La siguiente imagen describe un modelo de una *Biblioteca*, en la cual usuarios pueden pedir libros y revistas prestados, siempre y cuando estén registrados en ella. El diagrama muestra 5 clases:

- **Biblioteca**: con un atributo *name* único de tipo String, para definir su nombre.
- **Persona**: con otro atributo de tipo String, *nombre*.
- **Articulo**: una clase abstracta de la cual heredan:
  - **Libro**: con un String, *titulo*.
  - **Revista**: con un atributo Integer, *numero*, para indica su número de publicación.

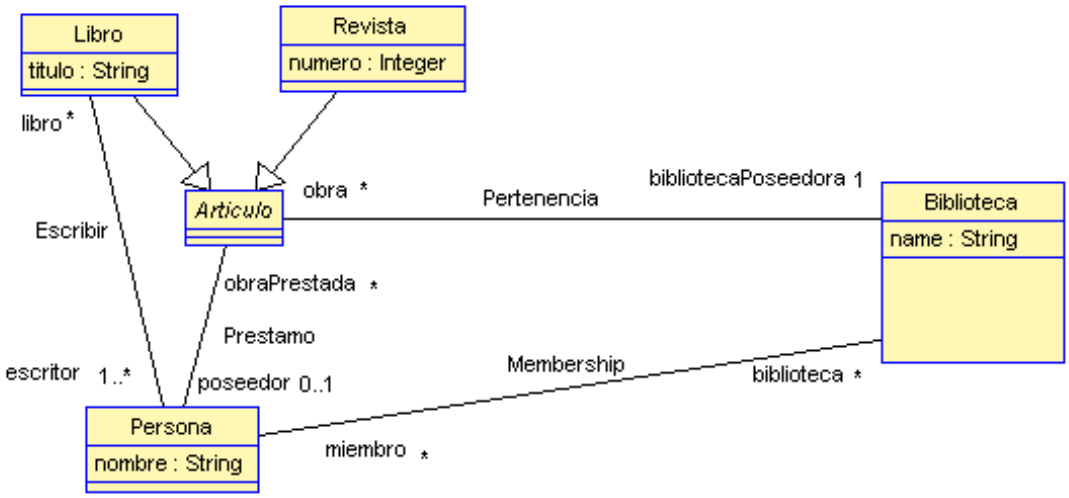

Imagen 1: Modelado en USE (GUI)

El modelo incluye las relaciones:

- ***Pertenencia,*** por la cual una biblioteca puede tener de 0 a varias obras en su repertorio, y una obra no podría existir sin estar almacenada en una biblioteca, y, aunque ésta fuera prestada, seguiría perteneciendo a dicha biblioteca.
- ***Membership:*** permitiendo así que haya varios usuarios registrados en una biblioteca, y a la vez, estos puedan ser socios de otras.
- ***Escribir:*** que posibilita que una persona escriba varios libros y que estos deban ser escritos por al menos una (o varias), dejando de lado los libros de carácter *anónimo,* ya que en este modelo no los consideraremos.
- ***Prestamo:*** por la cual una obra puede ser prestada a la vez, como máximo, a una persona, y pudiendo un esta tener varias en su disposición al mismo tiempo.

