Ingeniería del Software

Práctica 2: Análisis UML

Presentación

En esta práctica seguiremos trabajando sobre la aplicación que introdujimos en la Práctica 1. Y para ello debéis partir de la solución oficial publicada en la Práctica 1.

Esta Práctica profundiza en el análisis del sistema de Software. La actividad cubre los contenidos del módulo 4 de la asignatura.

En el calendario del aula tenéis una descripción completa de las competencias y objetivos que esta Práctica incluye.

Importante: Para responder a las preguntas, debéis basaros en la información que **se indica de forma explícita** en el enunciado. Por ejemplo, aunque sea lógico que un sistema ofrezca una funcionalidad para darse de baja como usuario, si ninguno de los stakeholders lo menciona, no se considerará una respuesta válida.

Nota previa

Recordad que, tal como se explica en el Plan docente de la asignatura, esta actividad debe resolverse de forma **individual**. En caso de detectar copias se penalizará la actividad con una D como nota.

Para garantizar que has resuelto esta actividad de forma individual, te pedimos que en tu solución entregada añadas el siguiente texto:

"*Certifico que he realizado la Práctica 2 de forma completamente individual y solo con la ayuda que el profesorado de esta asignatura considera oportuna según las FAQs sobre plagio.*"

Compromiso de autoresponsabilidad

Escribe aquí el texto con el compromiso de autoresponsabilidad que consta entrecomillado en la **nota previa. No escribirlo implica la no corrección de la Práctica y una calificación de D.**

"Certifico que he realizado la Práctica 2 de forma completamente individual y solo con la ayuda que el profesorado de esta asignatura considera oportuna según las FAQs sobre plagio."

Enunciado

Pregunta 1 (15%)

En la práctica anterior especificamos el caso de uso correspondiente a "Preparar el contenido de una saga", desde el momento en que el gestor de contenido propone trabajar sobre una saga hasta que ésta es completada y se publica. A continuación especificamos de nuevo el caso de uso:

Identificador de caso de uso: Preparar el contenido de una saga
Actor principal: Gestor de contenido
Actores de apoyo: Propietaria, Colaboradora fija, Colaborador invitado y Trabajadora
Nivel: General
Ámbito: Sistema
Escenario principal de éxito:

1. El gestor de contenido solicita preparar el contenido de una saga y para ello indica algunos datos básicos de la saga.
2. La propietaria aprueba la preparación del contenido de la saga propuesta.
3. La colaboradora fija completa la información de la saga propuesta.
4. La colaboradora fija invita a el colaborador invitado a participar en la preparación de la información de la saga propuesta
5. La colaboradora invitada completa la información de la saga propuesta.
6. La propietaria aprueba el contenido de la saga propuesta.
7. El gestor de contenido añade los enlaces a las historias de Instagram y al hilo de Twitter con la información de la saga.
8. La trabajadora sube las fotografías de lo que haya preparado en el aparador de la tienda.

9. El gestor de contenido publica el contenido en la plataforma y publica las historias de Instagram y el hilo de Twitter.
10. El caso de uso acaba.

Escenarios alternativos:

2a. La propietaria no aprueba la preparación del contenido de la saga propuesta.

2a1. El caso de uso continúa en el paso 10.

4a La colaboradora no invita a un colaborador invitado:

4a1 El caso de uso continúa en el paso 6.

6a La propietaria no aprueba el contenido de la saga propuesta:

