

# Ingeniería web

Tema 3

# Ingeniería web

## Tema 3

- Análisis y diseño de aplicaciones web
- Ingeniería web guiada por modelos
  - ¿Qué son los modelos?
  - Necesidad de modelos.
- Ejemplos de desarrollo:
  - Equipos de fútbol
  - ...

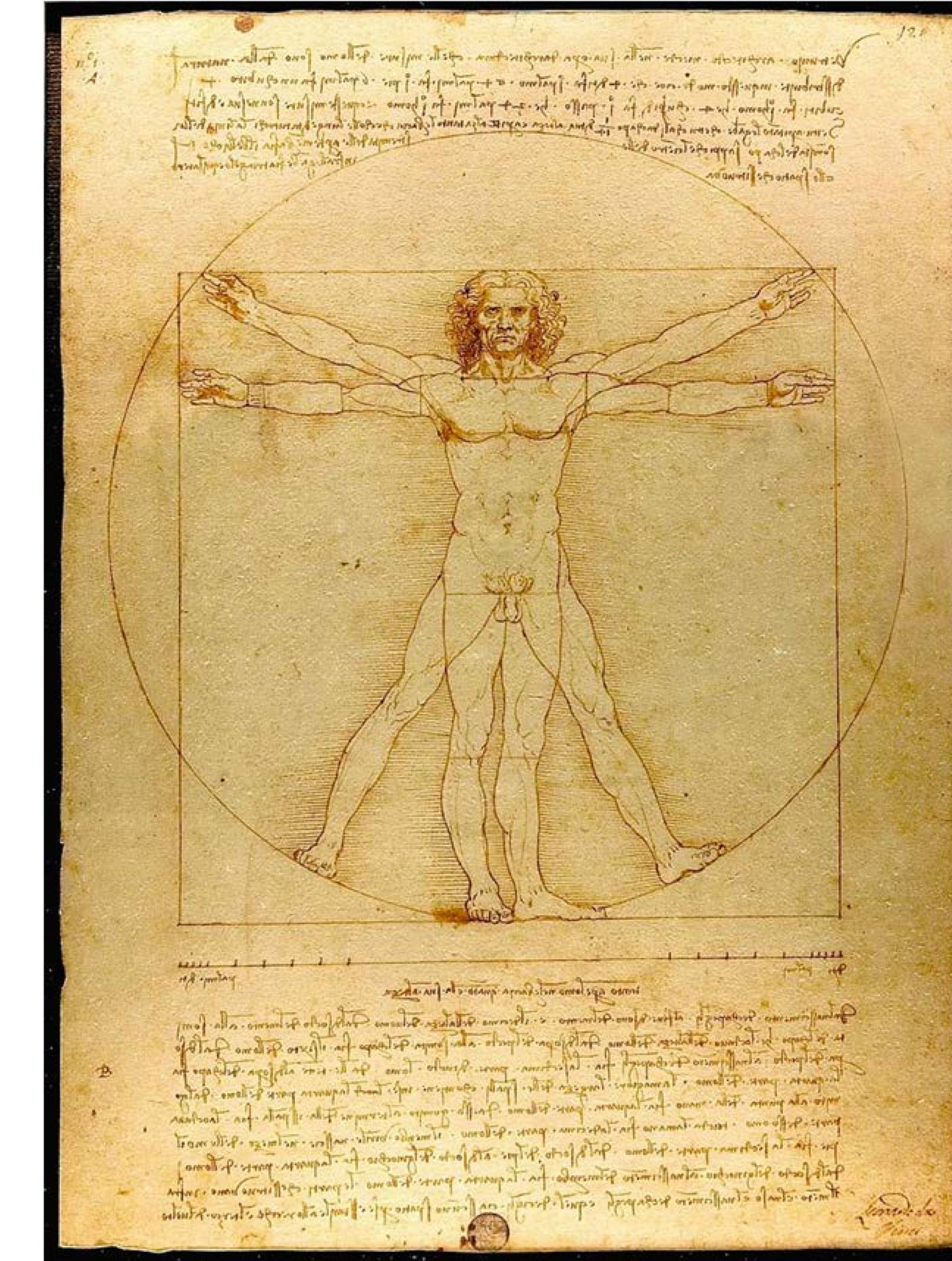
# 1. Desarrollo Dirigido por Modelos (MDD)

# Evolución del desarrollo software

- Programación en Ensamblador
  - X86
- Programación Estructurada
  - Pascal, C
- Orientación a Objetos
  - C++, Java
- Componentes software
  - J2EE, .NET
- Orientación a Aspectos
  - AspectJ, Spring Framework AOP
- Arquitecturas orientadas a Servicios
  - WSDL, SOAP

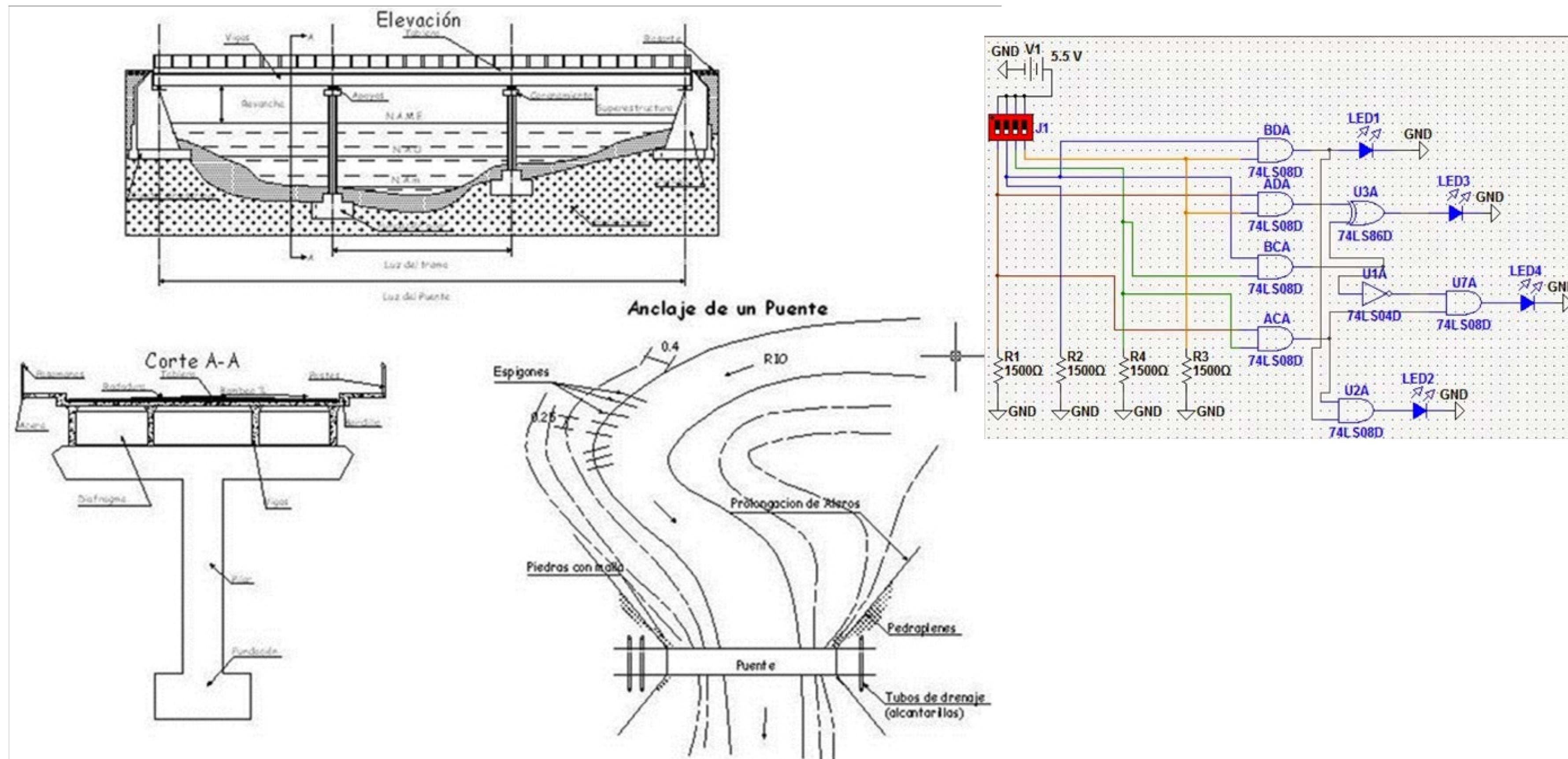
# Modelos en Ingeniería

- Hasta ahora la programación siempre ha sido el centro de atención
- Al igual que en otras ingenierías ¡hay que aumentar el nivel de abstracción!
- Los modelos ayudan a construir sistemas más complejos
- Tan antiguos como las ingenierías por ejemplo el Hombre de Vitruvio de Leonardo da Vinci



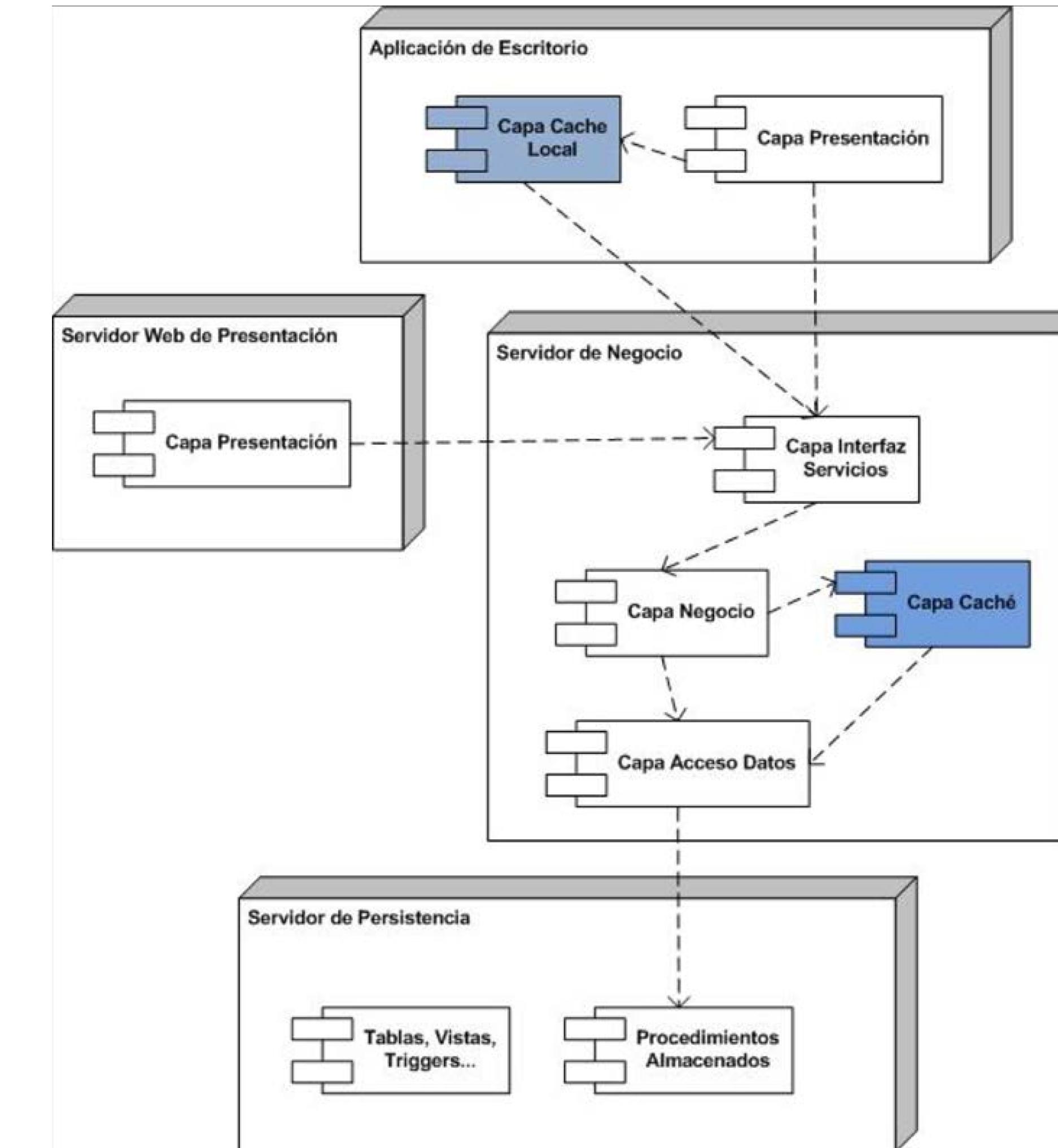
# Modelos en Ingeniería

- Los ingenieros “tradicionales” siempre construyen modelos antes de construir sus obras y artefactos



# Modelos en Ingeniería

- También en Ingeniería del Software



# Los modelos sirven para...

Especificar el sistema

Estructura, comportamiento, etc.

Comunicarse con los distintos stakeholders

Comprender el sistema (si ya existe)

Razonar y validar el sistema

Detectar errores y omisiones en el diseño

Prototipado (ejecutar el modelo)

Inferir y demostrar propiedades

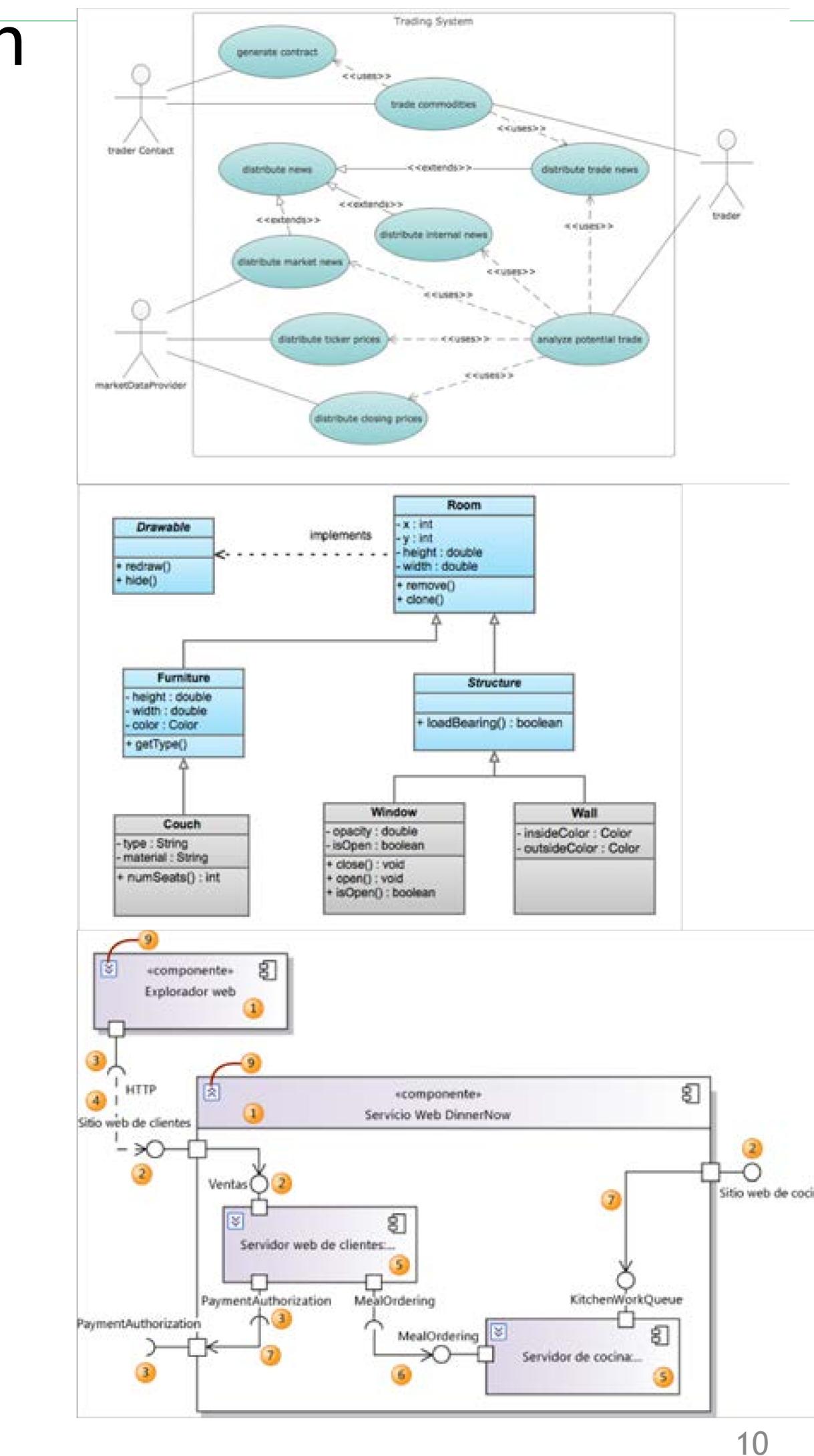
Guiar la implementación

# Características de los modelos

- Abstractos
  - Enfatizan ciertos aspectos, mientras que ocultan otros
- Comprensibles
  - Expresados en un lenguaje comprensible por los usuarios y *stakeholders*
- Precisos
  - Fieles representaciones del objeto o sistema modelado
- Predictivos
  - Deben de poder ser usados para inferir conclusiones correctas
- Baratos
  - Más fáciles y baratos de construir y estudiar que el propio sistema

# ¿Qué es un modelo de software?

- Descripción o especificación (de parte) de un sistema *software* desde un determinado punto de vista
- Una representación de parte de una funcionalidad, estructura o comportamiento de un sistema (Miller & Mukerji, 2001)
- Descripción o especificación de un sistema y su entorno para un determinado propósito. Los modelos suelen ser una combinación de esquemas y texto (Object Management Group, 2014)
- Una serie de declaraciones sobre un sistema, considerando una declaración una expresión que puede ser considerada verdadera o falsa (Seidewitz, 2003)



# Limitaciones actuales de los modelos de software

Solo se usan como documentación  
¡Que además no se actualiza!

Salto entre el modelo y la implementación del sistema

Grandes diferencias semánticas en los lenguajes respectivos

No hay herramientas de propagación automática de cambios:

Cambios en el modelo no se reflejan en el código  
Cambios en el código no se reflejan en el modelo (el modelo no vuelve a usarse jamás tras la primera implementación)

# Limitaciones actuales de los modelos de software

Los distintos modelos del sistema no se armonizan

Suponen vistas de un mismo sistema, pero no hay forma de relacionarlas

No hay herramientas de integración de modelos

Cada lenguaje de vista tiene una semántica distinta del resto

No hay ni lenguajes ni herramientas para manejar modelos Solo editores, pero no hay compiladores, optimizadores, validadores, transformadores de modelos, etc.

¿Se está realmente hablando de Ingeniería del Software?

# Software y modelos

Software has the rare property that it allows us to directly evolve models into full-fledged implementations without changing the engineering medium, tools, or methods

Bran Selic and John Hogg

- Esto facilita enormemente garantizar la fiabilidad entre los modelos y los sistemas producidos, puesto que todos viven en el mismo mundo.
- **Por tanto: El modelo es la implementación**
- **Pero para ello:** Solo si el modelo contiene toda la información necesaria para producir el sistema

# ¿Qué es MDD?

- MDD - *Model Driven Development*
- Un enfoque de desarrollo de *software* donde las entidades de primer nivel son los modelos y las transformaciones de modelos
  - Frente a los programas y los compiladores, que constituyeron el paradigma análogo hace treinta años
- MDD implica la generación (casi) automática de implementaciones a partir de modelos
- En MDD son claves los lenguajes, tanto de modelado como de transformación de modelos. Los modelos son conformes a meta-modelos
- MDA es la propuesta para MDD que hace OMG, usando sus estándares
  - MOF, UML, OCL, XMI, QVT
  - MOF y UML permiten definir nuevas familias de lenguajes

# Desarrollo dirigido por modelos

- El objetivo del MDD es *elevar el nivel de abstracción* (*mediante herramientas*) para proporcionar *altos* niveles de *automatización* (*en dominios específicos*)
- MDD **utiliza modelos** para capturar información de **alto nivel** y **automatizar su implementación** compilando modelos para *producir ejecutables* o usándolos para facilitar/guiar el desarrollo manual
- MDD puede incluso facilitar la *automatización* de actividades como la *depuración* y la *configuración* automática del *software*

# Desarrollo dirigido por modelos

## Lenguajes específicos del dominio (DSL/DSM)

- Desafortunadamente los *modelos* se han utilizado principalmente *como documentación* para consumo humano y no para ser procesables por máquinas
- Las **Fábricas de Software** están interesadas en modelos que puedan ser procesados por herramientas y proponen usarlos de la misma forma en la que se usa el código fuente. Estos modelos deben ser *precisos*
- Para elevar el nivel de abstracción, un lenguaje de modelado debe estar *orientado a dominios más reducidos* que aquellos que permite modelar un lenguaje de propósito general

# Desarrollo dirigido por modelos

## Lenguajes específicos del dominio (DSL/DSM)

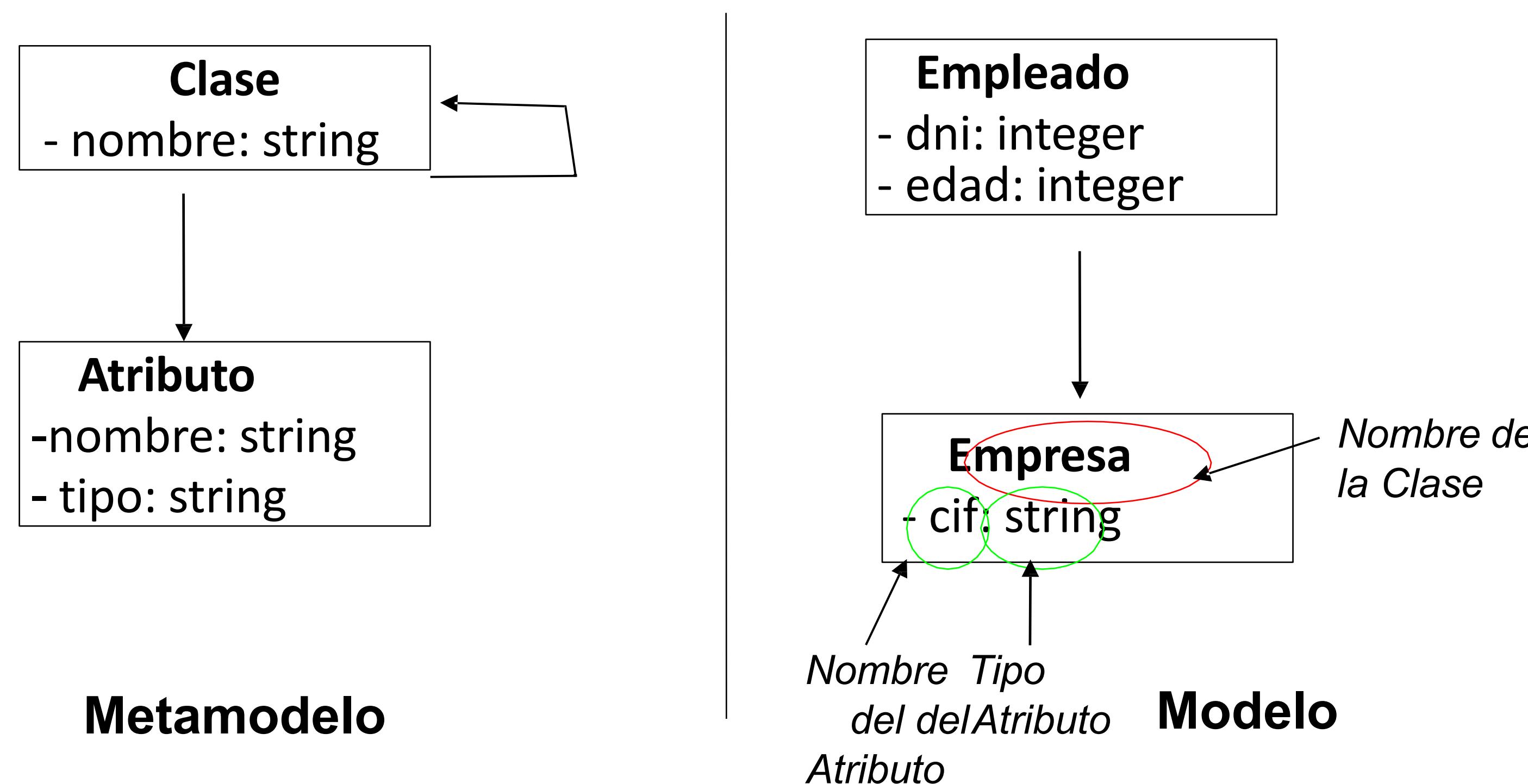
- El **propósito** para el que el lenguaje se ha diseñado debe **explicitarse**, de forma que un observador familiar con el dominio pueda evaluar el lenguaje
- El lenguaje debe usar **términos familiares** a la gente que trabaja con el **dominio**
- La **notación** del lenguaje, sea gráfica o textual, debe ser **fácil de usar**
- El lenguaje debe **tener una gramática** que gobierne la forma en la que se pueden combinar los conceptos **para construir expresiones**
- El significado de cada expresión bien formada debe estar bien definido, de forma que los usuarios puedan **construir modelos** que otros usuarios puedan **entender** y que las herramientas puedan **generar implementaciones** válidas a partir de los modelos
- Un lenguaje que cumple estos criterios se llama ***Domain-Specific Language (DSL)***, porque modela conceptos de un dominio específico. Un **DSL** se define con **mayor rigor** que un lenguaje de propósito general

