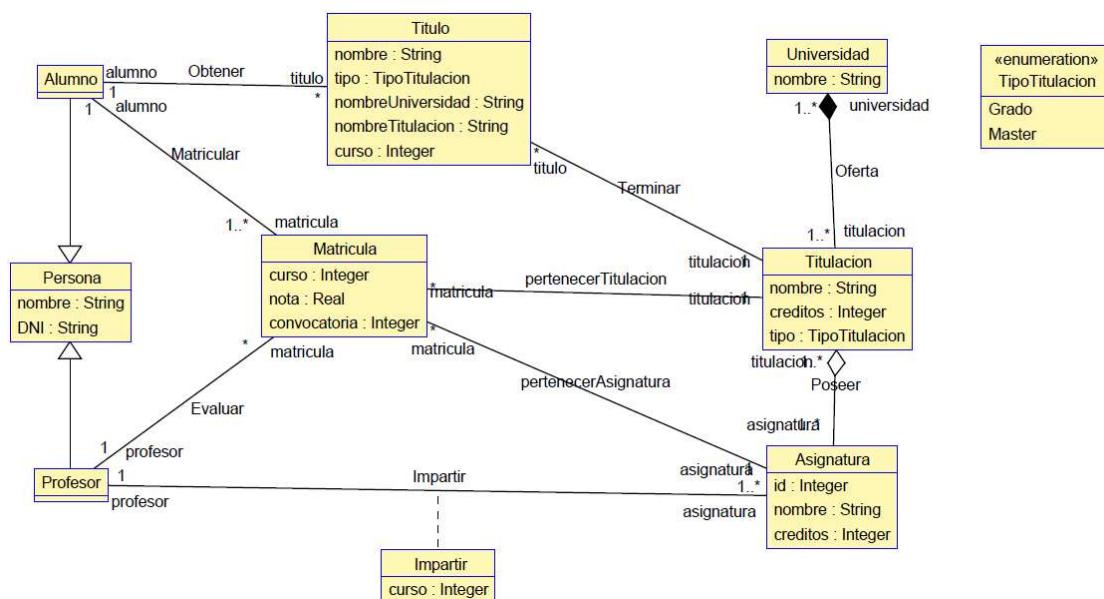
Modelo en MagicDraw:



Modelo en Papyrus:



Modelo en USE:



Análisis del dominio:

En este apartado pasamos a comentar las clases que forman el modelo estructural, así que sus atributos y las relaciones entre ellas, explicando el por qué de las multiplicidades, el tipo de relación, etc...

- Una Universidad debe tener un nombre identificatorio, y se relaciona con las titulaciones a través de la relación “oferta”, que es una relación de composición, ya que una titulación no puede existir sin pertenecer a una Universidad (y solo pueden pertenecer a esta), y una Universidad está compuesta por una (ya que no tiene sentido que exista una universidad que no ofrezca titulaciones) o muchas titulaciones. Esto está definido en las propias multiplicidades de la relación.
- tipoTitulacion es un enumerable que puede tomar dos valores: “Grado” o “Máster”.
- Una titulación debe tener un nombre identificatorio, un número de créditos y un atributo que define su tipo (tipoTitulacion, es decir, tomará valores de “Grado” o “Máster”), que depende de los créditos que tenga dicha titulación. Si tiene 240 créditos será un Grado y si tiene 60 créditos será un Máster.
- Titulación se relaciona con asignatura mediante la relación “poseer”. Una titulación está “compuesta” por una o más asignaturas (como define la multiplicidad de la relación) pero una asignatura puede pertenecer a una o más titulaciones (como define también la multiplicidad de esa misma relación), por lo que en lugar de ser una composición debe ser una agregación.
- Una asignatura es obligatoriamente impartida por un profesor en un curso concreto, que a su vez puede impartir una o muchas asignaturas en dicho curso. Esta relación está definida por la clase relacional “impartir”, que tiene de atributo un curso y define con las multiplicidades que una asignatura puede ser impartida solo por un profesor y que un profesor puede impartir una o más asignaturas.
- Tanto profesor como alumno heredan de persona, que tiene un nombre y un DNI que la identifica (ambos atributos son heredados por sus hijos).
- Un alumno está relacionado con un título a través de la relación “obtener”, y como podemos ver en las multiplicidades, un título solo puede pertenecer a un alumno concreto (y además debe existir un alumno que lo posea para que exista el título), y un alumno puede detener una cantidad indeterminada de título, incluyendo el caso de no tener ninguno.
- Un título tiene un nombre, un tipo (que define si es de Grado o Máster), el nombre de la Universidad, el nombre de la titulación y el curso en el que se obtuvo. Se relaciona, además de con alumno (ya explicado), con una titulación (desde la que obtendrá el valor de los atributos que define el nombre de la titulación y nombre de Universidad), mediante la relación “terminar”. Como definen las multiplicidades de dicha relación, un título debe estar obligatoriamente relacionado con una y solo una titulación para existir, y una titulación puede tener un número indeterminado número de títulos (que dependerá de los alumnos que hayan acabado la titulación).

- Tenemos además matrículas, que están relacionadas con titulaciones, asignaturas y alumnos.
- Una matrícula tiene un curso, una nota y una convocatoria.
- La idea de la clase matrícula surge de que un alumno debe tener una matrícula por cada asignatura y titulación en la que está. Es por eso que matrícula se relaciona con alumno a través de la relación “matricular”, que como definen sus multiplicidades obliga a que para que exista una matrícula, esta pertenezca a un alumno, que a su vez puede tener una o muchas matrículas (si no estuviera matriculado en nada no sería un alumno). Matrícula se relaciona también con titulación a través de la relación “pertenercerTitulacion” y con asignatura a través de la relación “pertenercerAsignatura”. Ambas relaciones obligan a que para que exista una matrícula, esta debe pertenecer a una y solo una titulación, y a una y solo una asignatura. A su vez, una titulación puede tener cero o muchas matrículas de alumnos, y una asignatura puede tener cero o muchas matrículas de alumnos (como definen las multiplicidades de las dos relaciones mencionadas).
- Matrícula se relaciona también con profesor, ya que este debe evaluar las matrículas de los alumnos para establecer su nota. Esta relación se hace a través de “evaluar”, que en sus multiplicidades obliga a que, si hay una matrícula de un alumno en una asignatura de una titulación, tenga que ser evaluada por un profesor obligatoriamente (mediante restricciones se controlará que sea el mismo profesor que da esa asignatura), y además un profesor puede evaluar una o muchas matrículas.

Restricciones:

En este apartado nos vamos a dedicar a definir las restricciones del modelo que hemos realizado:

- Un profesor no puede matricularse de una asignatura que está impartiendo:

```
context Matricula inv ProfesorNoAlumno:  
self.alumno.DNI <> self.profesor.DNI
```

- Un alumno solo podrá matricularse de una asignatura si no la ha aprobado anteriormente:

```
context Alumno inv AprobarUnaVez:  
self.matricula -> select (m: Matricula | m.nota >= 5) ->  
isUnique(asignatura.id)
```

```
context Alumno inv MatriculasAnterioresSuspensas:  
self.matricula -> forAll(m1,m2 | (m1 <> m2 and m1.asignatura.id =  
m2.asignatura.id and m1.nota >= 5) implies m1.curso > m2.curso)
```

- Una asignatura debe tener una nota entre 0 y 10:

```
context Matricula inv NotaCorrecta:  
(self.nota >= 0 and self.nota <= 10) or self.nota.isUndefined()
```

- Todas las asignaturas deben tener un id distinto:

```
context Asignatura inv AsignaturasDistintas:  
Asignatura.allInstances()->isUnique(id)
```

- Todos los profesores del sistema deben tener un DNI distinto, y todos los alumnos del sistema deben tener un DNI distinto. Un alumno y un profesor pueden tener el mismo DNI (Son la misma persona):

```
context Profesor inv ProfesoresDistintos:  
Profesor.allInstances()->isUnique(DNI)
```

```
context Alumno inv AlumnosDistintos:  
Alumno.allInstances()->isUnique(DNI)
```

- Una titulación tiene que tener 240 o 60 créditos, y dependiendo de los que tenga será Grado o un Máster respectivamente:

```
context Titulacion inv TipoTit:  
(self.creditos = 60 or self.creditos = 240) and  
if (self.creditos = 60) then  
    self.tipo=TipoTitulacion::Master  
else  
    self.tipo=TipoTitulacion::Grado  
endif
```

- Para que un alumno obtenga el título de una Titulación debe haber superado los créditos que tenga esa misma que consiste en aprobar ese número de créditos de asignaturas que componen estas titulaciones (Máster → 60 créditos | Grado → 240 créditos):

- ```
context Alumno inv CreditosParaGradoOMaster: self.titulo -> size() > 0
implies
(self.matricula -> select (m: Matricula | m.nota >= 5).asignatura ->
select(a: Asignatura | self.titulo.nombreTitulacion = a.titulacion.nombre)
-> collect (creditos) -> sum() = 240)
or
(self.matricula -> select (m: Matricula | m.nota >= 5).asignatura ->
select(a: Asignatura | self.titulo.nombreTitulacion = a.titulacion.nombre)
-> collect (creditos) -> sum() = 60)
```

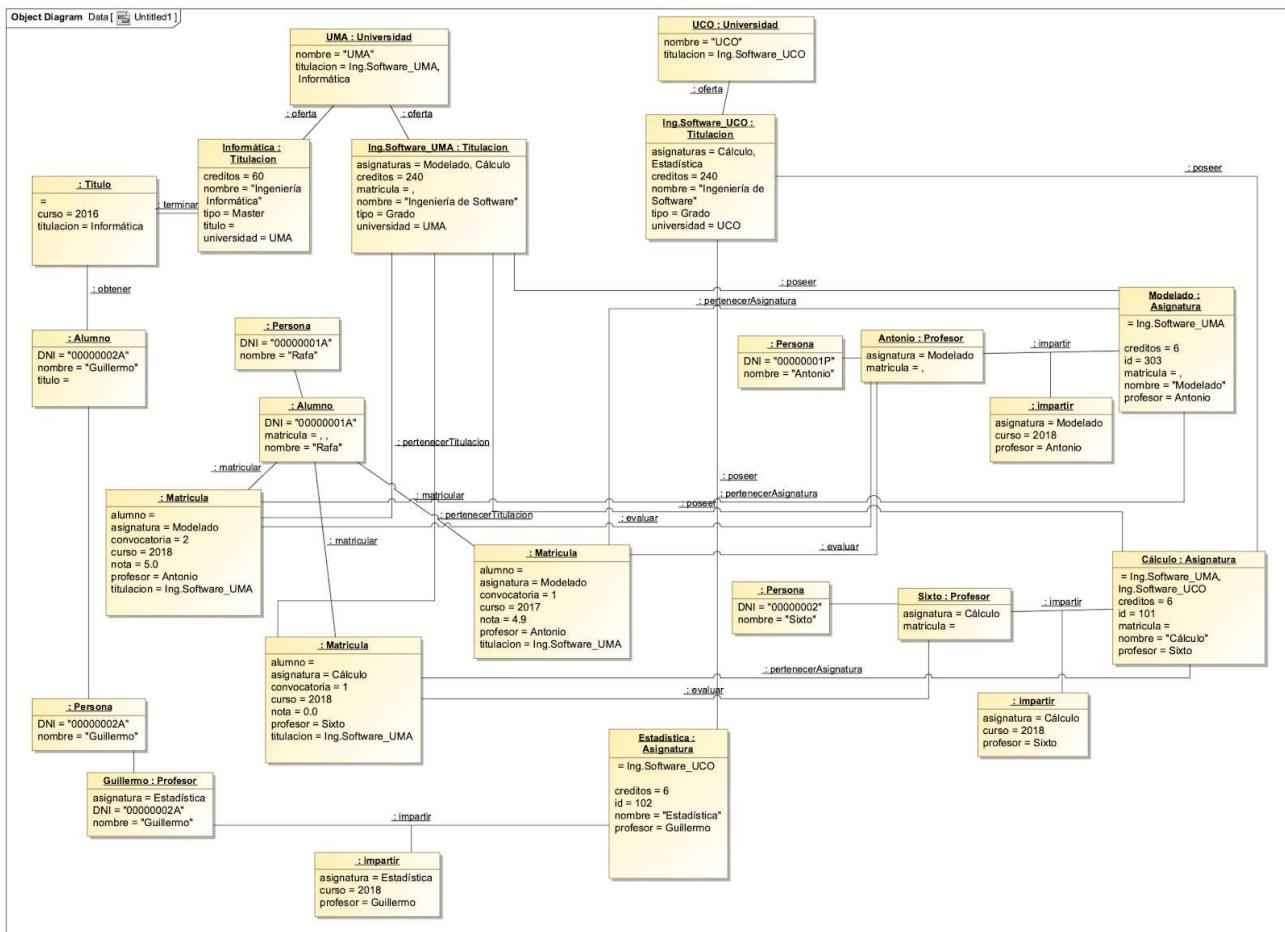
- Un alumno no puede tener un título de Máster o estar matriculado en un Máster si no tiene un título de Grado:

```

context Alumno inv GradoParaPosgrado:
self.matricula -> exists (m1: Matricula | m1.titulacion.tipo =
TipoTitulacion::Master)
or
self.titulo -> exists (t1: Titulo | t1.tipo = TipoTitulacion::Master)
implies
self.titulo -> exists (t1: Titulo | t1.tipo = TipoTitulacion::Grado)

```

## Diagrama de objetos en MagicDraw:



En este modelo representamos un caso real algo reducido, pero bien definido, en el que se detallan bien como trabajan las relaciones y restricciones:

Universidades diferentes ofrecen diferentes grados y máster que poseen asignaturas. En nuestro caso: UMA y UCO, cada una ofertando un grado de Ingeniería de Software.

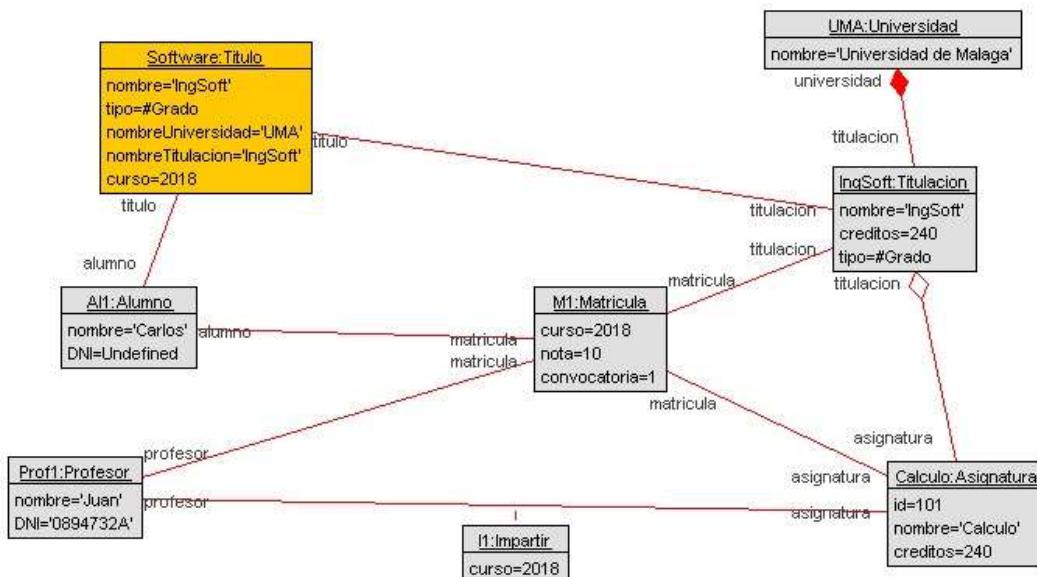
Además de los típicos casos simples de las asignaturas que conforman las titulaciones o las relaciones entre alumno, profesor, asignatura y matricular, se detallan casos un poco más curiosos:

-Podemos ver como Rafa se matricula por segunda vez en la asignatura Modelado por haberla suspendido en un curso anterior (Nota < 5).

-Mostramos también el caso de un profesor de la UCO que fue alumno en el Máster de Informática que ofrece la UMA.

## Diagrama de objetos y comprobación de restricciones en USE:

En este modelo representamos otro caso real, pero en USE, comprobando además que cumple todas las restricciones que hemos creado. Como no se cumplía el número de créditos mínimo para tener un título porque no tenemos tantas asignaturas creadas para llegar a tal número, hemos puesto que la única asignatura que hay tenga 240 créditos para que se cumpla la restricción del título.



```

checking structure...
checked structure in 2ms.
checking invariants...
checking invariant (1) `Alumno::AlumnosDistintos': OK.
checking invariant (2) `Alumno::AprobarUnaVez': OK.
checking invariant (3) `Alumno::CreditosParaGradoOMaster': OK.
checking invariant (4) `Alumno::GradoParaPosgrado': OK.
checking invariant (5) `Alumno::MatriculasAnterioresSuspensas': OK.
checking invariant (6) `Asignatura::AsignaturasDistintas': OK.
checking invariant (7) `Matricula::NotaCorrecta': OK.
checking invariant (8) `Matricula::ProfesorNoAlumno': OK.
checking invariant (9) `Profesor::ProfesoresDistintos': OK.
checking invariant (10) `Titulacion::TipoTit': OK.
checked 10 invariants in 0.034s, 0 failures.

```

# Cinco

## Práctica 1

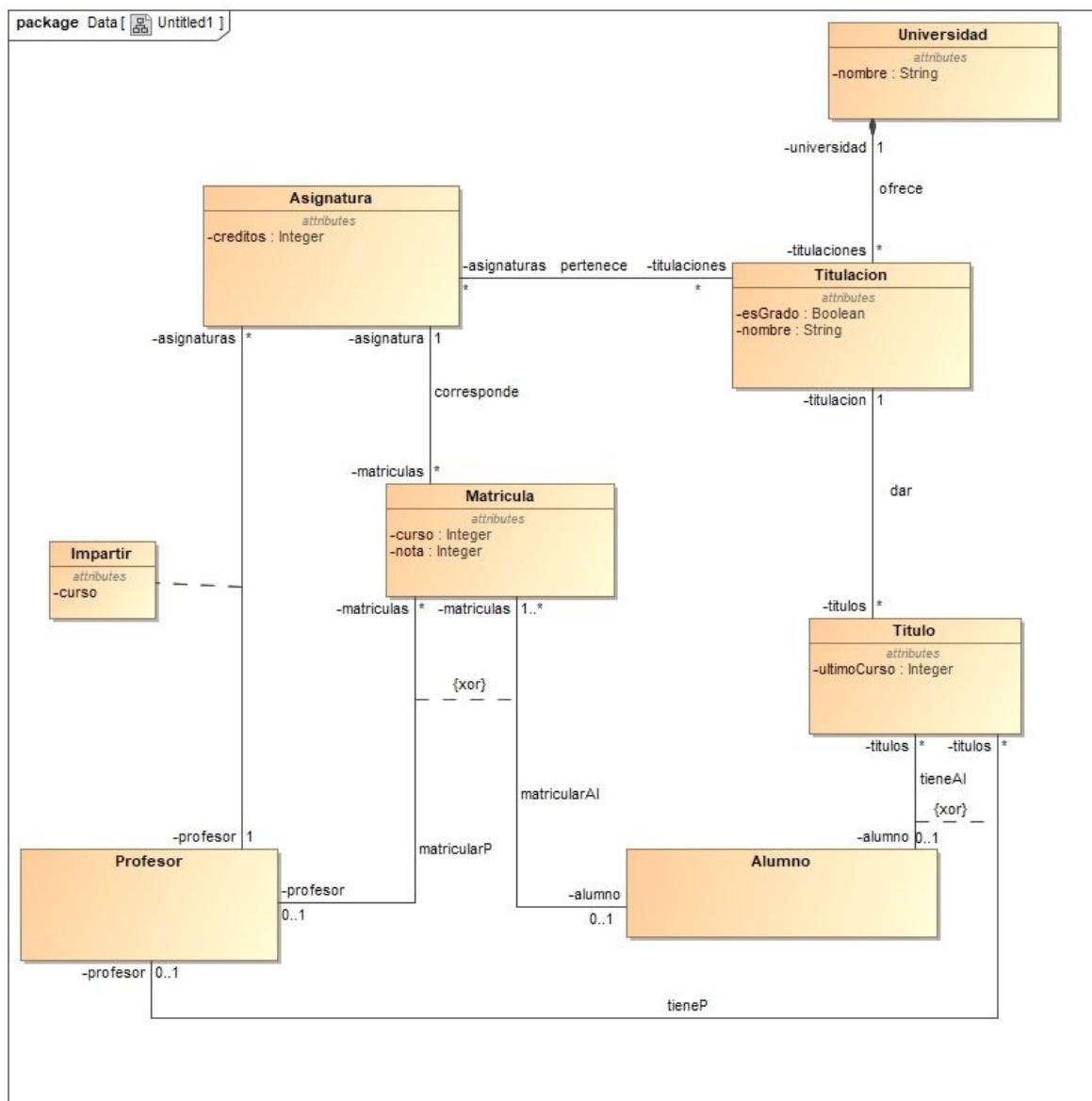
### **Explicación:**

Hemos modelado la clase Matrícula dada la necesidad de distinguir entre matrículas de alumnos y profesores, creando relaciones separadas para ambas. La anotación {xor} hace referencia a que una misma matrícula o bien corresponde a un alumno o a un profesor, no pudiendo carecer de ambos ni estar relacionada con ambos. La anotación {xor} en las relaciones entre Profesor y Alumno y Título tiene el mismo significado.