6a1 La propietaria añade notas para la corrección del contenido

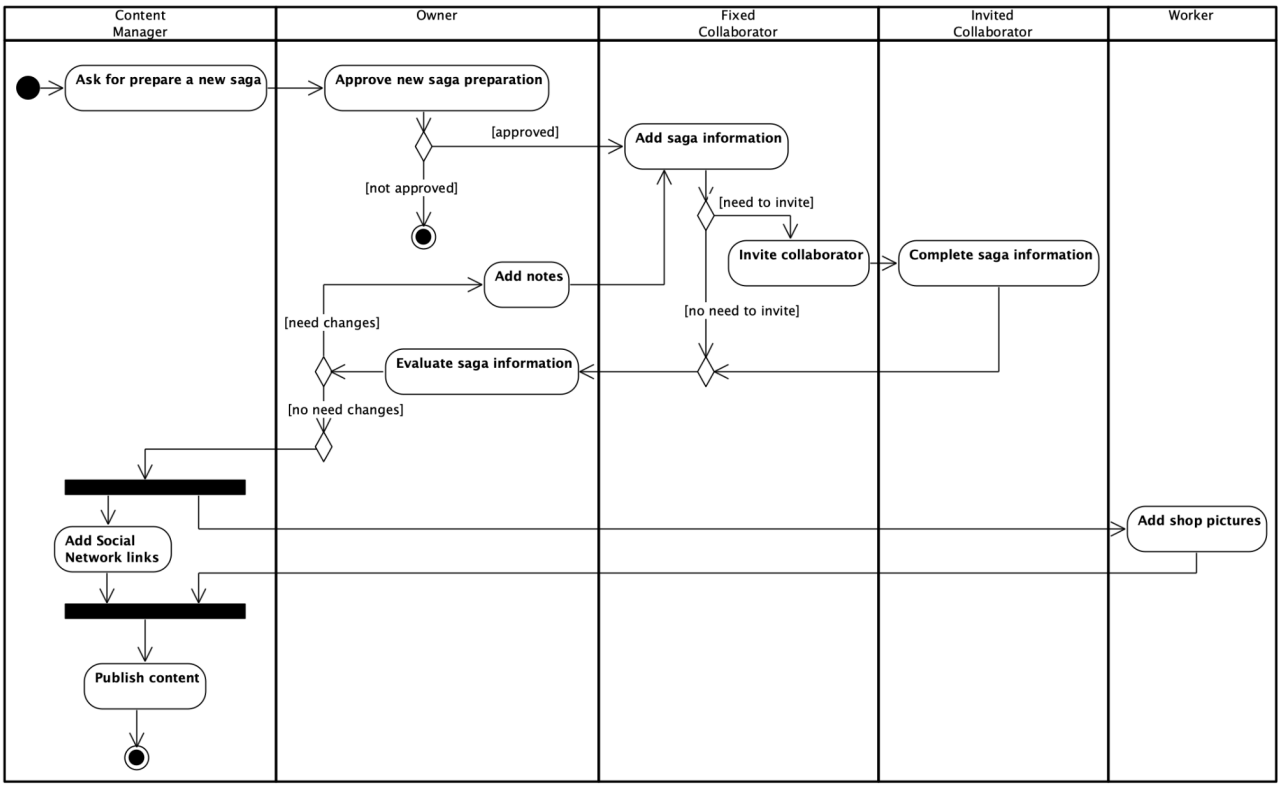
6a2 El caso de uso continúa en el paso 3.

Nota: Los puntos 6 y 7 del escenario principal de éxito aparecían así en el enunciado de la práctica 1: “Y llegados a este punto sólo nos quedarán un par de tareas que siempre haremos a la vez. Por mi parte, prepararé una historia de Instagram y un hilo de Twitter en el que publicaré la información de la saga que habremos definido (...). Y Ada, por su parte, se encargará de preparar la tienda con un escaparate chulo en la librería o lo que se le ocurra en referencia a la saga (...) Cuando todo esto esté terminado (...)”. Tenedlo en cuenta para modelizar la solución.