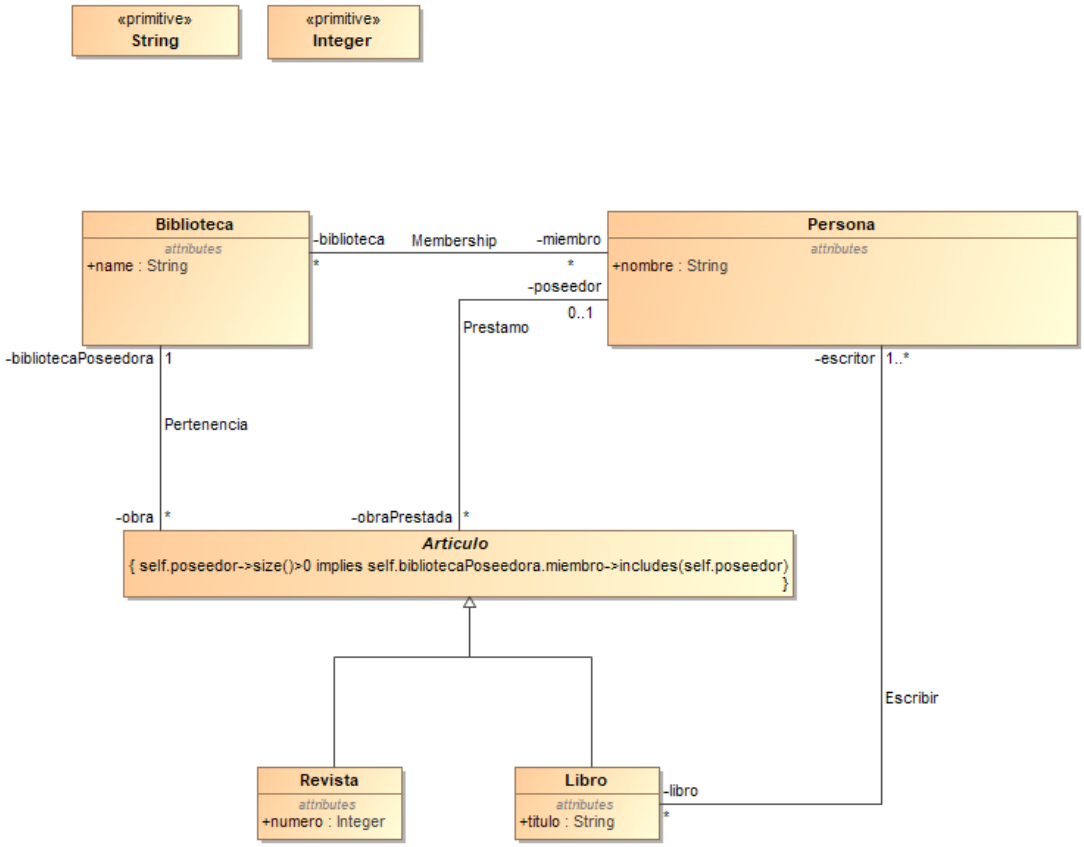

Imagen 2: Modelado usando MagicDraw

Para conseguir que un usuario pueda obtener un libro prestado de la biblioteca se requiere que este y el usuario que lo quiera estén registrado en su base de datos. Para ello creamos el constraint *debeEstarRegistrado:*

```
constraints

context Articulo inv debeEstarRegistrado:
  self.poseedor->size()>0 implies self.bibliotecaPoseedora.miembro->includes(self.poseedor)
```

Imagen 3: Constraint

A continuación, crearemos una instancia del modelo para probarlo y ver que sus restricciones de integridad se mantienen. (siguiente pag.)

CASO 1:
Modelo válido.

```
!new Persona('sergio')
!set sergio.nombre:='Sergio'

!new Persona('olive')
!set olive.nombre:='Olive'

!new Biblioteca('jabega')
!set jabega.name:='UMA_Jabega'

!new Libro('keyToPass')
!set keyToPass.titulo:='Conceptual Modeling of Information Systems'

!insert (olive, keyToPass) into Escribir
!insert (jabega, keyToPass) into Pertenencia
!insert (sergio, jabega) into Membership
!insert (sergio, keyToPass) into Prestamo
```

Imagen 4: Código #1

Al introducir la instancia con sus relaciones y el comando *check*, obtenemos lo siguiente en consola:

```
use> info state
State: state#1
class      : #objects + #objects in subclasses
---------------------------------------------
(Articulo) :        0                       1
Biblioteca :        1                       1
Libro      :        1                       1
Persona    :        2                       2
Revista    :        0                       0
---------------------------------------------
total      :        4

association : #links
--------------------
Escribir    :      1
Membership  :      1
Pertenencia :      1
Prestamo    :      1
--------------------
total       :      4
use> check
checking structure...
checked structure in 2ms.
checking invariants...
checking invariant (1) `Persona::debeEstarRegistrado': OK.
checked 1 invariant in 0.005s, 0 failures.
```

Resultado en consola 1

CASO 2: No cumple el constraint exigido.

```
!new Persona('sergio')
!set sergio.nombre:='Sergio'

!new Persona('olive')
!set olive.nombre:='Olive'

!new Biblioteca('jabega')
!set jabega.name:='UMA_Jabega'

!new Libro('keyToPass')
!set keyToPass.titulo:='Conceptual Modeling of Information Systems'

!insert (olive, keyToPass) into Escribir
!insert (jabega, keyToPass) into Pertenencia
!insert (sergio, keyToPass) into Prestamo
```

Imagen 5: Código #2

Al ejecutar el comando *check* obtenemos que el constraint no se cumple:

```
use> info state
State: state#1
class      : #objects + #objects in subclasses
---------------------------------------------
(Articulo) :        0                       1
Biblioteca :        1                       1
Libro      :        1                       1
Persona    :        2                       2
Revista    :        0                       0
---------------------------------------------
total      :        4

association : #links
--------------------
Escribir    :      1
Membership  :      0
Pertenencia :      1
Prestamo    :      1
--------------------
total       :      3
use> check
checking structure...
checked structure in 5ms.
checking invariants...
checking invariant (1) `Persona::debeEstarRegistrado': FAILED.
  -> false : Boolean
checked 1 invariant in 0.006s, 1 failure.
```

Resultado en consola 2