# Metamodelado

- Un metamodelo describe la estructura posible de los modelos (o lenguajes) de forma abstracta
  - Define las construcciones o elementos de un lenguaje de modelado y sus relaciones
  - Define las restricciones y las reglas de modelado sobre los elementos
  - No define la sintaxis concreta del lenguaje
- Un metamodelo define la sintaxis abstracta y la semántica estática de un lenguaje de modelado

# Metamodelado

Un modelo para describir un lenguaje de modelado



# Metamodelado

- Los metamodelos y los modelos tienen una relación clase-instanci
- Cada modelo es una instancia de un metamodelo
- Para definir un metamodelo se necesita un lenguaje de metamodelado que se describe con un meta metamodelo

# Hacia MOF...

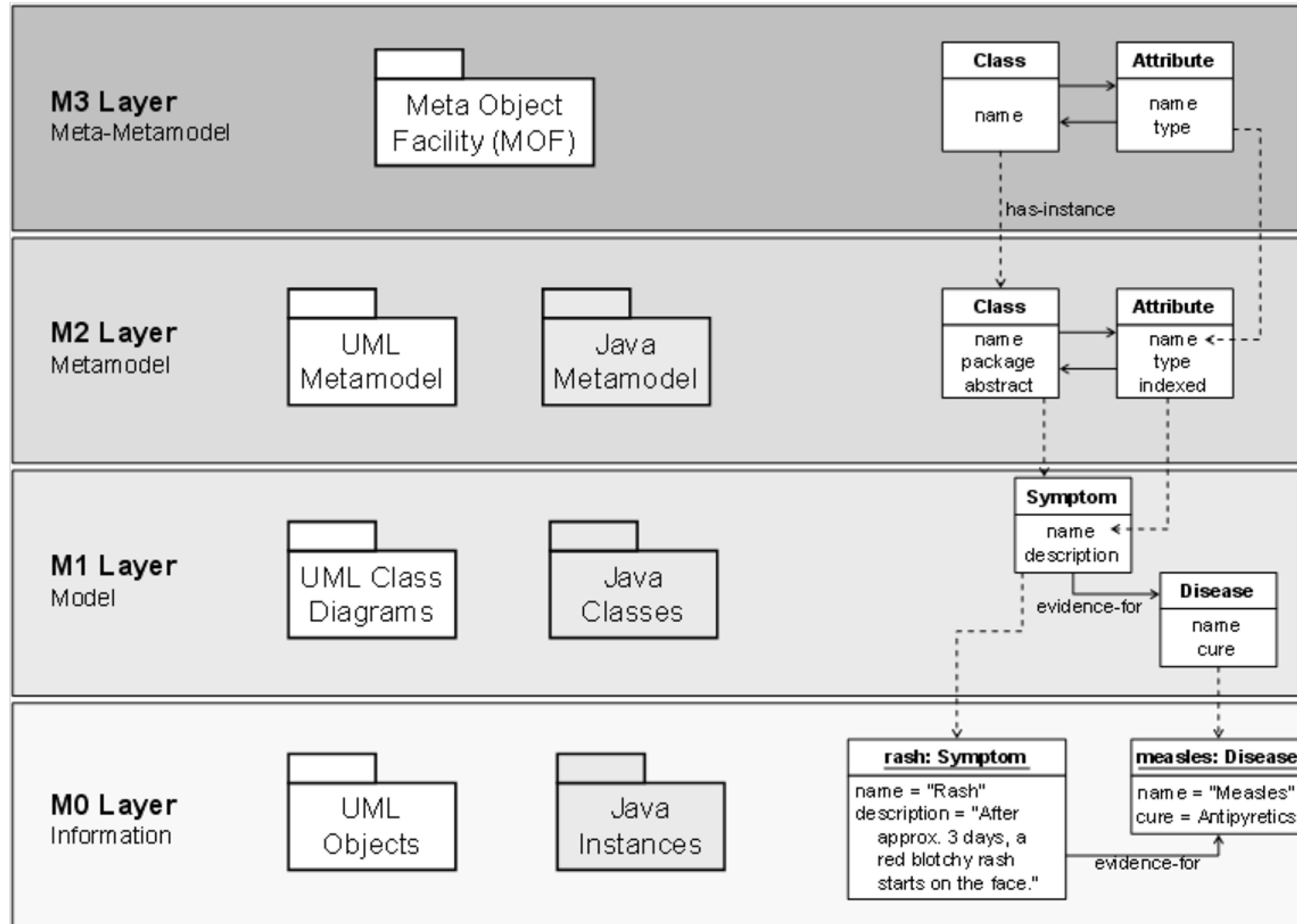
- ¿Cómo definir un lenguaje de metamodelado?
  - Con un lenguaje de meta metamodelado
  - ¿Y cómo definir un lenguaje de meta metamodelado?
    - Con un lenguaje de meta metametamodelado
    - ¿Y cómo definir...?
- ¿Hasta cuándo seguir?
  - Hasta disponer de un lenguaje capaz de describir cualquier otro lenguaje (¡¡incluso a él mismo!!)

# Meta Object Facility (MOF)

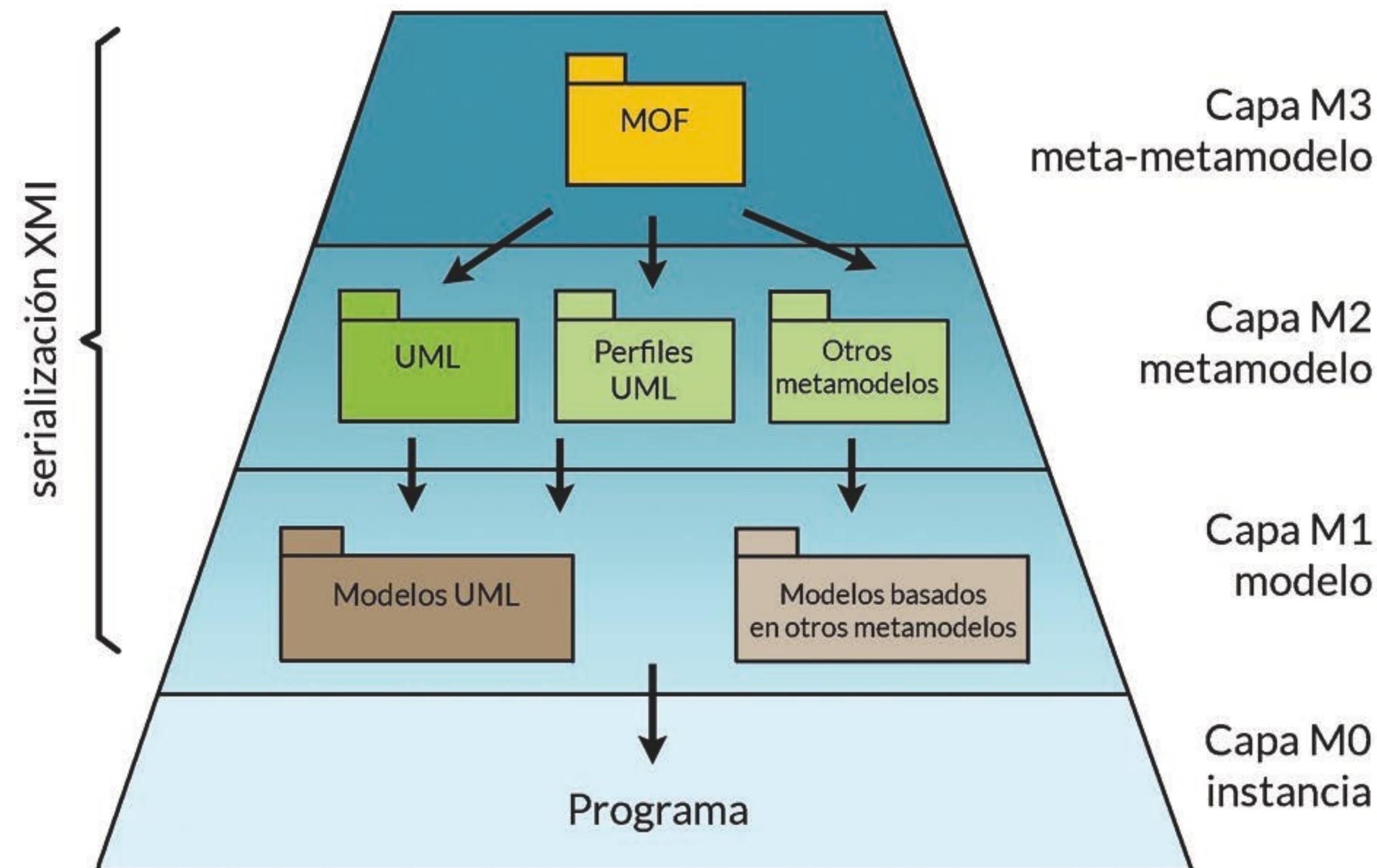


- Objetivo: Lenguaje “estándar” para especificar metamodelos
- Subconjunto de UML
- Se utiliza para definir los metamodelos de OMG (como por ejemplo el de UML)
- Versión actual: 2.5.1 (basado en UML 2)
  - <http://www.omg.org/spec/MOF/2.5.1/>
  - (Object Management Group, 2017)

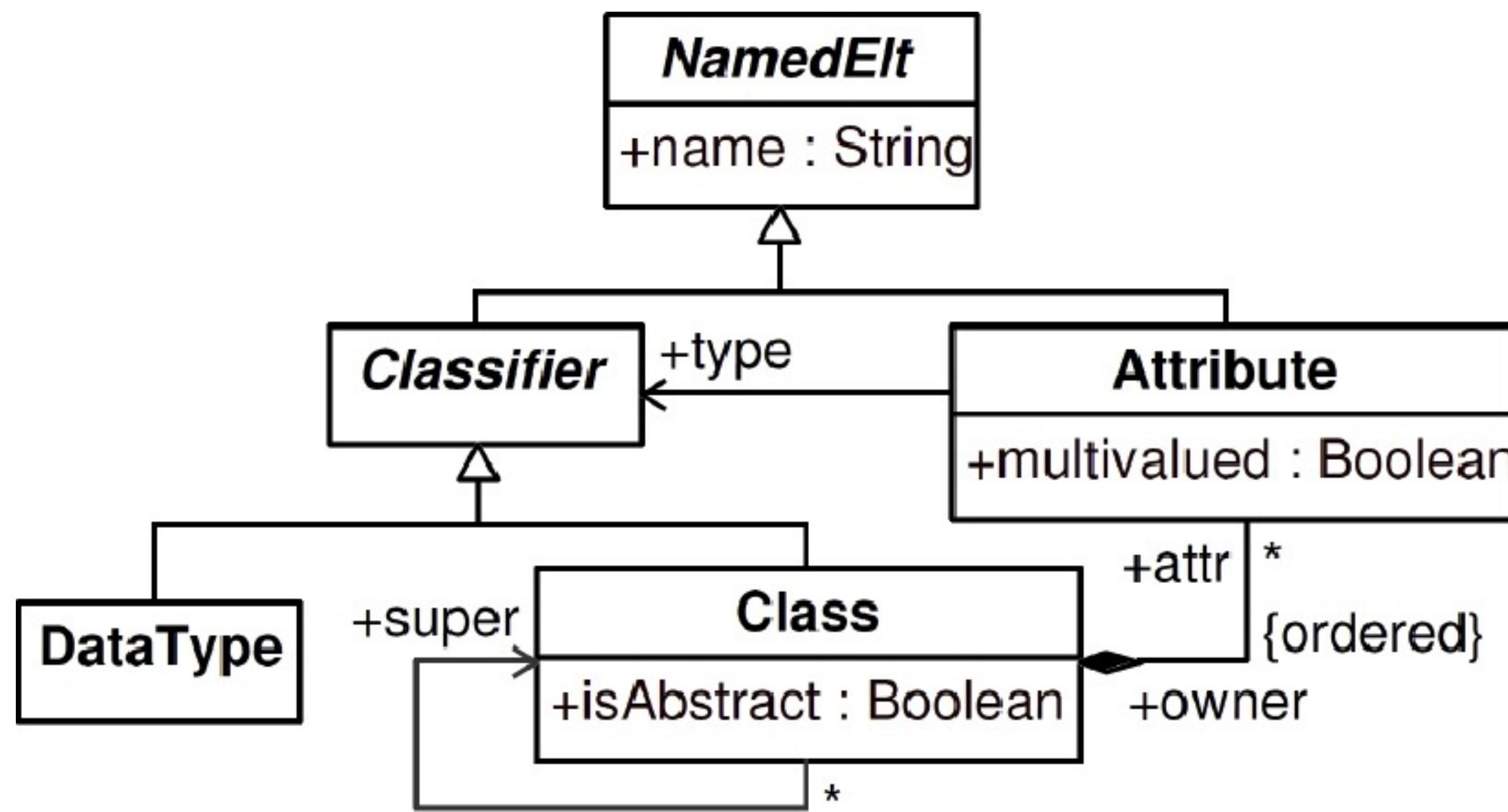
# Meta Object Facility (MOF)



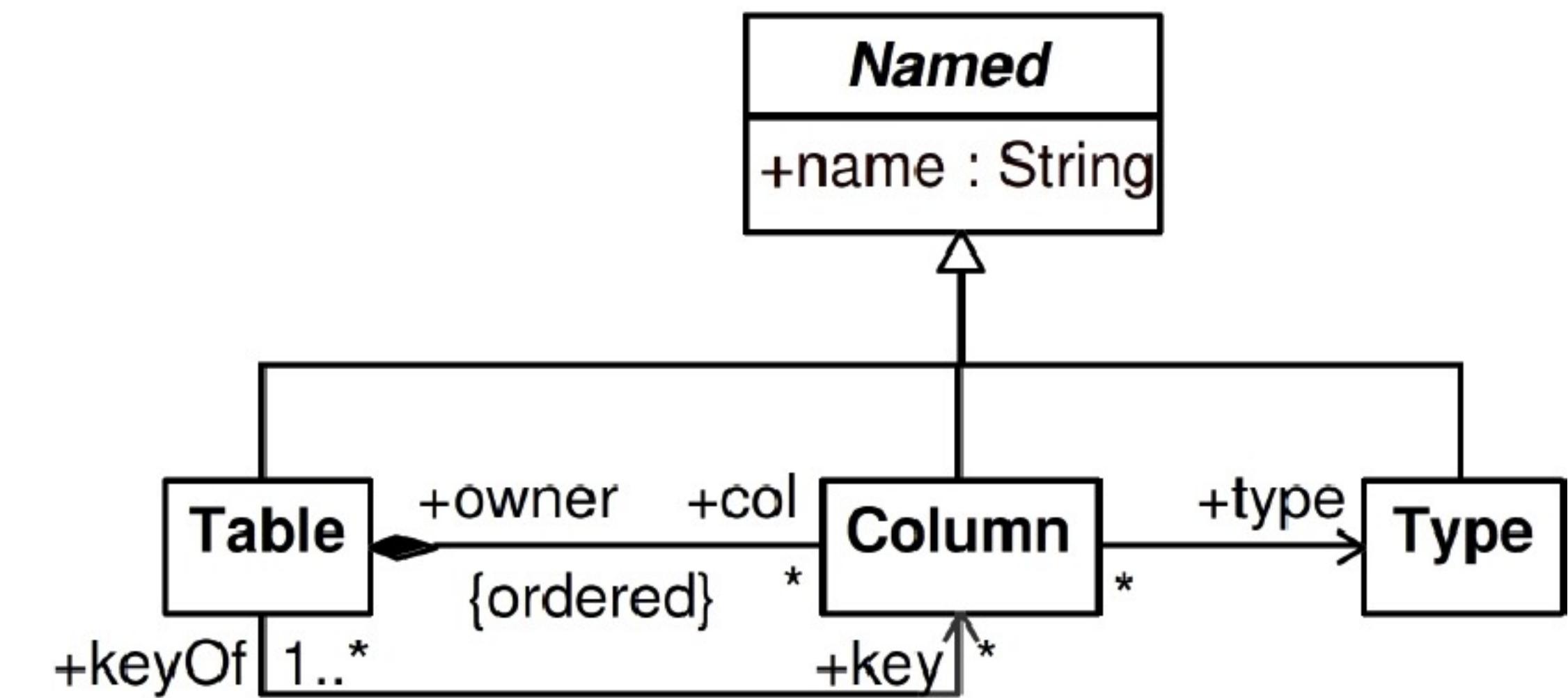
# Meta Object Facility (MOF)



# MOF



(a) Source metamodel.



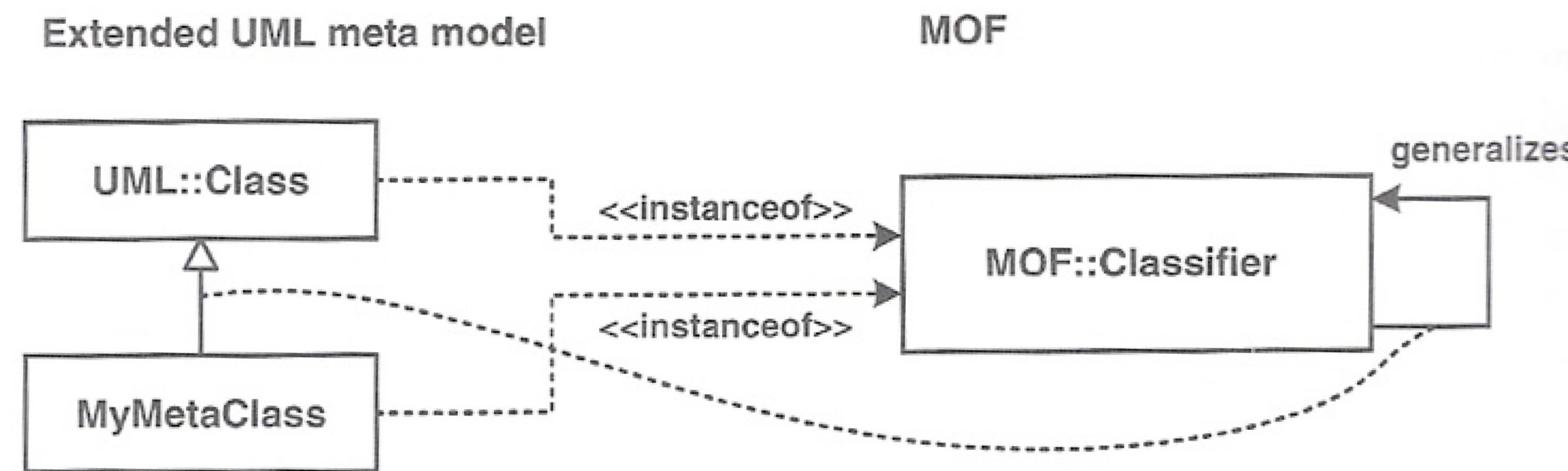
(b) Target metamodel.

# Extensión de UML

- Extensión basada en el Metamodelo
- Extensión mediante Perfiles UML
  - Mediante UML 1.X
  - Mediante UML 2.X

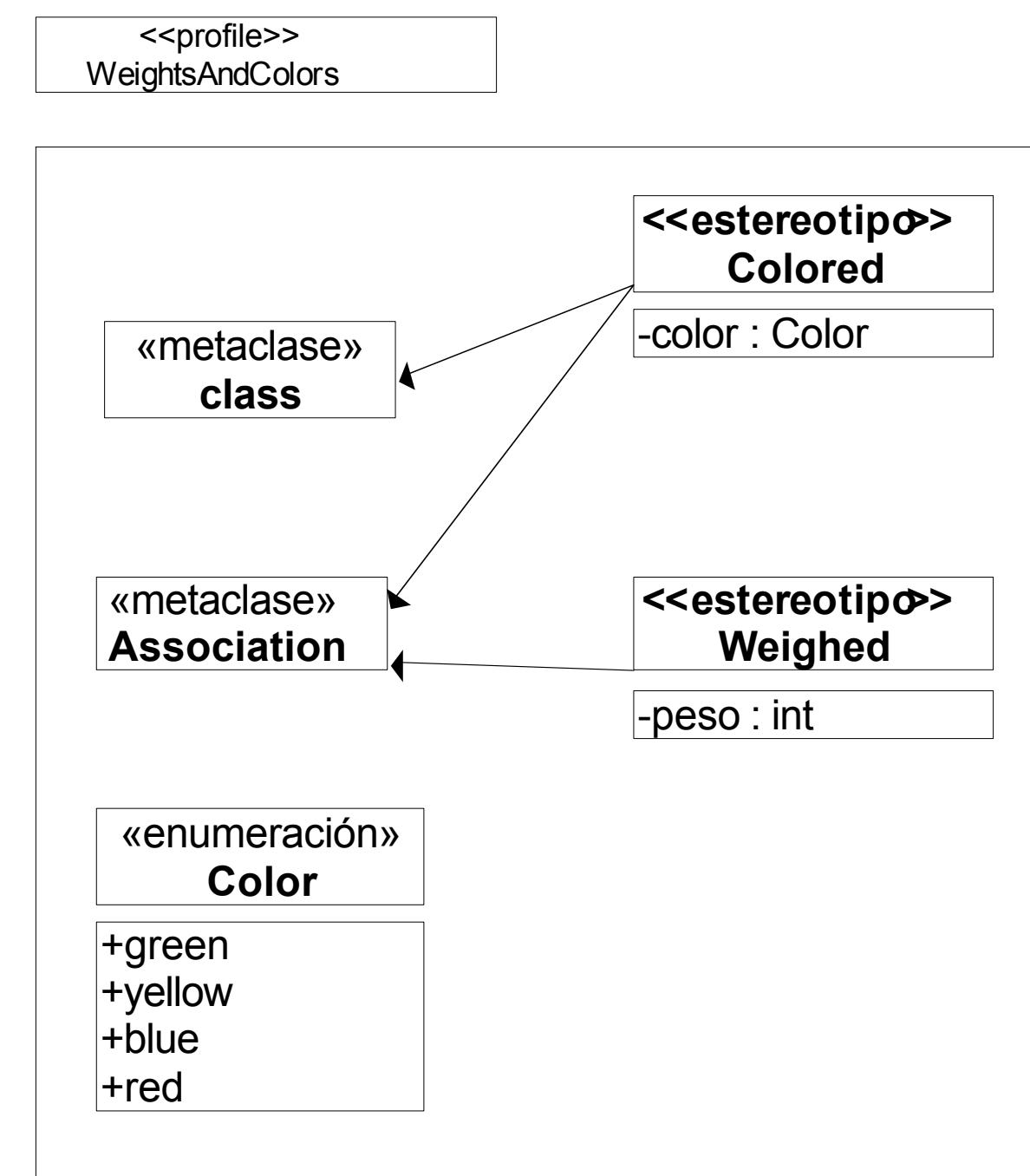
# Extensión basada en el metamodelo

- Este tipo de extensión extiende el metamodelo UML
- La herramienta de soporte debe poder manipular un metamodelo de UML basado en MOF



# Perfiles UML

- Un **perfil UML** es una agrupación de elementos de modelado (UML) que han sido **adaptados** para un propósito específico