Realizad el **diagrama de actividades UML** de este caso de uso. Tened en cuenta que:

- El diagrama tiene que mostrar claramente qué actor hace cada actividad
- **No es necesario indicar las actividades que hace el sistema**

Solución

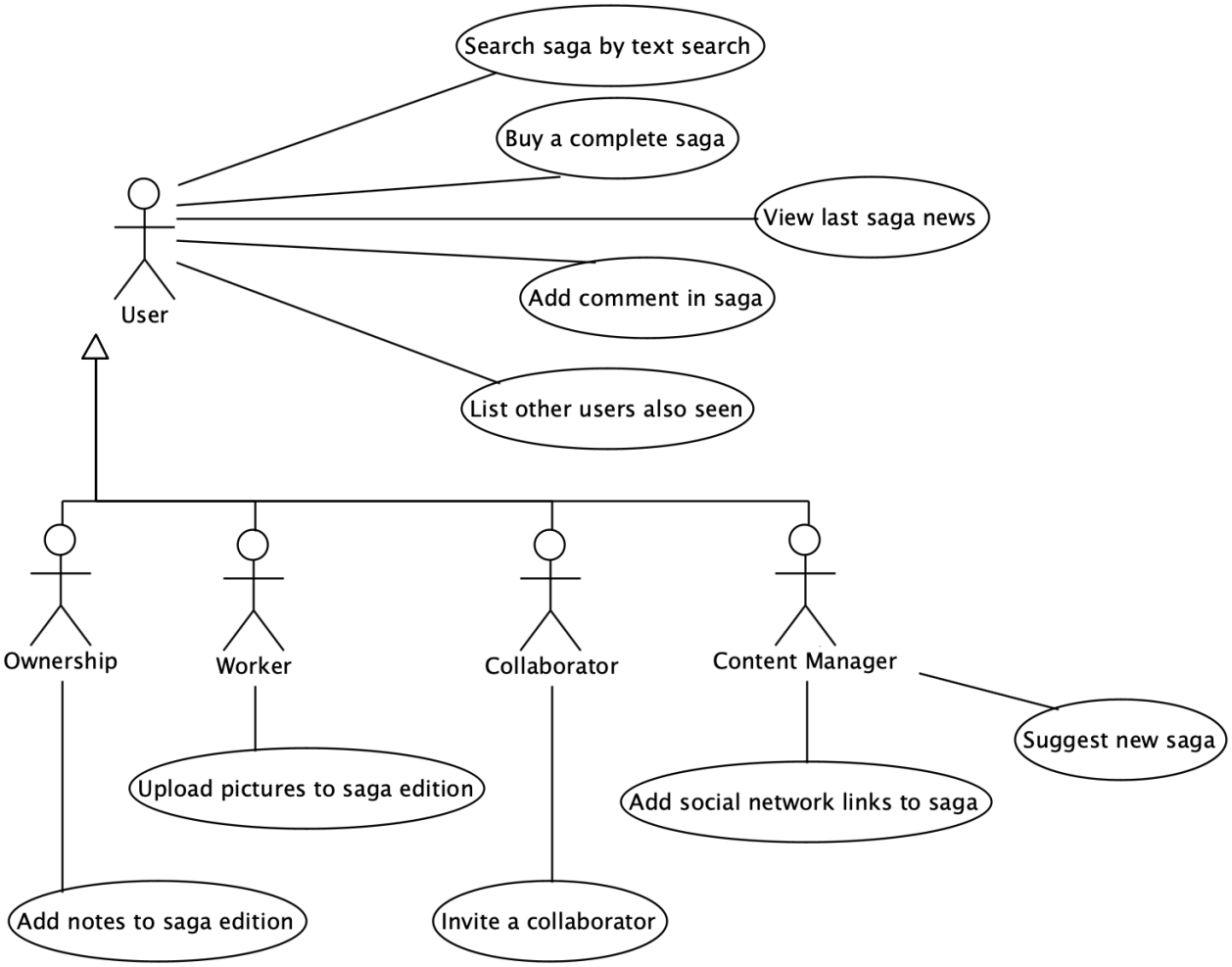


Pregunta 2 (10%)

Haced un **diagrama de casos de uso de nivel de usuario** que incluya diez casos de uso que podáis identificar revisando las entrevistas indicadas en la práctica 1.

