Fase 1.2 – Domain Driven Design

Tecnologías para el Desarrollo de Aplicaciones Empresariales sobre Internet

(332)

Máster Universitario en Ingeniería Informática

Autor: Noé Ruano Gutiérrez

Identificación de elementos en el modelo de dominio

- **Entities:**

***Usuario*:** es preciso poder identificar de manera unívoca a los usuarios de la plataforma por medio de sus nombres puesto que, de no ser así, la lógica del aplicativo no permitiría llevar a cabo todas las operaciones especificadas en el documento de requisitos sobre los usuarios. Además, no es concebible un aplicativo funcional y acorde a los requisitos si los usuarios no son claramente distinguibles entre sí por medio de un identificador unívoco.

***Serie*:** al igual que ocurre con los usuarios, las series ofertadas en la plataforma deben poder distinguirse entre sí de manera unívoca puesto que, de no ser así, la facturación por la visualización de los capítulos no podría llevarse a cabo. Si no se distinguen las series entre sí no puede recuperarse el tipo de serie y, por ende, no puede asociarse ningún cargo por la visualización de un capítulo, salvo en el caso en que el usuario está suscrito a la cuota fija.

***Temporada*:** las temporadas de cada serie requieren de identificación para poder distinguirse unas de otras puesto que, de no ser así, no podrían recuperarse los capítulos de una temporada concreta de una serie.

***Capítulo*:** los capítulos requieren identificación puesto que dentro del dominio debe poder realizarse operaciones sobre capítulos concretos, como por ejemplo en el cobro por visualizaciones. A partir del capítulo se recuperan en este orden la temporada y serie a los que pertenece, de forma que pueda llevarse a cabo el cobro del importe correspondiente al tipo de serie.

***Recibo*:** un recibo no describe tan solo una característica de un usuario, sino que toma una parte activa en su ciclo de vida además de que, dentro del suyo propio, cambiará conforme se vaya computando la facturación del usuario, es decir, que las instancias de la clase Recibo no permanecerán inmutables durante su ciclo de vida.

- **Value-Objects:**

***PersonalSerie*:** no se requiere, al menos en la implementación del problema abordado, que se le asigne un identificador con el que referirse de manera unívoca a los creadores/actores de cada serie, puesto que tan solo describen una característica de la serie que no va a desempeñar un papel crítico en el funcionamiento de la lógica del modelo. Además, una vez se instancia un objeto de esta clase, no se requerirán modificaciones de este, es decir, que permanecerá inmutable durante toda su ciclo de vida.

***Cargo*:** una vez creada una instancia de la clase *Cargo*, esta permanecerá inmutable durante todo su ciclo de vida puesto que, a partir de la creación de un cargo, en un principio, la lógica de la aplicación no requiere realizar modificaciones de este.

- **Services:**

No se ha identificado ningún elemento del dominio que pueda ser clasificado como servicio. No obstante, podría contemplarse la existencia de un servicio que permita llevar a cabo las operaciones sobre las cuentas bancarias de los usuarios de cara a realizar los cobros.

Identificación de aggregates

Se tendría, por un lado, el *aggregate* que engloba las clases *Usuario*, *Recibo* y *Cargo*, siendo el *aggregate root* la clase *Usuario*, que es la *entity* cuya existencia controla el ciclo de vida del resto de elementos del *aggregate*, puesto que no se entiende la existencia de un recibo y sus correspondientes cargos si estos no pertenecen a un usuario concreto.

Por otro lado, se tendría el *aggregate* que engloba a las clases *Serie*, *PersonalSerie*, *Temporada* y *Capítulo*, siendo el *aggregate root* la clase *Serie* la cual, de nuevo, controla el ciclo de vida del resto de elementos del *aggregate*.

Refactorización del modelo de dominio

El modelo diseñado inicialmente, presentado en la Figura 1, no cumplía con uno de los principios esenciales del diseño dirigido por modelos, y es que existían referencias de elementos internos a un *aggregate*, a elementos externos a ese *aggregate* pero sin referenciar al *aggregate root*. Estas referencias son las que apuntan a la clase *Capítulo* desde las clases *Usuario* y *Cargo*.