# Perfiles UML

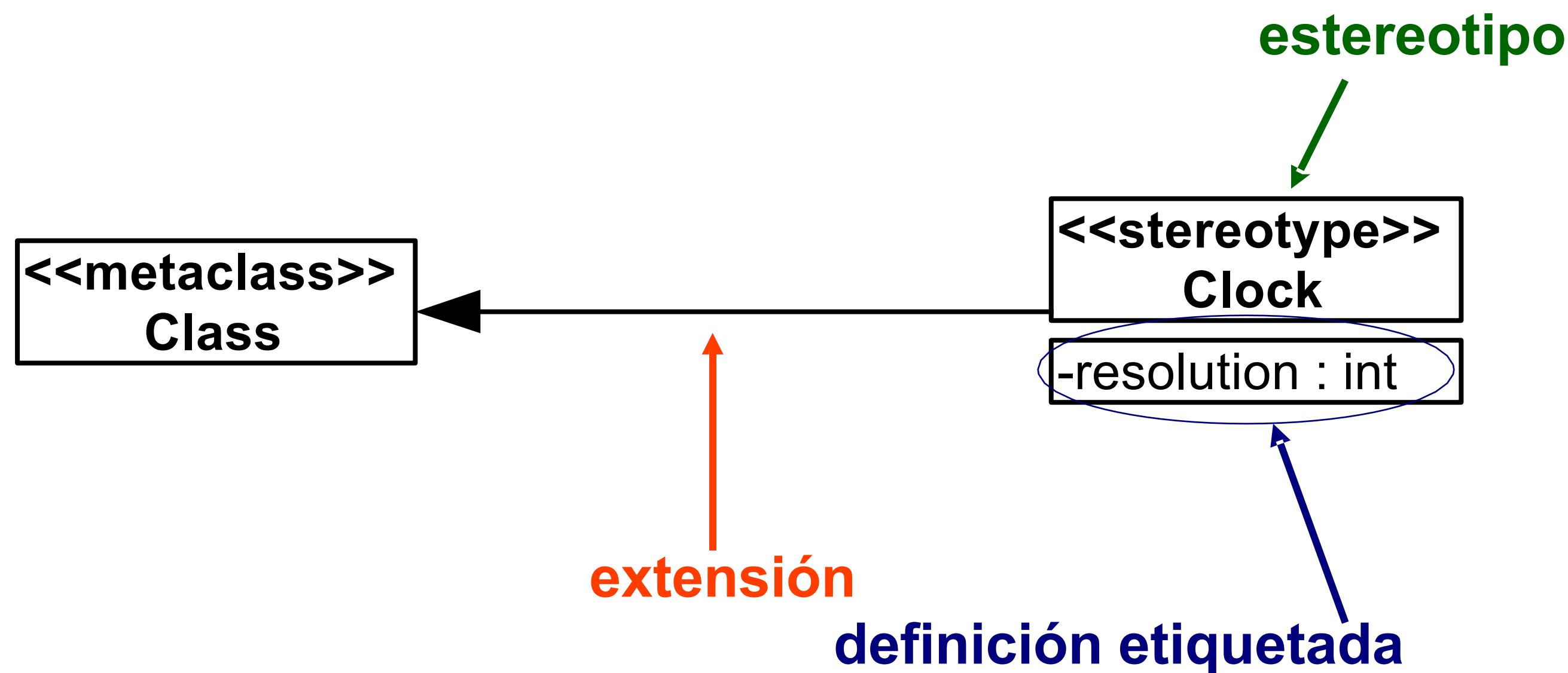
- El **objetivo** de los perfiles es
  - Proporcionar un mecanismo directo para **adaptar** un metamodelo existente con constructores que son específicos para un dominio, plataforma o método particular. Dichas adaptaciones se agrupan en un perfil
- **No es posible** **eliminar** ninguna restricción de los metamodelos de UML, pero **es posible** **añadir** nuevas restricciones que son específicas para el perfil

# Perfiles UML

- Elementos que se utilizan en la definición de perfiles
  - **Estereotipo.** Define cómo se puede extender una metacategoría de UML
  - **Extensión.** Se utiliza para indicar que las propiedades de una metacategoría se extienden a través de un estereotipo
  - **Definición Etiquetada.** Especifican nuevos tipos de propiedades que pueden asociarse a elementos de modelado
  - **Restricción.** Pueden asociarse a cualquier elemento de modelado para redefinir su semántica

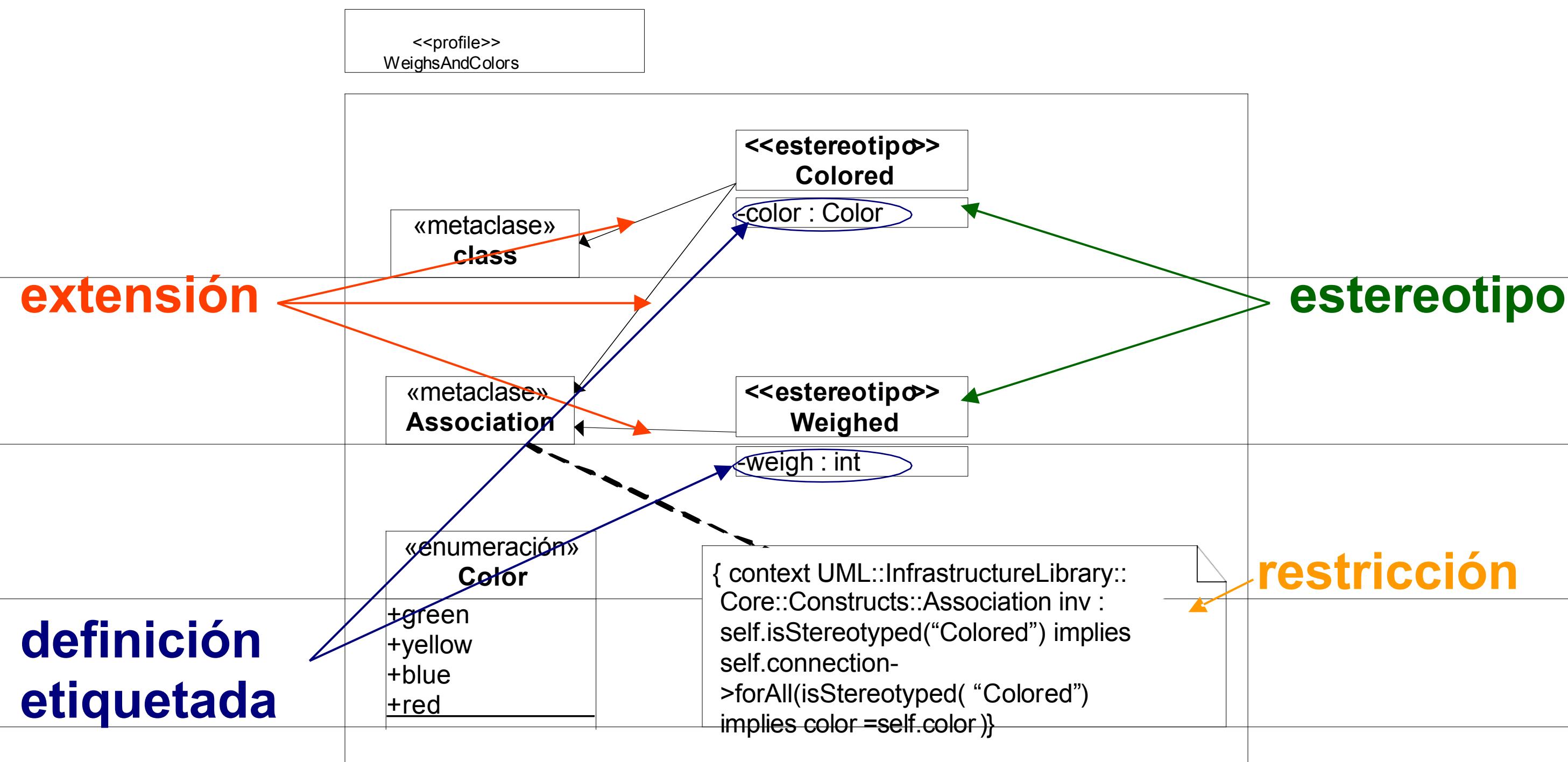
# Perfiles UML

- Ejemplo sencillo de un Perfil UML



# Perfiles UML

- Un ejemplo más complejo



# Perfiles UML

- **Estereotipos**

- **<<Colored>>** Proporcionan color a un elemento UML. Solo las clases y las asociaciones pueden colorearse
  - **Definición Etiquetada:** `color`, de tipo Color (tipo Enumerado definido en el perfil). Indica el color de cada clase o asociación que haya sido etiquetada como Colored
- **<<Weighed>>** Proporcionan peso a un elemento UML. Solo las asociaciones pueden tener asociado un peso
  - **Definición Etiquetada:** `weigh`, de tipo integer e indica el peso de cada asociación que haya sido estereotipada como Weighted

- **Restricción sobre la Asociación**

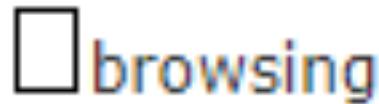
- *“Si dos o más clases están unidas por una asociación coloreada, el color de las clases debe coincidir con el de la asociación”*

# **UWE: diagramas y estereotipos**

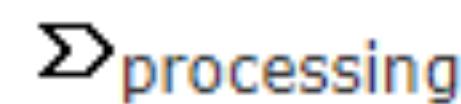
Modelado de requisitos:

Casos de uso:

Estereotipos Browsing and processing



browsing



processing

Estereotipo Browsing: No hay modificación de datos de persistencia.

Estereotipo processing: SI hay modificación de datos de persistencia.

# UWE

Modelado de requisitos:

Diagrama de actividades:  
cada caso de uso: descrito más detalladamente mediante un proceso.

Las acciones parte de un caso de uso así como los datos presentados al usuario y aquellos requeridos como entrada de datos pueden ser modelados con precisión como actividades.

>UserAction	SystemAction
DisplayAction	NavigationAction
DisplayPin	InteractionPin

## Diagrama de actividades:

Esterotipos «user Action» y «system Action» : análogamente al flujo de procesos.

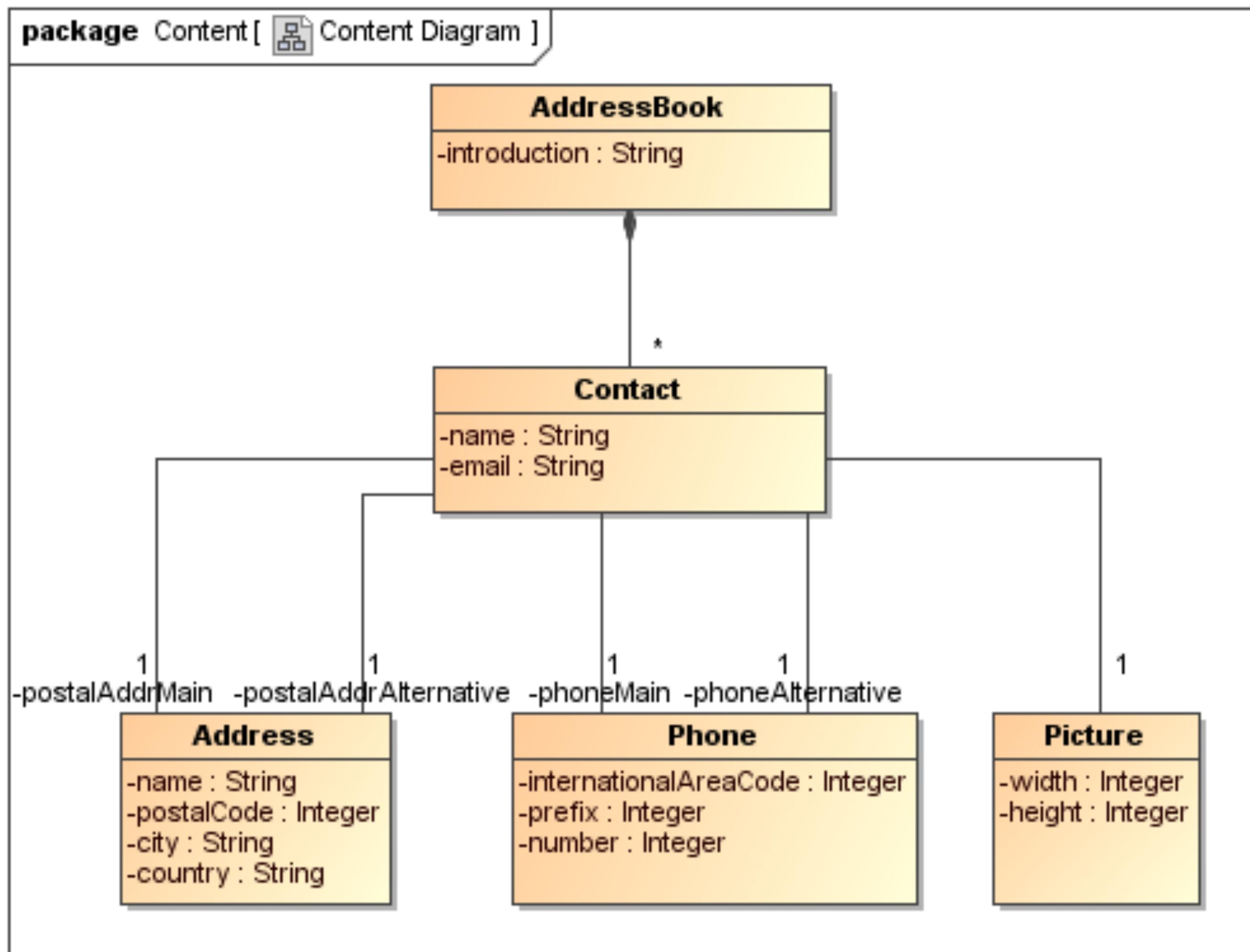
El estereotipo «user Action» es usado para indicar interacciones de usuario en la página web iniciando un proceso o respondiendo to un explícito requisito de información.

«system Action» describe acciones que son ejecutadas por el sistema.

el estereotipo «display Action», mientras que los dos pines de acción estereotipados «interaction Pin» y «display Pin» son usados para modelar la entrada y la salida de datos.

Finalmente el estereotipo «navigationAction», puede ser usado para modelar opciones de navegación y los elementos asociados de presentación.

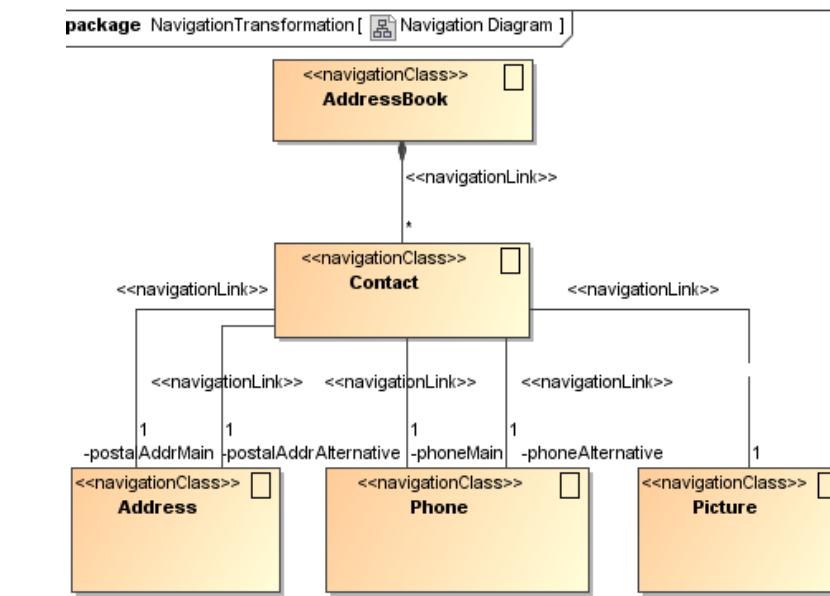
## Diagrama de “contenido”: Diagrama de clases “normal”:



# UWE

En un sistema para la web es útil saber cómo están enlazadas las páginas. Necesitamos un diagrama conteniendo nodos (nodes) y enlaces (links).

UWE provee diferentes estereotipos, ejemplo Para los nodos y enlaces son usados los estereotipos «navigationClass» y «navigationLink»:



## nombres de estereotipos y sus íconos

<input type="checkbox"/> clase de navegación		menú
<input type="checkbox"/> índice		pregunta
<input type="checkbox"/> visita guiada		clase de proceso
<input type="checkbox"/> nodo externo		

# UWE Modelo de presentación

El modelo de navegación no indica cómo aparece cada elemento en la web. Para ello se usa el modelo de presentación en ellos para expresar, que elemento está ubicado en qué parte de una web.

Estereotipos UWE:

## nombres de estereotipos y sus iconos

	grupo de presentación		página de presentación
	texto		entrada de texto
	ancla		fileUpload
	botón		imagen
	formulario		componente de cliente
	alternativas de presentación		selección

# UWE Modelo de procesos

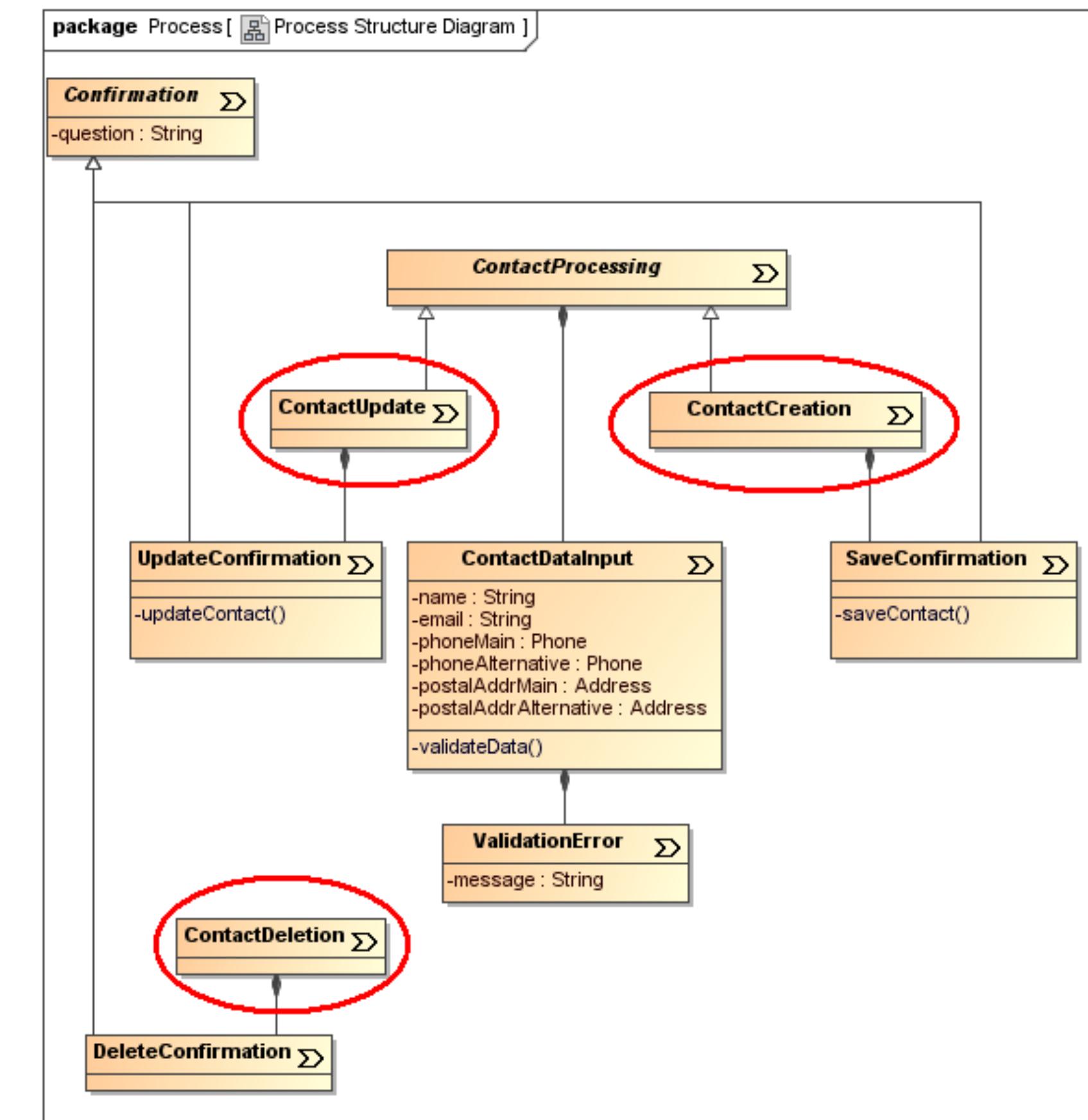
no hemos hablado en ningún momento de qué aspecto tienen las acciones de nuestras clases de proceso.

El Modelo de Proceso comprende:

el Modelo de Estructura del Proceso que describe las relaciones entre las diferentes clases de proceso y

el Modelo de Flujo del Proceso que especifica las actividades conectadas con cada «processClass».

escribir las relaciones entre las diferentes clases de proceso, se usa un diagrama de clases



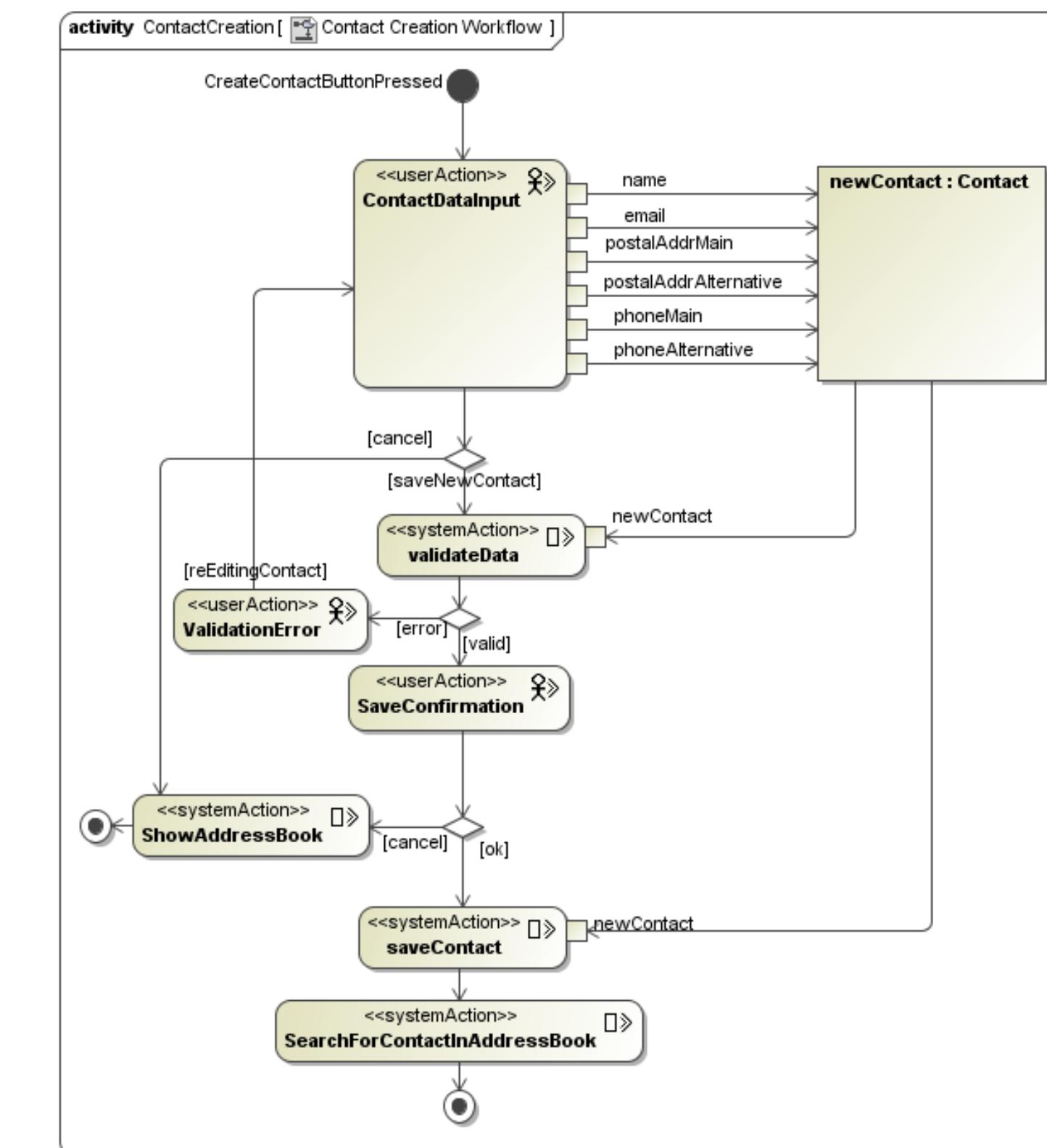
# UWE Modelo de flujo de procesos

Diagrama de actividades, describiendo el comportamiento de una clase de proceso, por ejemplo que sucede en detalle, cuando el usuario navega a una clase de proceso  
El estereotipo «user Action» es usado para indicar interacciones de usuario con la página web iniciando un proceso o respondiendo a un requerimiento explícito de información. Por el contrario, «system Action» describe acciones, que son ejecutadas por el sistema.

## nombres de estereotipos y sus iconos

♀» acción de usuario

□» acción de sistema



# Ejemplos UWE

Equipos de fútbol

# Problema

- Hacer un sistema web en el cual podamos cargar los distintos clubes de fútbol, con su información particular.
- Cargar diferentes ciudades.
- Asociar clubes con ciudades.
- Obtener el informe de los campeonatos logrados por cada club.
- Registro de usuario y *login* al sistema.