- Debéis incluir casos de uso cuyo actor sea cualquier tipo de actor que ya identificamos en la práctica 1: Propietaria, Colaborador, Colaborador invitado, Gestor de contenido, Trabajadora y Usuario.
    - Los actores distintos de Usuario pueden realizar lo que puede hacer un actor Usuario más sus propios casos de uso.
- Tened en cuenta que algunos de estos casos ya han sido identificados en la pregunta 6 de la Práctica 1 (podéis incluirlos).
- No hay que hacer la especificación textual de los casos de uso. Sólo el diagrama UML.
- Debéis mostrar casos de uso de un mínimo de dos tipos de actores.

Solución



Nota: En la solución se han mostrado varios casos de uso de las entrevistas de la Práctica 1, aunque se pueden extraer algunos más. Si vuestra solución contiene otros casos de uso de nivel usuario se considerará correcta.

Pregunta 3 (30%)

Seguiremos con el caso de OnlySF, y nos adentraremos en la fase de análisis y modelado de clases de la aplicación. En concreto, queremos centrarnos en la modelización del core de la información de la aplicación web, las sagas.

De una saga necesitaremos conocer el título, una descripción, y el tipo de ciencia ficción que se le aplica, que puede ser de dos tipos: Ciencia Ficción Soft o Ciencia Ficción Hard. El título tendrá que ser único, no queremos generar confusiones. Una saga la forman libros, de los que queremos tener el título, una sinopsis, la edad mínima recomendada (que puede ser opcional), el año de publicación y el isbn, que nos permite identificarlo. Curiosamente, hay libros que pueden formar parte de varias sagas, con lo que tendremos que tenerlo en cuenta.

Cuando un libro está en una saga, es necesario tener registrado el orden en el que este libro tendría que ser leído. Y entendemos por orden, un simple valor desde 1 hasta el número de libros que componen la saga. Este número de libros que forman la saga también será interesante tenerlo en cuenta y registrarlo.

Cada libro es escrito por uno o varios autores, de los que necesariamente hemos de conocer su nombre y apellidos, la fecha de nacimiento y la fecha de defunción, en caso que esté muerto. Claro, un autor también puede escribir muchos libros, y no necesariamente tenemos que tener algún libro suyo documentado. Como una saga puede tener libros de varios autores, necesitamos saber el autor principal con el que se identifica la saga, ya que muchas veces es por este autor que podemos encontrar la saga a la que un cliente hace referencia.

Por último, para dar soporte al workflow de edición de sagas que OnlySF tiene planteado, necesitamos que una saga tenga un estado. Por ahora (quizás en el futuro habrá más), una saga podrá estar Publicada, en Borrador o Descartada. En el primer caso, Publicada, necesitaremos saber en qué fecha se ha publicado. Para el caso de Borrador, en este estado, la administradora puede crear notas sobre aspectos que quiere que sean trabajados por parte de los colaboradores. Estas notas tendrán un comentario y se podrán marcar como solucionadas cuando corresponda. Por último, cuando una saga se descarte, tendremos que guardar la fecha en la que se ha descartado.