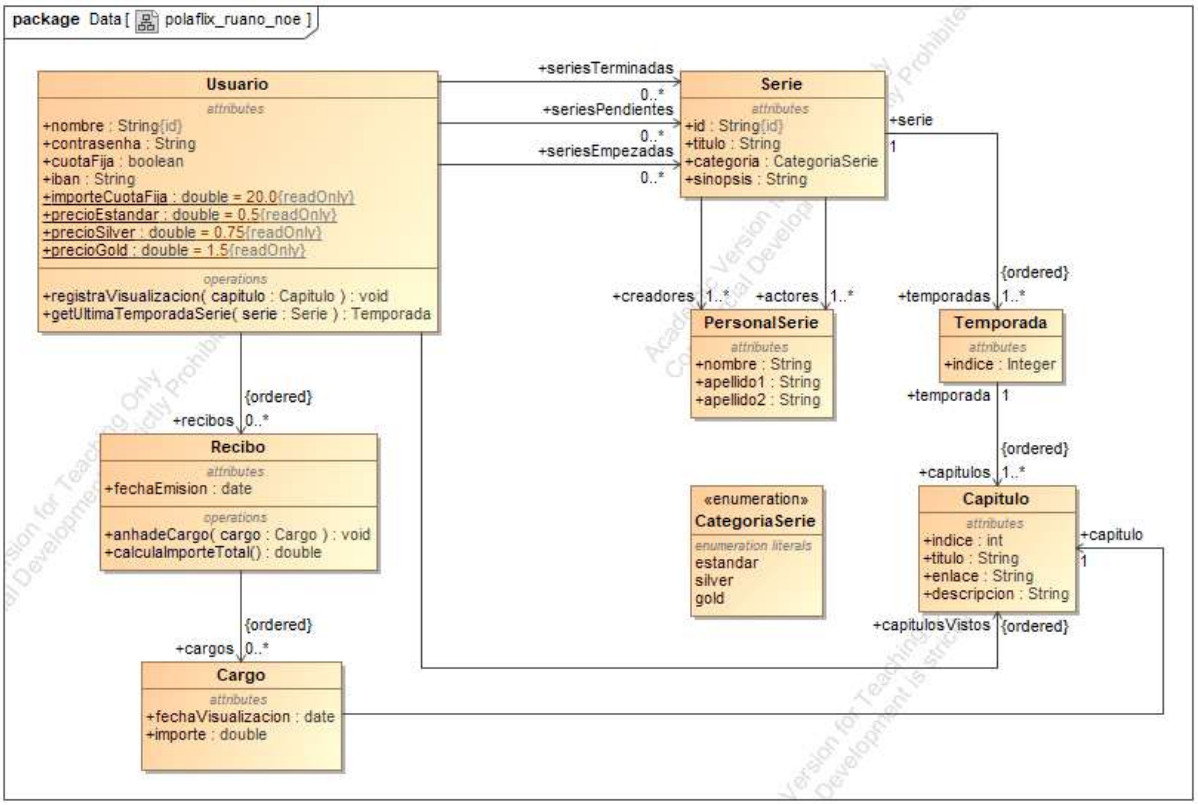

Figura 1: Primera versión del modelo de dominio diseñado

Se muestra en la Figura 2 el modelo refactorizado conforme a las reglas del diseño dirigido por modelos, prescindiendo de cualquier tipo de referencia por parte de elementos internos a los *aggregates*, hacia elementos de otros *aggregates* de manera directa sin pasar por el *aggregate root*.

Se eliminan las mencionadas referencias incorporando a la clase *Capítulo* un id de forma que la clase *Cargo* tan solo deba incorporar un atributo que albergue el id del capítulo visualizado. Además, dentro de la clase *Usuario* se mantiene una estructura de datos con los id's de los capítulos ya visualizados.

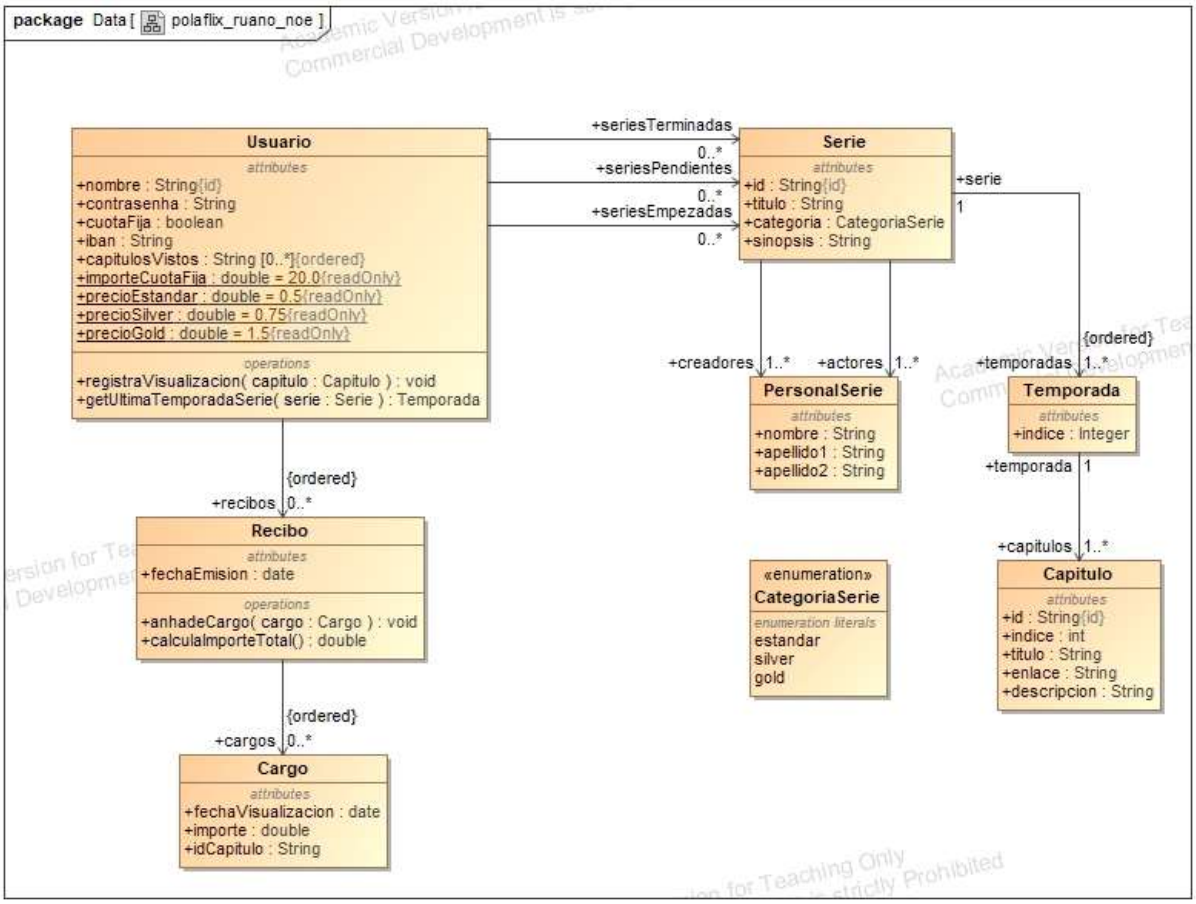

Figura 2: Modelo de dominio refactorizado