# Tareas a realizar

1. Definir actores
2. Definir relaciones entre actores
3. Definir Casos de Uso para cada actor
4. Definir capa de contenido
5. Definir capa de navegación
6. Definir capa de presentación

# 1. Actores

- Definiremos dos tipos de actores:
  - Usuario no registrado
  - Usuario registrado
- El no registrado podrá leer información.
- El registrado podrá hacer lo mismo que el no registrado, e introducir información al sistema.

# 1. Actores

Actores	Descripción
Usuario no registrado	El usuario no registrado representa al usuario que no posee login, que no... etc.
Usuario registrado	Este actor representa a los usuarios que no .. y tampoco .. Etc. etc.

## 2. Relaciones entre actores



- El usuario registrado puede hacer todo lo que hace el no registrado, además de sus propias funcionalidades.
- Debe haber un diagrama exclusivamente para denotar esto.
- Este diagrama, por sí solo, ocuparía la segunda página.

### 3. Casos de Uso por actor

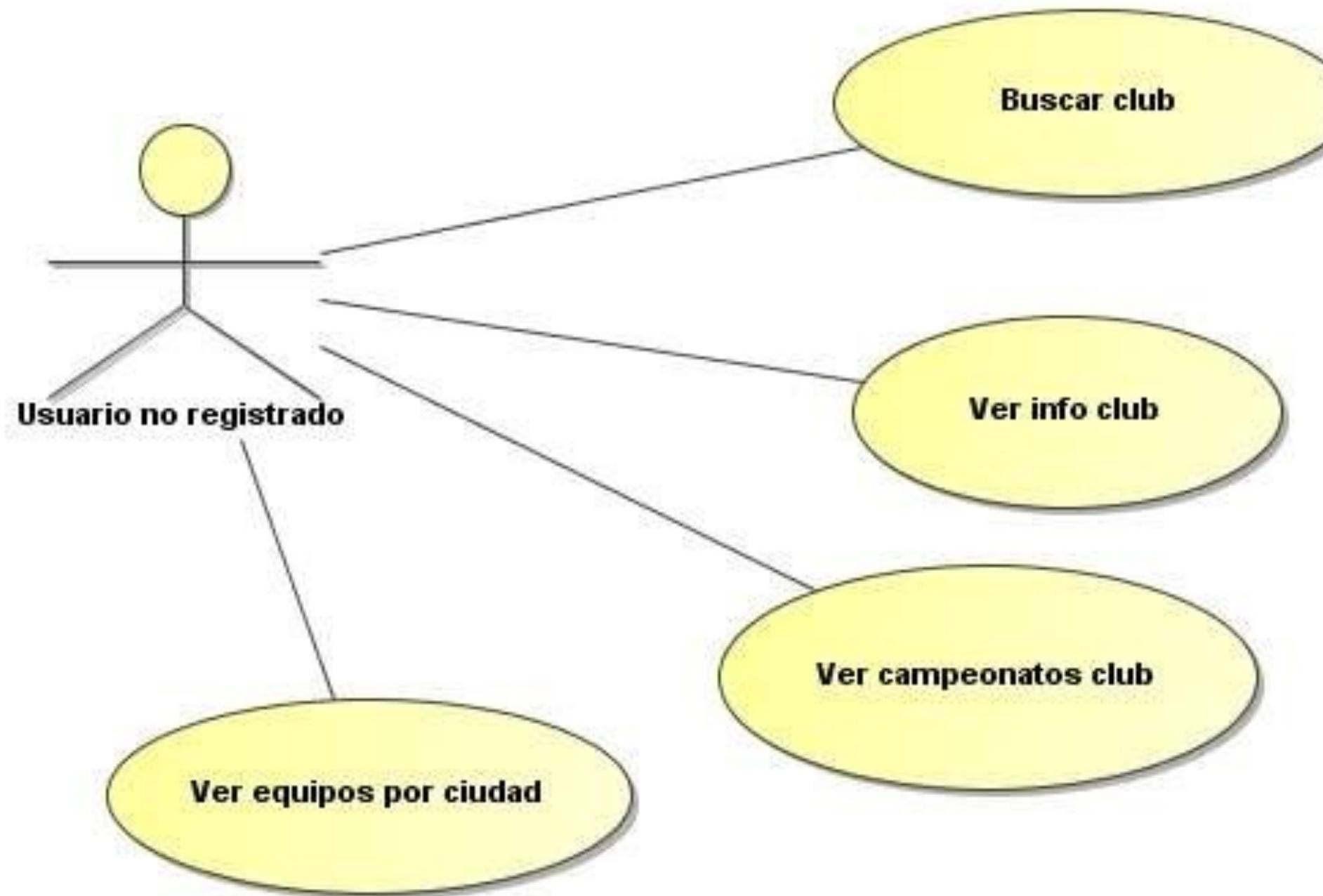
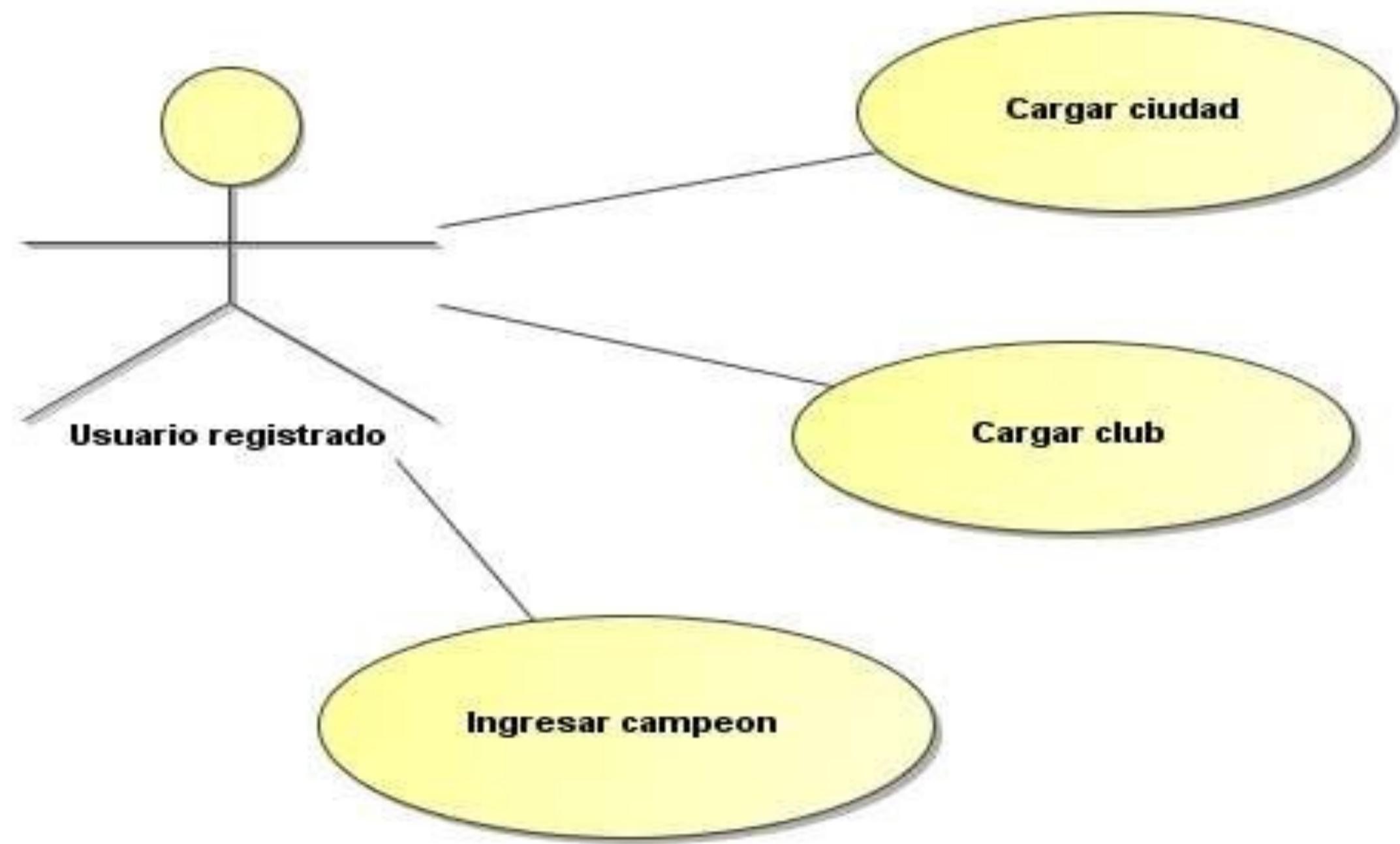


Diagrama de  
un actor en particular.  
El diagrama comprende solo los Casos  
de Uso particulares a este actor.

# 3. Casos de Uso por actor



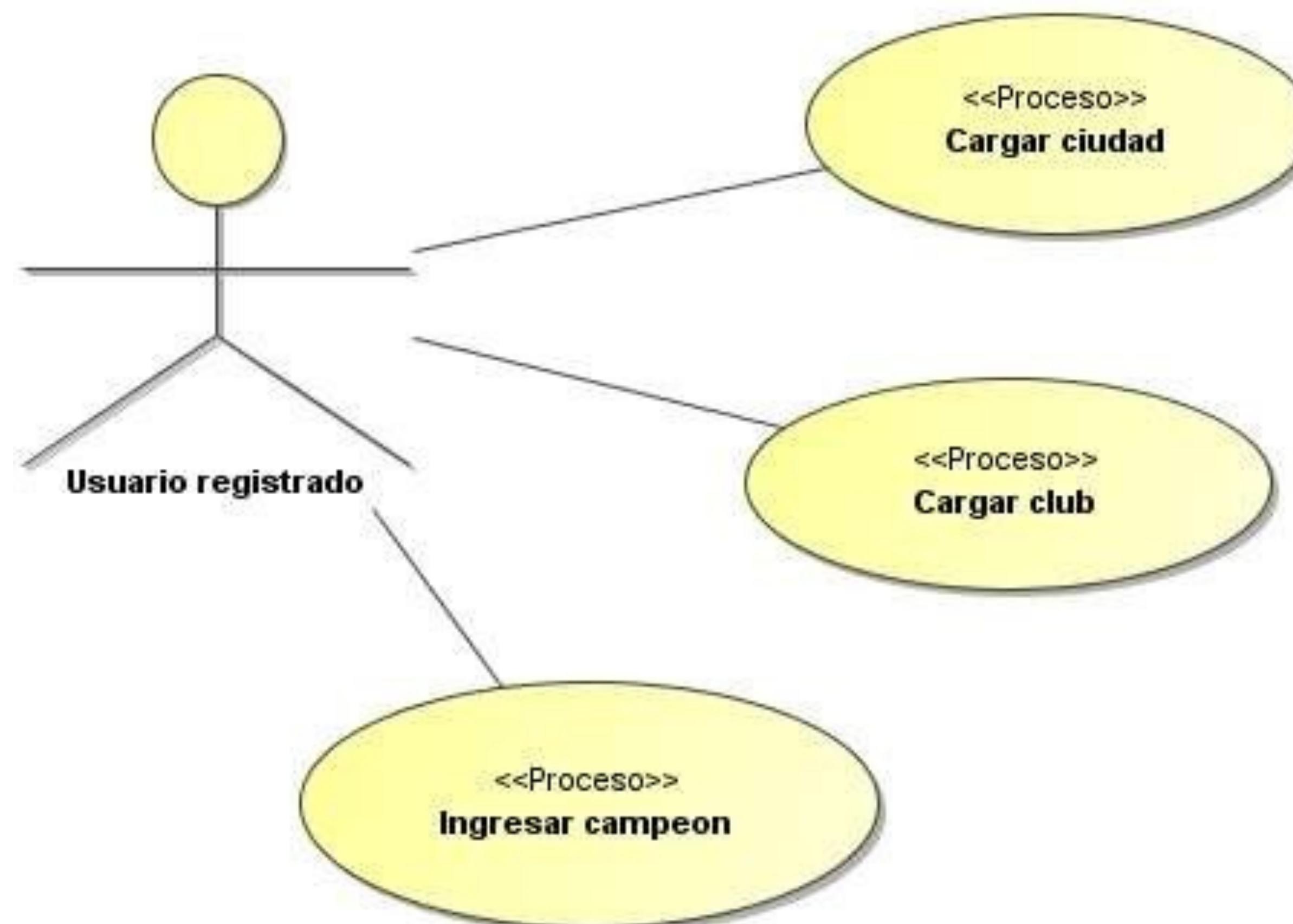
### 3. Casos de uso por actor

- Una vez definidos esto, pasamos agregar los estereotipos para casos de uso definidos por UWE (Navegación, Proceso).

### 3. Casos de uso por actor

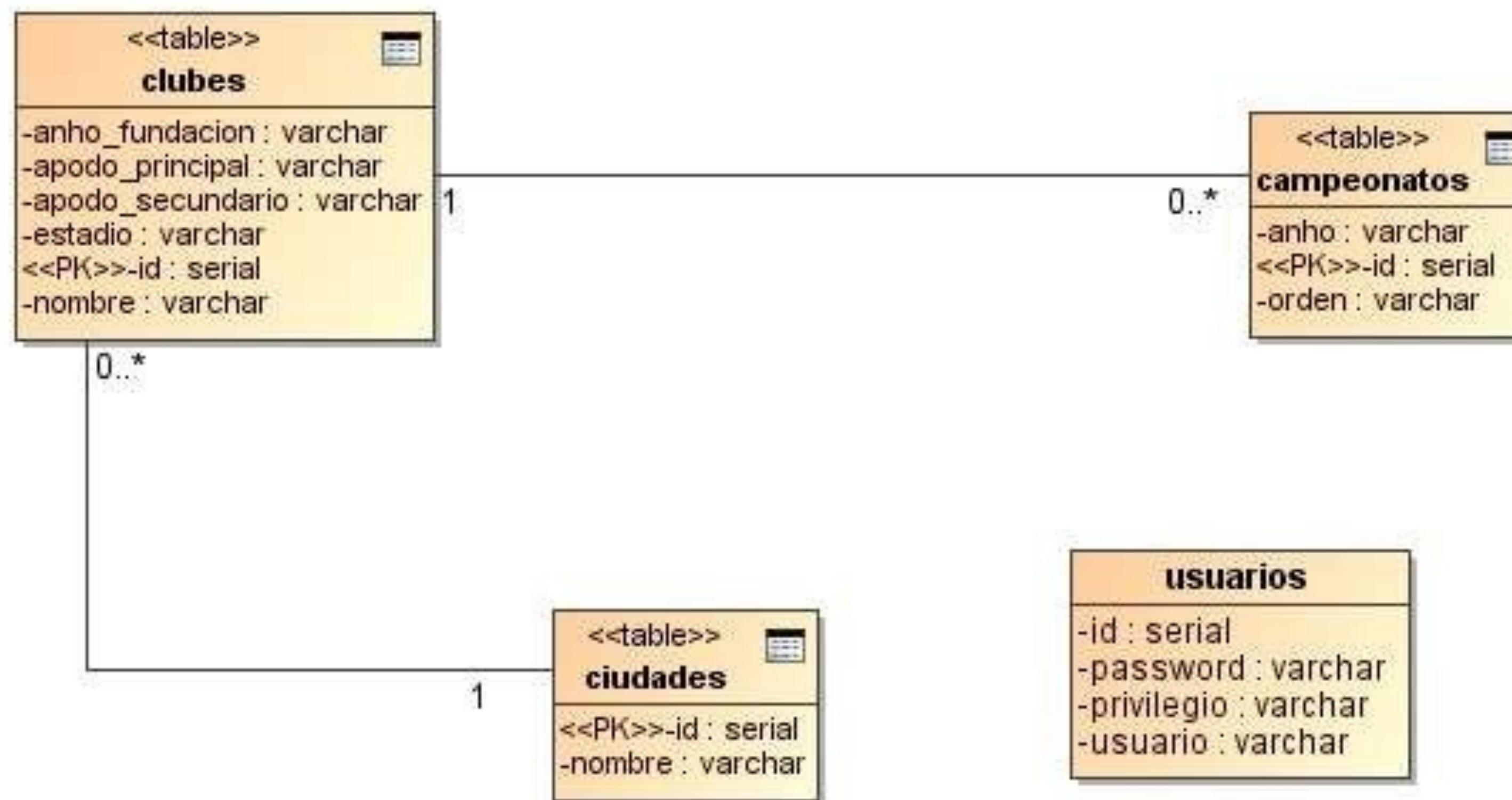


### 3. Casos de uso por actor



# 4. Capa de contenido

- La capa de contenido nos expresa la información. En ese caso, ER con notación UML



# 5. Capa de navegación

- Ahora pasaremos a ver un procedimiento básico para la capa de navegación.
- Se utilizarán los estereotipos de UWE
- En nuestro ejemplo, de los requisitos y funcionalidades de CU sabemos que tenemos que buscar clubes, cargar clubes nuevos, cargar ciudades, cargar campeonatos, etc.
- Nos basaremos en todas las funcionalidades para obtener la navegación.

# 5. Capa de navegación

- Conceptualmente, partiremos que el usuario se validará primero.
- Posteriormente podrá buscar un club, ver los campeonatos de un club, ver los equipos en cada ciudad.
- Además, si es usuario registrado, podrá cargar un club nuevo, cargar un campeonato nuevo, o cargar una ciudad.
- Se omitió el login y registro de nuevo usuario en los CU.

# 5. Capa de navegación

- En algún momento tendremos que poder llegar (navegar) hasta una página de un club.
- O una página con los campeonatos de un club.
- O una página con los clubes de una ciudad.
- Basar en esto para saber las páginas (nodos) a los cuales se llegará y refinaremos hacia atrás.
- Todos estos serán nodos navegacionales.
- Son los casos de uso que definimos con el estereotipo <<navegacional>>

# 5. Capa de navegación

Además tendremos nodos de proceso.

Los casos de uso que definimos como <>proceso>> derivarán en algún momento en un nodo de proceso.

Es decir donde hay algún tipo de lógica de programación, que terminará en algún tipo de modificación de datos.

En otras palabras, los casos de uso cargar club implica en algún momento poder navegar hasta una página donde cargaremos un nuevo club.

Cargar ciudad, igual al punto anterior. etc.

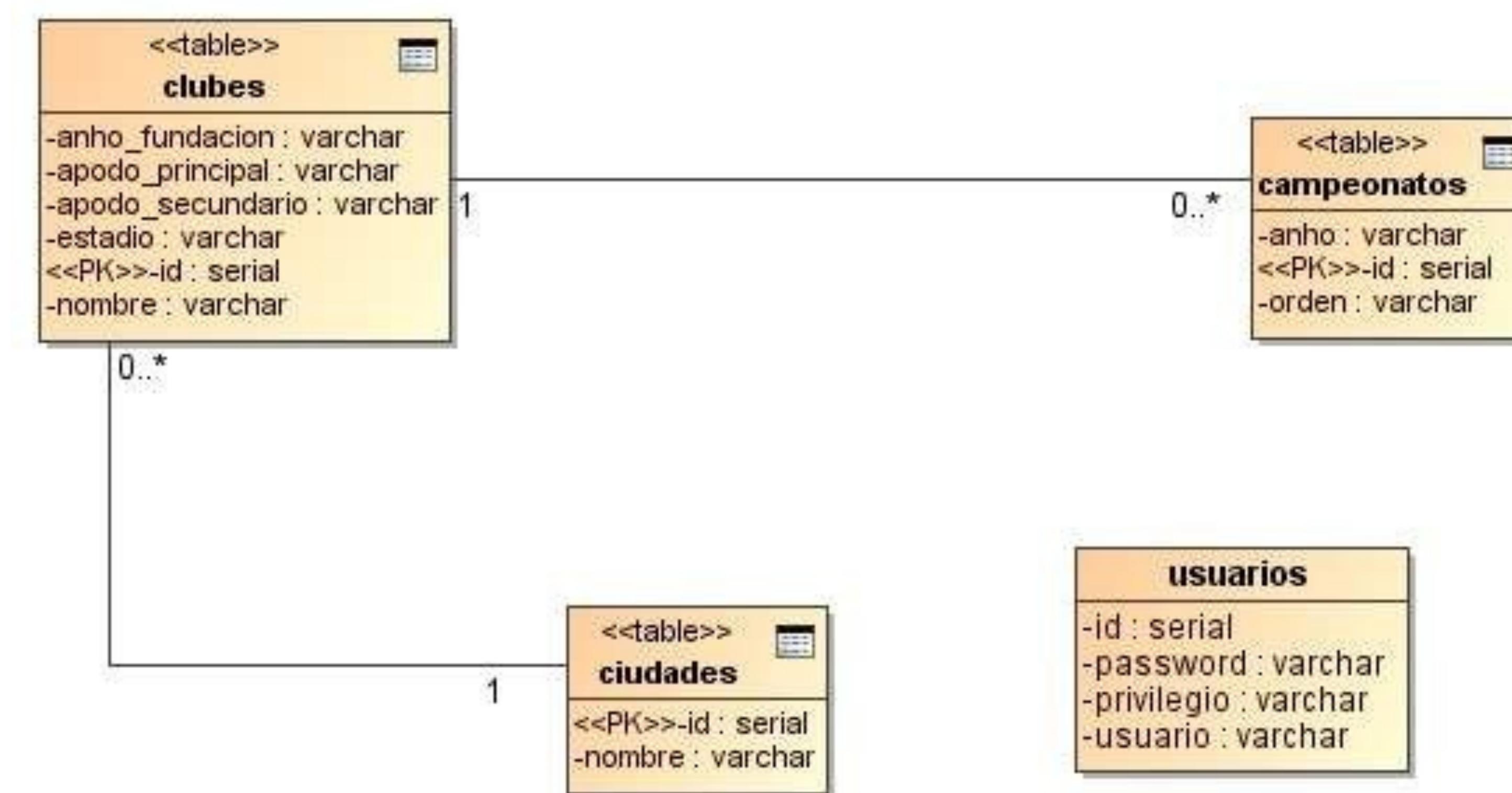
Un CU de <>proceso>> eventualmente implicará dos nodos:

Uno de navegación (la página de carga), y el otro un nodo de proceso (la lógica de negocio donde se realiza el proceso de carga y los controles necesarios para esto).

Y debe haber una asociación de proceso entre estos.

# 5. Capa de navegación

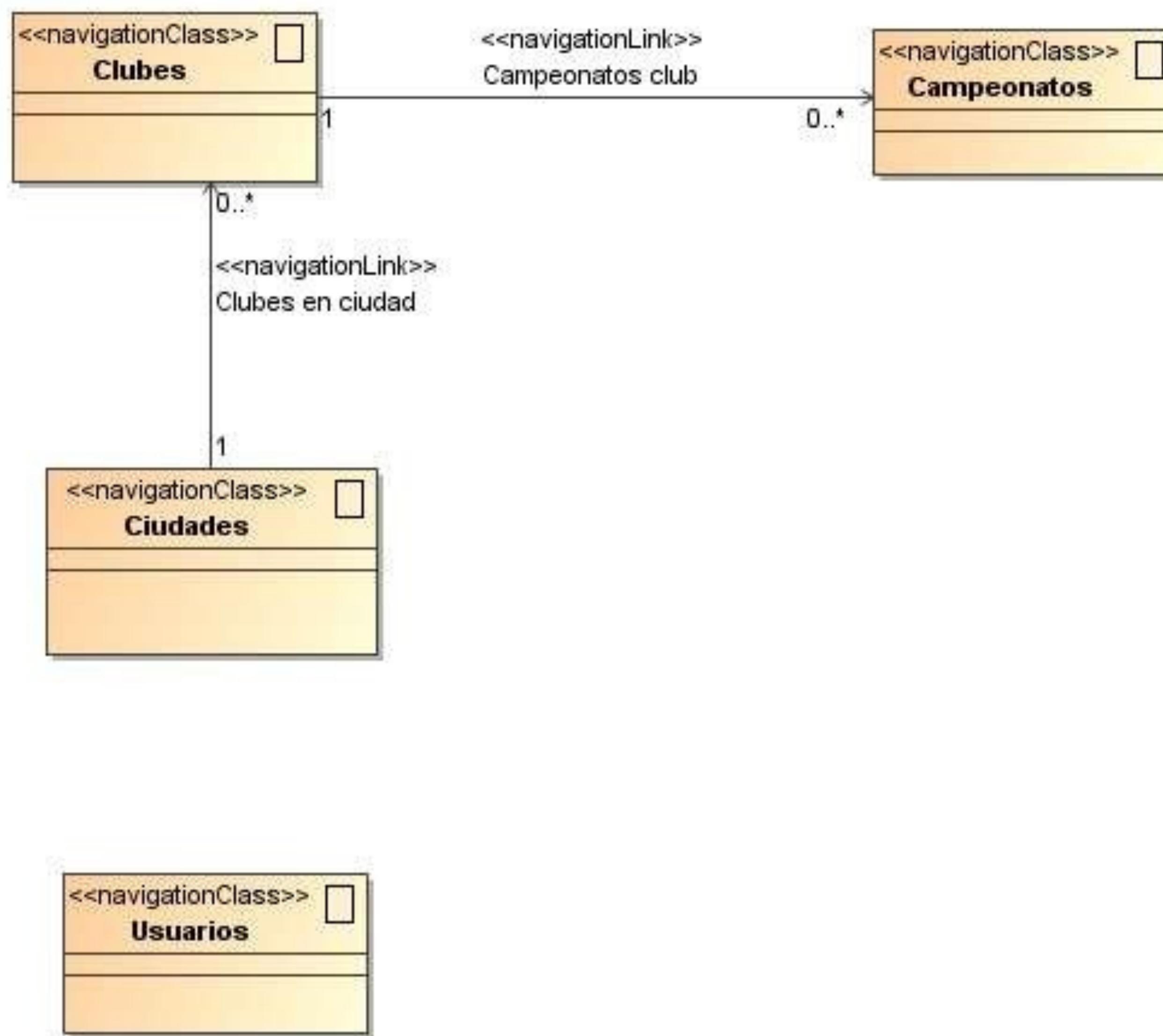
- Partimos de nuestra capa de contenido (diagrama BD)



# 5. Capa de navegación

- Vemos cuales de las tablas son importantes para nuestra navegación y las colocamos en el diagrama de navegación, junto con sus asociaciones.
- Los links de navegación son dirigidos.
- Si consideramos es necesario un link de ida y vuelta entre dos nodos, se deben colocar dos links de asociación.