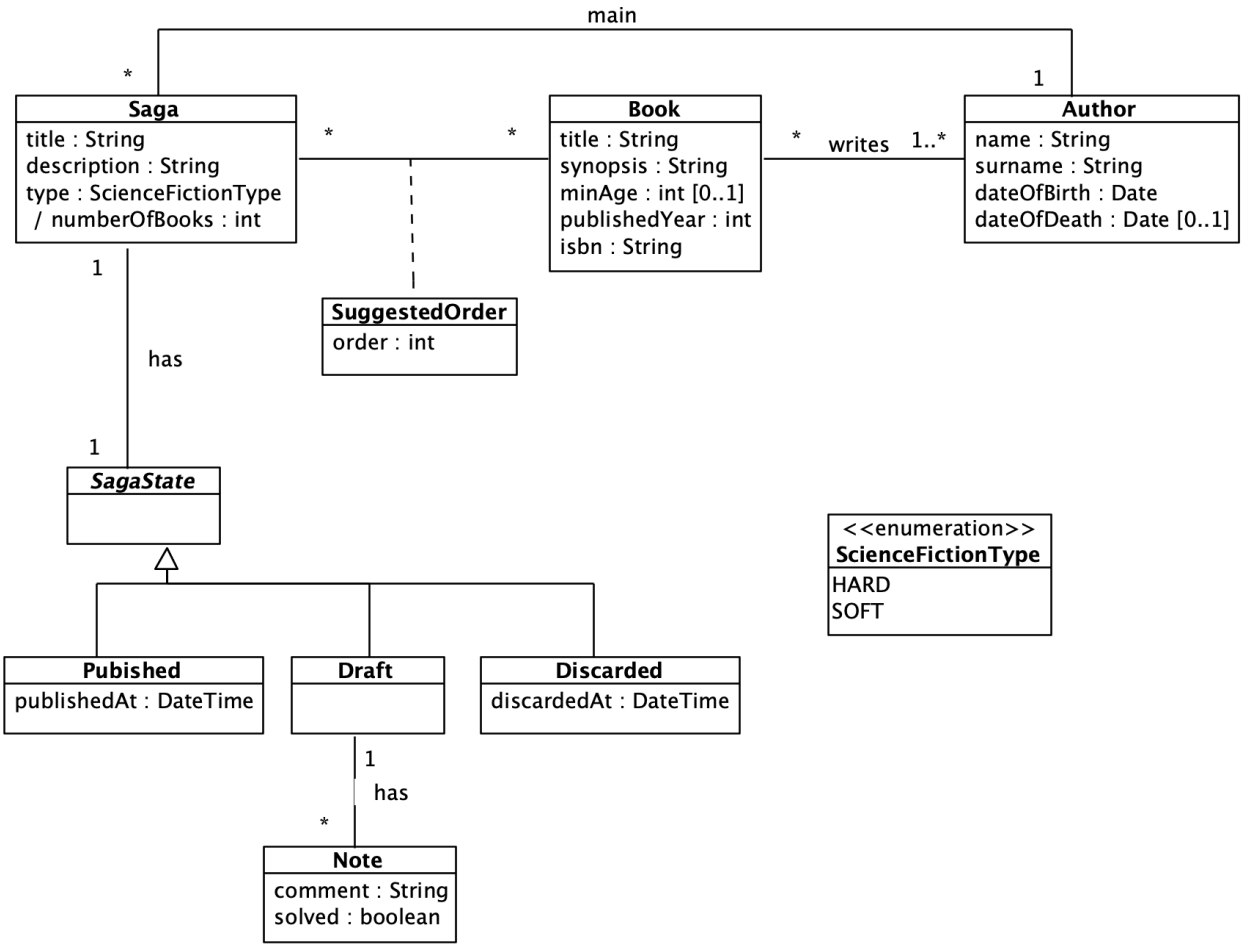
Se pide:

a) Haced el **diagrama de clases UML** del modelo de análisis a partir de la información anterior.

b) Indicad las **claves de las clases del dominio** y **otras restricciones de integridad textuales** que pueda haber. En caso de haber **información derivada**, indicad también **a partir de qué información se deriva**.

Solución

a) Diagrama de clases



Hemos optado por modelar el estado de la *Saga* creando una nueva clase. Una alternativa aceptable, aunque menos óptima, sería con el uso de herencias o con el uso de atributos opcionales en la propia clase *Saga*.

b) Claves de las clases del dominio, restricciones de integridad e información derivada

- **Claves de las clases de dominio:**
  - *Saga: title*
  - *Book*: isbn
- **Otras restricciones de integridad:**
  - Los valores de *order* en la clase asociativa *SuggestedOrder* tienen que ser valores entre 1 y el número de libros asociados con la misma Saga.
  - El autor *main* de una saga tiene que ser el autor de alguno de los libros que componen la saga.
- **Información derivada:**
  - El atributo *numberOfbooks* de la clase *Saga* es el número de libros asociados a la instancia de saga.