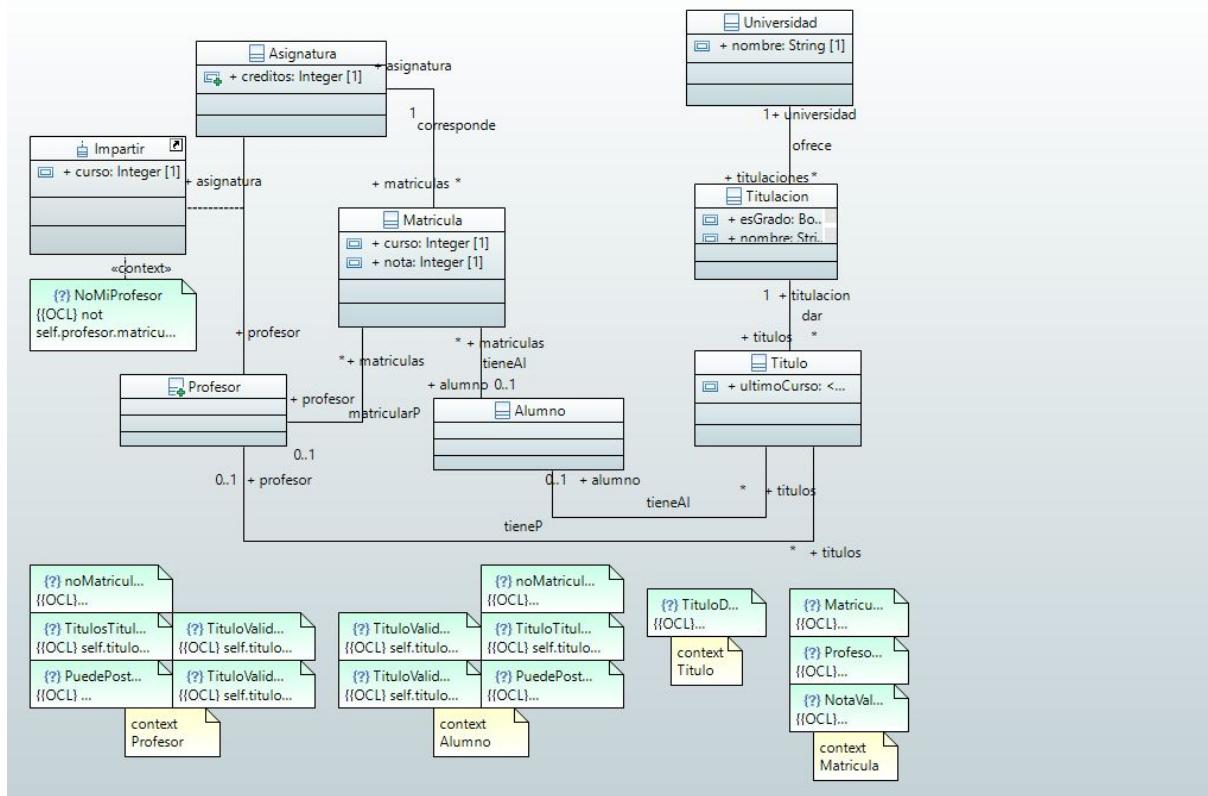
Hemos definido las siguientes restricciones en función de las necesidades de nuestro sistema:

- Un alumno/profesor no puede matricularse en una asignatura que ya ha aprobado.
- Un alumno/profesor puede tener un título si ha aprobado 240 créditos de la titulación correspondiente si es de grado o 60 si es de posgrado.
- La nota tiene que estar entre 0 y 10.
- Una matrícula pertenece a un solo alumno/profesor.
- Un título pertenece a un solo alumno/profesor.
- Para que un alumno/profesor se matricule en un postgrado tiene que tener un título de un grado.
- Un alumno/profesor no puede tener más títulos que titulaciones ofrece la universidad.
- Un profesor no puede ser su propio alumno.

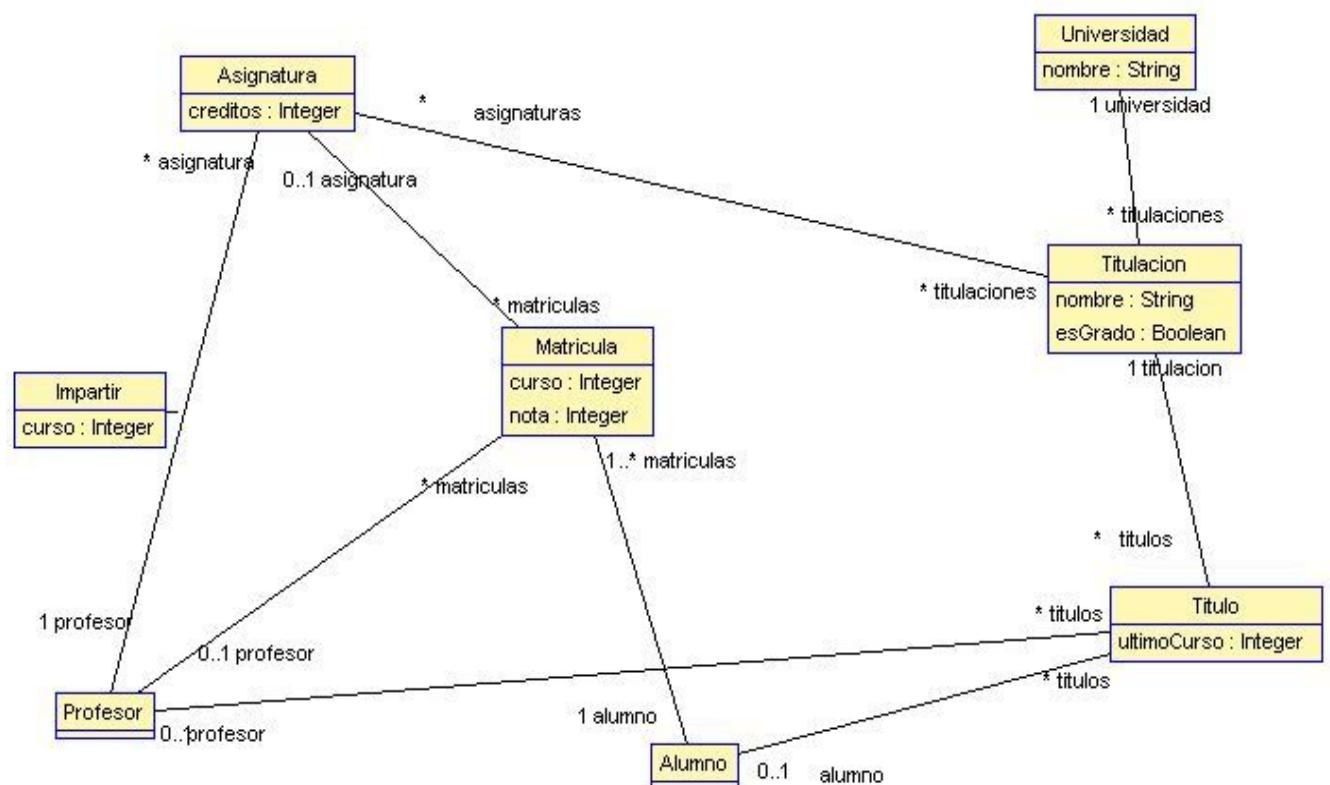
## Modelo en MagicDraw



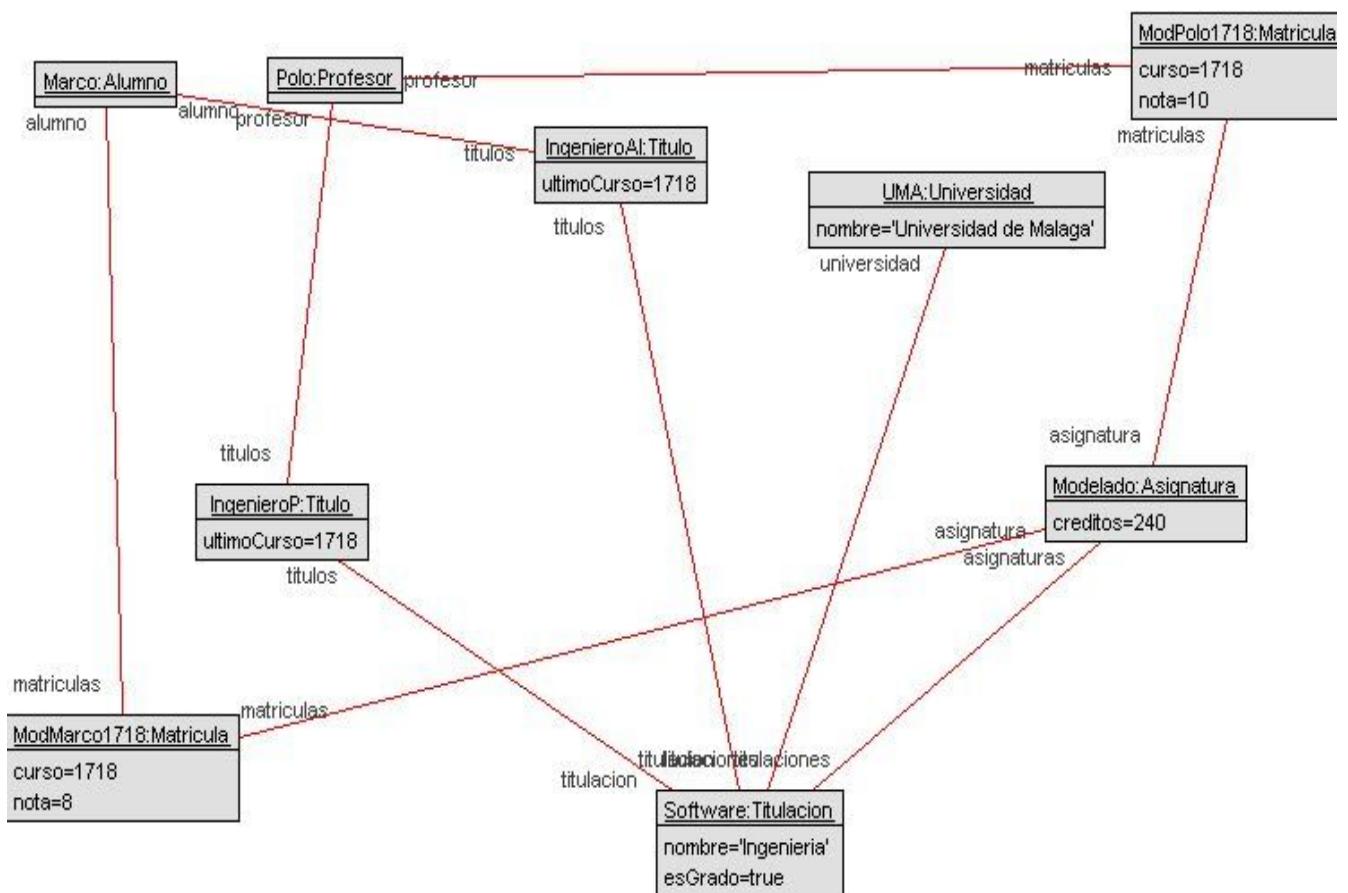
## Modelo en Papyrus



## Modelo en USE



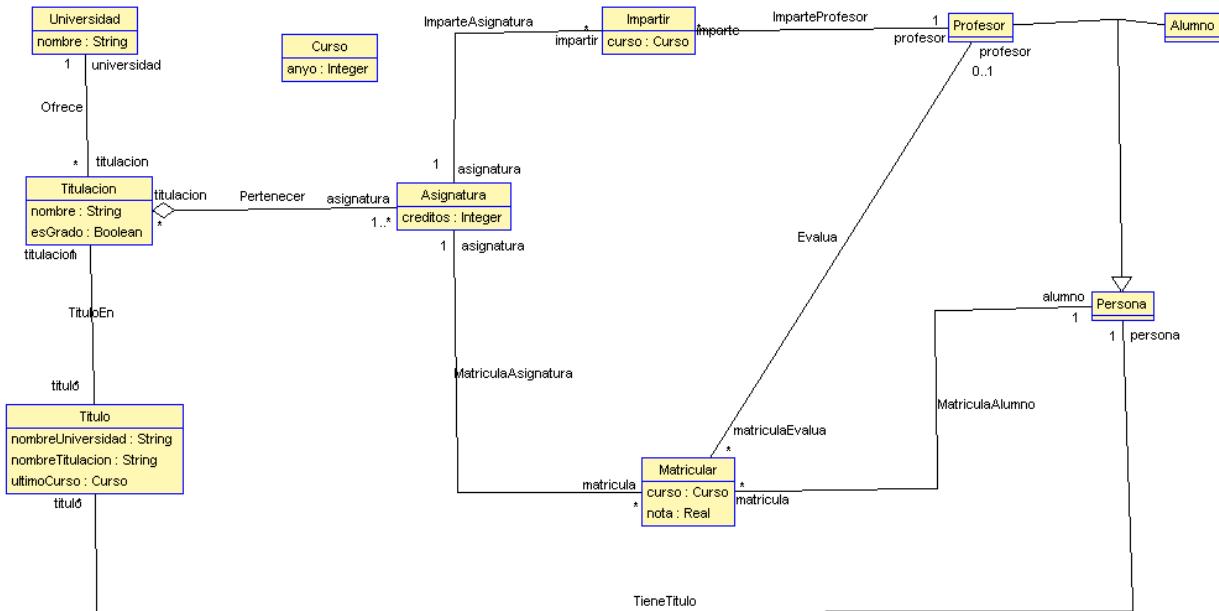
## Modelo de objetos en USE (para probar restricciones)



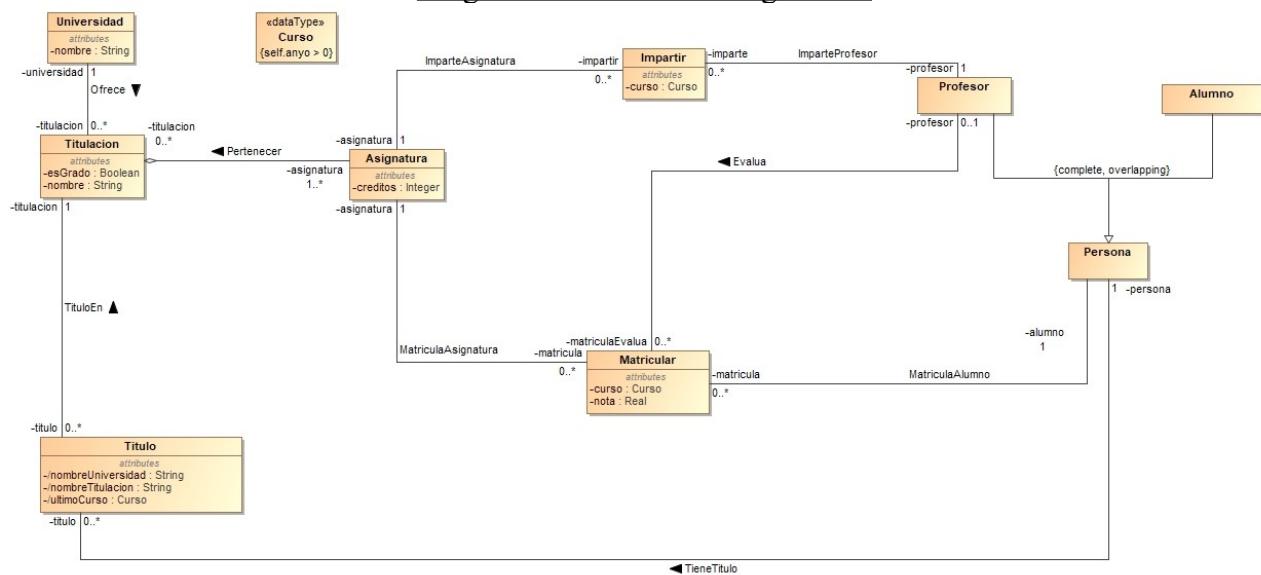
**Seis**

## -Diagramas de Clases:

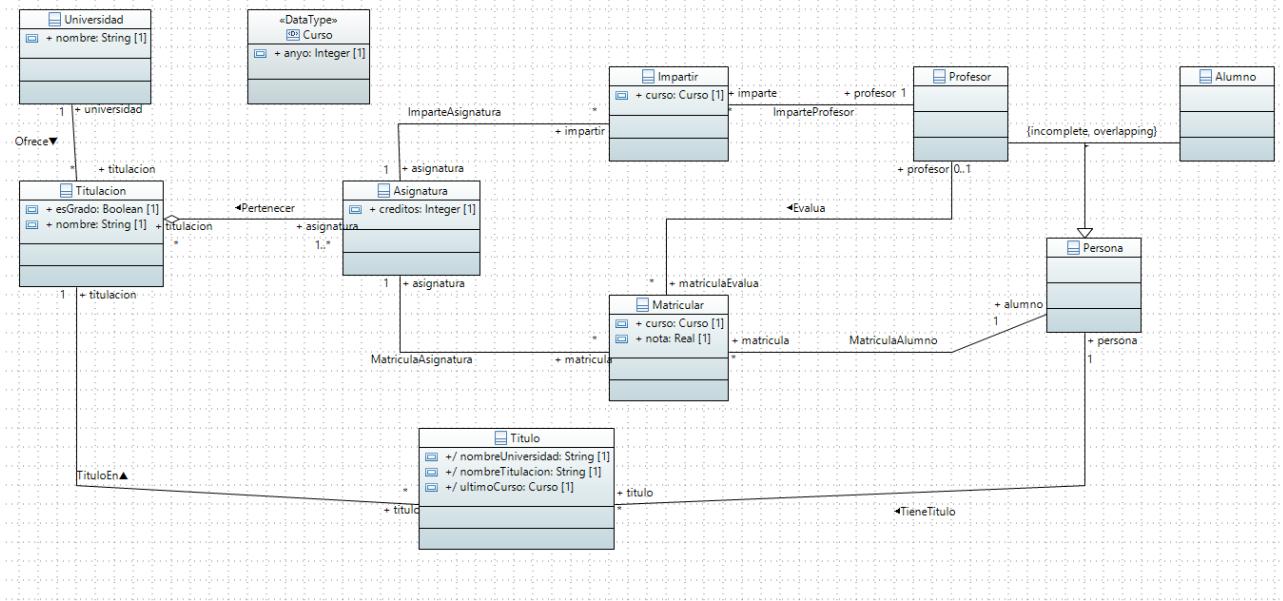
## Diagrama de Clases en UML



## Diagrama de Clases en MagicDraw

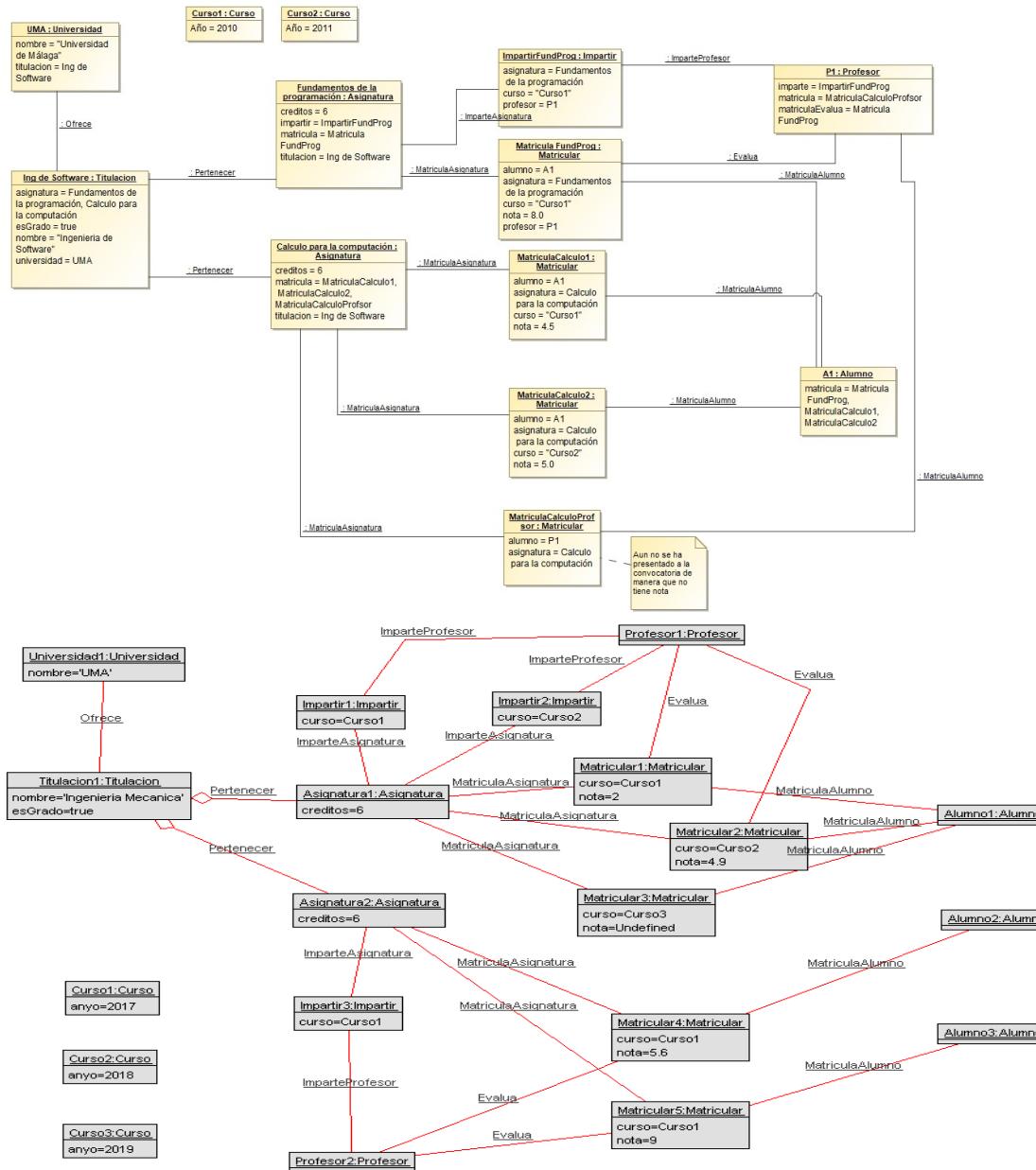


## Diagrama de Clases en Papyrus



## -Diagrama de Instancias:

Aquí se exponen dos ejemplos de diagramas de instancias:



## -Descripción de los puntos más importantes del modelo:

### A)Entidades:

- Se ha creado el tipo de dato Curso que simplemente almacena el año con un Integer. Ese año se refiere a una fecha (Ej.: 2017) y no al número del curso(Ej.: Primer curso, Quinto curso). Al almacenar solo un Integer se podría eliminar del modelo, pero consideramos que facilita la compresión del modelo.
- En la entidad Titulación se ha añadido un atributo booleano “esGrado” que indica si la titulación es de grado o postGrado. Consideramos que es la mejor forma de modelar esta idea debido a que Titulación no tendría funcionalidades diferentes según este aspecto. Simplemente en las restricciones se diferencia entre el número de créditos para obtener el título entre ambos tipos de titulaciones.
- Las entidades Impartir y Matricular realmente “relacionan” las entidades Asignatura y Persona. Entonces, ¿por qué no se han modelado como relaciones?. Primero porque necesitan guardar cierta información como curso y nota, pero la explicación a esta pregunta se encuentra desarrollada en la primera incidencia recogida más abajo en este documento.