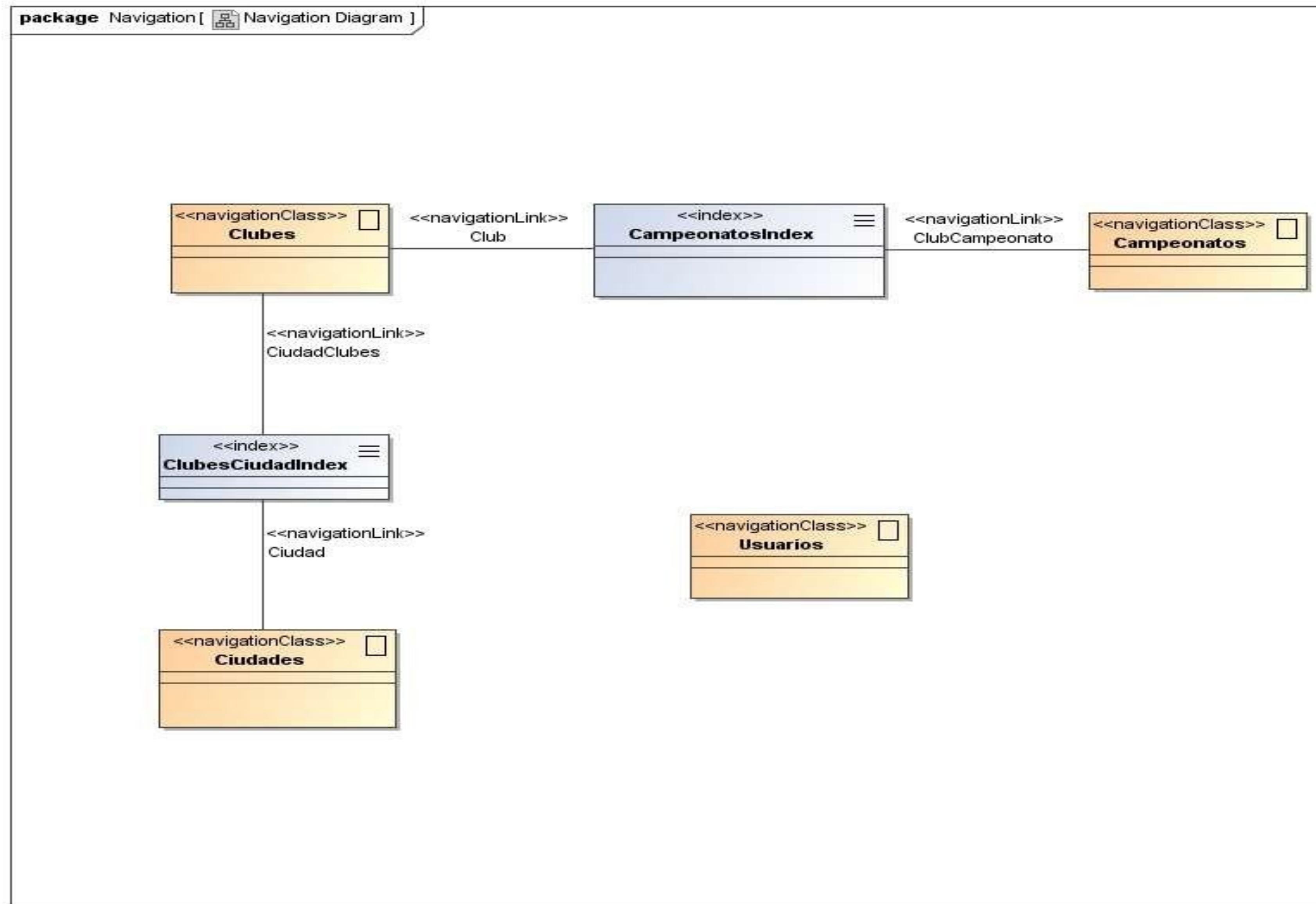
# 5. Capa de navegación



# 5. Capa de navegación

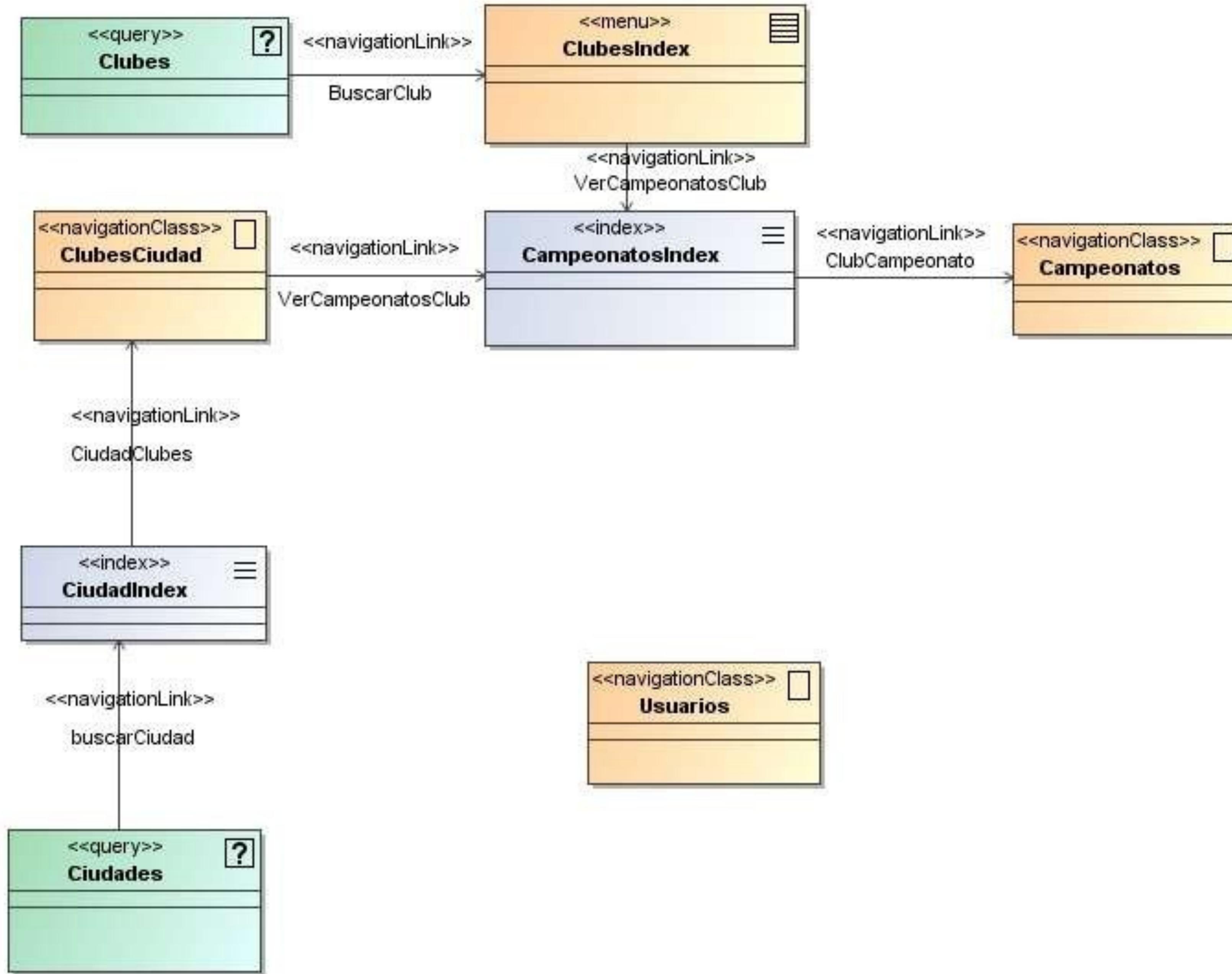
- Ahora debemos eliminar las multiplicidades.
- Donde desde un nodo haya más de un link de navegación de salida se utiliza un *menú*.
- Si la multiplicidad del lado final del link de navegación es mayor a uno se utiliza:
  - Index
  - Guided tour
  - Querie

# 5. Capa de navegación



# 5. Capa de navegación

- En la imagen anterior reemplazamos asociaciones con multiplicidad por nodos con primitivas de acceso.
- En la imagen vemos de color grises las clases con el estereotipo *index*.
- Ahora debemos agregar la dirección en las asociaciones.
- En el ejemplo anterior desde un nodo de navegación Ciudad general, pasamos a un Indice de Ciudades, de donde elegimos una y vemos los clubes de dicha Ciudad.
- Lo mismo de un nodo Clubes, vamos a un nodo Indice de campeonatos de Club, de donde podemos elegir uno en particular.
- Normalmente queríamos buscar una ciudad primero, o un club, o grupos de los mismos.
- Desde los resultados de la búsqueda, tener una o varias opciones de elección para ver la información de los mismo. Ahí utilizaremos *queries*.



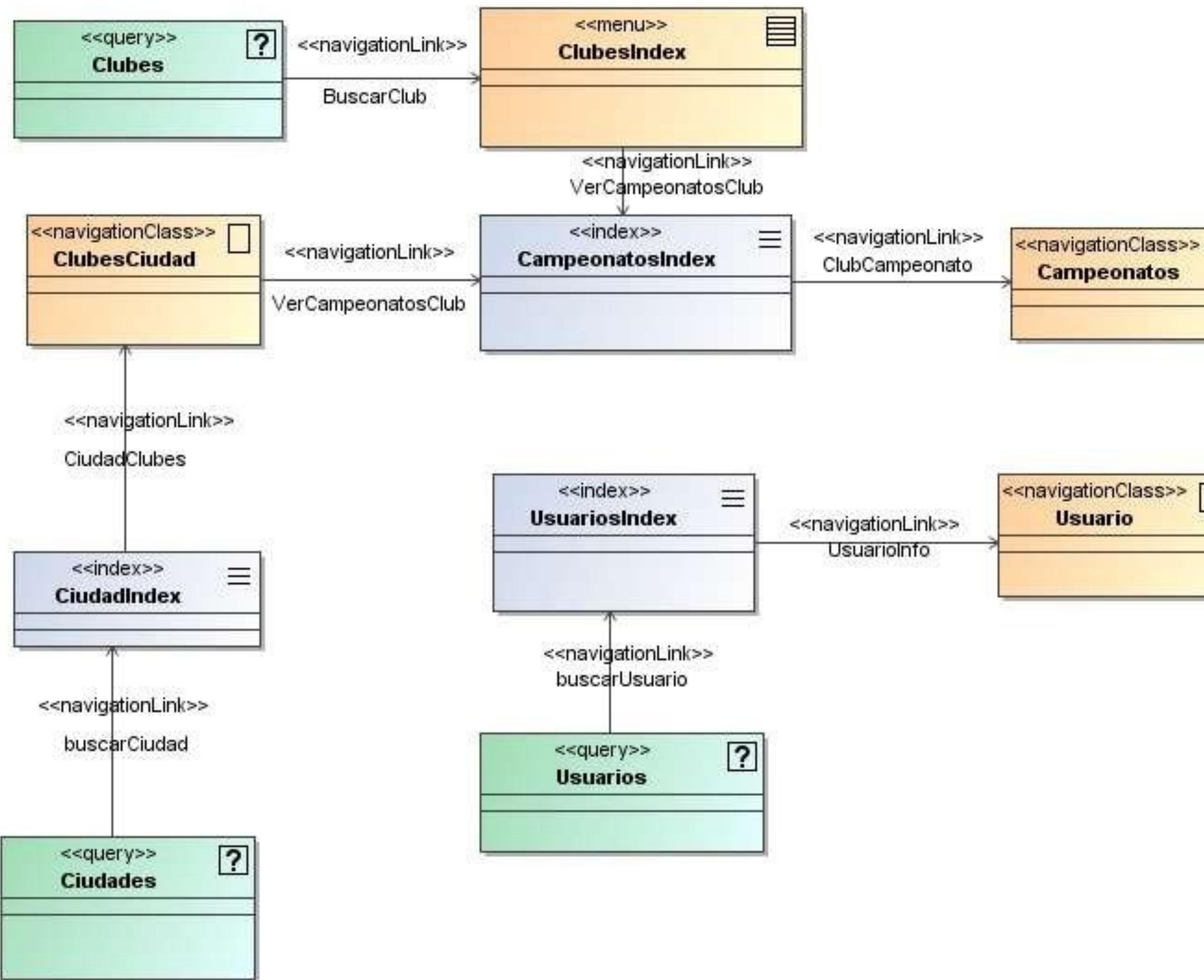
# 5. Capa de navegación

- Ahora en verde tenemos las clases con estereotipo *query*.
- Las clases verdes indican un nodo donde se realiza una búsqueda (*query*).
- La misma puede tener desde cero a varios resultados. Por lo tanto se utiliza un menú (nodos con links de salida con multiplicidad mayor a uno).
- Desde la lista de Ciudades, podemos elegir cualquiera, para ver los clubes en dicha ciudad.
- Desde la lista de clubes, podemos elegir cualquiera para ver sus campeonatos.
- Tanto la lista de clubes, como la lista de ClubCiudad tienen el mismo contenido (un grupo de clubes)
- Por lo tanto en ClubCiudad podemos también elegir cualquier Club y ver sus campeonatos, lo que nos crea un link de navegación hacia CampeonatosIndex.
- CampeonatosIndex es un indice porque puede tener multiplicidad mayor a uno en el lado final o de salida (un club con varios campeonatos).

# 5. Capa de navegación

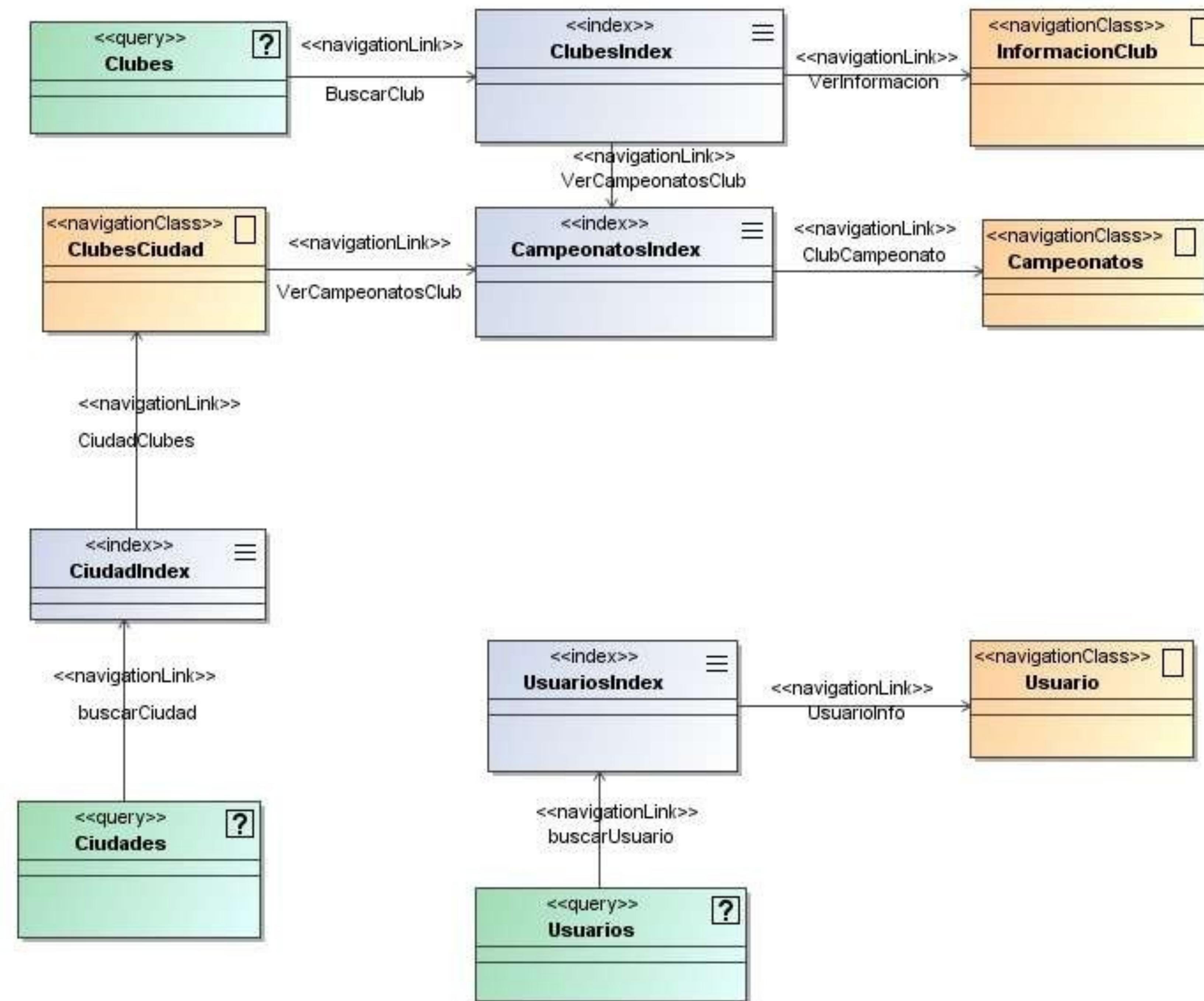
- También debemos ver información de usuarios, buscar usuarios, elegir un usuario de una lista, etc.
- De esta manera seguimos refinando el diagrama de navegación.

package Navigation [  Navigation Diagram ]



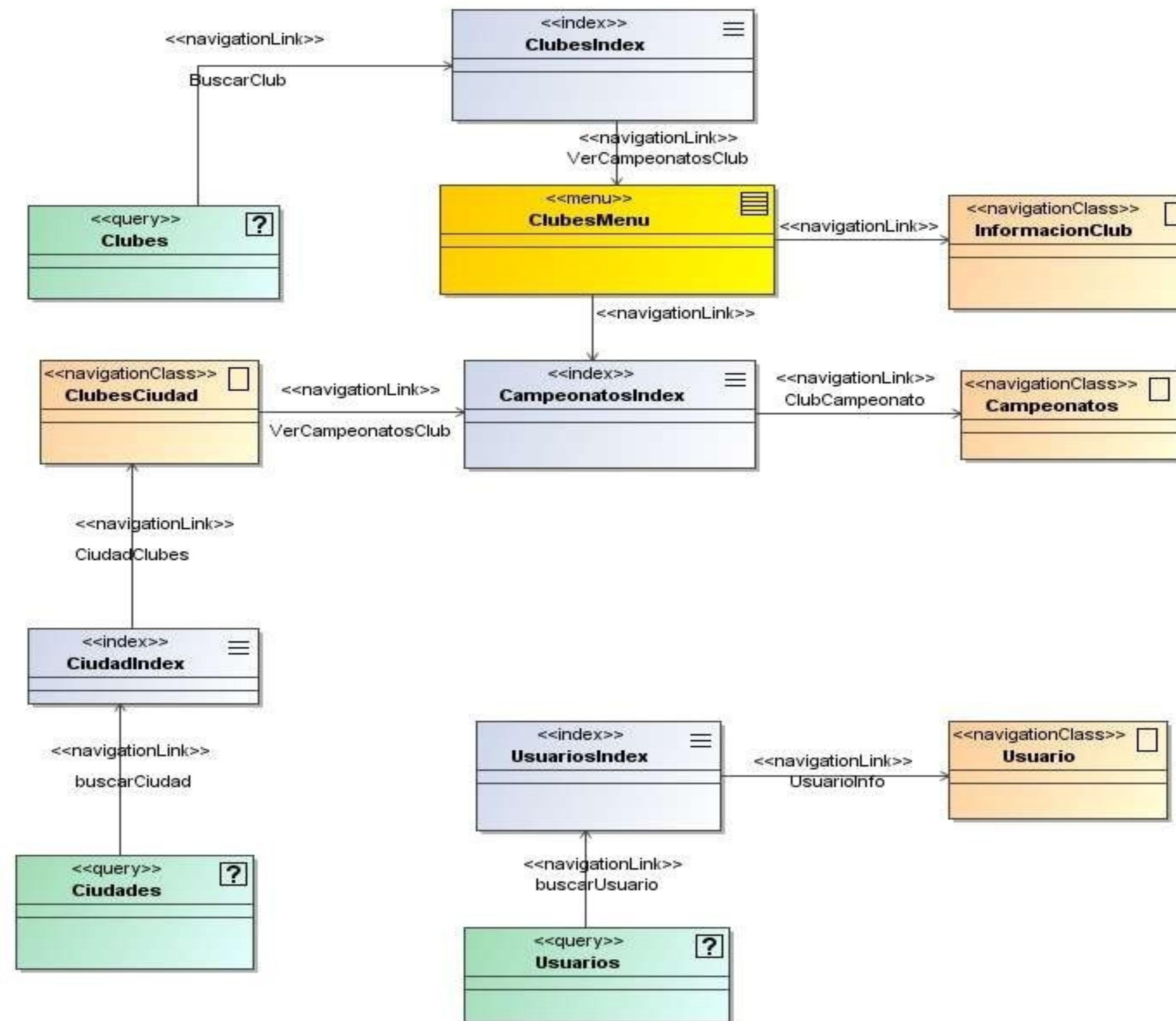
## 5. Capa de navegación

- Podemos ver la información general de un club desde el índice de clubes.
- Sin embargo ahora tendremos dos links de navegación de salida de un mismo nodo.



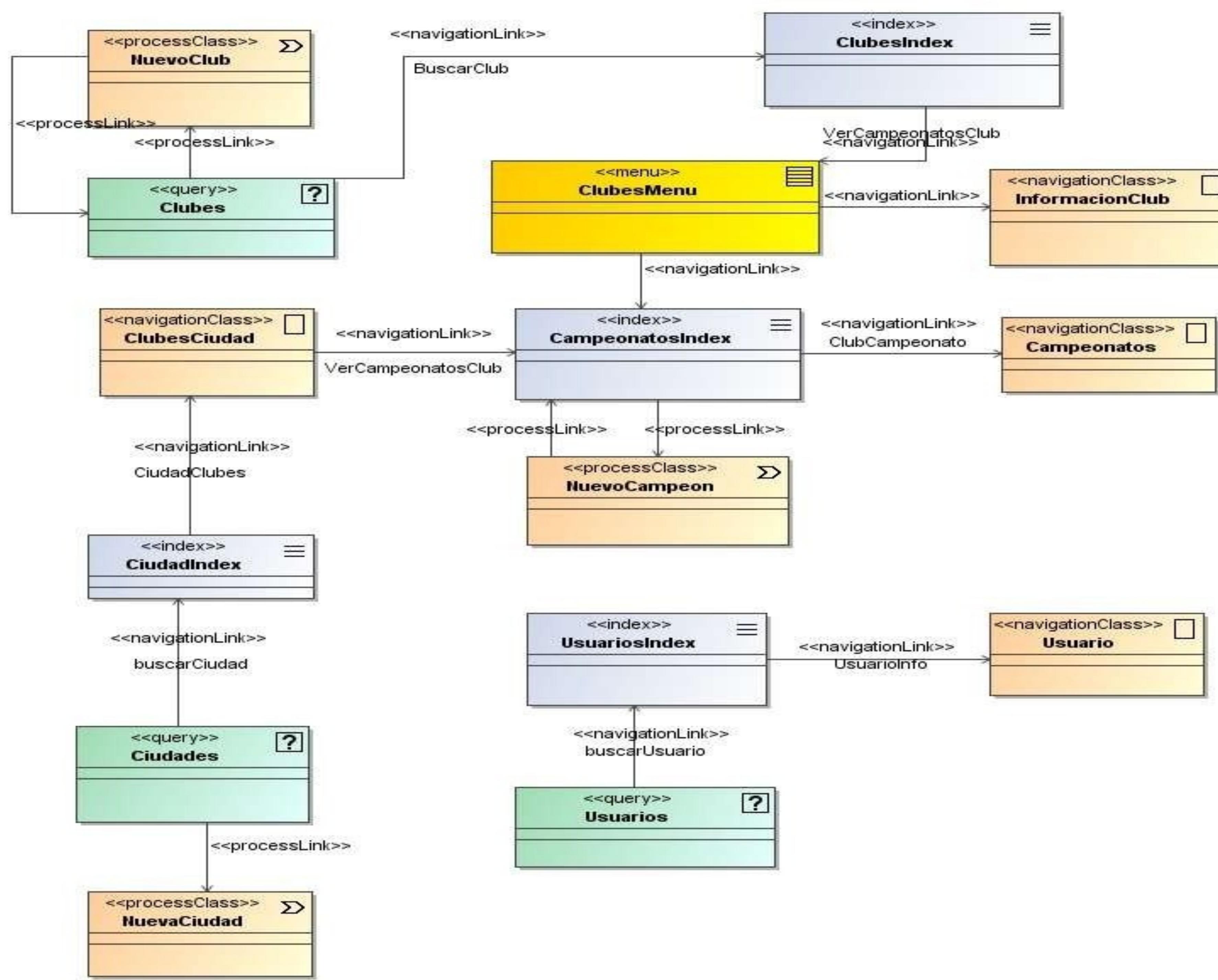
# 5. Capa de navegación

- De nuevo, nodos con más de un link de navegación se pueden transformar en menús.