Pregunta 4 (25%)

Seguimos con el análisis y vamos a centrarnos ahora en los usuarios de esta aplicación. De cada usuario necesitaremos almacenar su email y un password. Cada vez que se identifique en el sistema, nos guardaremos también la fecha de este último login.

En las entrevistas anteriores, hemos visto que tendremos varios tipos de usuarios, pero también nos han indicado que un mismo usuario podrá ser de varios tipos a la vez. Así que modelamos los tipos de usuarios en forma de roles, de manera que un usuario podrá tener varios roles a la vez. De cada rol necesitaremos conocer el tipo, que es único y que coincidirá con un valor de los que se han citado en el ejercicio 2 y una descripción.

Ya sabemos que los usuarios podrán realizar comentarios a las sagas, tantos como quieran. Y también sabemos que tendremos que registrar qué sagas han sido vistas por parte de un usuario, ya que esto será una opción disponible en la interfaz gráfica para navegar entre sagas.

Lo que no sabíamos y nos han contado recientemente es que quieren que los usuarios puedan invitar a otros usuarios a formar parte de la aplicación. Cuando un usuario invita a otro usuario, se genera un código que identificará a esta invitación. En el caso que la invitación sea aceptada y el usuario invitado se registre usando el código proporcionado para ello, el sistema generará dos cupones. Estos cupones tendrán una fecha de validez límite, un porcentaje asociado de descuento (que puede ser diferente para cada usuario, con lo que necesariamente tendremos que almacenar qué cupón corresponde a cada usuario participante en la invitación, es decir, un cupón para el emisor de la invitación y el otro para el receptor de la misma), y cuando el cupón se use, tendremos que registrar la fecha en la que se ha usado.

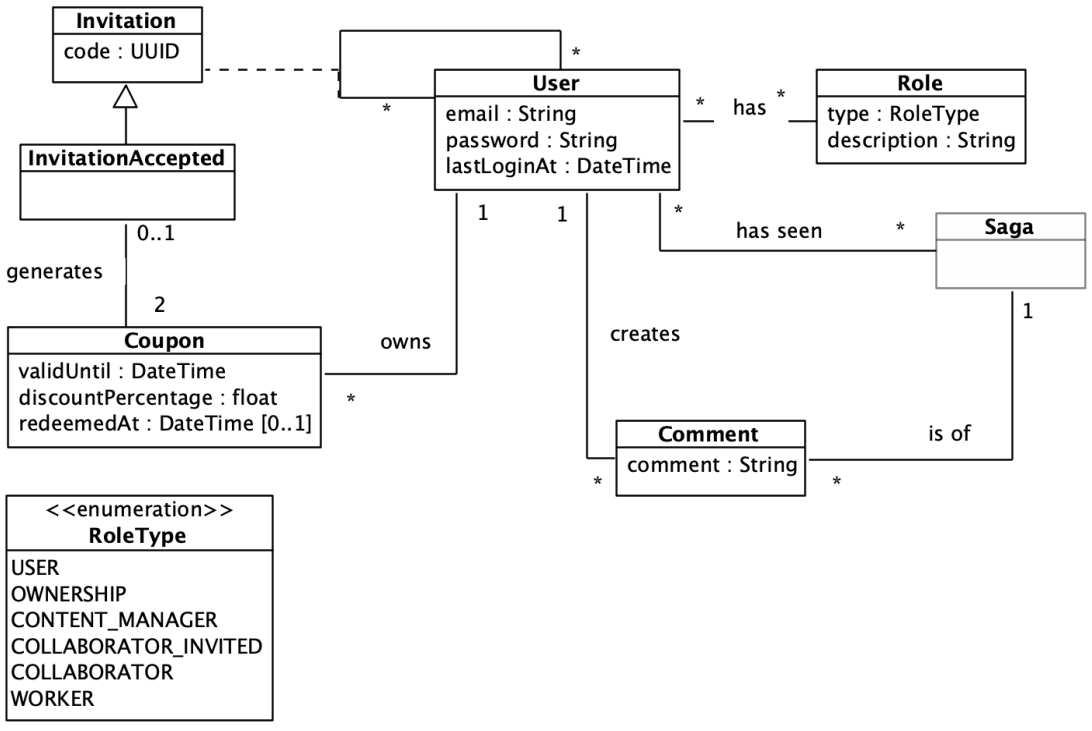
Este sistema de cupones se usará para otros flujos que no son sólo el que proviene de la invitación, con lo que un usuario podrá tener muchos cupones a lo largo de su vida como usuario.