### B)Relaciones y multiplicidades:

- Las multiplicidades de los roles de Asignatura y Persona/Profesor en las relaciones que involucran a estas entidades con Matricular e Impartir son siempre de 1. De modo que se obliga a la existencia de instancias de asignatura y persona/profesor para poder crear una instancia de Impartir o Matricular.
- Se ha establecido una relación de agregación entre Titulación y Asignatura, ya que las titulaciones harán referencia a las asignaturas de las que estén compuestas. Pero al eliminarse una titulación, las asignaturas deben de seguir “viviendo” puesto que estas se pueden encontrar en otras titulaciones.
- La generalización entre Profesor y Alumno con Persona es solapada y completa, esto es así porque todas las personas del modelo deben ser un profesor, un alumno o ambas cosas al mismo tiempo.

### C)Restricciones:

No se han incluido las restricciones en las fotos de los modelos debido a que ocuparían mucho espacio en este, así que a continuación se enumeran las que consideramos más relevantes o interesantes.

- “Un alumno no puede matricularse de una asignatura que ya tenga aprobada”
- “Un alumno no se puede matricular de la misma asignatura dos veces en el mismo curso”
- “Un profesor solo puede evaluar las matrículas de las asignaturas que él esté impartiendo en ese curso”
- “Un profesor no se puede matricular de una asignatura que él esté impartiendo durante ese curso”
- “Para obtener el título de un grado/postgrado es necesario tener aprobados un total de 240/60 créditos o más”
- “Una persona no puede tener un título en la misma titulación más de una vez”
- “Para poder obtener un título de postGrado es necesario tener al menos un título de grado”

## -Incidencias:

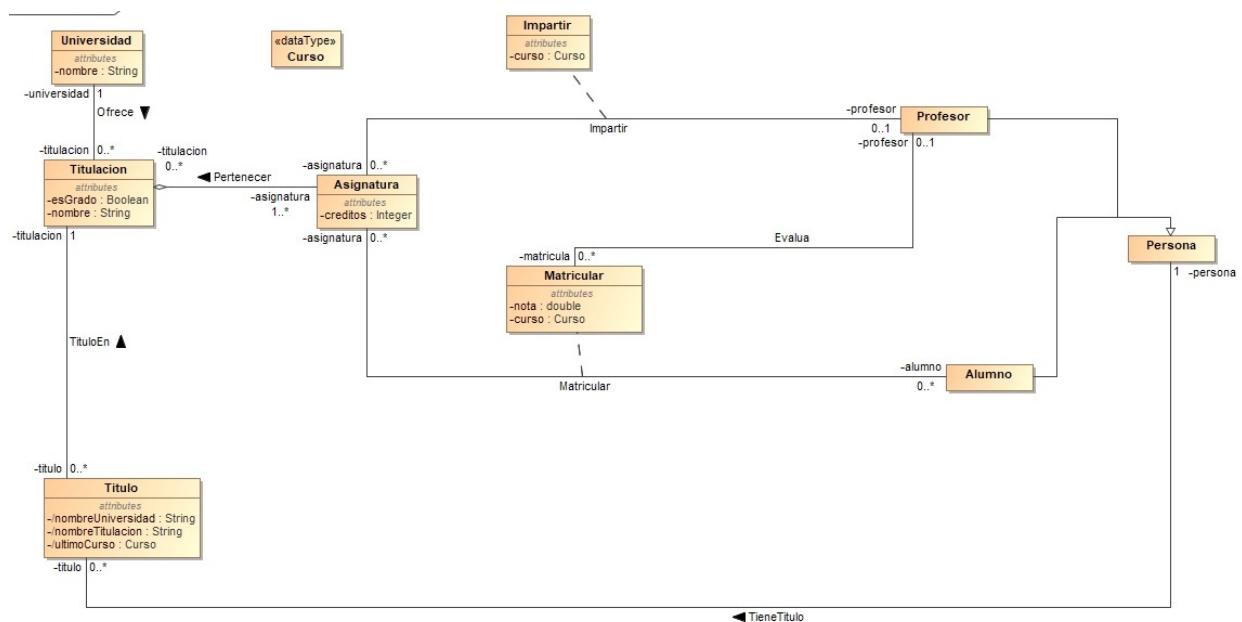
Durante el proceso de diseño del modelo han surgido diferentes errores o incidencias. A continuación se exponen las que consideramos más interesantes:

- Creación de las entidades Impartir y Matricula:

En un primer modelo usamos reificación para crear las relaciones de Matrícula e Impartir, debido a que necesitábamos añadirle los atributos curso y nota.

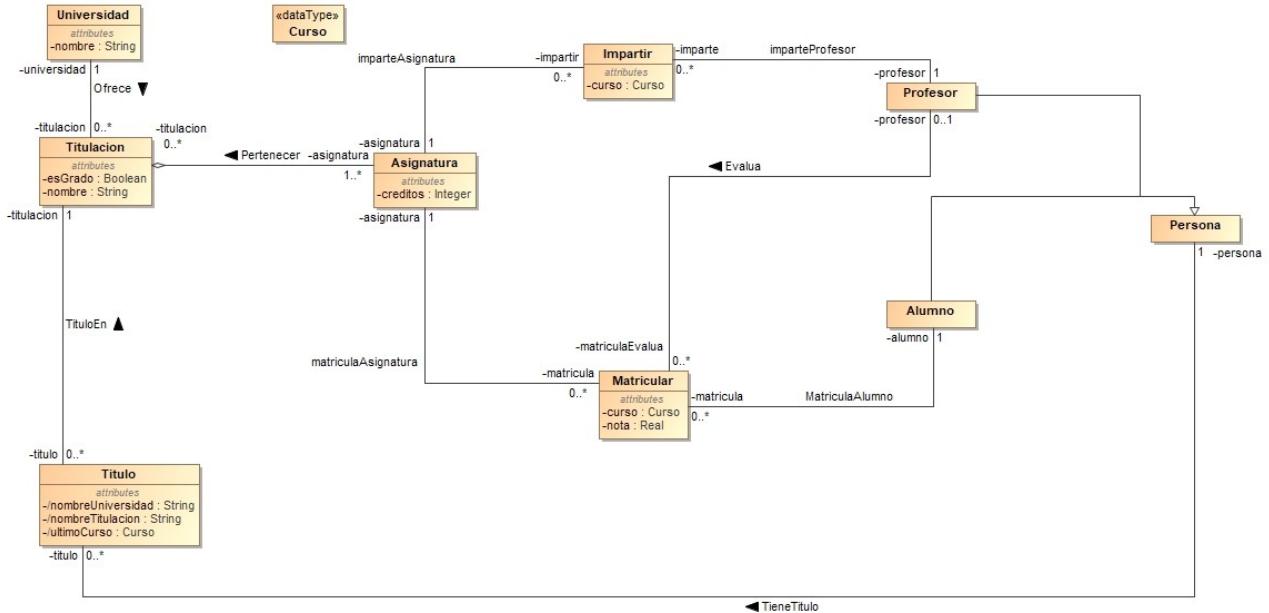
El problema llegó cuando tratamos de hacer un modelo de instancias en el que nos dimos cuenta de que no se nos permitía crear dos matrículas diferentes entre un alumno y una asignatura (“Un alumno se matricula de la misma asignatura en dos cursos diferentes”). Lo cual tiene todo el sentido del mundo ya que no se puede tener dos relaciones del mismo tipo entre las mismas instancias (Teoría de conjuntos: no elementos repetidos).

Nuestro problema era que no teníamos muy claro lo que era una reificación, pero este error nos ha ayudado a entenderla mejor. Y para este caso las sustituimos por las entidades Matricular e Impartir, que sí nos soluciona este problema.



- Cambio de relación matricular desde Alumno:

En un principio se planteó la relación MatriculaAlumno entre las entidades Matricular y Alumno, pero durante la especificación de la restricción: “Un profesor no se matricula en una asignatura que imparte durante ese curso”; vimos que nuestro modelo complicaba excesivamente la especificación de esta. Por lo tanto establecimos la relación entre Persona y Matricular, lo que facilitó las cosas.

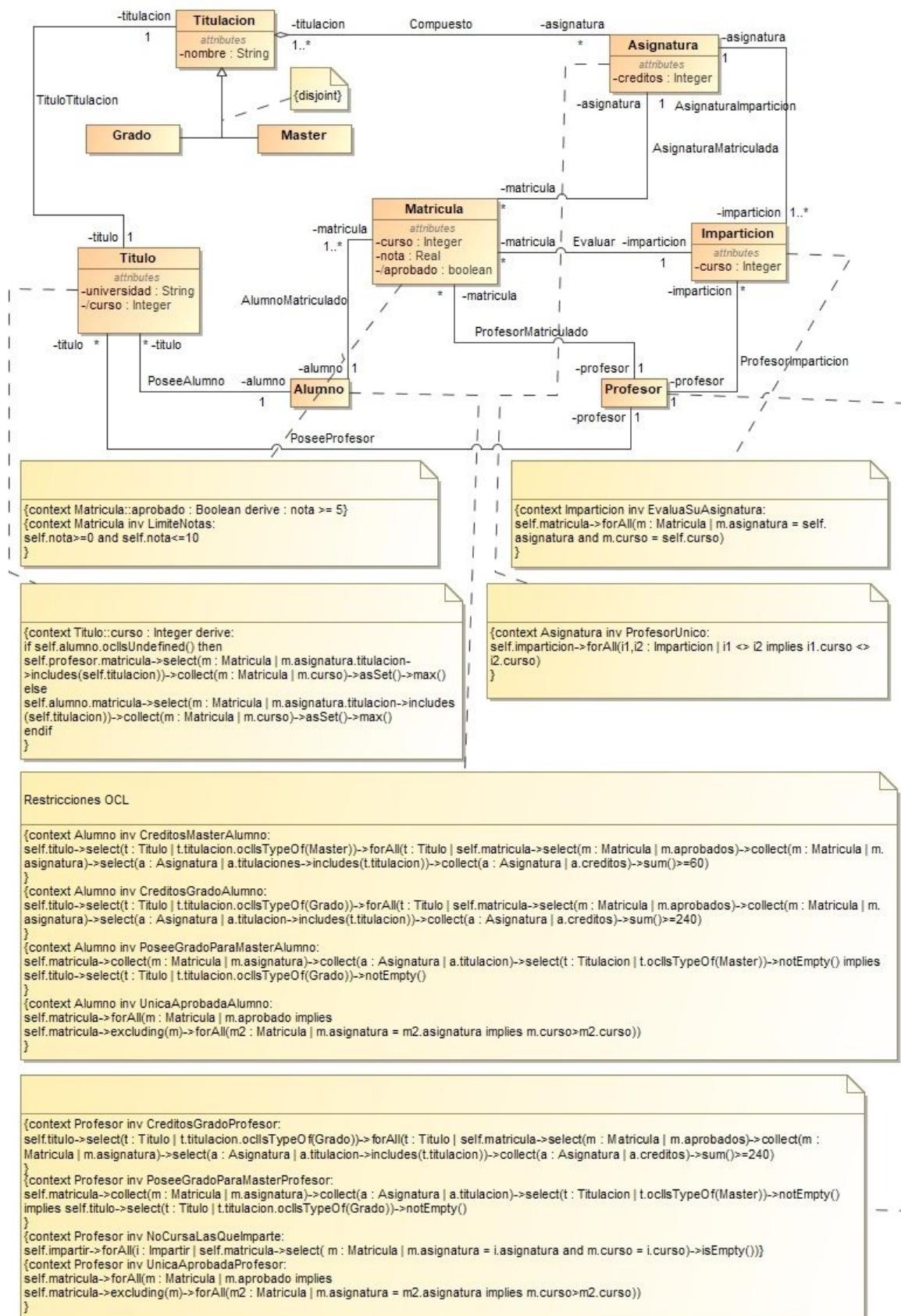


- Restricción inferida: Durante la especificación de restricciones nos dimos cuenta de que la restricción: “Un profesor no puede evaluarse a sí mismo”, listada entre las restricciones que encontramos para nuestro modelo, se puede inferir de otras dos que son: “Un profesor no puede matricularse e impartir la misma asignatura durante un curso” y “Un profesor sólo puede evaluar matrículas de asignaturas que imparte en ese curso”; por esta razón no se encuentra especificada en el conjunto de restricciones.
- Derivación en USE: Cuando pasamos el modelo a USE tuvimos el problema de que no pudimos especificar atributos derivados, debido a que en la documentación no aparece nada o al menos nosotros no fuimos capaces de encontrarlo.

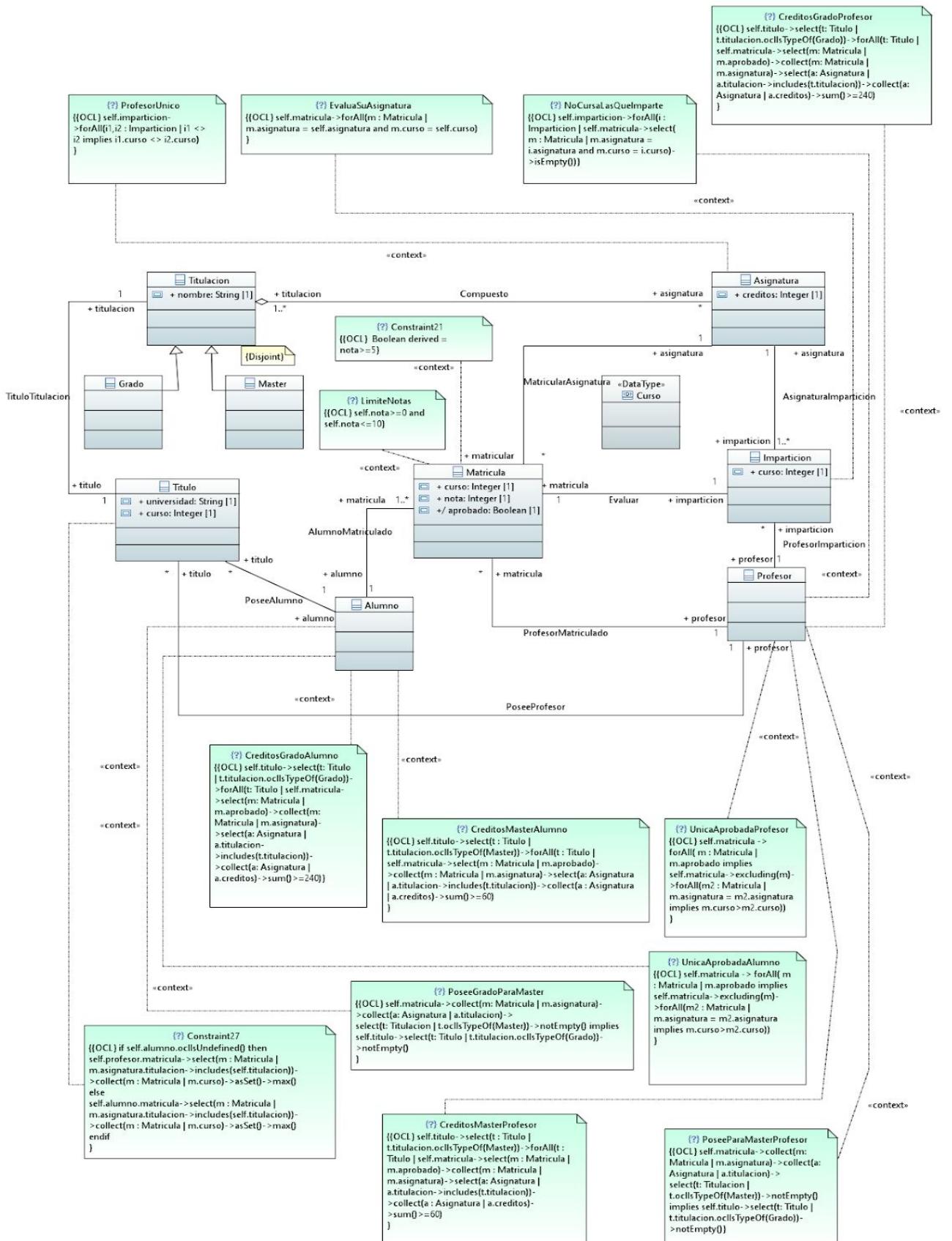
**Siete**

# PRÁCTICA 1

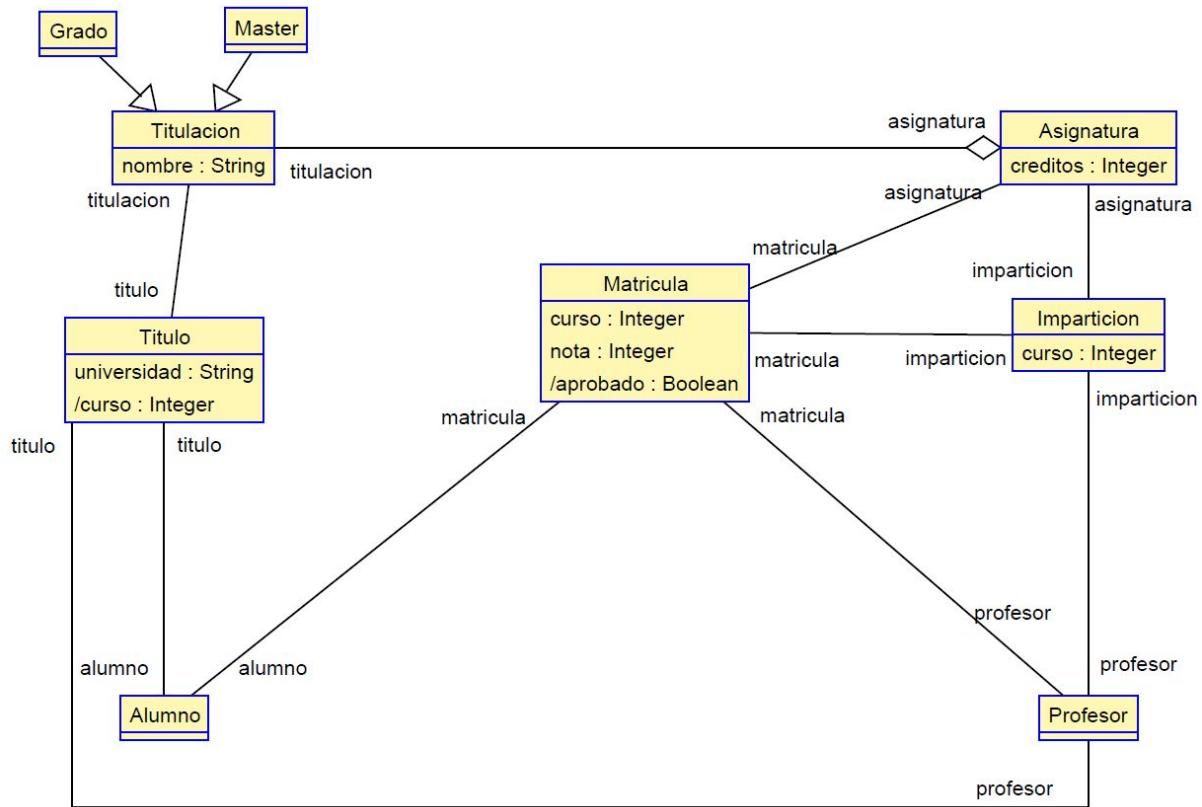
## Diagrama en Magicdraw:



## Diagrama en Papyrus:



## Diagrama en USE:



## Descripción:

En este modelo UML de la Universidad hemos creado la clase Titulación, que es una clase abstracta de la que derivan Grado y Máster, la cual se relaciona con la clase Asignatura mediante una agregación, ya que una misma asignatura puede impartirse en distintas titulaciones y una asignatura sigue existiendo en el caso de que la titulación desaparezca.

La clase Matricular es otra clase asociación de la relación entre Asignatura y Alumno. Un alumno puede matricularse en muchas asignaturas. De la clase Alumno deriva Profesor que puede impartir una asignatura en un curso mediante la clase asociación Impartir.