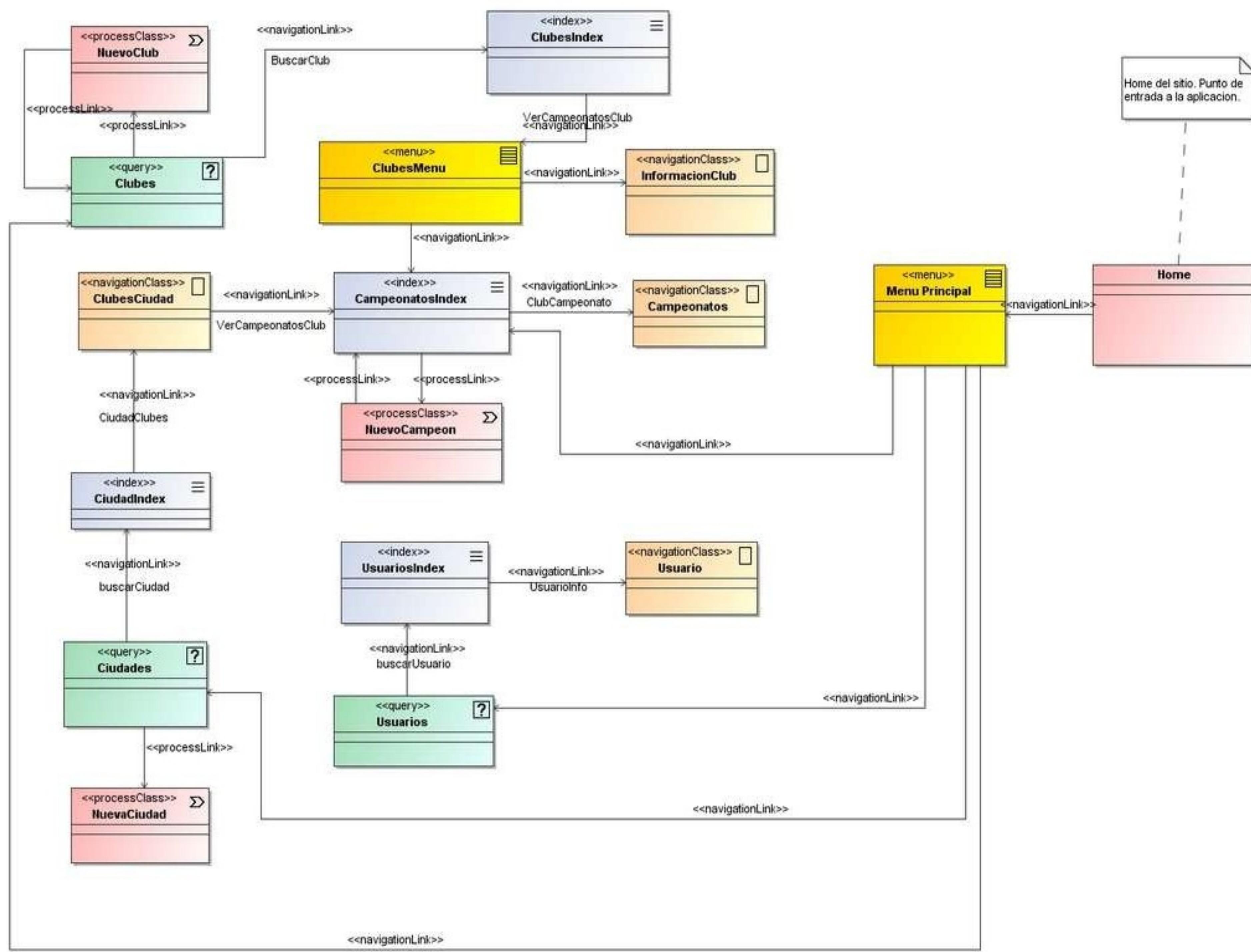
## 5. Capa de navegación

- Seguimos refinando la navegación de esta manera.
- Luego agregamos los nodos de proceso.
- Estos salen de los Casos de Uso de proceso.
- Los mismos implican situaciones donde haya lógica de negocio, transacciones con los datos subyacentes, etc.



## 5. Capa de navegación

- De nuevo, refinar los nodos con más de un link de salida.
- Y luego, definir un punto de entrada a la aplicación que será simbolizado como el *home*.
- No dejar nodos sin conexión, ya que implicaría que no se puede navegar hasta ellos.



# 5. Capa de navegación

- Debemos así refinar el diagrama hasta cubrir todas las funcionalidades, items en los requerimientos, etc.
- Si el diagrama crece demasiado, se puede separar en subdiagramas
- Otra opción por navegación de cada actor.
- Ver también hay tres nodos de proceso, uno por cada CU de proceso

# 6. Capade presentación

Provee una visión abstracta de la interfaz del usuario:  
Esta basada en el modelo de navegación.

No tener en cuenta temas como colores, letras, posicionamiento de los elementos dentro de la página, etc.

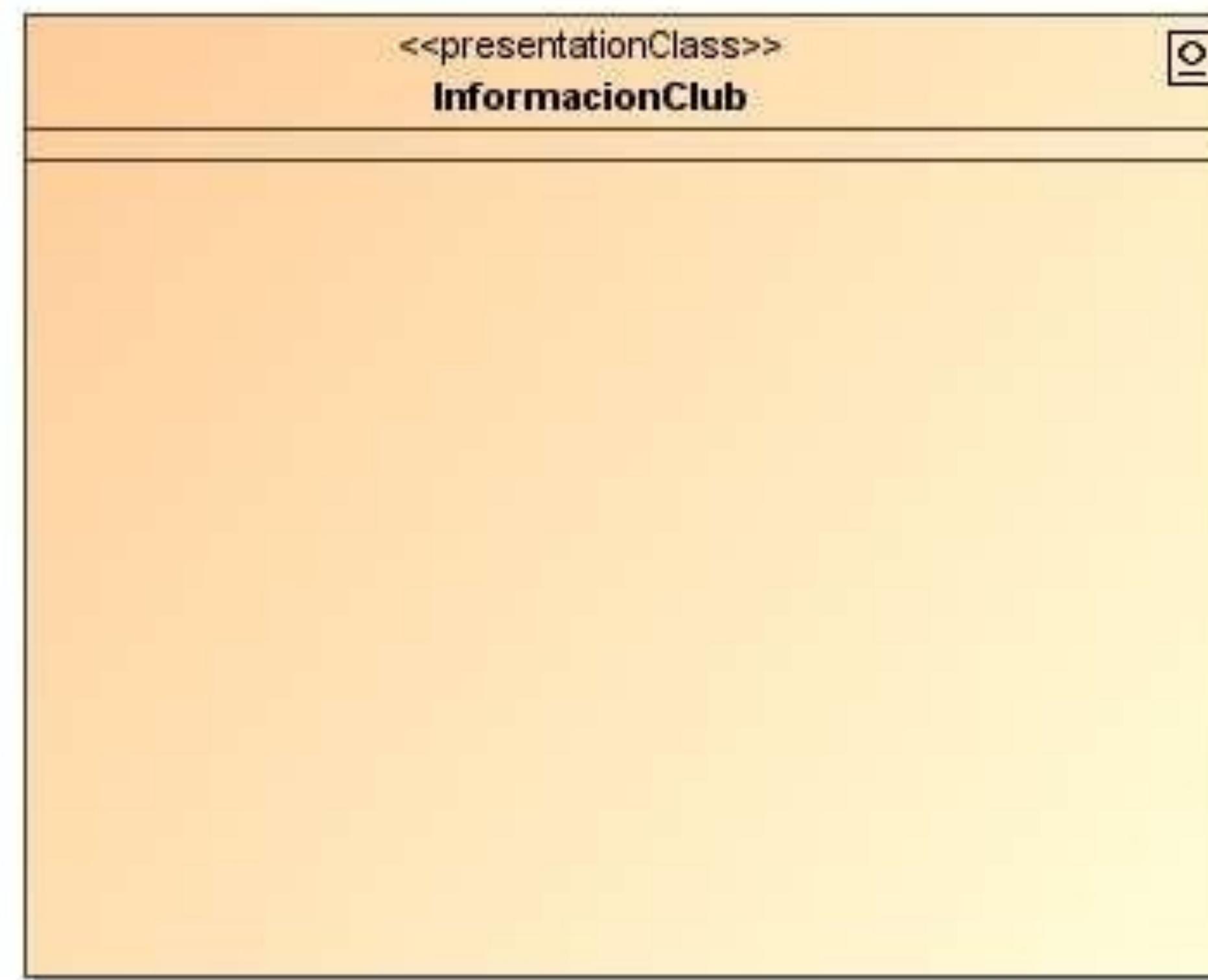
Las clases de presentación se basan los nodos del diagrama de navegación.  
Es decir, tendremos una clase de presentación por cada menú, clase de navegación, primitivas de acceso (query, guided tour, index), y clases de proceso.

Para facilitar la ubicación de la clase de presentación de un nodo de navegación se recomienda utilizar los mismos nombres para los mismos.

Tomemos como ejemplo el nodo de navegación “InformaciónClub” del diagrama de Navegación. A continuación, vemos la clase de representación que corresponde a este nodo.

# 6. Capa de presentación

- Clase de presentación del nodo de navegación del mismo nombre.

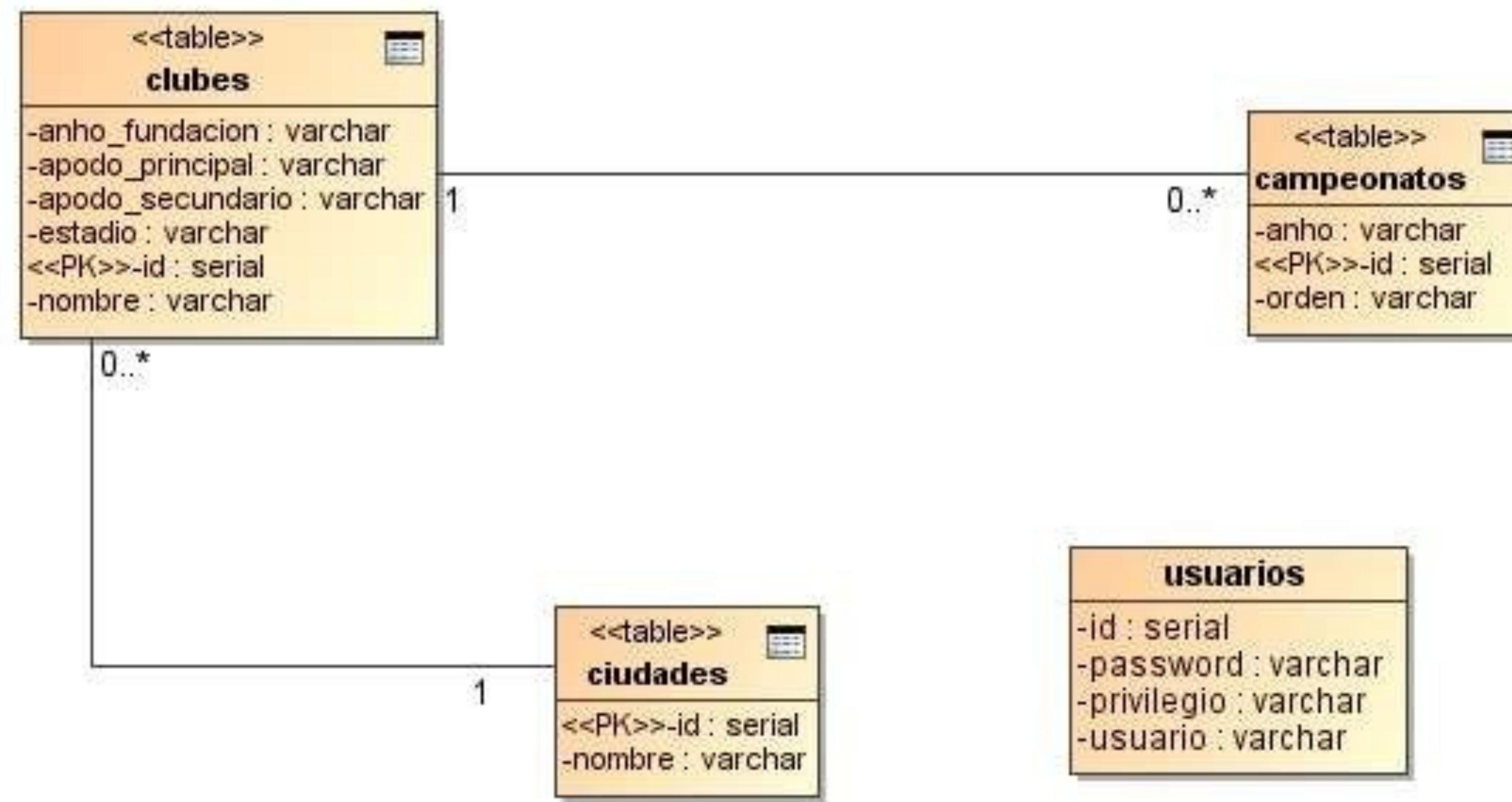


# 6. Capa de presentación

- Ya tenemos una clase de presentación vacía.
- Para llenarla, partimos de nuestra Base de Datos.
- ¿Qué atributos tiene, en la BD, la tabla que produce el nodo de navegación InformacionClub, y posteriormente la clase de presentación InformacionClub?

# 6. Capa de presentación

- Vemos de nuevo nuestro diagrama de BD.



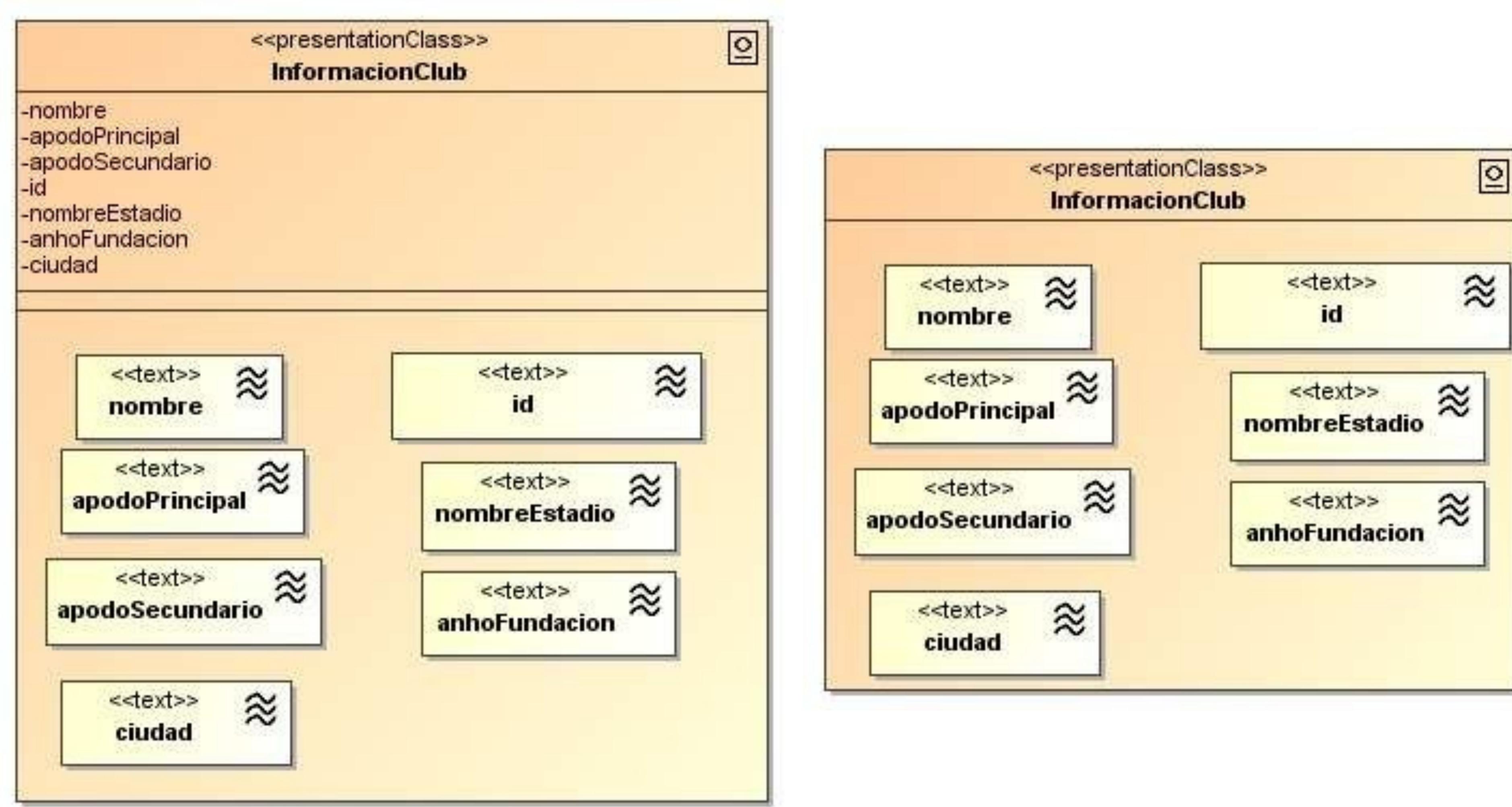
# 6. Capa de presentación

- En el nodo de presentación veremos la información relacionada a un club.
- De la tabla tenemos los atributos:
  - Id
  - Nombre
  - Apodo Principal
  - Apodo Secundario
  - Estadio
  - Año de fundación
  - Id de la ciudad a la que pertenece (esta es una clave foranea que viaja desde la tabla ciudad)

# 6. Capa de presentación

- Cada atributo de la clase será representado con un elemento de UI.
- El MD y el plug-in UWE proporciona todos los esterotipos y elementos visuales e iconos necesarios para la clase de presentación.
- Así como un diagrama para crear los mismos. Entonces nuestra clase de presentación
- InformaciónClub se verá
  - un elemento, el nombre de la ciudad, viene de otra tabla (Ciudad), mediante un Join, con la información que se tiene con el id de Ciudad.
- El MD permite dibujar los elementos dentro de la clase que lo contiene, facilitando la visualización y entendimiento de los mismos.

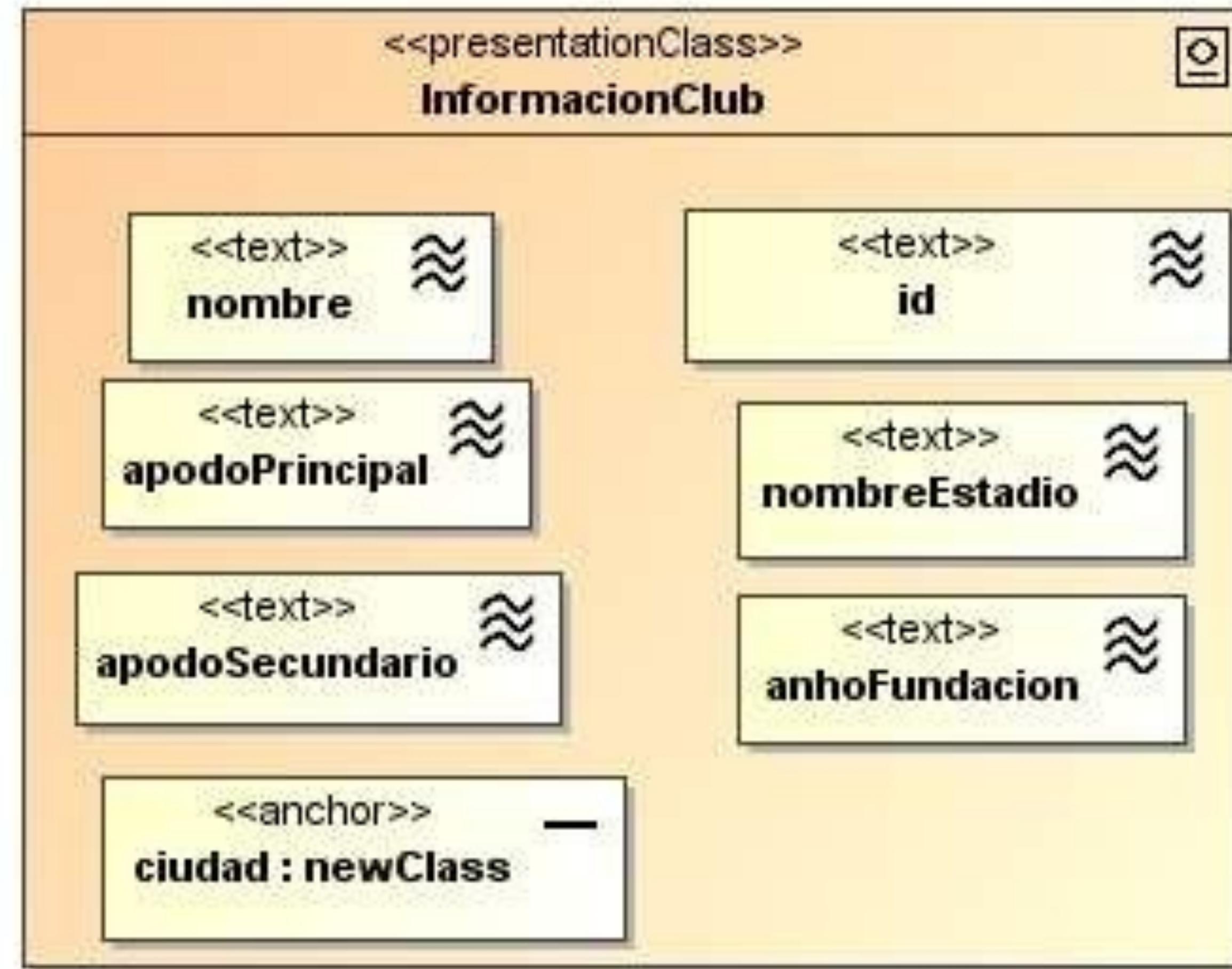
# 6. Capa de presentación



# 6. Capa de presentación

- Un club muestra, en su presentación, el nombre de la ciudad a la que pertenece.
- Quizás sería conveniente que del nombre de la ciudad podamos llegar a la lista de clubes de dicha ciudad.
- Deberíamos modificar nuestro diagrama navegacional permitiendo un link de salida desde InformacionClub hacia ClubesCiudad.
- Hecho esto, “ciudad” en nuestra clase de presentación ya no sería solo texto, sino un link.