Se pide:

a) Haced el **diagrama de clases UML** del modelo de análisis a partir de la información anterior. En caso de que tengáis que reutilizar clases del ejercicio anterior, mostradlas de nuevo en este diagrama. Si hay algún cambio en ellas especificadlo.
b) Indicad las **claves de las clases del dominio** y **otras restricciones de integridad textuales** que pueda haber. En caso de haber **información derivada**, indicad también **a partir de qué información se deriva**.

Solución

a) Diagrama de clases



La clase *Saga* es la misma que en el ejercicio anterior, pero para una mayor claridad se muestra sin atributos.

La clase *Comment* es una clase asociativa expresada en forma de clase y asociaciones (y no en forma de clase asociativa como en otros puntos del diagrama). El motivo es que necesitamos permitir que un usuario pueda hacer más de un comentario a una saga, y el modelado usando una clase asociativa no nos lo permitiría.

b) Claves de las clases del dominio, restricciones de integridad e información derivada. Señalamos únicamente las nuevas claves, restricciones e información derivada

- **Claves de las clases de dominio:**
  - User: email
  - Role: type
  - Invitation: code
- **Otras restricciones de integridad:**
  - *redeemedAt* será anterior a *validUntil* en Coupon
  - La asociación *owns* entre *User* y *Coupon*, en el caso que el cupón se haya generado vía una invitación aceptada, será con los dos usuarios que han participado en la invitación aceptada.
- **Información derivada:**
  - No hay

Pregunta 5 (20%)

Finalmente queremos centrarnos en la parte que permitirá a los trabajadores que usen la aplicación encontrar con facilidad las sagas.

Para ello añadiremos información de películas que estén basadas en uno o varios libros de los que ya teníamos registro. Una película puede tomar argumentos de varios libros, y para nuestra aplicación, solo tendremos documentadas películas de las que tenemos algún libro.

Tanto para las películas como para los libros, queremos registrar posibles alias de los mismos. Un alias sólo contendrá un nombre y, opcionalmente, el idioma en que esté expresado este alias. Los alias se pueden repetir siempre que sean de películas y libros diferentes.

Por último, y también para dar más juego a los trabajadores, hemos decidido crear una taxonomía para clasificar los libros. En la taxonomía aparecerán áreas de conocimiento y para cada área de conocimiento habrá dominios y otras áreas de conocimiento. Por ejemplo, en el área de Ciencias, habrá el dominio de las matemáticas y el área de conocimiento de ciencias de la naturaleza. Y en esta última habrá los dominios de ecología y de biología. De manera visual se puede representar de la siguiente manera:

- Ciencias *(área)*
  - Matemáticas *(dominio)*
  - Ciencias de la Naturaleza *(área)*
    - Ecología *(dominio)*
    - Biología *(dominio)*

Sea una área de conocimiento o un dominio, sólo nos interesa saber su nombre, que lo identifica.

Un libro puede tratar sobre varios dominios de la taxonomía. Para cada uno de los dominios que son tratados en un libro habrá que especificar el porcentaje. Así, un libro puede tratar en un 25% de filosofía aristotélica, en un 30% de psicología y en un 40% de matemáticas (con una suma de 100%, y podemos considerar suficiente indicar un valor entero en los porcentajes).

Con esta información, un libro quedará etiquetado en una área de conocimiento gracias a los porcentajes de los dominios, ya que la suma de los porcentajes de los dominios de cada libro nos dará una mayoría en una de las áreas.