## Restricciones:

- Si la nota es mayor o igual a 5 es aprobado:

```

context Matricula::aprobado : Boolean derive:
 nota >= 5

```

- Si un alumno aprueba una asignatura, dicho alumno no puede matricularse otra vez de esa asignatura:

```

context Alumno inv UnicaAprobadaAlumno:
 self.matricula->forAll(m : Matricula | m.aprobado implies
 self.matricula->excluding(m)->forAll(m2 : Matricula | m.asignatura =
 m2.asignatura implies m.curso>m2.curso))

```

- Si un profesor aprueba una asignatura, dicho profesor no puede matricularse otra vez de esa asignatura:

```
context Profesor inv UnicaAprobadaProfesor:
self.matricula->forAll(m : Matricula | m.aprobado implies
self.matricula->excluding(m)->forAll(m2 : Matricula | m.asignatura =
m2.asignatura implies m.curso>m2.curso))
```

- Nota comprendida entre los valores 0 y 10:

```
context Matricula inv LimiteNotas:
self.nota>=0 and self.nota<=10
```

- Un profesor no puede matricularse de la asignatura que está impartiendo en ese curso:

```
context Profesor inv NoCursaLasQueImparte:
self.imparticion->forAll(i : Imparticion | self.matricula->select(m :
Matricula | m.asignatura = i.asignatura and m.curso =
i.curso)->isEmpty())
```

- Hay un total de 240 créditos de las asignaturas de Grado del alumno:

```
context Alumno inv CreditosGradoAlumno:
self.titulo->select(t : Titulo |
t.titulacion.oclIsTypeOf(Grado))->forAll(t : Titulo |
self.matricula->select(m : Matricula | m.aprobados)->collect(m :
Matricula | m.asignatura)->select(a : Asignatura |
a.titulacion->includes(t.titulacion))->collect(a : Asignatura |
a.creditos)->sum()>=240)
```

- Hay un total de 240 créditos de las asignaturas de Grado del profesor:

```
context Profesor inv CreditosGradoProfesor:
self.titulo->select(t : Titulo |
t.titulacion.oclIsTypeOf(Grado))->forAll(t : Titulo |
self.matricula->select(m : Matricula | m.aprobados)->collect(m :
Matricula | m.asignatura)->select(a : Asignatura |
a.titulacion->includes(t.titulacion))->collect(a : Asignatura |
a.creditos)->sum()>=240)
```

- Hay un total de 60 créditos de las asignaturas de Master del alumno:

```
context Alumno inv CreditosMasterAlumno:
self.titulo->select(t : Titulo |
t.titulacion.oclIsTypeOf(Master))->forAll(t : Titulo |
```

```

self.matricula->select(m : Matricula | m.aprobados)->collect(m :
Matricula | m.asignatura)->select(a : Asignatura |
a.titulaciones->includes(t.titulacion))->collect(a : Asignatura |
a.creditos)->sum(>=60)

```

- Hay un total de 60 créditos de las asignaturas de Master del profesor:

```

context Profesor inv CreditosMasterProfesor:
self.titulo->select(t : Titulo |
t.titulacion.oclIsTypeOf(Master))->forAll(t : Titulo |
self.matricula->select(m : Matricula | m.aprobados)->collect(m :
Matricula | m.asignatura)->select(a : Asignatura |
a.titulaciones->includes(t.titulacion))->collect(a : Asignatura |
a.creditos)->sum(>=60)

```

- Un alumno no puede matricularse de un Master si no posee un título de Grado:

```

context Alumno inv PoseeGradoParaMasterAlumno:
self.matricula->collect(m : Matricula | m.asignatura)->collect(a :
Asignatura | a.titulacion)->select(t : Titulacion |
t.oclIsTypeOf(Master))->notEmpty() implies self.titulo->select(t :
Titulo | t.titulacion.oclIsTypeOf(Grado))->notEmpty()

```

- Un profesor no puede matricularse de un Master si no posee un título de Grado:

```

context Profesor inv PoseeGradoParaMasterProfesor:
self.matricula->collect(m : Matricula | m.asignatura)->collect(a :
Asignatura | a.titulacion)->select(t : Titulacion |
t.oclIsTypeOf(Master))->notEmpty() implies self.titulo->select(t :
Titulo | t.titulacion.oclIsTypeOf(Grado))->notEmpty()

```

- Una evaluación debe hacerse para las matrículas de la misma asignatura y el mismo curso de la impartición.

```

context Imparticion inv EvaluaSuAsignatura:
self.matricula->forAll(m : Matricula | m.asignatura =
self.asignatura and m.curso = self.curso)

```

- Cada asignatura está impartida por un sólo profesor cada curso.

```

context Asignatura inv ProfesorUnico:
self.imparticion->forAll(i1,i2 : Imparticion | i1 <> i2 implies
i1.curso <> i2.curso)

```

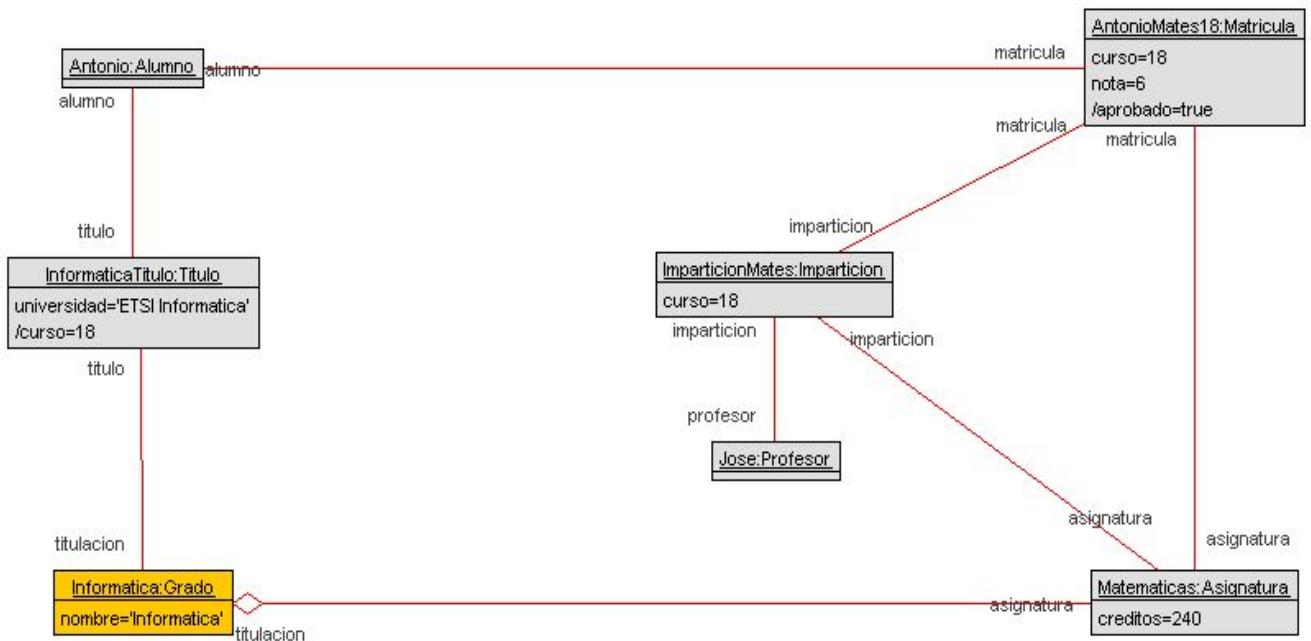
- El curso del título ha de corresponderse al de la última asignatura aprobada que sumase los créditos necesarios para obtenerlo.

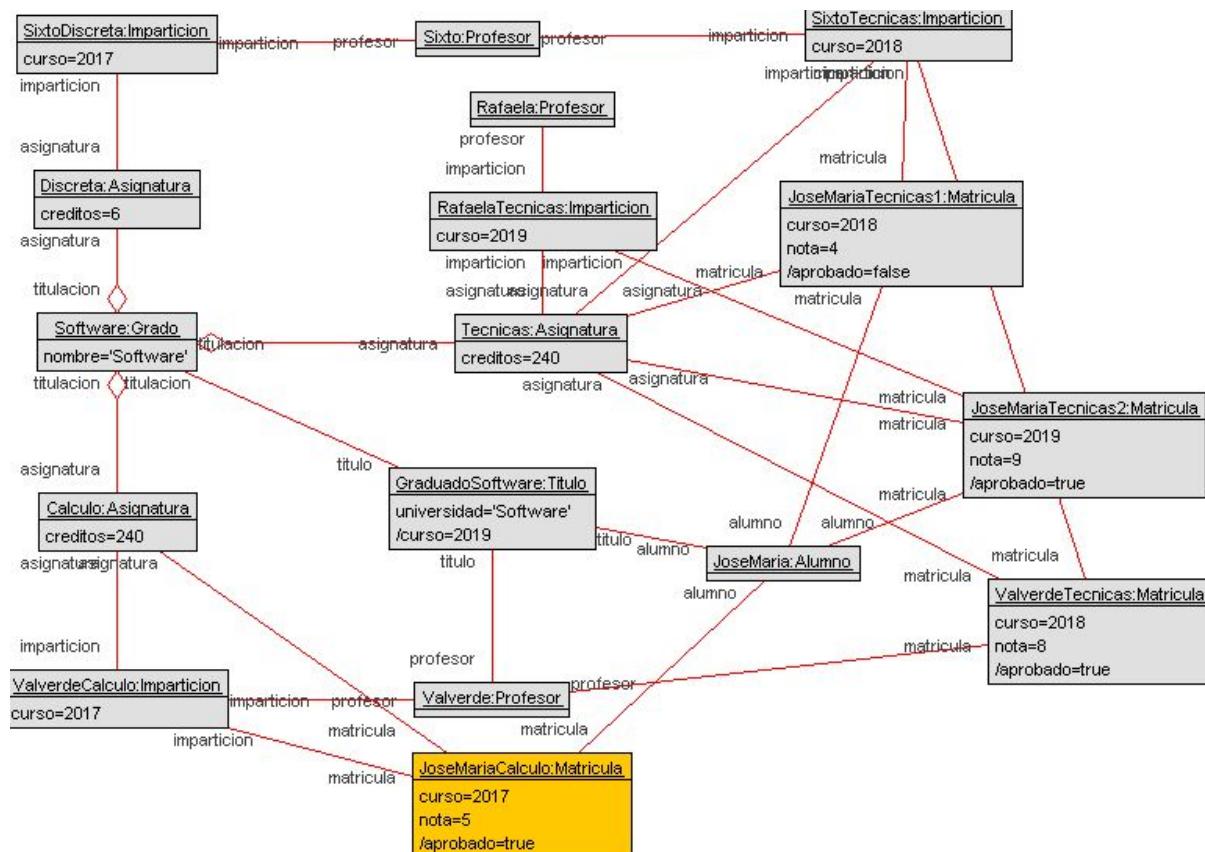
```

context Titulo::curso : Integer derive:
if self.alumno.oclIsUndefined() then
 self.profesor.matricula->select(m : Matricula |
m.asignatura.titulacion->includes(self.titulacion))->collect(m :
Matricula | m.curso)->asSet()->max()
else
 self.alumno.matricula->select(m : Matricula |
m.asignatura.titulacion->includes(self.titulacion))->collect(m :
Matricula | m.curso)->asSet()->max()
endif

```

### Ejemplos en diagramas de objetos:



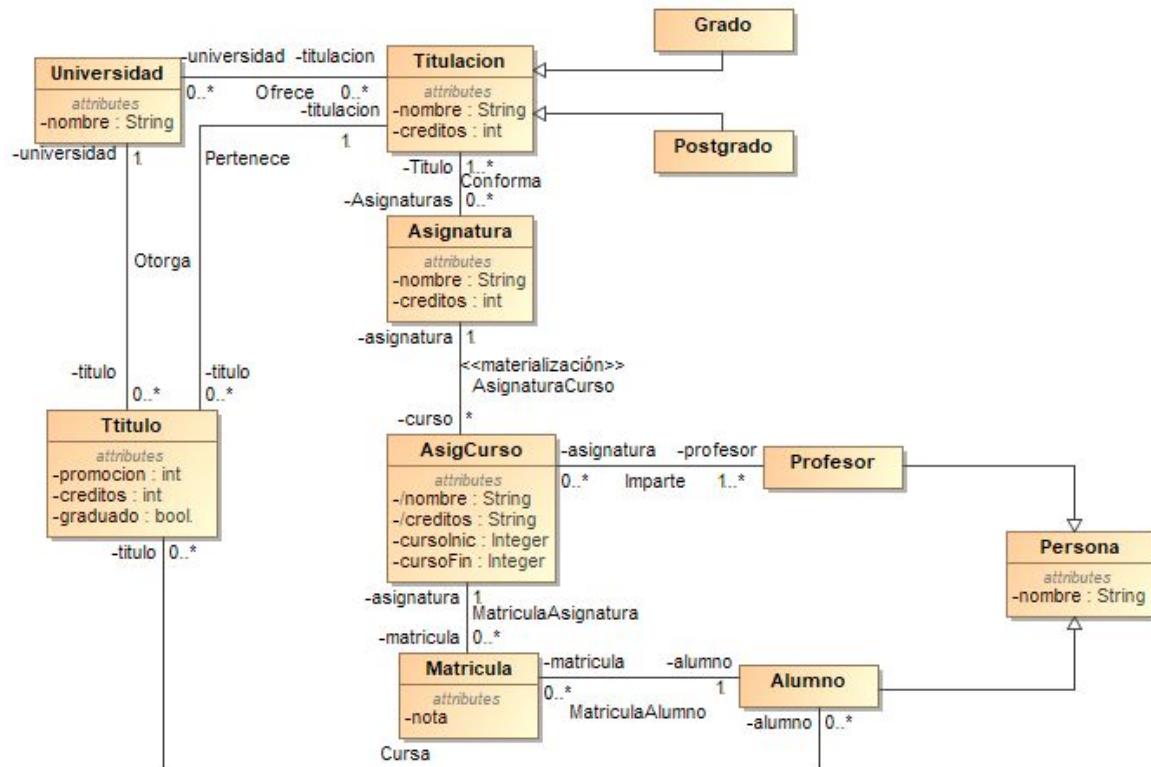


Ocho

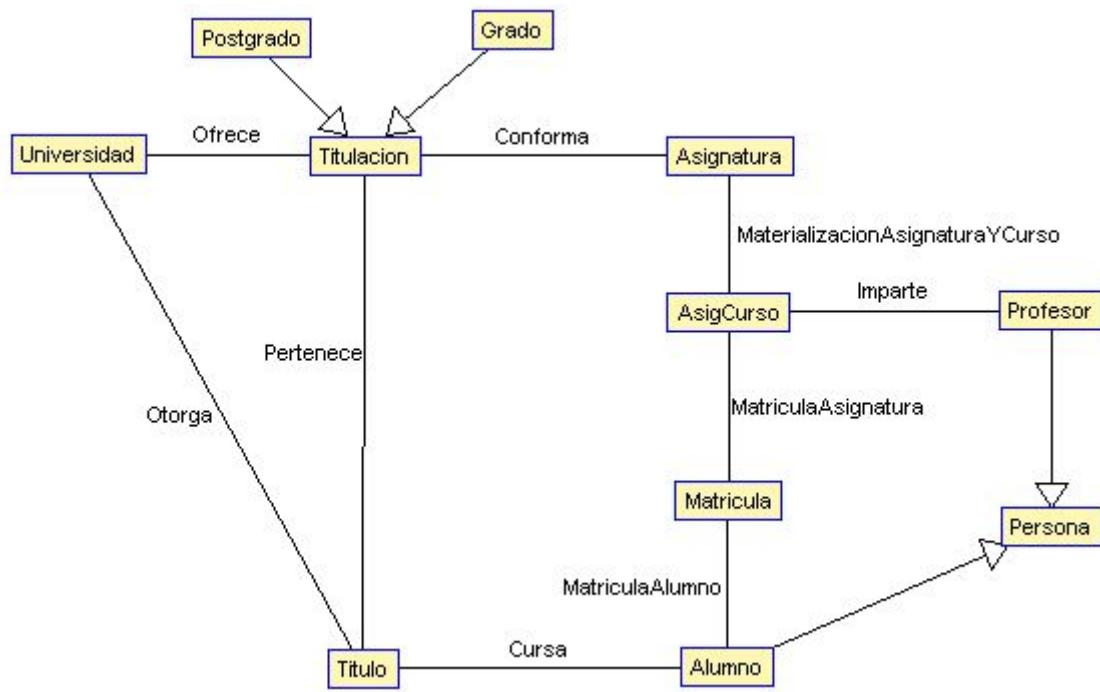
# Practica 1

## Modelado y Diseño de Software

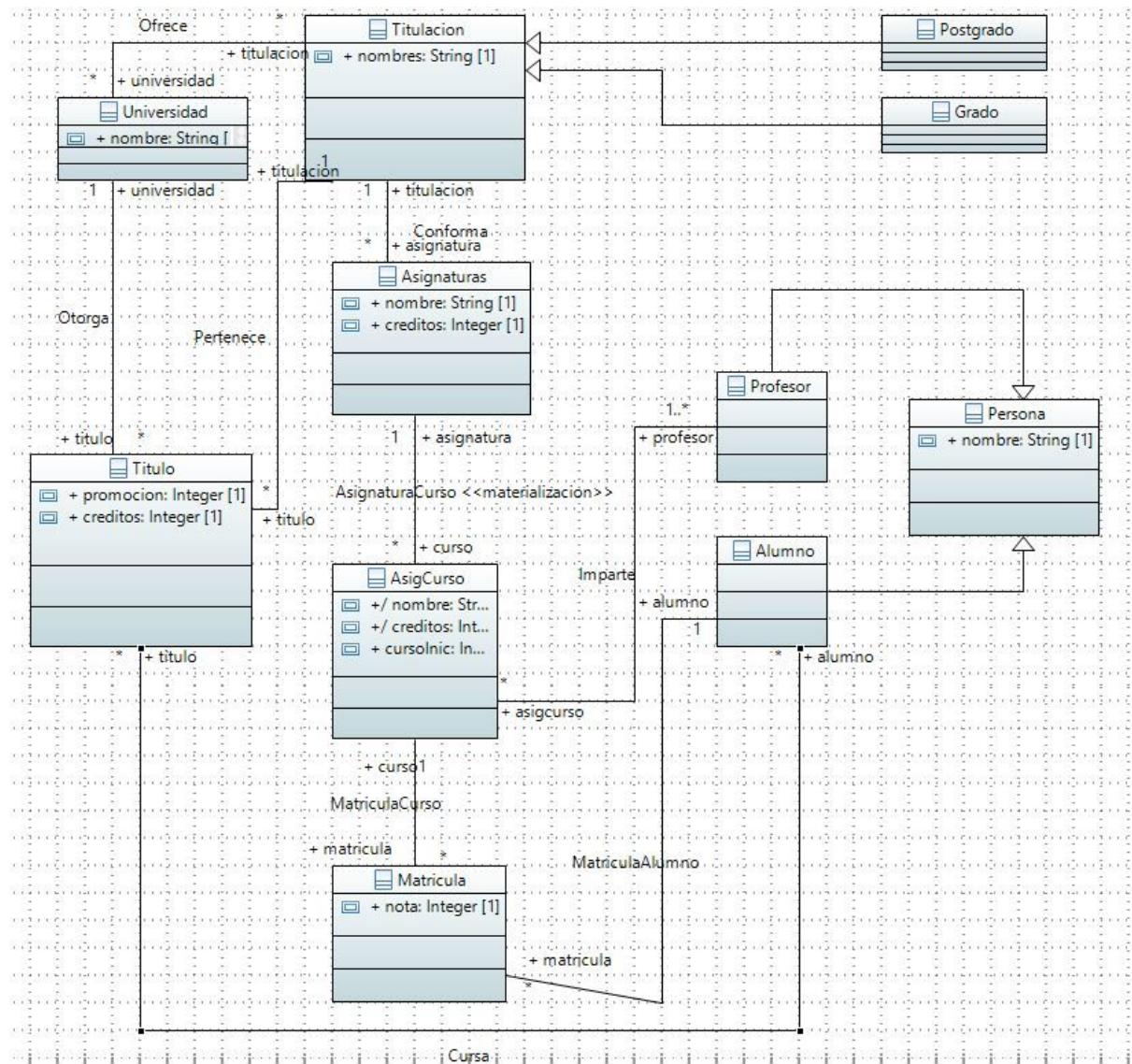
### MagicDraw:



USE:



## Papyrus:



Las entidades más relevantes son Universidad, Título, Titulación, Grado, Postgrado, Asignaturas, AsigCurso, Matrícula, Profesor, Alumno y Persona.

Las entidades se relacionan como se puede observar en los modelos.

## Restricciones:

**context Alumno inv NolparteMatriculado:**

**self.matricula.asignatura.profesor -> not includes(self)**

Un profesor no puede estar matriculado en la asignatura que imparte. Esta versión recorre desde Alumno hasta profesor, así no tenemos conjuntos dentro de conjuntos

**//matriculas -> 1 asignatura -> 1 profesor**

```
context Profesor inv NolmparteMatriculado:
self.asignatura.matricula.alumno -> not includes(self)
Esta versión recorre *asignaturas -> * matriculas -> 1alumno
```

```
context AsigCurso::nombre: String
derive: self.asignatura.nombre
```

```
context AsigCurso::creditos: int
derive: self.asignatura.creditos
```

```
context Titulo::promocion: int
derive:
```

**MAX(titulo -> titulación -> asignatura -> asignatura curso = titulo -> alumno -> matricula ->AsignaturaCurso**

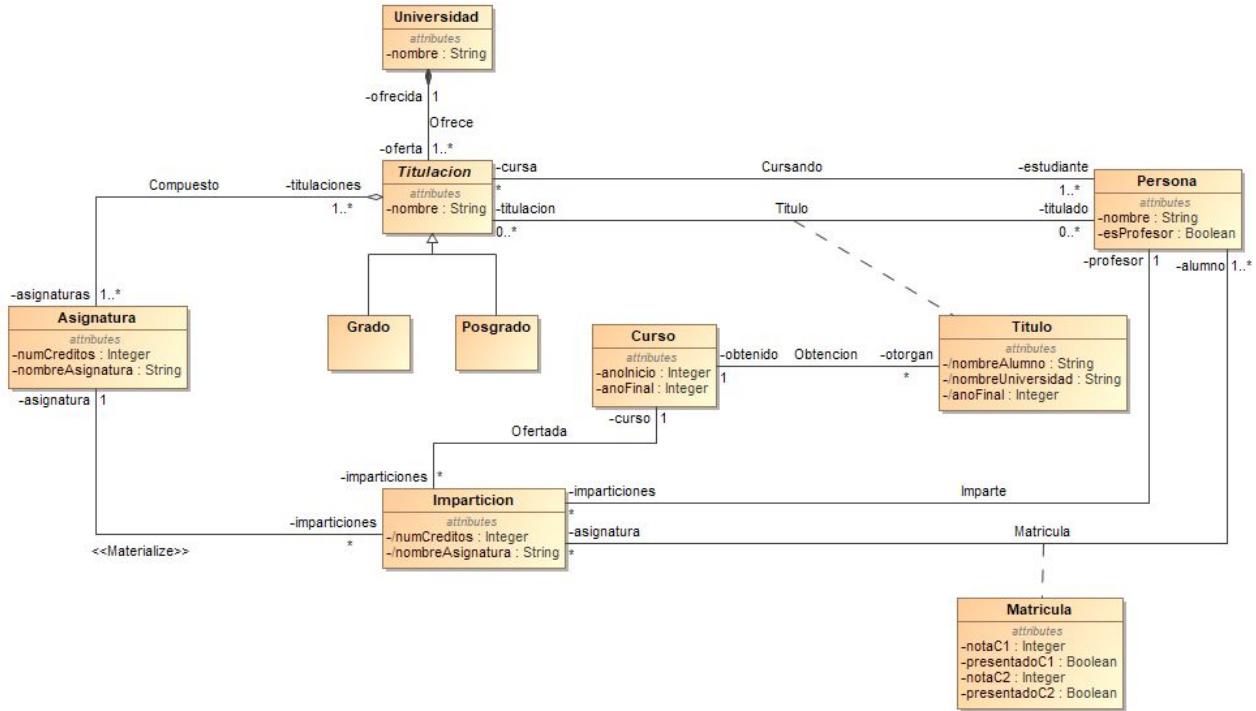
```
post: result->forAll(elem | self.titulo.asignaturas.curso->includes(elem) and
self.alumno.matricula -> select(m | m.nota >=5).asignatura ->includes(elem))
```

Los profesores evalúan a los alumnos matriculados en las asignaturas que imparten, asignándoles una nota entre 0 y 10. A partir de una calificación de 5 la asignatura se considera aprobada.