# 6. Capa de presentación



# 6. Capa de presentación

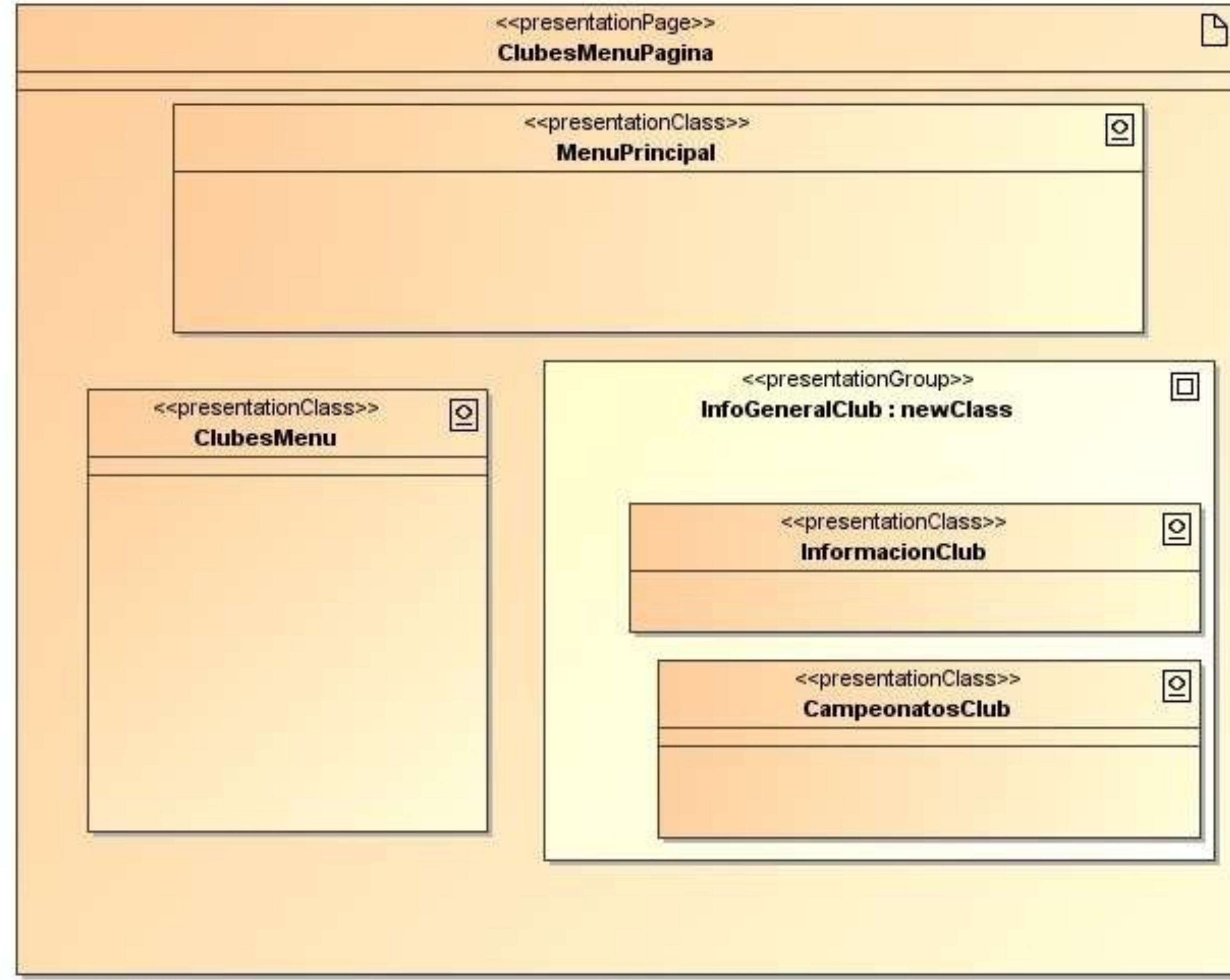
- De esta manera, vamos tomando los diferentes nodos de navegación, y construyendo su correspondiente clase de presentación.
- Usualmente, en una misma página se presentará información de varios nodos de navegación.
- Para esto usamos el estereotipo *page*.
- Una página puede contener varias clases de presentación, así como grupos de presentación.
- Los grupos de presentación son contenedores que pueden tener grupos de presentación y clases de presentación.
- A continuación veremos un ejemplo de la página de ClubesMenu.

# 6. Capa de presentación



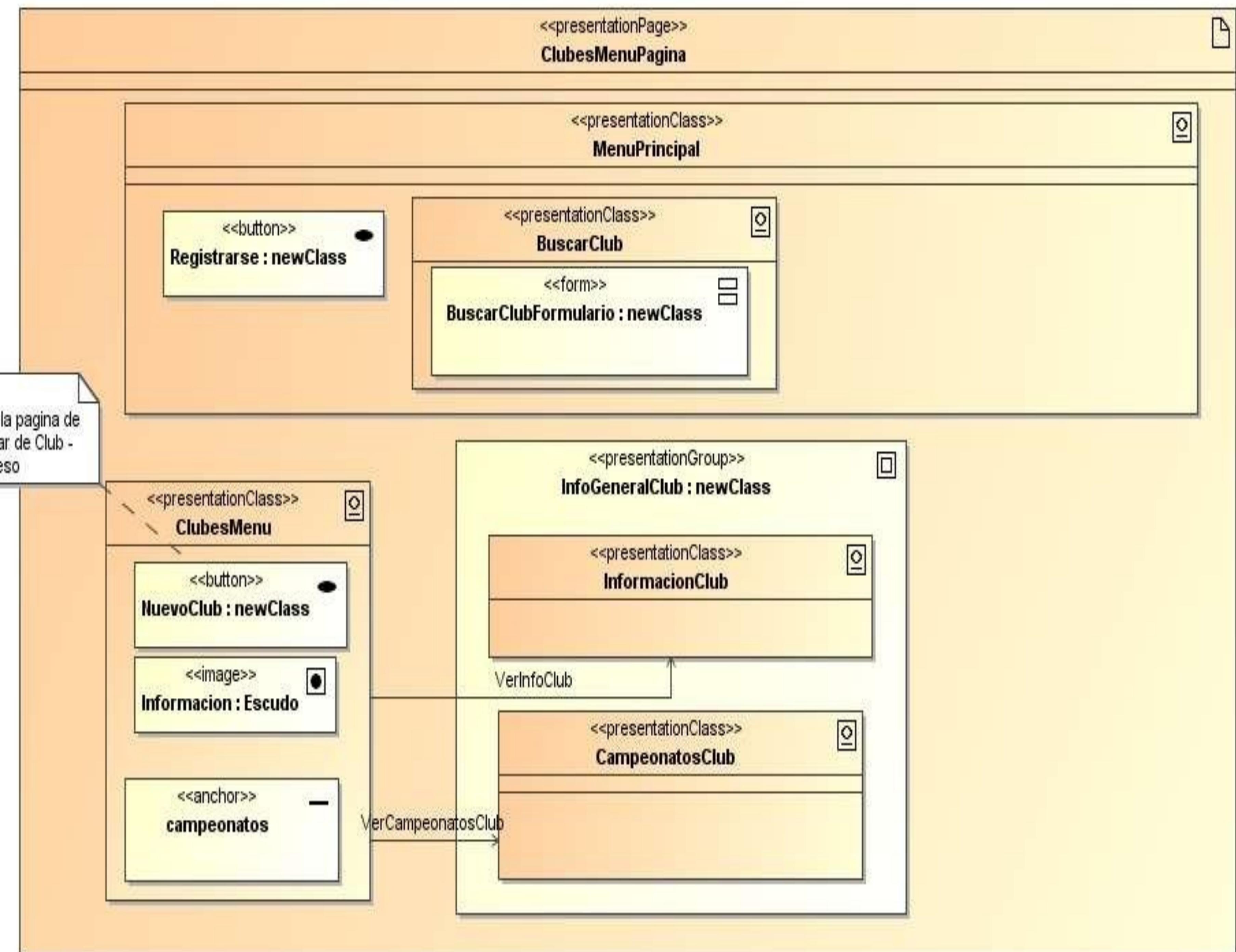
# 6. Capa de presentación

- En esta página queremos ver:
  - El menú principal
  - El menú de clubes
  - Información relacionada al club (información propia, campeonatos, etc).
- Por lo tanto, agregamos las clases de presentación de dichos elementos a la clase (queda a cargo del alumno hacer los mismos).



# 6. Capa de presentación

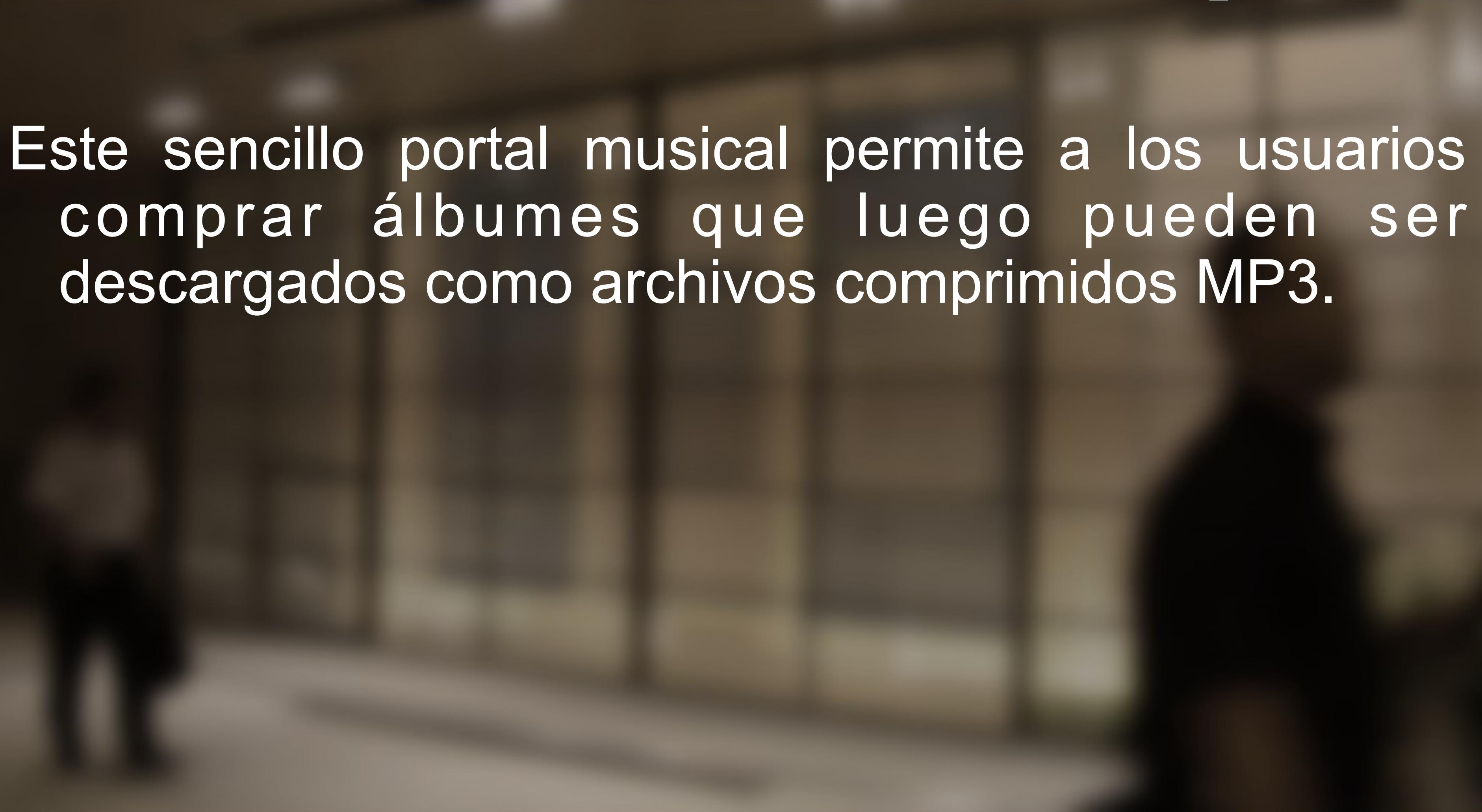
- Como en los pasos previos ya deberíamos de haber creado todas las clases de presentación individuales, simplemente completamos las páginas con las clases que la componen.
- Los atributos salen de la BD.
- Y además, la navegación se hace en base a los links de navegación entre los nodos en el diagrama de navegación.
- También quizás necesitemos modificar nuestra navegación y tener un link a crear un nuevo club.
- Por lo tanto necesitamos añadir esto también a nuestros Casos de Uso, etc.
- Así de informacionClub puedo llegar a informacionClub y también a campeonatosIndex (diagrama de navegacion). Por lo tanto:



# 6. Capa de presentación

- Como se ve, normalmente, los *query* se ven en una clase con un formulario dentro.
- Un atributo que lleva a otro nodo, un *anchor* o ancla (link).
- Debemos realizar las clases de presentación individuales.
- Y posteriormente, agrupar las necesarias para ir presentando las páginas.
- ¿Cuántas páginas hacer? Las que representen los UC más importantes para nuestro sistema.

# Portal musical simple.



Este sencillo portal musical permite a los usuarios comprar álbumes que luego pueden ser descargados como archivos comprimidos MP3.

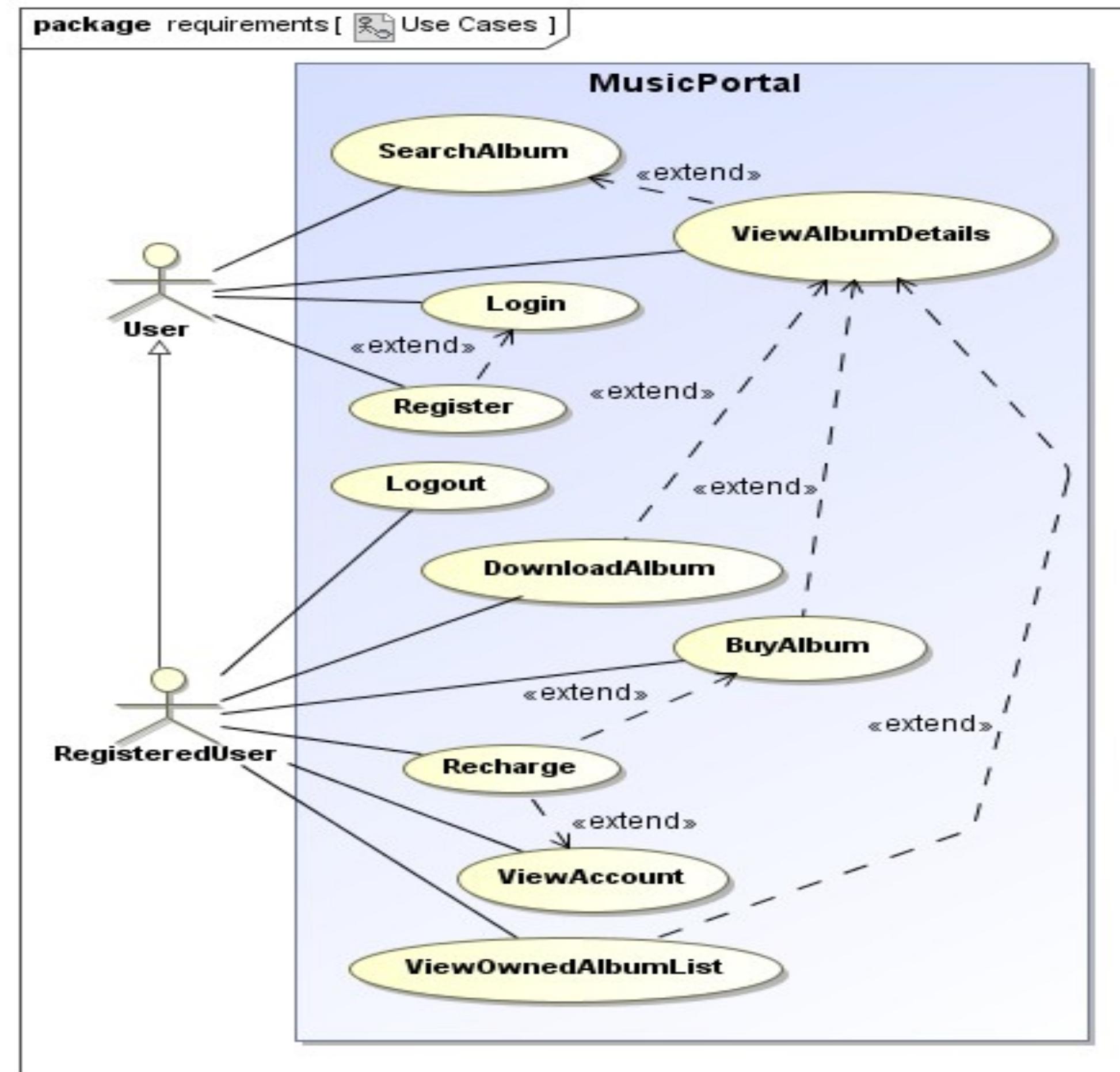
## Requisitos

- λ Se hace una distinción entre los usuarios y los usuarios registrados, un usuario se convierte en usuario registrado una vez que accede con su loguin. Sólo los usuarios registrados pueden comprar o descargar discos. Los usuarios no registrados pueden registrarse con un nombre de usuario que no ha sido utilizado por otro usuario y una contraseña elegida libremente.
- λ Cada usuario puede buscar discos por su nombre. Otros métodos de búsqueda no se ofrecen. El resultado de la búsqueda se presenta como una lista de los álbumes coincidentes que proporciona enlaces a una página de detalle de cada álbum. Las páginas de detalles del álbum deben mostrar el título del álbum, el nombre del artista, la lista de las canciones y el precio del álbum. Si el usuario ya ha comprado el álbum un enlace de descarga se muestra a continuación. De lo contrario, habrá un enlace para comprar el álbum.
- λ Sólo se pueden descargar los discos completos.

# Requisitos

- λ En este ejemplo simplificado, cada disco tiene sólo un artista. Esta restricción se hace para reducir la complejidad de la navegación y modelos de presentación. Sería fácil para añadir soporte para múltiples artistas la adición de un índice en el modelo de navegación y una colección anclada en el modelo de presentación, respectivamente.
- λ Cada usuario registrado tiene una cuenta de crédito que se utiliza para comprar discos. La cuenta de crédito puede ser recargada por el pago con tarjeta de crédito. Para ello, el usuario tiene que introducir sus datos de tarjeta de crédito y el monto a cargar. Estos datos son validados y el usuario tiene que confirmar la transacción antes de que se cargue en la tarjeta de crédito. Los detalles de manipulación de tarjetas de crédito no se modelan en este ejemplo.
- λ Si un usuario está conectado, puede navegar a una página de cuenta que muestra los créditos del usuario y la lista de los álbumes que ha comprado en el pasado.
- λ Los enlaces para ingresar o salir, registrarse en la página y la cuenta del usuario se muestran siempre. Esto también es válido para el cuadro de búsqueda de álbum.

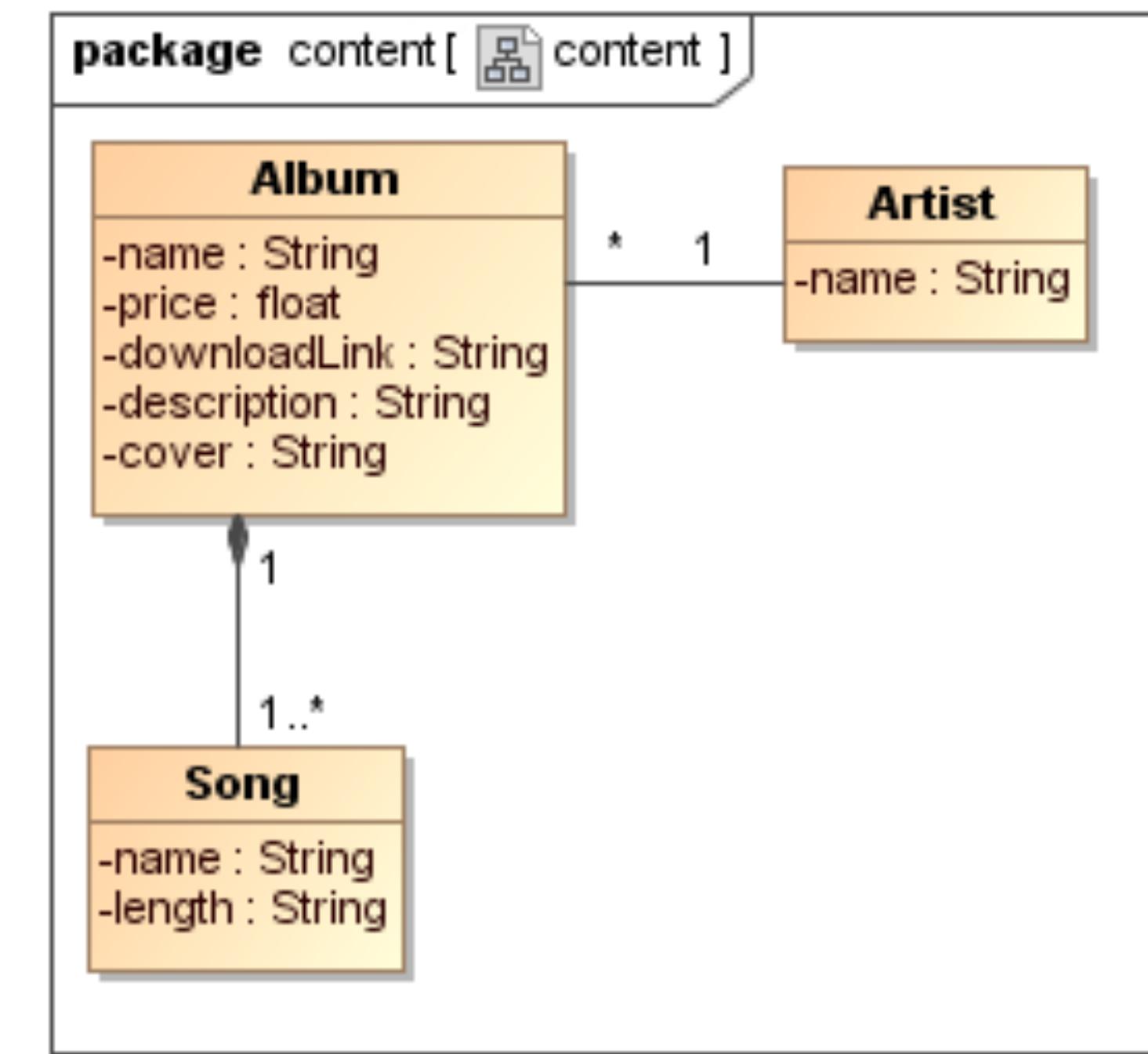
# Casos de Uso



# Diagrama de Contenidos

El modelo de contenido visualiza la información de dominio relevante para el sistema de Web que incluye principalmente el contenido de la aplicación Web.

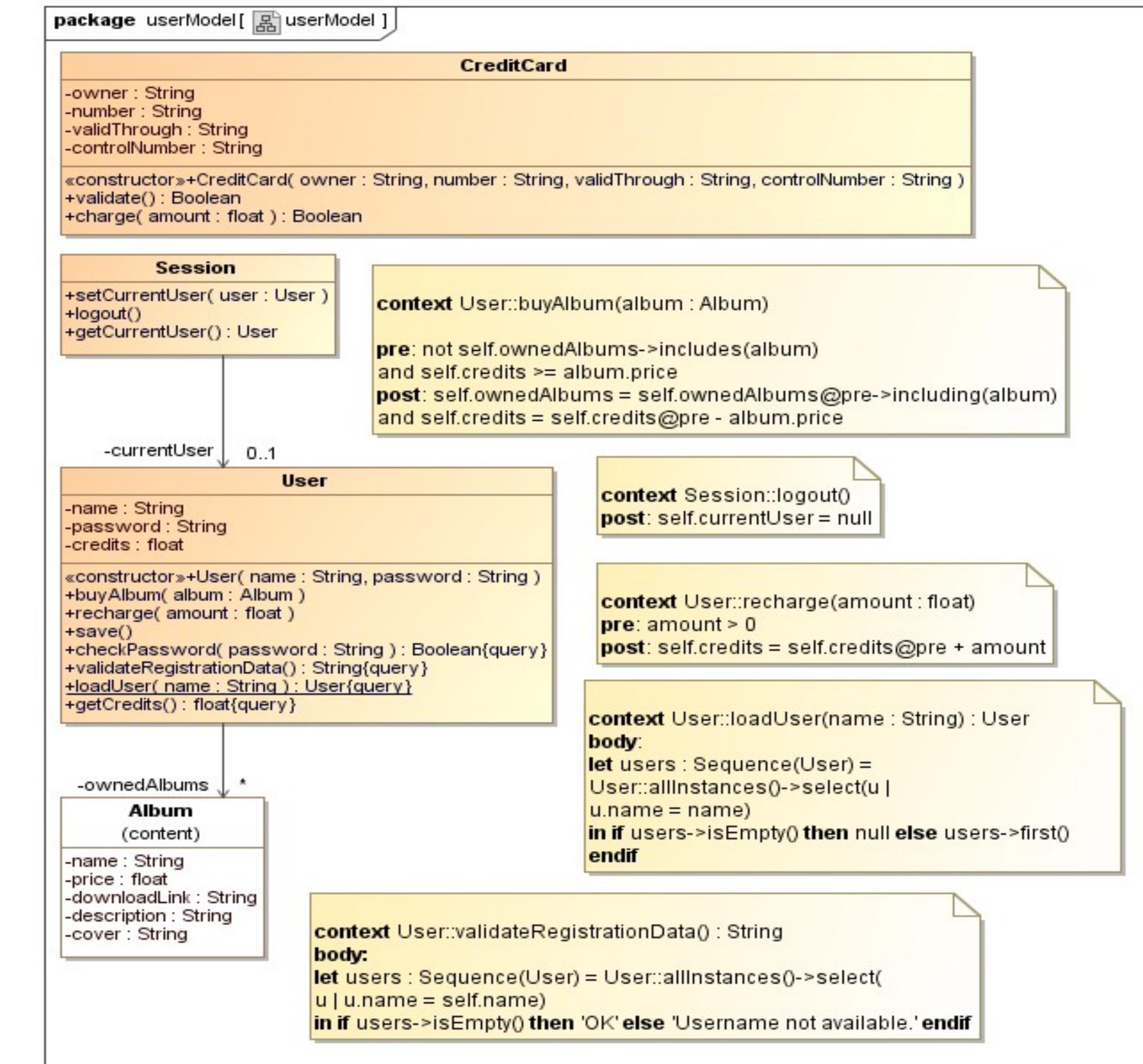
En nuestro ejemplo, la información es proporcionada por el Álbum de clases, artista y canción.