Se pide:

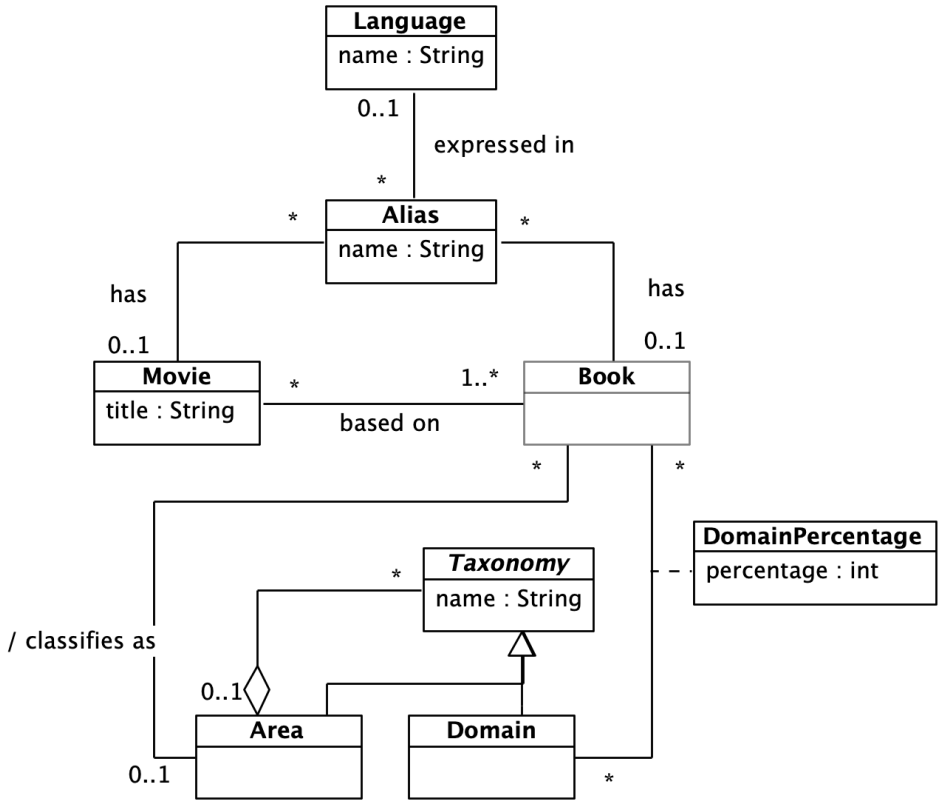
a) Haced el **diagrama de clases UML** del modelo de análisis a partir de la información anterior. En caso de que tengáis que reutilizar clases del ejercicio anterior, mostradlas de nuevo en este diagrama. Si hay algún cambio en ellas especificadlo.

b) Indicad las **claves de las clases del dominio** y otras restricciones de integridad textuales que pueda haber. En caso de haber **información derivada**, indicad también **a partir de qué información se deriva**.

Nota: Para la representación de la taxonomía podéis acceder a la información presentada en esta página: https://en.wikipedia.org/wiki/Composite_pattern, donde se describe un patrón de análisis que se puede aplicar en estos contextos como el que hemos descrito.

Solución

a) Diagrama de clases



La clase *Book* es la misma que en el ejercicio anterior, pero para una mayor claridad se muestra sin atributos.

La clase *Language* se podría modelar como un atributo en la clase *Alias*.

b) Claves de las clases del dominio, restricciones de integridad e información derivada. Señalamos únicamente las nuevas claves, restricciones e información derivada

- **Claves de las clases de dominio:**
  - Taxonomy: name
  - Language: name
- **Otras restricciones de integridad:**
  - No pueden existir dos alias con el mismo nombre que hagan referencia a una misma película o un mismo libro
- **Información derivada:**
  - La asociación *classifies as* es derivada a partir de las clases asociaciativas *DomainPercentage* evaluando la suma de porcentajes individuales.