Nueve

# MODELO UNIVERSIDAD

En este modelo se describe la estructura del sistema universitario. A continuación, se muestra una captura del modelo en MagicDraw:



Una universidad ofrece una o más titulaciones siendo estas exclusivas de dicha universidad. Esas titulaciones pueden ser de grado o de posgrado.

Una titulación está formada por al menos una asignatura que aporta un número de créditos. La asignatura no es exclusiva de la titulación, ya que puede aparecer en más de una (no puede existir una asignatura que no pertenezca a ninguna titulación).

Una impartición es una materialización de una asignatura para un curso en concreto, un profesor que la imparte y un grupo de alumnos.

Una persona debe estar cursando al menos una titulación para considerarse un estudiante. Al alcanzar el número de créditos necesarios para acabar la titulación, este recibe un título y pasa a considerarse un titulado. El título contiene el nombre del alumno, el de la universidad y el curso en el que aprobó la última asignatura que necesitaba para obtenerlo.

También hemos añadido algunas restricciones al modelo:

- Si un alumno ha aprobado una asignatura no puede matricularse de nuevo en la misma.
  - Las notas que se le asigna al alumno debe estar entre 0 y 10.
  - La nota que se le asigna a un alumno que no se ha presentado al examen es 0.
  - Un mismo alumno no puede estar matriculado 2 veces en la misma asignatura el mismo curso.
  - Un profesor no puede cursar una asignatura que sea impartida por él mismo.
  - No puede haber dos imparticiones de una misma asignatura en un mismo curso.
  - Los años de impartición no serán nulos.
  - Cualquier persona debe ser al menos estudiante o profesor.

- Un alumno podrá cursar una titulación de posgrado siempre y cuando haya obtenido algún título de grado.
- Un alumno tiene que haber superado 240 créditos para obtener un título de grado y 60 créditos para obtener un título de posgrado.
- Una titulación debe tener el mínimo de créditos (240 para grados, 60 para postgrados) para poder existir.
- Los alumnos de una titulación deben estar matriculados en alguna asignatura de la misma.
- Los alumnos matriculados en una asignatura deben estar cursando alguna de las titulaciones a las que pertenece la asignatura.
- Si un alumno aprueba una asignatura en primera convocatoria, éste no podrá presentarse a la segunda convocatoria.
- Los cursos deben contener años de inicio y fin lógicos, es decir, la diferencia entre ambos debe ser de un año y no pueden ser menores que 0.

A lo largo del planteamiento y desarrollo del modelo, hemos tenido diferentes ideas para enfocarlo. Una de ellas es considerar profesor como un subconjunto de estudiantes, ya que como un profesor podría estar matriculado de una asignatura, consideramos que un profesor era un estudiante que daba clases. Sin embargo, dado que un profesor puede no estar matriculado en ninguna asignatura, nos obligaba a poner la relación de estudiante y asignaturas como 0 o muchos y podríamos llegar a tener estudiantes que no estén matriculados en ninguna asignatura también y no tenía sentido. Por ello, se han creado dos roles en persona, una de estudiante y otra de profesor.

También pensamos en poner la clase “Impartición” como una clase de asociación entre asignatura y estudiante; sin embargo se descartó cuando eliminamos la clase “Estudiante” y pusimos “Persona”. Además, frente a la materialización, ésta solución aumentaba la complejidad de las restricciones en OCL.

Además de esas dos opciones, también estuvimos pensando acerca de materializar la asignatura en forma de curso, pero ésto creaba 1 curso por cada asignatura cuando realmente el curso es común en todas las asignaturas de todas las titulaciones.

Otra de las opciones que estuvimos barajando fue la de crear Título como una clase independiente, sin embargo, de ese modo, hubiéramos necesitado crear una relación entre Título-Titulación, Título-Curso y Título-Persona y al ver que con una clase de asociación se simplificaban las restricciones, decidimos utilizar el segundo método.

Además del resto de opciones, se pensó en tener simplemente una nota por cada matrícula, sin embargo, con esta idea no podíamos saber cuántas convocatorias ha gastado un estudiante, por ello decidimos añadir las notas de cada convocatoria, además de un booleano para saber si se había presentado o no a dicha convocatoria.

A pesar de todas las ventajas respecto al resto de opciones, el modelo presenta un par de desventajas, como son el uso de 2 integer y 2 booleanos para representar las notas y si se ha presentado o no en ambas convocatorias, además de ello, no podemos saber qué persona es profesor, por ello se añadió un booleano de esProfesor. Esto último se podría solucionar si se crearan 2 clases de roles subrogados respecto la clase Persona, creando 2 roles distintos y diferenciados, pero tendríamos problemas para algunas restricciones, así que se decidió dejar como está

**Diez**

# Modelo de Universidad

## 1. Introducción

En esta memoria se van a describir los modelos que hemos realizado en cada una de las herramientas y las restricciones que hemos incluido. Además, se detallarán las entidades y las relaciones entre las mismas.

Nuestro modelo se compone de siete entidades, de las cuales tres de ellas heredan de otras entidades, de las relaciones que unen estas entidades y de las restricciones que tiene cada clase y que son necesarias para asegurar la integridad del modelo y que lo hacen satisfacible.

También tiene los siguientes tipos de datos:

- **TipoMiembro:** es un *enumeration* que puede ser Profesor o Alumno.
- **TipoTitulación:** es un *enumeration* que puede ser Grado o Posgrado.
- **Curso:** es un *dataType* que contiene los datos del año académico.

En los próximos apartados se hará una descripción de cada entidad, relación y restricción que tiene nuestro modelo del sistema.

## 2. Descripción de entidades

- **Universidad:** contiene como atributo el nombre de la universidad.
- **Titulación:** tiene como atributos *nombre*, *creditosMax*, que es el número de créditos que hay que aprobar para tener esa titulación, y *tipoTitulacion* que puede ser grado o posgrado. De esta clase heredan las clases de **Grado** y **Posgrado**.
- **Grado:** tiene como restricción que el número máximo de créditos sea 240 y que el tipo de titulación sea grado.
- **Posgrado:** tiene como restricción que el número máximo de créditos sea 60 y que el tipo de titulación sea grado.
- **Asignatura:** tiene como atributos el número de créditos de esa asignatura, *creditos*, y el curso en el que se imparten, *curso*, de tipo Curso.
- **Miembro:** tiene el atributo *tipoMiembro* que puede ser Profesor o Alumno.
- **Profesor:** hereda de la clase Miembro y tiene la restricción de que el tipoMiembro es Profesor.

## 3. Descripción de relaciones

- **Universidad-Titulacion:** una universidad “ofrece” una o muchas titulaciones.
- **Titulacion-Asignatura:** una titulación “se compone” de asignaturas.
- **Profesor-Asignatura:** un profesor “imparte” cero o muchas asignaturas.
- **Miembro-Asignatura:** cero o muchos miembros “se matriculan” en una o muchas asignaturas. Si el miembro es un profesor, no se puede matricular en una asignatura que imparte, esto se pondrá como restricción del sistema.

- **Miembro-Asignatura:** cero o un alumno/profesor “aprueba” cero o muchas asignaturas. En este caso, el alumno/profesor no podrá aprobar una asignatura en la que no se ha matriculado, esto también se pondrá como restricción del sistema.
- **Miembro-Titulación:** uno o muchos miembros “obtienen” cero o muchas titulaciones. En el título deben aparecer el nombre de la universidad y el curso en el que se aprobaron los últimos créditos necesarios para obtenerlo, de ahí que aparezca la clase *obtener*.
- **Profesor-Miembro:** uno o muchos profesores “evalúan” a uno o muchos alumnos. Al evaluar al alumno, el profesor debe poner una nota para la asignatura en la que le da clase, de ahí la tabla *Evaluar*. Esta relación tiene la restricción de que el profesor debe impartir la asignatura en la que el alumno se ha matriculado para poder ponerle la nota.

#### 4. Descripción de restricciones

- **Un alumno solo podrá aprobar una asignatura una vez, no puede matricularse de asignaturas que ya ha aprobado en cursos anteriores:**

```
context Miembro
inv noMatriculaSiAprobado: self.asignaturaApr->intersection(self.asignaturaMatr)->size() = 0
```

- **Los profesores no pueden matricularse en aquellas asignaturas que imparten:**

```
context Profesor
inv noMatriculaSiEsProfesor: self.asignatura->intersection(self.profesor.asignaturaMatr)->size() = 0
```

- **A partir de la calificación de 5 la asignatura se considera aprobada:**

```
context evalua
inv calificacionAprobada: self.calificacion >= 5
```

- **Para obtener un título de grado, hay que tener aprobados al menos 240 créditos:**

```
context Miembro
inv tituloGrado: self.asignaturaApr->collect(creditos)->sum() >= 240
```

- **Para obtener un título de posgrado, hay que tener aprobados al menos 60 créditos:**

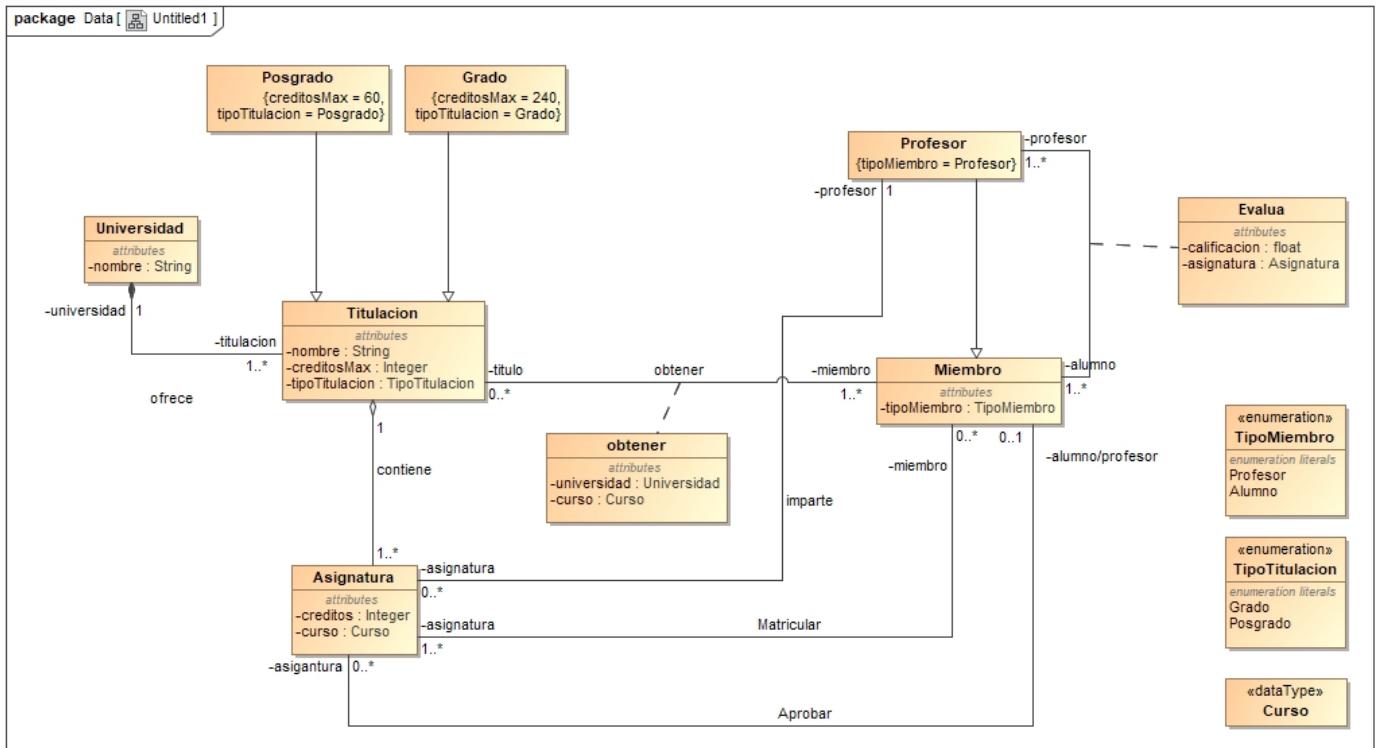
```
context Miembro
inv tituloPosgrado: self.asignaturaApr->collect(creditos)->sum() >= 60
```

- **Para poder matricularse en una asignatura de posgrado, el alumno debe tener un título de grado:** esta restricción debería ser que si el alumno se matriculó antes en un grado y tiene 240 créditos o más, se puede matricular en una asignatura del posgrado.

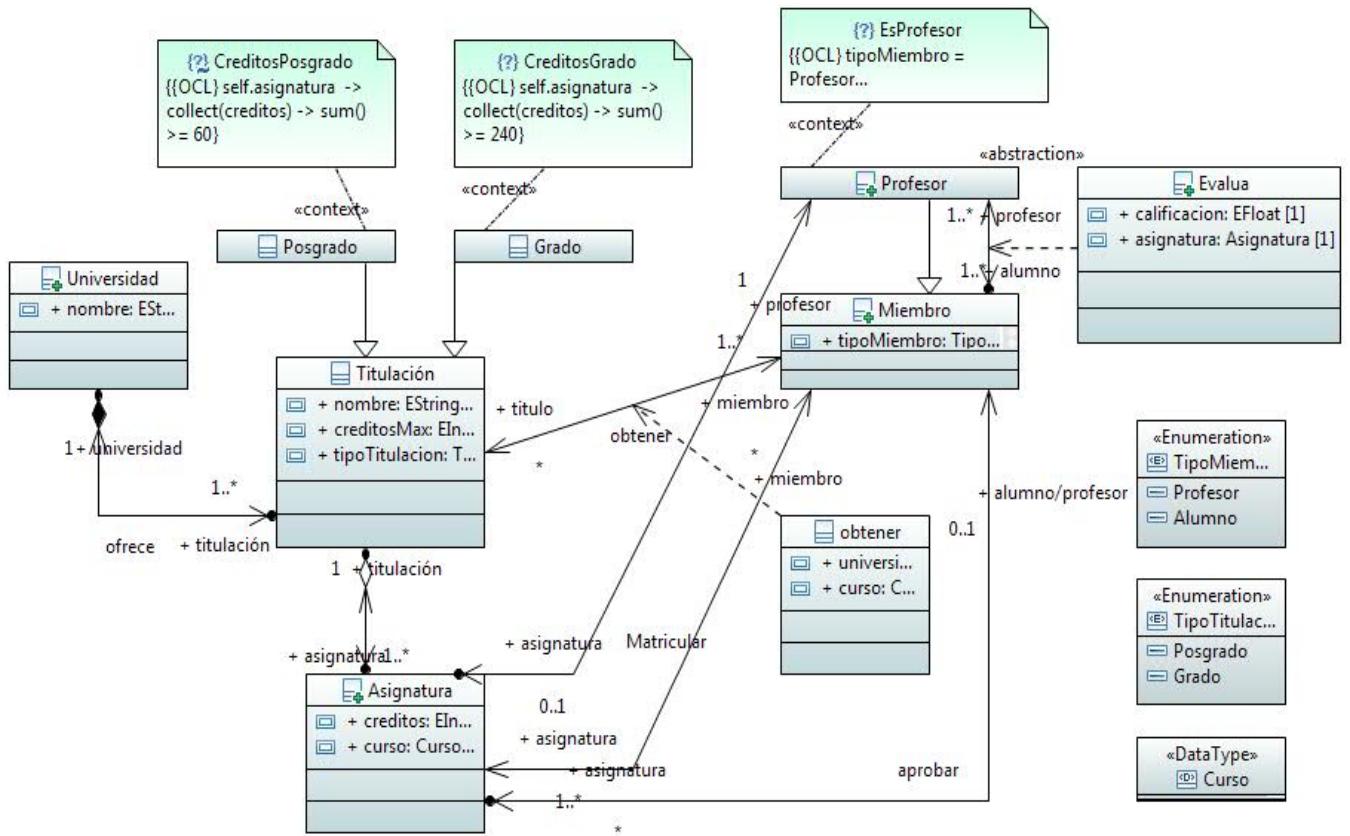
- **El profesor debe impartir la asignatura en la que el alumno se ha matriculado para poder ponerle la nota:**

```
context evalua
inv profesorDaClaseAlumno: self.profesor.asignatura = self.miembroEva.asignaturaMatr
```

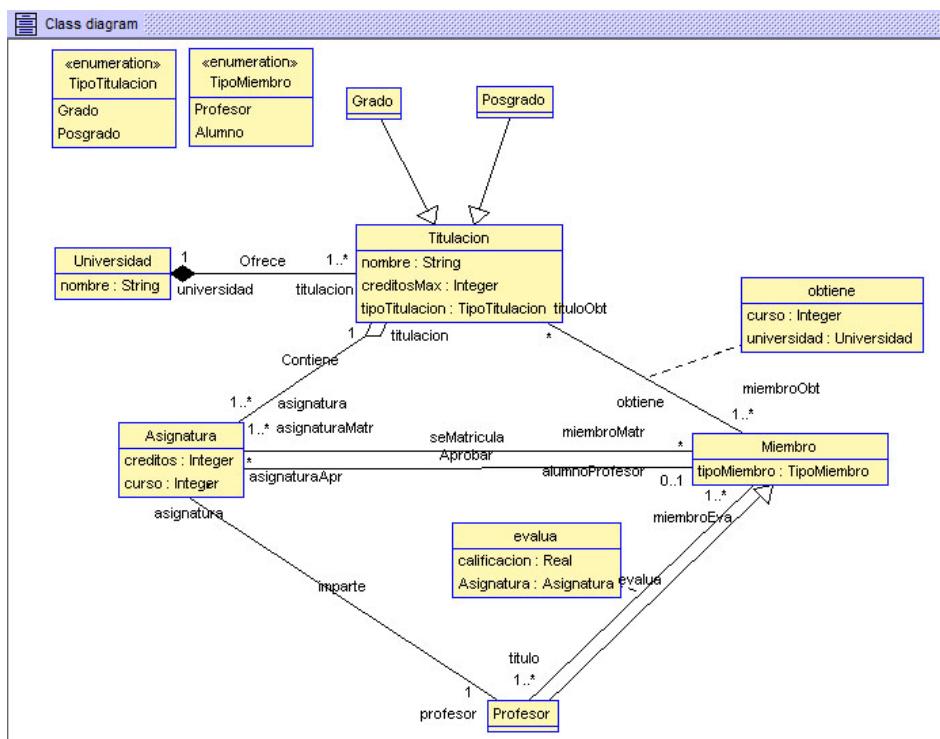
## 5. Modelo en Magic Draw



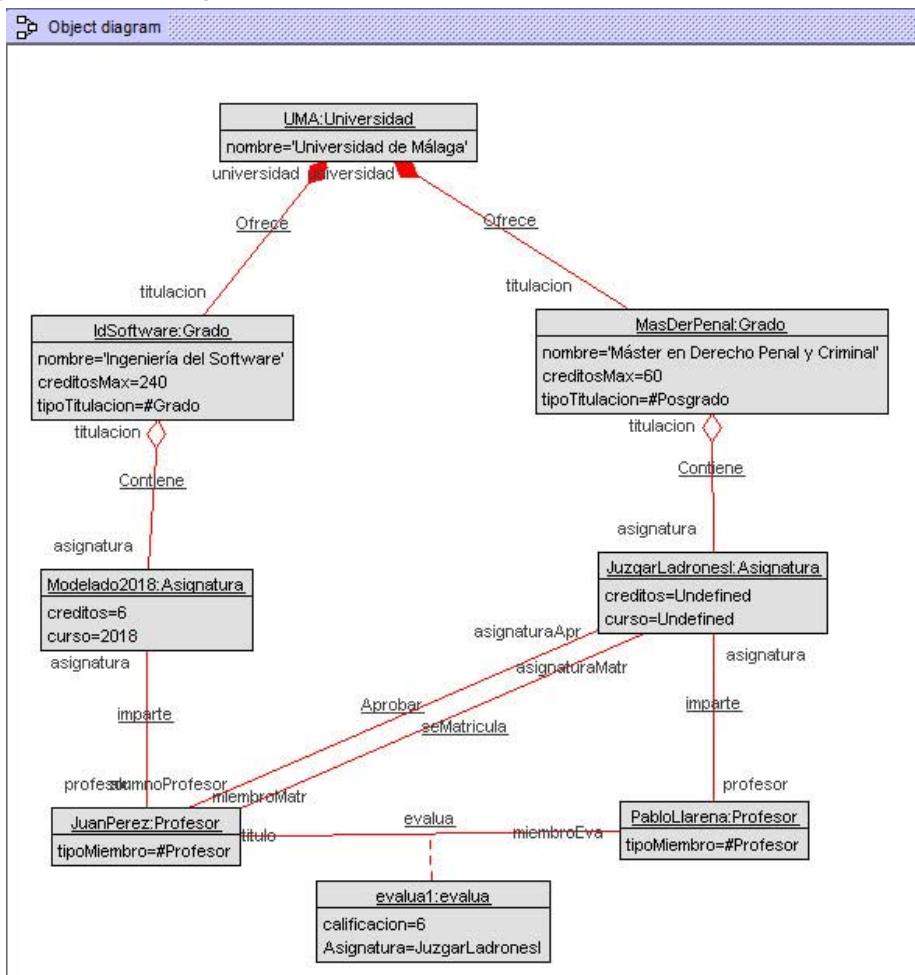
## 6. Modelo en Papyrus



## 7. Modelo en USE

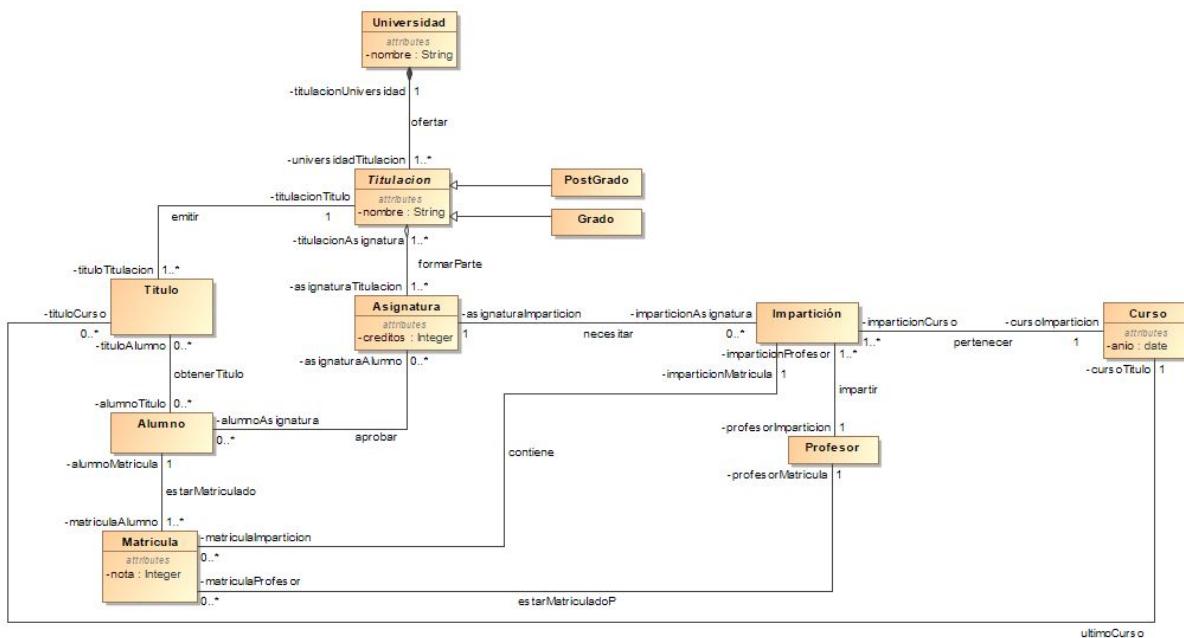


## 8. Diagrama de objetos



Once

## DISEÑO DEL MODELO



En nuestro modelo podemos ver que la clase **Universidad** tiene una relación de *composición* con la clase **Titulación**, ya que si desaparece Universidad no tiene mucho sentido tener Titulaciones.

La clase Universidad se relaciona con la clase Titulación mediante la relación **Ofertar**. Respecto a la multiplicidad entre ambas clases sabemos que en una Universidad se ofertan *una o muchas Titulaciones* y una Titulación sólo puede ser ofertada por *una Universidad*.

La clase **Titulación** es una *generalización* de la clase **Grado** y **PostGrado**, ya que una titulación o es Grado o es PostGrado. Ambas clases heredan el atributo *nombre* de la clase Titulación, al ser ésta una *clase abstracta*.

La clase **Titulación** tiene una relación de *agregación* con la clase **Asignatura**, ya que si desaparece la Titulación no tienen porqué desaparecer Asignatura, ya que hay Asignaturas que pueden pertenecer a varias Titulaciones.

La clase **Titulación** se relaciona con la clase **Asignatura** mediante la relación **FormarParte**. Respecto a la multiplicidad entre ambas clases tenemos que una Titulación se compone de *una o muchas Asignaturas* y una Asignatura forma parte de *una o muchas Titulaciones*.

La clase **Titulación** se relaciona con la clase **Título** mediante la relación **Emitir**. Respecto a la multiplicidad entre ambas clases tenemos que una Titulación emite *uno o muchos Títulos* y un Título es emitido por *una Titulación*.

La clase **Título** se relaciona con la clase **Alumno** mediante la relación **obtenerTítulo**. Un Título puede ser obtenido por *ceros o muchos Alumnos* y un Alumno puede obtener *ceros o muchos Títulos*. Es cero la obtención de un título debido a que si se crea una nueva titulación, el título no se obtendrá hasta pasados todos los años que dure esa titulación.

La clase **Titulo** también se relaciona con la clase **Curso** mediante la relación **UltimoCurso**, ya que tenemos que saber el Curso en el que se obtiene. Sabemos que un Título se obtiene en *un solo Curso* y un Curso puede ser el último curso de *cero o muchos Títulos*.

En la clase **Asignatura** vamos a tener un atributo llamado *créditos*, donde vamos a poder guardar el número de créditos de dicha asignatura y otro atributo llamado *id*, que va a ser nuestro identificador para cada asignatura.

En la clase **Curso** vamos a guardar el año de un Curso, mediante el atributo *año*.

La clase **Asignatura** se relaciona con la clase **Alumno** mediante la relación **Aprobar**. Un Alumno puede aprobar *cero o muchas Asignaturas* y una Asignatura puede ser aprobada por *cero o muchos Alumnos*.

La clase **Asignatura** se relaciona también con la clase **Impartición** mediante la relación **Necesitar**. Una Impartición necesita *una sola Asignatura* y una Asignatura puede ser necesitada en *cero o muchas Imparticiones*.

La clase **Alumno** se relaciona con la clase **Matrícula** mediante la relación **EstarMatriculado**. Un Alumno puede estar matriculado en *una o muchas Matrículas* y en una Matrícula sólo puede estar *matriculado un Alumno*.

La clase **Matrícula** va a tener un atributo *nota* que va a guardar la calificación del alumno para dicha Matrícula.

La clase **Matrícula** se relaciona también con la clase **Impartición** mediante la relación **Contener**. Una Impartición puede estar contenida en *una o muchas Matrículas* y una Matrícula puede contener sólo *una Impartición*.

La clase **Impartición** se relaciona con la clase **Profesor** mediante la relación **Impartir**. Una Impartición es impartida por *un solo Profesor* y un Profesor puede impartir *una o muchas Imparticiones*, ya que para ser Profesor mínimo tiene que tener una Impartición.

La clase **Impartición** también se relaciona con la clase **Curso** mediante la relación **Pertenecer**. Una Impartición pertenece a *sólo un Curso* y a un Curso pueden pertenecer *una o muchas Imparticiones*, ya que no tiene mucho sentido tener un Curso sin Imparticiones.

La clase **Profesor** se relaciona con la clase **Matrícula** mediante la relación **EstarMatriculadoP**, ya que tenemos que contemplar también la posibilidad de que un Profesor se pueda matricular también. Un Profesor puede estar matriculado en *cero o muchas matrículas* y una matrícula es sólo de *un Profesor*.

## RESTRICCIONES

### Universidad:

Restricción para controlar que el nombre de la universidad no sea nulo.

- context Universidad inv NombreUniversidad:
 

```
Universidad.allInstances()->forAll(u
 u.nombre.oclIsUndefined() = false)
```

Restricción para que el nombre de la universidad se único.

- inv NombreUniversidadUnico:
 

```
Universidad.allInstances -> collect(nombre)-> size()
 =Universidad.allInstances -> collect(nombre) ->
 asSet()->size()
```

### Titulación:

Restricción para que las asignaturas no repitan “id”, es decir, cada asignatura tiene asignado su propio identificador.

- context Titulacion inv idUnico:
 

```
self.asignaturaTitulacion->collect(id)->size()=
 self.asignaturaTitulacion -> collect(id) -> asSet()->size()
```

Restricción para que el campo de nombre de nuestra titulación no sea nulo.

- context Titulacion inv NombreTitulacionUndefined:
 

```
Titulacion.allInstances()->forAll(t|t.nombre.oclIsUndefined()=
 false)
```

### Asignatura:

Restricción que nos obliga a que las asignaturas tengan “id”.

- context Asignatura inv IDNoVacio:
 

```
Asignatura.allInstances()-
 forAll(asig|asig.id.oclIsUndefined()=false)
```

Restricción para que la asignatura tenga asignada creditos, es decir, no sea nulo.

- context Asignatura inv CreditosNoVacio:
 

```
Asignatura.allInstances()-
 forAll(a|a.creditos.oclIsUndefined()=false)
```

### Alumno:

Restricción para que una asignatura este aprobada, tiene que tener una nota mayor a cinco.

- context Alumno inv Si\_Aprobada :
 

```
self.asignaturaAlumno-> forAll(a | self.matriculaAlumno->
 exists (m | m.imparticionMatricula.asignaturaImparticion = a
 and m.nota >= 5))
```

Restricción para que un alumno solamente tenga una matrícula aprobada de una asignatura.

- context Alumno inv NoVuelveMatricularse :

```
self.asignaturaAlumno-> forAll(a| self.matriculaAlumno->
select (m | m.imparticionMatricula.asignaturaImparticion = a
and m.nota >= 5) -> size()<=1)
```

Restricción que nos obliga a tener 240 créditos para poder tener el título de “Grado”.

- context Alumno inv creditosTituloGrado :

```
(self.tituloAlumno->
select(t|t.titulacionTitulo.oclIsKindOf(Grado)))->
forAll(title|(self.asignaturaAlumno->
select(asig|asig.titulacionAsignatura->
forAll(nombre=title.titulacionTitulo.nombre)))->
collect(creditos) -> sum()>=240)
```

Restricción que nos obliga a tener 60 créditos para poder tener el título de “Posgrado”.

- context Alumno inv creditosTituloPosgrado :

```
(self.tituloAlumno->
select(t|t.titulacionTitulo.oclIsKindOf(Posgrado)))->
forAll(title|(self.asignaturaAlumno->
select(asig|asig.titulacionAsignatura->
forAll(nombre=title.titulacionTitulo.nombre)))->
collect(creditos)-> sum()>=60)
```

Restricción que me obliga a tener un “Grado” para poder matricularme de un “Posgrado”

- context Alumno inv MatricularseEnPosgrado:

```
self.matriculaAlumno->
forAll(m|m.imparticionMatricula.asignaturaImparticion.titulacionAsignatura->exists(t|t.oclIsKindOf(Posgrado))implies
self.tituloAlumno->
exists(ttitle|ttitle.titulacionTitulo.oclIsKindOf(Grado)))
```

Restricción para que un alumno no se pueda matricular de una asignatura una vez aprobada

- context Alumno inv soloUnaAprobada:

```
self.asignaturaAlumno-> forAll(a|self.matriculaAlumno-> select
(m|m.imparticionMatricula.asignaturaImparticion=a and
m.nota>=5)-> size()<=1) and self.asignaturaAlumno->
forAll(a|self.matriculaAlumno->
select(m|m.imparticionMatricula.asignaturaImparticion = a and
m.nota>=5)->
```

```

collect(c|c.imparticionMatricula.cursoImparticion.anio)->
asOrderedSet()->first()=(Matricula.allInstances->
select(m|m.imparticionMatricula.asignaturaImparticion=a)->
collect(c|c.imparticionMatricula.cursoImparticion.anio)->
max()))

```

**Matricula:**

Restricción para que nuestro campo nota tenga un valor entre cero y diez.

- context Matricula inv Entre0y10:
 

```
self.nota >= 0 and self.nota <= 10
```

**Profesor:**

Restricción que controla que un profesor no se puede matricular de una asignatura que imparte él.

- context Profesor inv NoMatriculadoeImpartir :
 

```
self.imparticionProfesor->
forAll(d|d.matriculaImparticion.profesorMatricula->
excludes(self))
```

**Curso:**

Restricción que obliga al campo anio a tener algún valor.

- context Curso inv CursoUndefined:
 

```
Curso.allInstances()->forAll(c|c.anio.oclIsUndefined()= false)
```

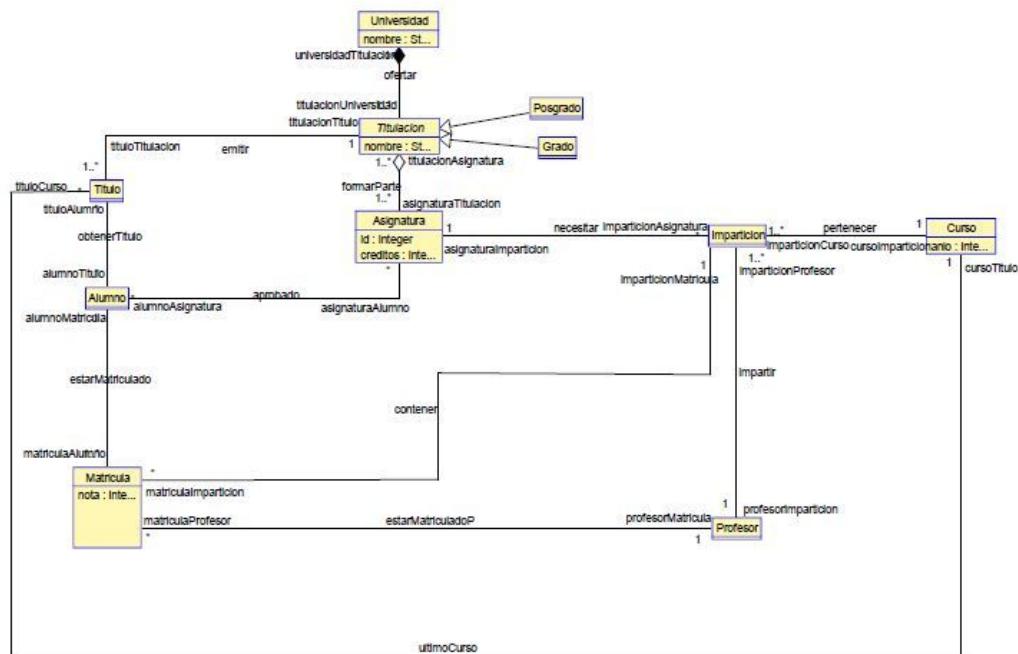
**Título:**

Restricción que obliga a que el título tenga el curso de la última asignatura aprobada

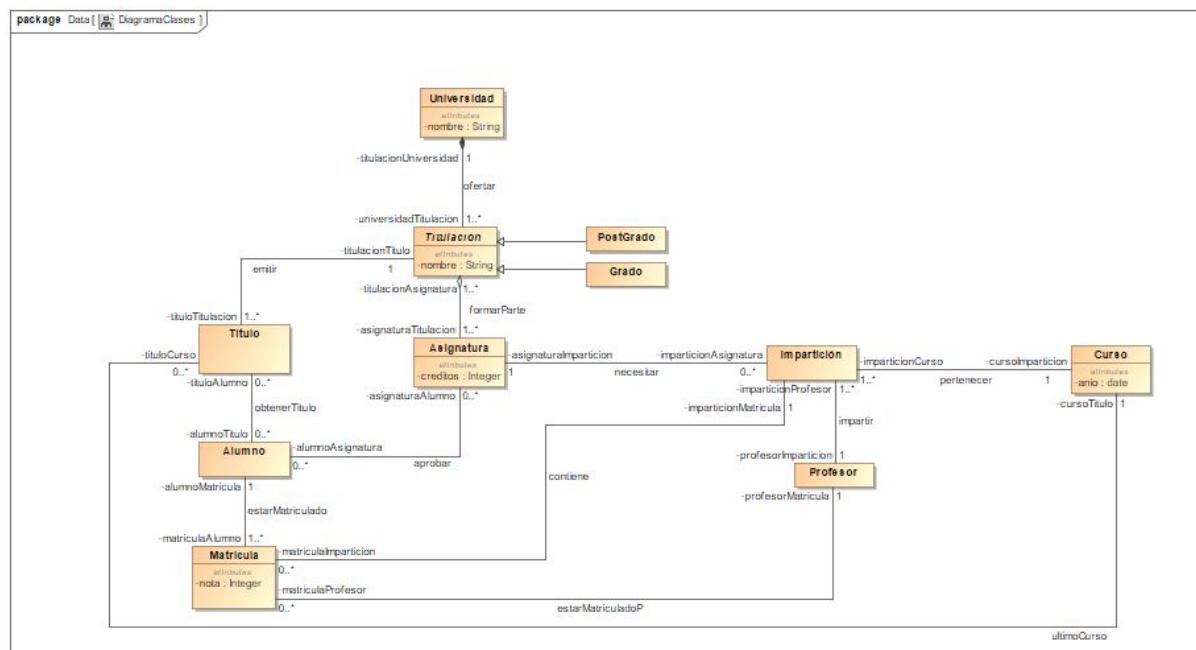
- context Titulo inv AsignaturaCursoTitulo:
 

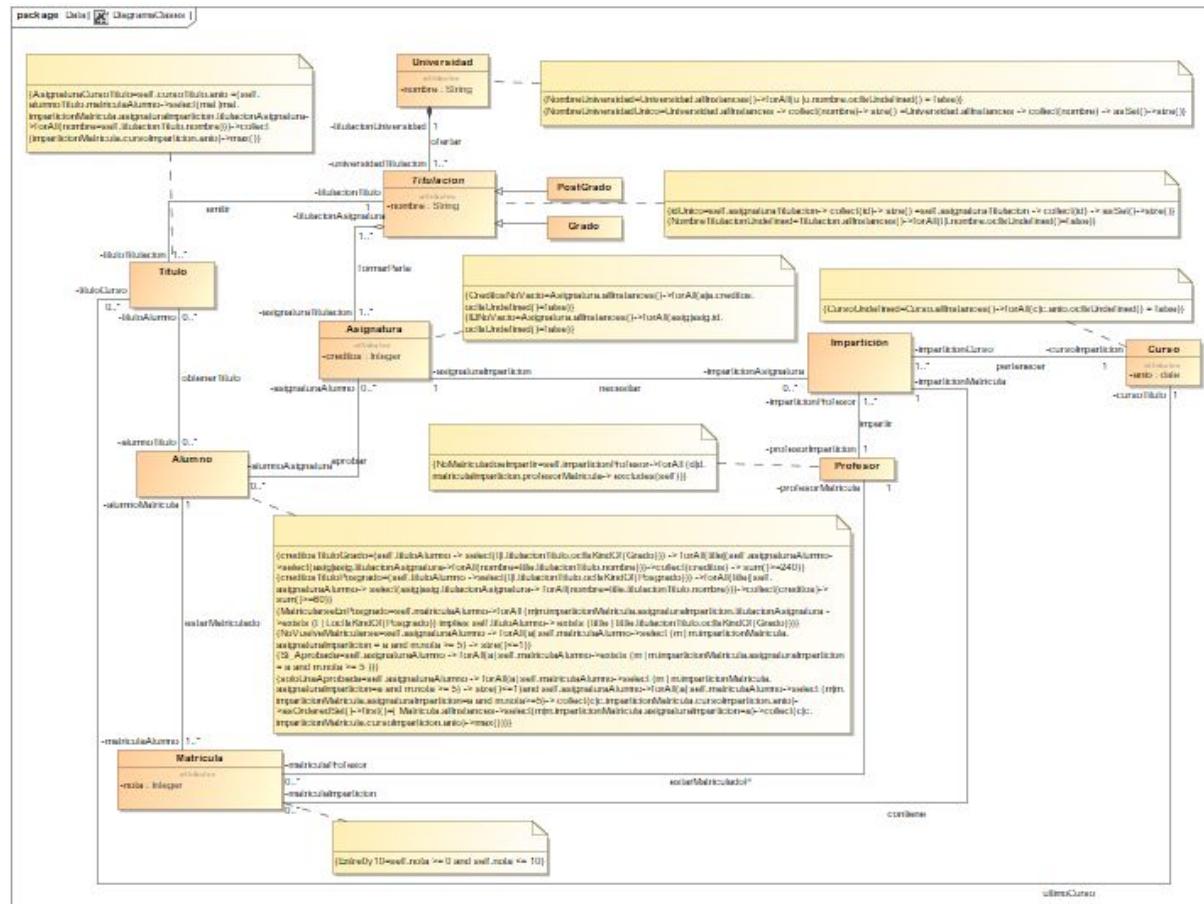
```
self.cursoTitulo.anio=(self.alumnoTitulo.matriculaAlumno->
select(mat|mat.imparticionMatricula.asignaturaImparticion.titulacionAsignatura->
forAll(nombre=self.titulacionTitulo.nombre)))->
collect(imparticionMatricula.cursoImparticion.anio)->max()
```

## MODELO EN USE

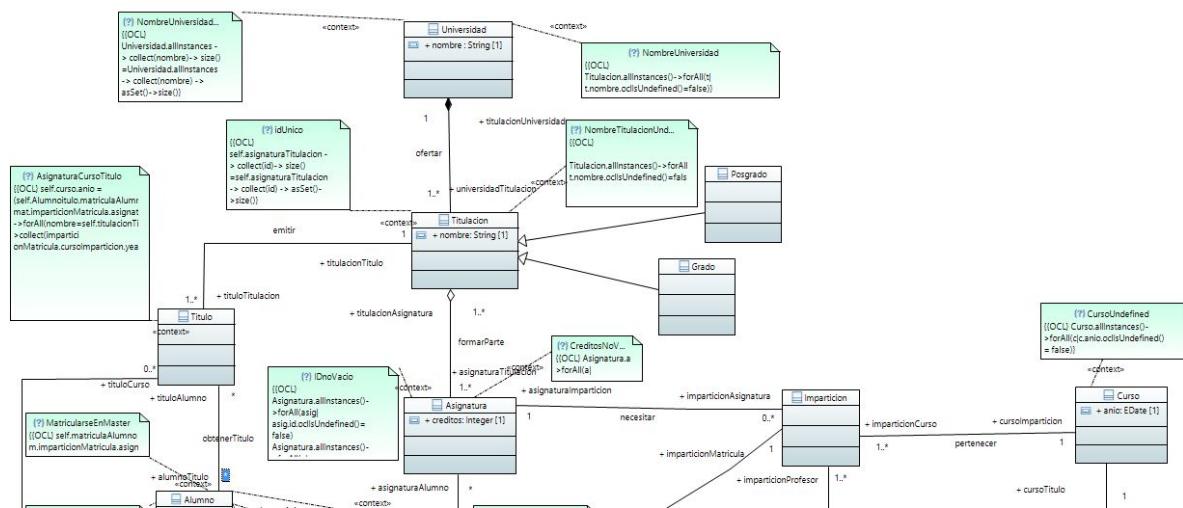


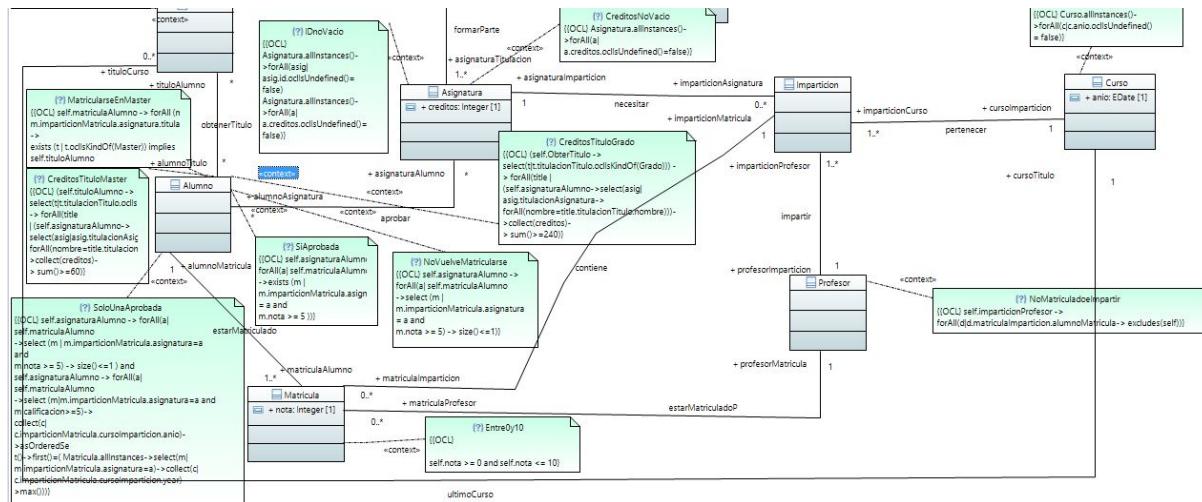
## MODELO EN MAGICDRAW



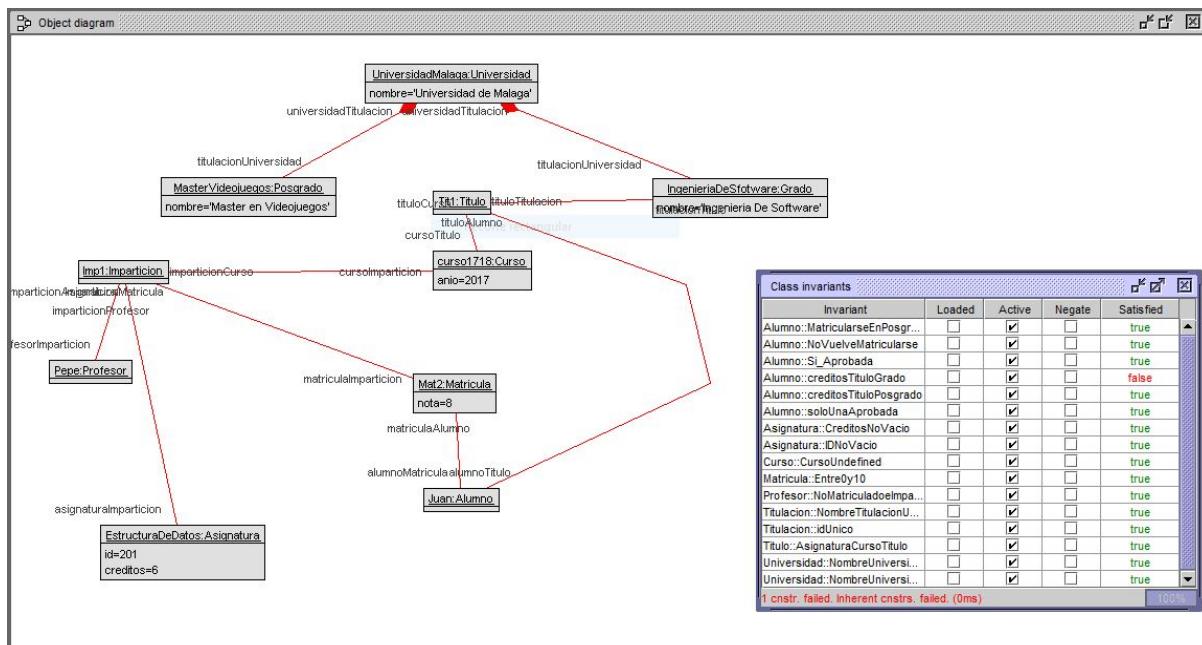


## **MODELO EN PAPYRUS**





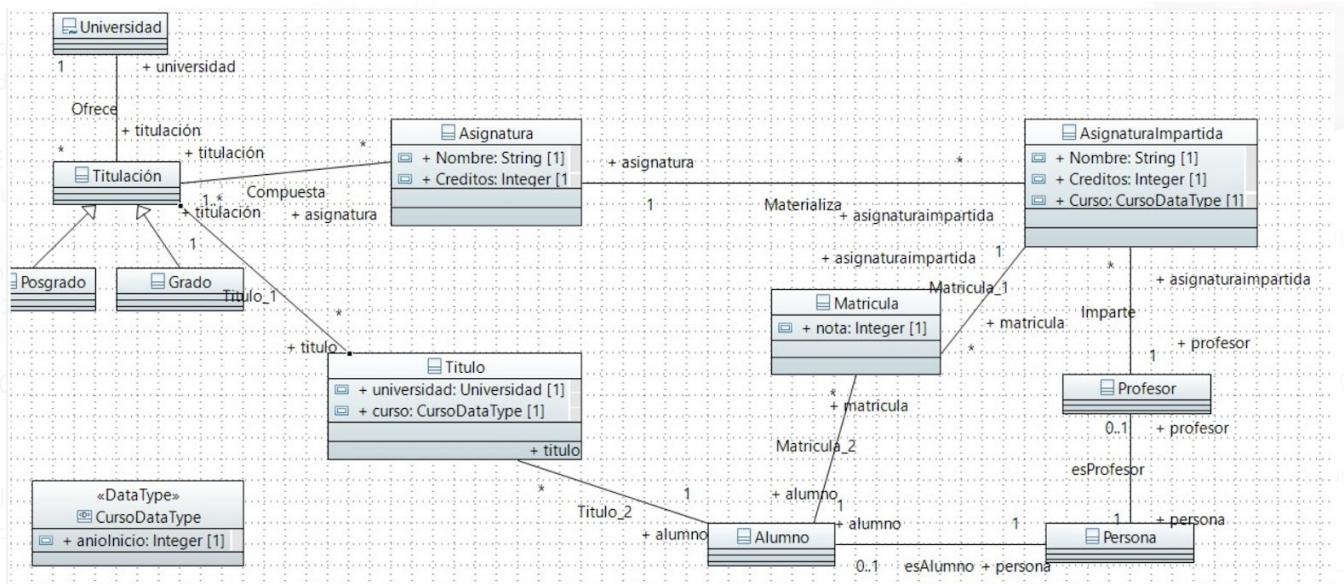
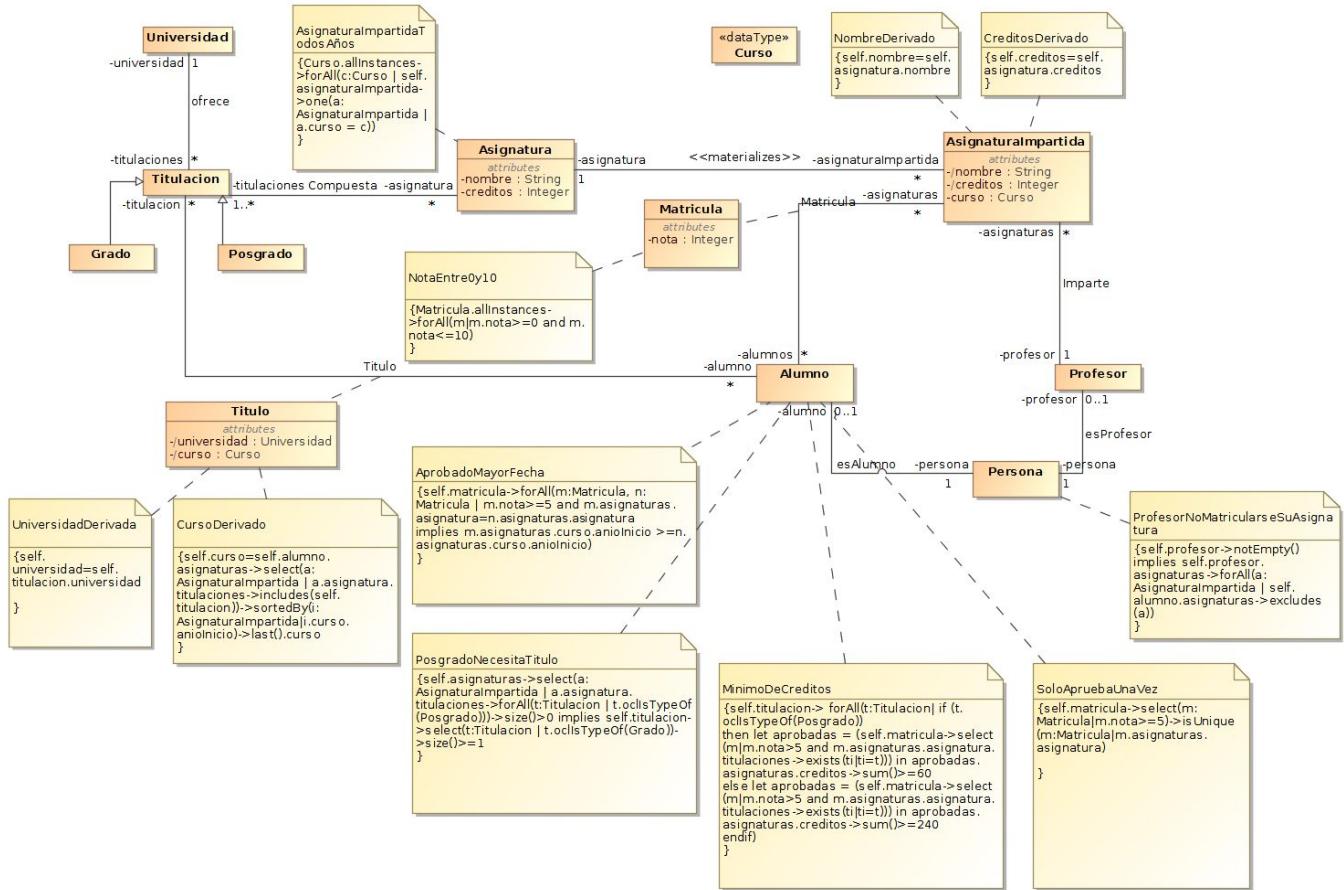
## MODELO DE OBJETOS

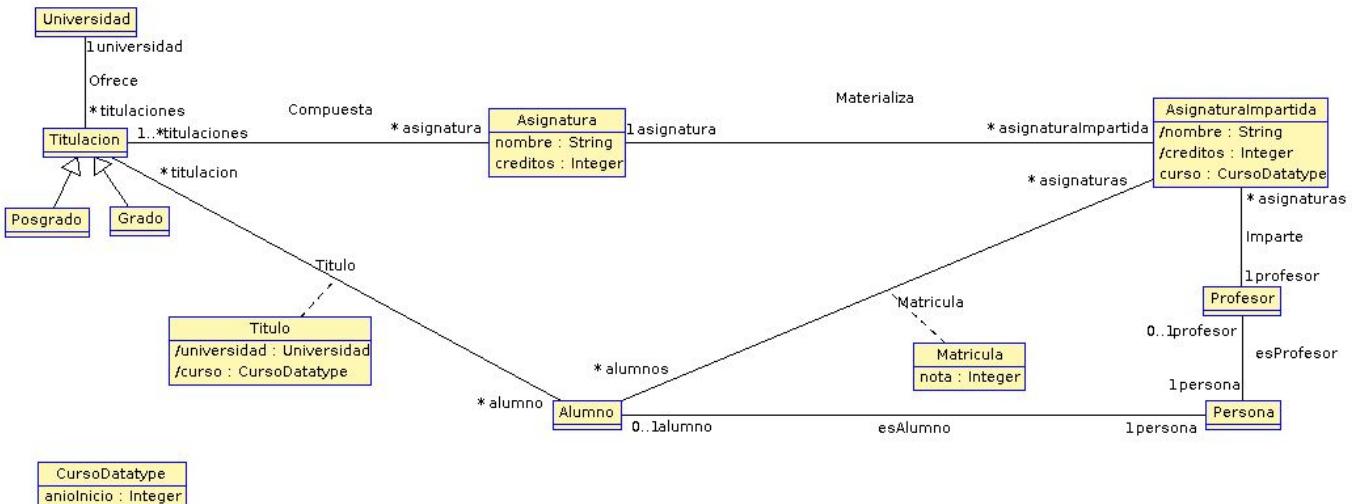


Cumple todos los invariantes a excepción del invariante “creditosTituloGrado” debido a que le hemos dado el título “Tit1” a “Juan”, y para poder obtenerlo tendría que haber superado los 240 créditos, pero como observamos en el diagrama, solo tiene una asignatura superada.

**Doce**

## MODELADO DE UN SISTEMA GESTOR DE UNA UNIVERSIDAD





*El modelo de objetos .soil utilizado para verificar todas las restricciones no se incluye ya que no se ve bien en una imagen por la cantidad de objetos que hay que incluir para que cumpla todas las restricciones, pero está incluido en la entrega*

A continuación se detalla el porqué de las decisiones tomadas en nuestro modelo del sistema y se explicará cada uno de los elementos implicados en el mismo.

La primera frase del enunciado “Una Universidad ofrece distintas titulaciones de grado y de posgrado (másters).”, nos hace plantearnos la primera cuestión en el modelado de nuestro sistema. ¿Debemos considerar al sistema Universidad una clase de nuestro modelo?. En principio lo dejamos en duda, no obstante más adelante el enunciado dice “El título recoge el nombre de la universidad” lo cual nos hace pensar que habrá más de una universidad por lo cual decidimos incluirla a nuestro modelo.

Por tanto partiendo de Universidad, tenemos que una universidad ofrece una serie de Titulaciones, las cuales pueden ser o bien una Titulación de Grado o bien una Titulación de Posgrado.

La relación de Universidad y Titulación la denominamos **ofrece** y las multiplicidades indican que una Titulación es ofrecida únicamente en una universidad mientras que una universidad puede ofrecer multitud de titulaciones.

La multiplicidad unaria es porque una titulación en distintas universidad puede ser distinta. Por ejemplo , Ingeniería del software en la UMA no se compone de las mismas asignaturas que la titulación de mismo nombre en la UCA por ejemplo. Es decir en nuestro caso Titulación, no conceptualiza únicamente el nombre de la misma sino que se entiende como un conjunto de asignaturas, que pueden ser distintas entre universidades distintas.

Tal y como hemos dicho una titulación se compone de un conjunto de asignaturas, así como una asignatura puede pertenecer a más de una titulación, pero al menos debe pertenecer a una. Esto lo hemos modelado así pues entendemos que puede ser útil permitir crear una titulación y después añadirle las asignaturas correspondientes. No obstante, una asignatura debe ser creada para una titulación concreta al menos.

En cuanto a los atributos, contamos únicamente con los requeridos en el enunciado: nombre y créditos.

Si continuamos con nuestro modelo observamos la clase Asignatura Impartida, que mantiene una relación de materialización con la clase asignatura.

La clase Asignatura representa un temario, una programación docente, un nombre y unos créditos, es decir se trata de una plantilla que debe ser aplicada cada año mediante su impartición. Es decir debe ser materializada.

En esta materialización, esta asignatura es impartida por un profesor concreto, en un curso concreto y a unos alumnos concretos.

Todo esto se concreta de la forma que observamos en el modelo.

Por un lado una Asignatura Impartida contiene una serie de atributos de los cuales dos de ellos son derivados de la Asignatura a la que materializan (nombre y créditos) y un tercero que llamamos curso y que hemos considerado de un nuevo dataType Curso, para garantizar el formato "17/18", "18/19".

De esta forma una asignatura impartida tendrá exactamente el mismo nombre y número de créditos que la asignatura a la que materializa y especificará un curso en el que será impartida.

Para conocer el número de alumnos matriculados en una asignatura impartida, disponemos de la relación matrícula con Alumno.

Se trata de una relación con atributos, a través de la cual los profesores los evalúan asignándoles una nota entre 0 y 10. Las multiplicidades son en ambos casos 0..\*

La impartición por otro lado es llevada a cabo por un Profesor, y sólo uno, el cual puede cambiar de año en año. Además un mismo profesor puede impartir más de una asignatura en el mismo curso o en cursos diferentes.

A pesar de que diferenciamos entre Alumno y Profesor como clases, estos son en realidad roles de una Persona. Esto es porque una misma persona puede ser profesor en algunas asignaturas pero también se le permite ser alumno en otras, por lo que en algunas ocasiones se comporta de una forma o de otra, es decir en algunas ocasiones mantiene un rol y en otras otro.

Podríamos haber simplemente utilizado dos relaciones sin clases, donde usáramos ambos roles, no obstante aunque no se especifique entendemos que un profesor aunque es una persona igual que un alumno, puede hacer cosas distintas como evaluar, realizar exámenes... por lo que estimamos conveniente distinguir dos clases para permitir distinción futura.

Es importante también las multiplicidades entre Persona y estas dos clases: 0..1 nos asegura que una persona puede ser o no alumno pero no tiene sentido hablar de ser más de un alumno y lo mismo ocurre con profesor. Además del otro lado de la relación la multiplicidad es 1, pues un alumno es una persona y sólo una.

Para finalizar contamos con la relación con atributos: Título, la cual conceptualiza la acreditación que un alumno obtiene al superar un número concreto de créditos, de una titulación, ofrecida en una universidad determinada.

Esta dispone de dos atributos, universidad y curso, ambos derivados, para asegurar la consistencia de los datos de tal forma que los valores coincidan exactamente con la universidad en la que lo consiguió en alumno y el curso en el que lo hizo.

Las restricciones de nuestro modelo:

| Lenguaje natural                                                                                                                                                       | ocl                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cada asignatura debe ser impartida todos los años                                                                                                                      | context Asignatura inv AsignaturaImpartidaTodosAnios:<br>CursoDatatype.allInstances->forAll(c:CursoDatatype  <br>self.asignaturaImpartida->one(a:AsignaturaImpartida  <br>a.curso = c))                                                                                                                                                                                                                                                                                          |
| Un alumno solo puede aprobar una asignatura una vez, no pudiendo matricularse de asignaturas que ya ha aprobado en cursos anteriores.                                  | context Alumno<br>inv SoloApruebaUnaVez:<br>self.matricula->select(m:Matricula m.nota>=5)->isUnique(m:<br>Matricula m.asignaturas.asignatura)<br>inv AprobadoMayorFecha:<br>self.matricula->forAll(m:Matricula, n:Matricula   m.nota>=5<br>and m.asignaturas.asignatura=n.asignaturas.asignatura<br>implies m.asignaturas.curso.anioInicio<br>>=n.asignaturas.curso.anioInicio)                                                                                                  |
| Las notas tendrán un valor entre 0 y 10                                                                                                                                | context Matricula inv NotaEntreCeroYDiez:<br>Matricula.allInstances->forAll(m m.nota>=0 and<br>m.nota<=10)                                                                                                                                                                                                                                                                                                                                                                       |
| Los profesores no pueden matricularse en aquellas asignaturas que imparten este curso, aunque sí en otras.                                                             | context Persona inv ProfesorNoMatricularseSuAsignatura:<br>self.profesor->notEmpty() implies<br>self.profesor.asignaturas->forAll(a:AsignaturaImpartida  <br>self.alumno.asignaturas->excludes(a))                                                                                                                                                                                                                                                                               |
| Para obtener un título es preciso tener aprobados al menos 240 créditos de las asignaturas que componen una titulación de grado, o los 60 que componen una de posgrado | context Alumno inv MinimoDeCreditos:<br>self.titulacion-> forAll(t:Titulacion  if<br>(t.ocellsTypeOf(Posgrado))<br>then let aprobadas = (self.matricula->select(m m.nota>5<br>and m.asignaturas.asignatura.titulaciones->exists(ti ti=t))) in<br>aprobadas.asignaturas.creditos->sum()>=60<br>else let aprobadas = (self.matricula->select(m m.nota>5<br>and m.asignaturas.asignatura.titulaciones->exists(ti ti=t))) in<br>aprobadas.asignaturas.creditos->sum()>=240<br>endif) |
| para poder matricularse de una asignatura de posgrado, el alumno ha de estar en posesión de un título de grado (de esa universidad o de otra).                         | context Alumno inv PosgradoNecesitaTitulo:<br>self.asignaturas->select(a:AsignaturaImpartida  <br>a.asignatura.titulaciones->forAll(t:Titulacion  <br>t.ocellsTypeOf(Posgrado)))->size()>0 implies<br>self.titulacion->select(t:Titulacion  <br>t.ocellsTypeOf(Grado))->size()>=1                                                                                                                                                                                                |

# Trece

# Práctica 1

## Restricciones:

- noTitulacionDeUniversidad

```
context Alumno inv NoTitulacionDeUniversidad:
 if (self.titulo -> notEmpty) then (self.titulo.titulacion.universidad->intersection(self.universidad))->notEmpty()
 else true endif
```

Comprueba que un alumno está matriculado en una titulación que pertenece a la universidad en la que está suscripto.

- posgrado

```
context Alumno inv posGrado:
 self.titulacion-> forAll(t | t.esPosgrado implies self.titulo->exists(tenerTitulo and not(titulacion.esPosgrado)))
```

Comprueba que un alumno que se matricula en una titulación de posgrado este en posesión de un título de una titulación de posGrado

- Restricción NolmpartirDos

```
context Catedra inv NolmpartirDos:
 Catedra.allInstances() -> forAll(c1,c2 | c1<>c2 and (c1.asignatura = c2.asignatura) implies c1.curso <> c2.curso)
```

Comprueba que una asignatura en un curso no sea impartida por más de un profesor

- Restricción NoAsignaturaDeTitulación

```
context Matricula inv NoAsignaturaDeTitulacion:
 (self.asignatura.titulacion->intersection(self.alumno.titulo.titulacion))->notEmpty()
```

Comprueba que un alumno no se matricule a una asignatura que no pertenece a la titulación en la que está inscrito

- Restricción NoDobleMatricula

```
context Matricula inv NoDobleMatricula:
 if(self.nota >=5) then(Matricula.allInstances()-> forAll(m | m <> self and m.asignatura = self.asignatura
 and self.alumno = m.alumno implies m.curso < self.curso)) else (Matricula.allInstances()-> forAll(m |
 m <> self and m.asignatura = self.asignatura and self.alumno = m.alumno implies m.curso <> self.curso)) endif
```

Comprueba que un alumno no pueda matricularse a una asignatura que ha aprobado en un curso anterior y que no pueda matricularse dos veces en una asignatura en el mismo curso.

- Restricción NotaCorrecta

```
context Matricula inv NotaCorrecta:
 self.nota <=10 and self.nota >=0
```

Comprueba que la nota asignada a un alumno este comprendida entre cero y diez.

- Restricción NoAutoimparticion

```
context Catedra inv NoAutoimparticion:
 Catedra.allInstances() -> forAll(a1 | Matricula.allInstances() -> forAll(m1 | (m1.asignatura = a1.asignatura and
 m1.alumno = a1.profesor) implies m1.curso <> a1.curso))
```

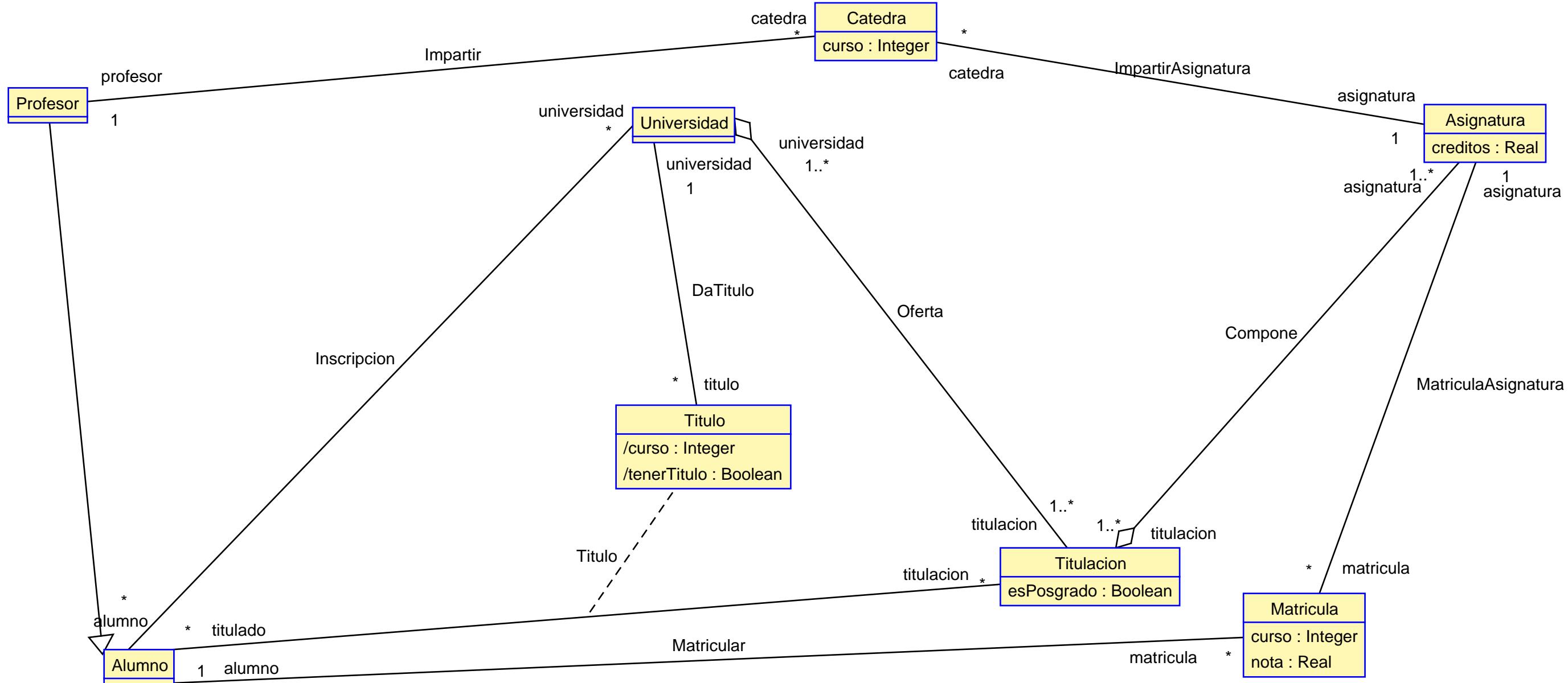
Comprueba que un profesor no se matricule como alumno en una asignatura en un curso en el cual está impartiendo la misma asignatura

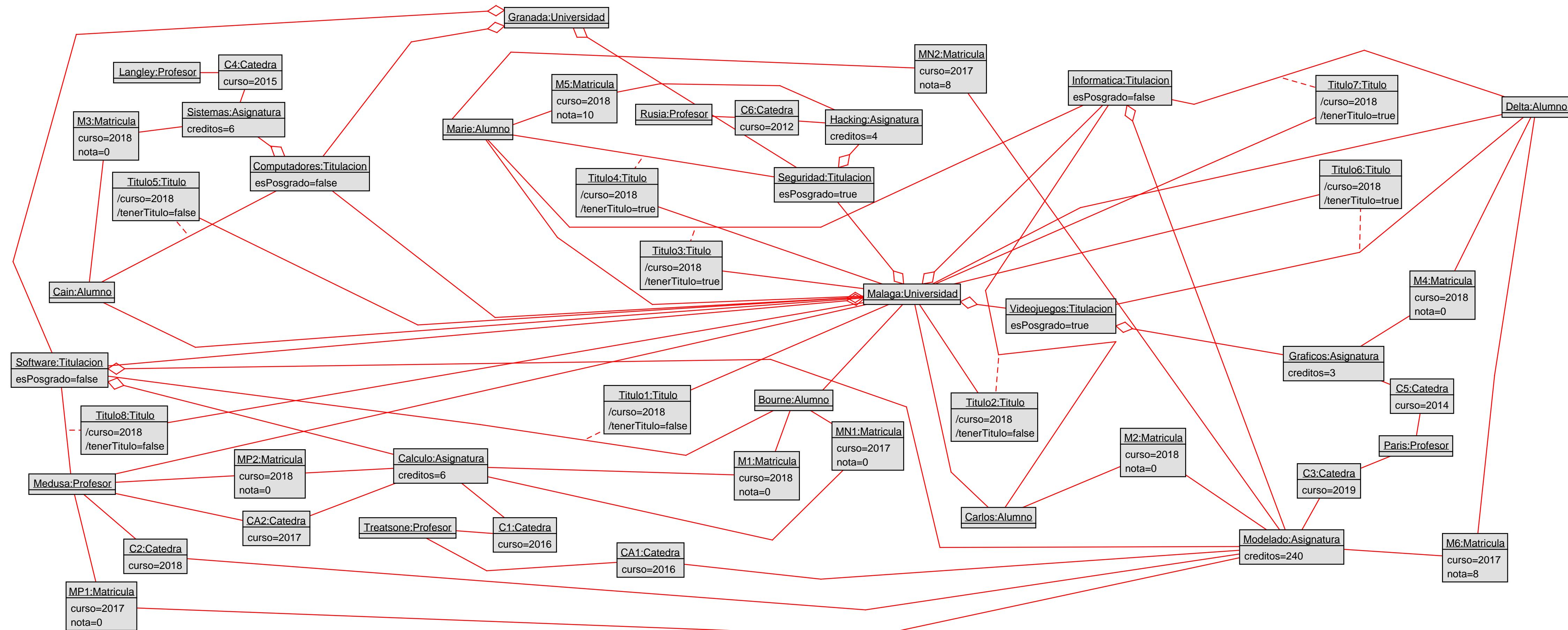
- Restriccion NoUniversidadSinTitulacion

```
context Titulo inv NoUniversidadSinTitulacion:
 ((self.titulacion.universidad->intersection(self.titulado.universidad))->includes(self.universidad))
```

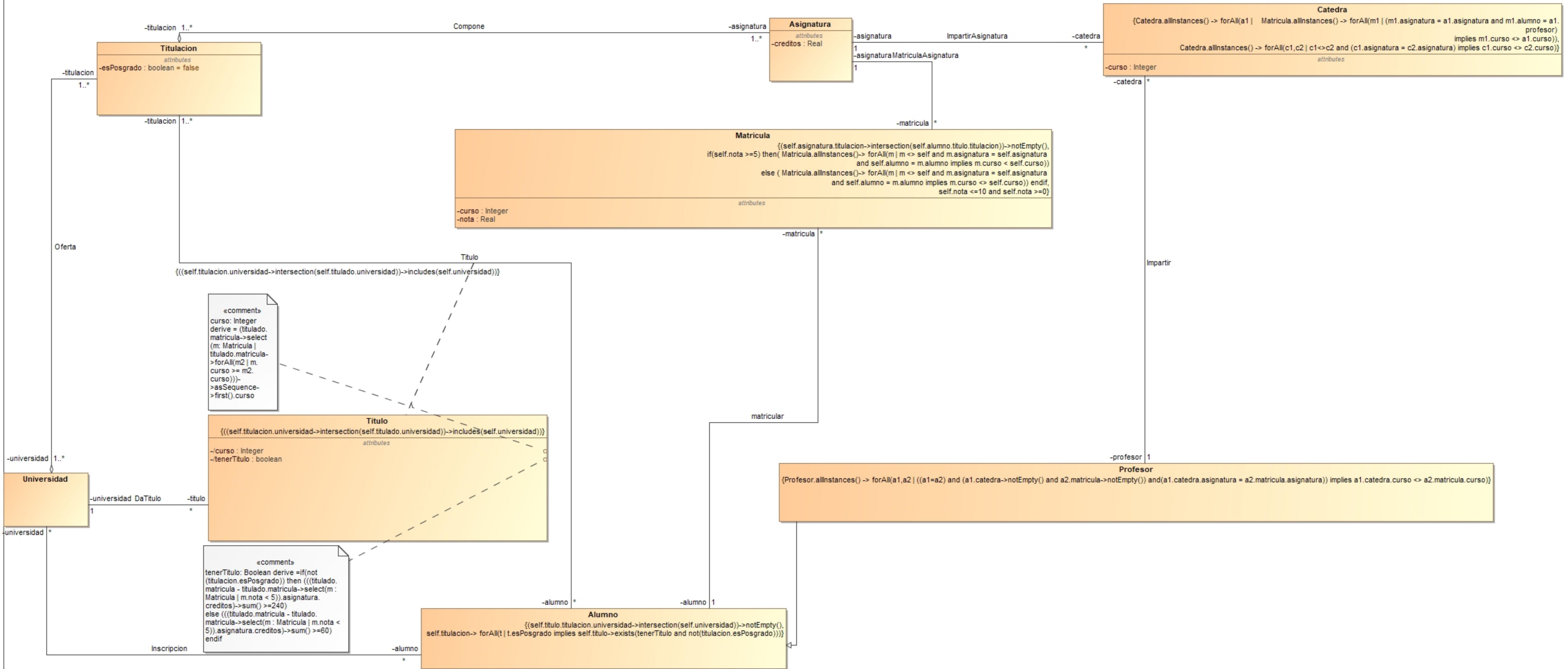
Comprueba que un alumno no se matricule en una titulación que no pertenece a la universidad en la que está matriculado.

**Modelo en USE**

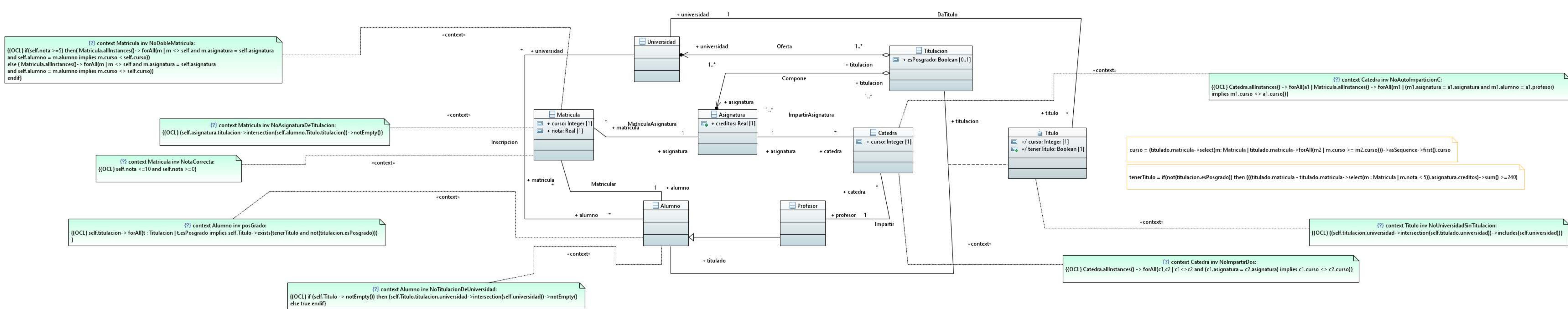




## **Modelo en MagicDraw**

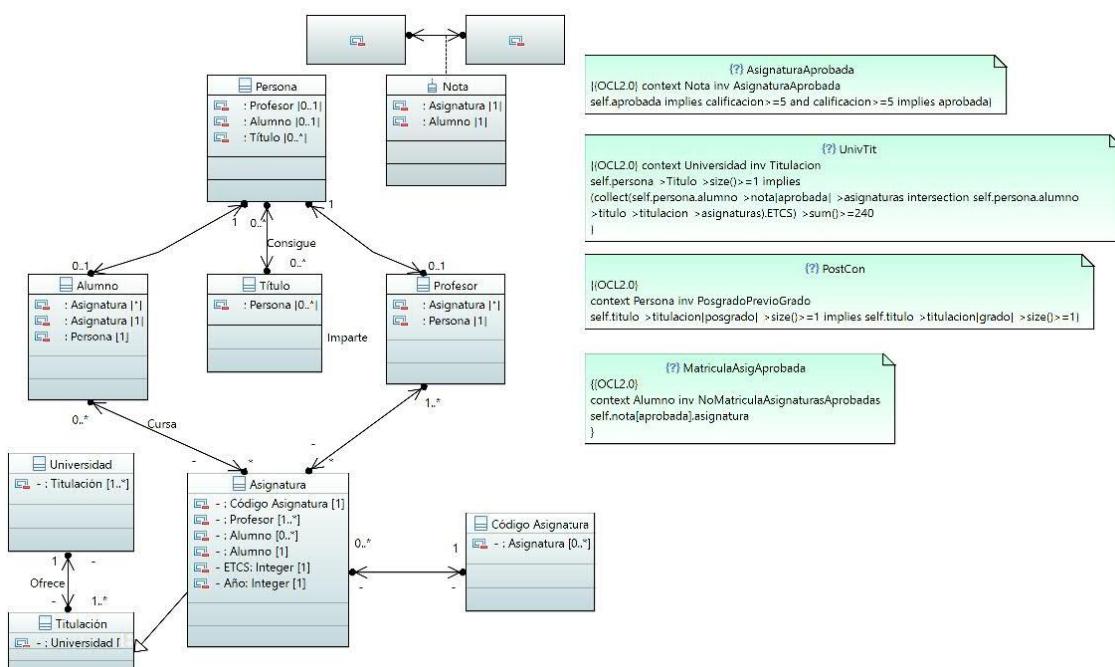
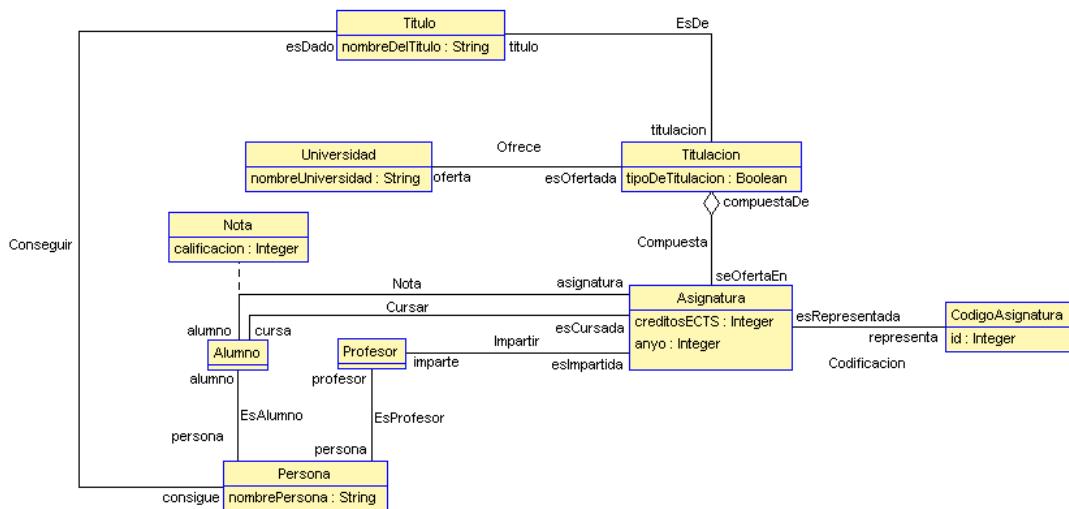
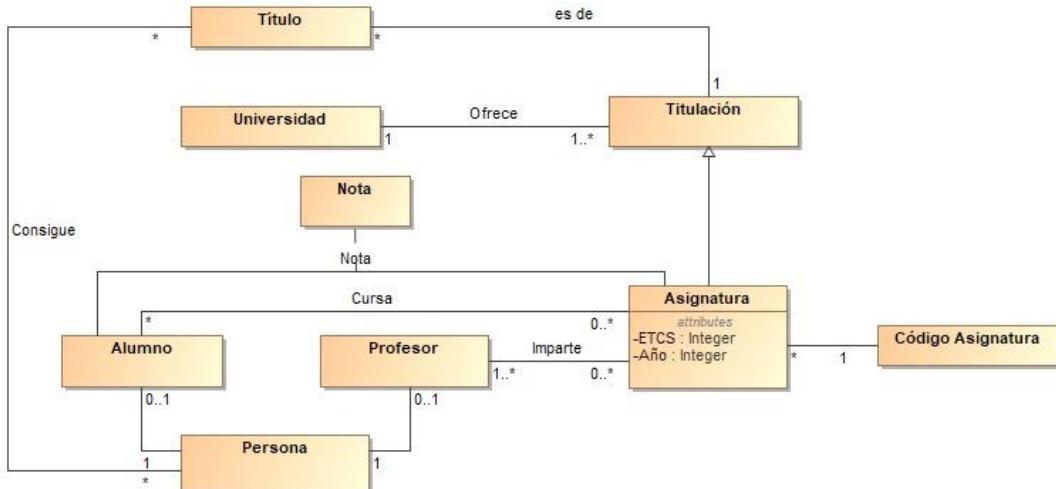


## **Modelo en Papyrus**



# Catorce

## Práctica 1. Universidad.



Para empezar, hemos creído necesaria la aparición de una clase universidad, dado que son múltiples las universidades de las que es posible obtener un título, de modo que es necesario poder diferenciar entre unas y otras.

La universidad ofrecerá plazas para diversas titulaciones, entre las que se distinguirá entre grado y posgrado a través de un atributo booleano que según un valor u otro, indicará en cuál de los dos casos nos encontramos.

Una titulación estará compuesta por una serie de asignaturas, que podrán además pertenecer a varias titulaciones. Para diferenciar unas asignaturas de otras como por ejemplo “Modelado” y “Requisitos”, hemos recurrido a una clase que les otorgue un código diferenciador, que además servirá para ver que asignaturas están ya aprobadas, pues la clase asignatura será dinámica, de una forma “Modelado18/19”. Los profesores, como es de esperar, serán los que imparten las clases y, por tanto, evalúen a los alumnos.

Dado que una persona puede ser a la vez profesor y alumno de una asignatura diferente, hemos recurrido a la clase persona que englobe a ambos roles, que estará relacionada con dos clases que precisamente representen los roles mencionados.

Luego habrá un grupo de alumnos que cursen dicha asignatura, y para expresar las notas que cada uno ha obtenido, se ha recurrido a una clase de asociación “Notas”, que relacionará a cada alumno con las asignaturas que curse o haya cursado, proporcionando su nota para por ejemplo comprobar cuantos créditos tiene aprobados, cuantos le faltan para terminar y ese tipo de operaciones.

Por último, los títulos que una persona posee se representan mediante la relación “conseguir”, donde una vez se haya alcanzado la cantidad de créditos necesarios, esa persona obtenga un título del grado o posgrado que esté cursando.