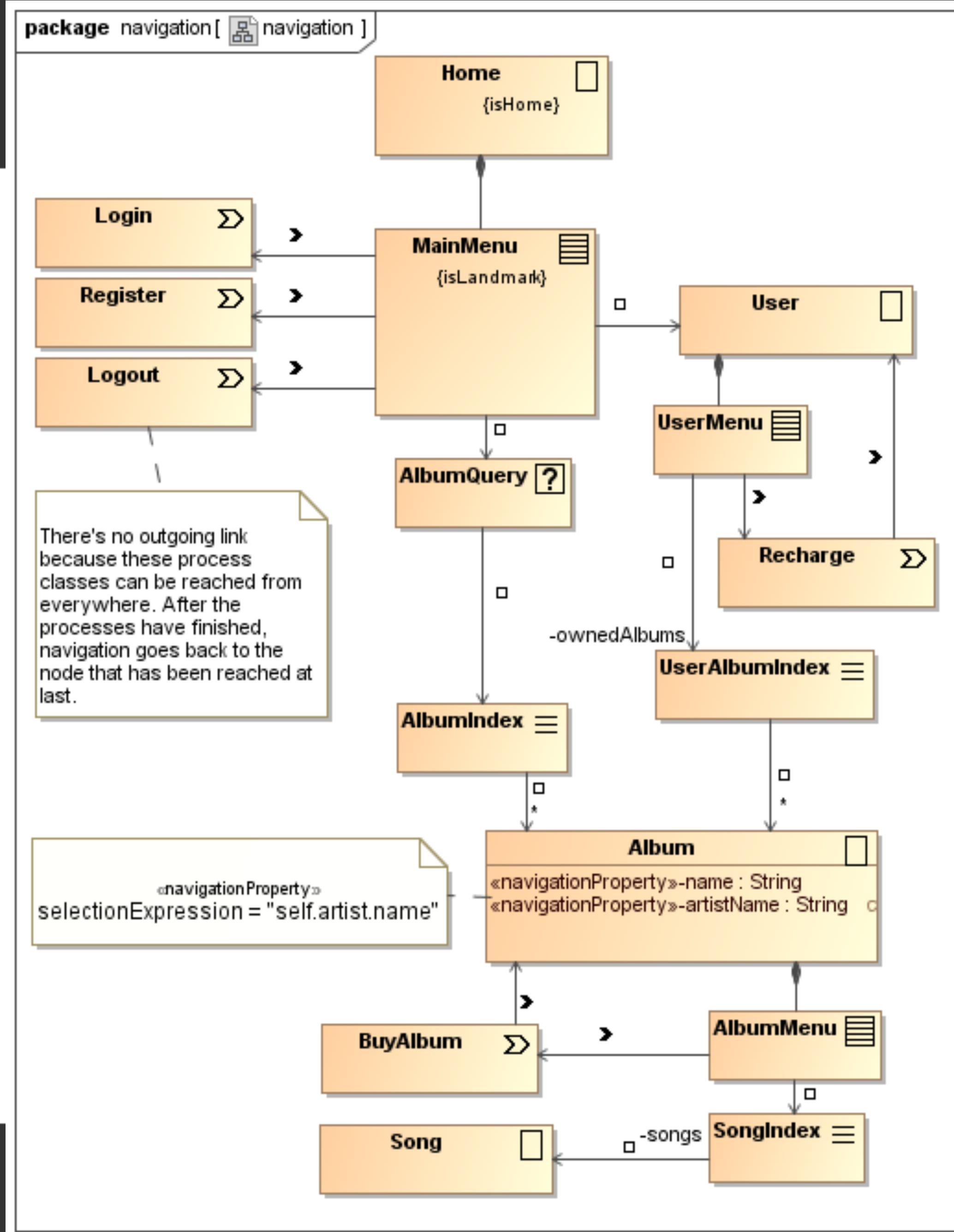
# Modelo Usuario

El modelo de usuario sirve para dos propósitos diferentes.

- Por un lado, contiene clases que definen qué información se almacena en el contexto de una sesión. En este ejemplo, vemos que una sesión puede tener un usuario actual que puede tener una colección de discos de su propiedad.
- Por otra parte, las clases en el modelo de usuario proporcionan operaciones que se pueden utilizar en los procesos de negocio.



# Modelo de Navegación



## Navigation

- navigationClass
- ≡ menu
- externalNode
- ? query
- guidedTour
- ≡ index
- navigationLink

## Process

- Σ processClass
- ⌚ userAction
- ⌚ systemAction
- ▶ processLink

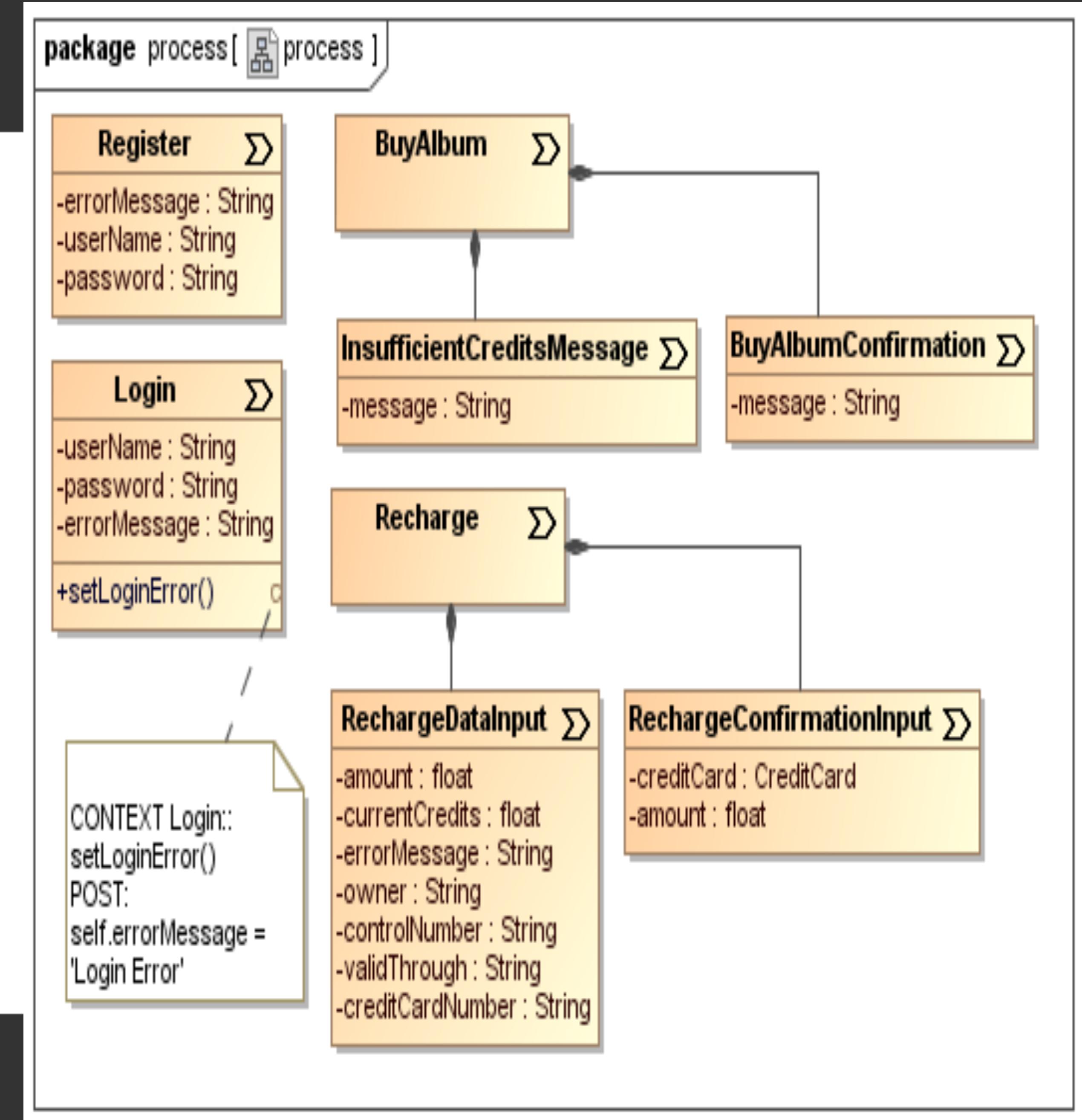
# Modelo de Proceso

## Navigation

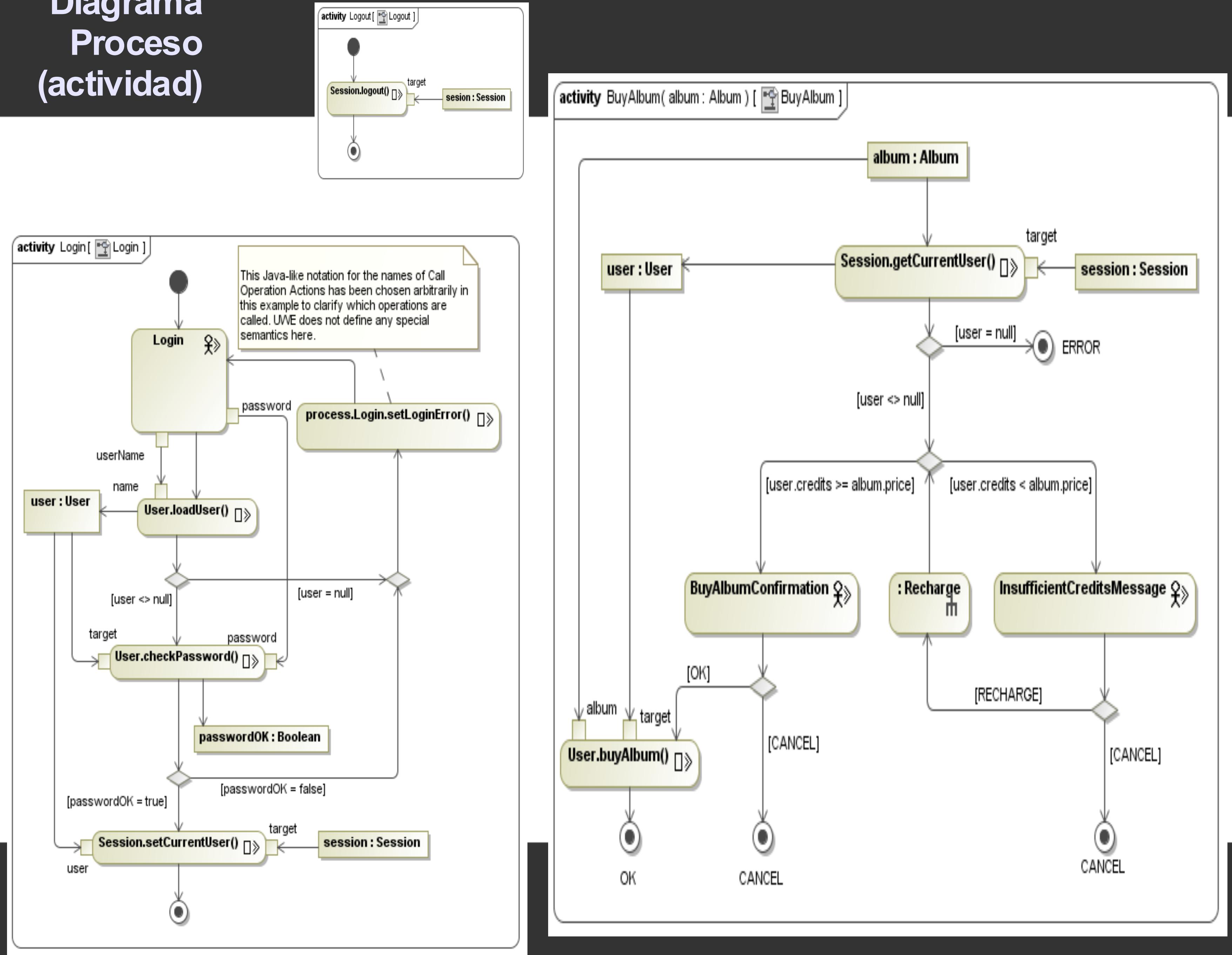
- navigationClass
- ☰ menu
- ↗ externalNode
- ?
- query
- guidedTour
- ≡ index
- navigationLink

## Process

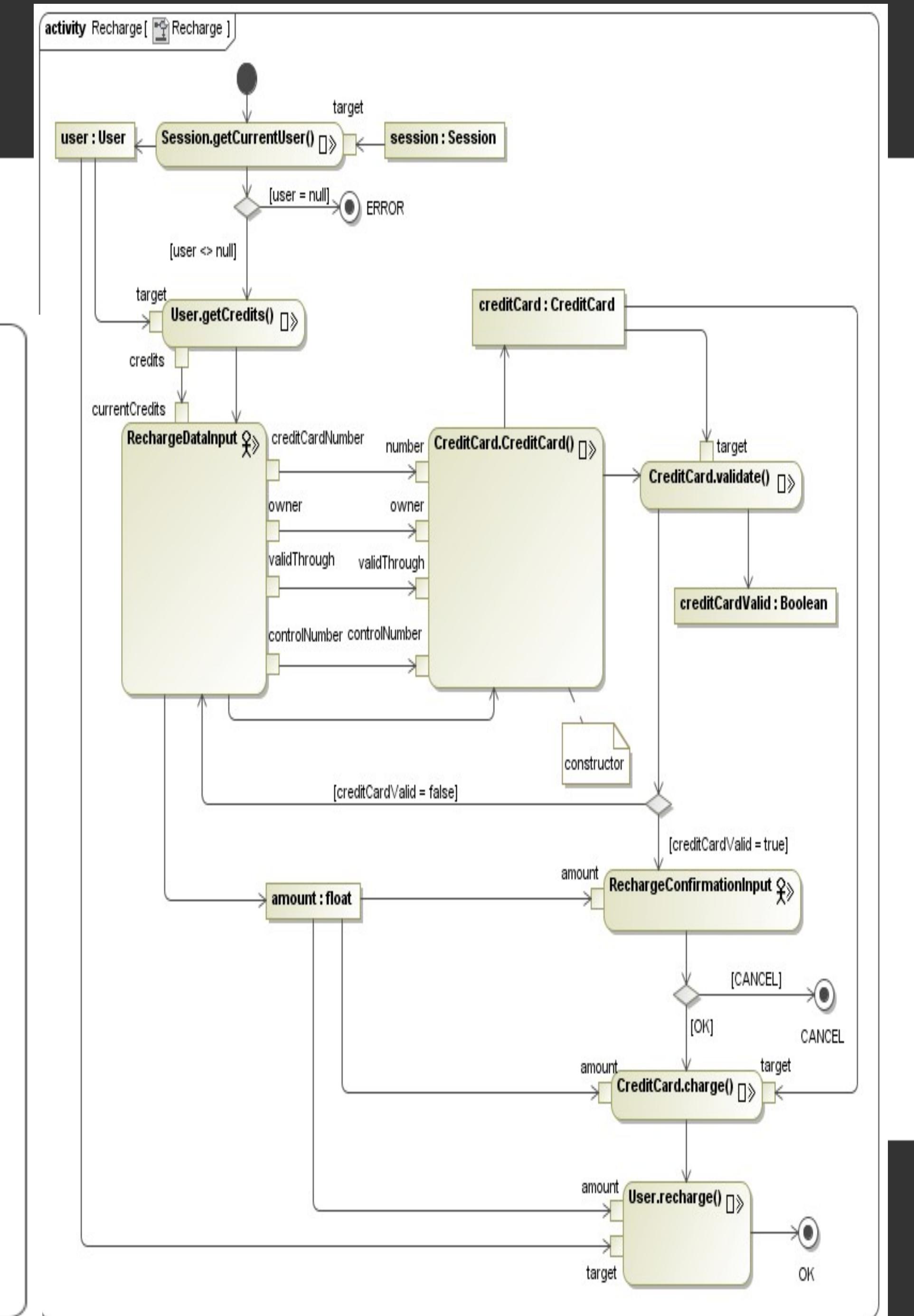
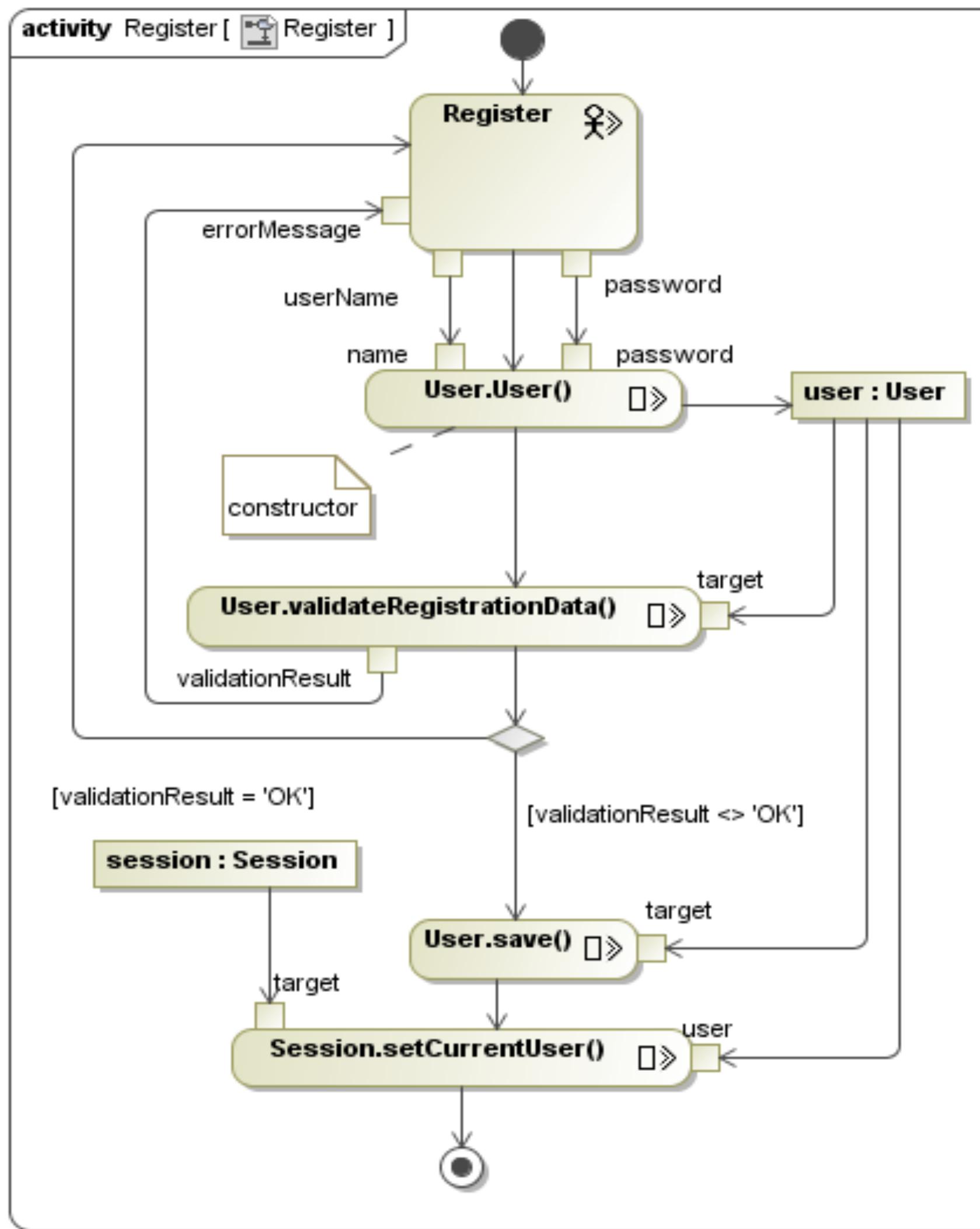
- Σ processClass
- ✖ userAction
- ✖ systemAction
- processLink



# Diagrama Proceso (actividad)



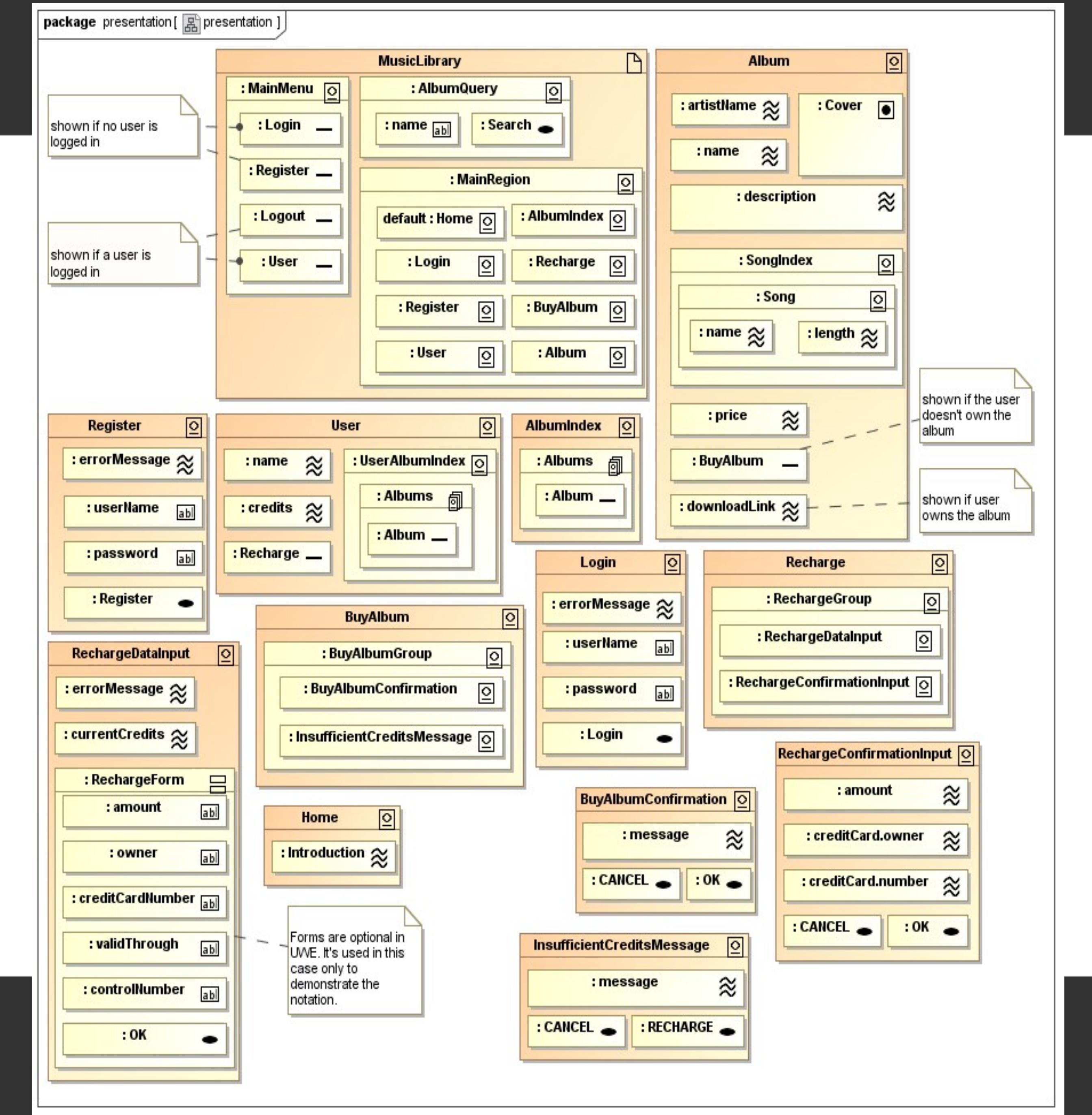
# Diagrama Proceso (actividad)



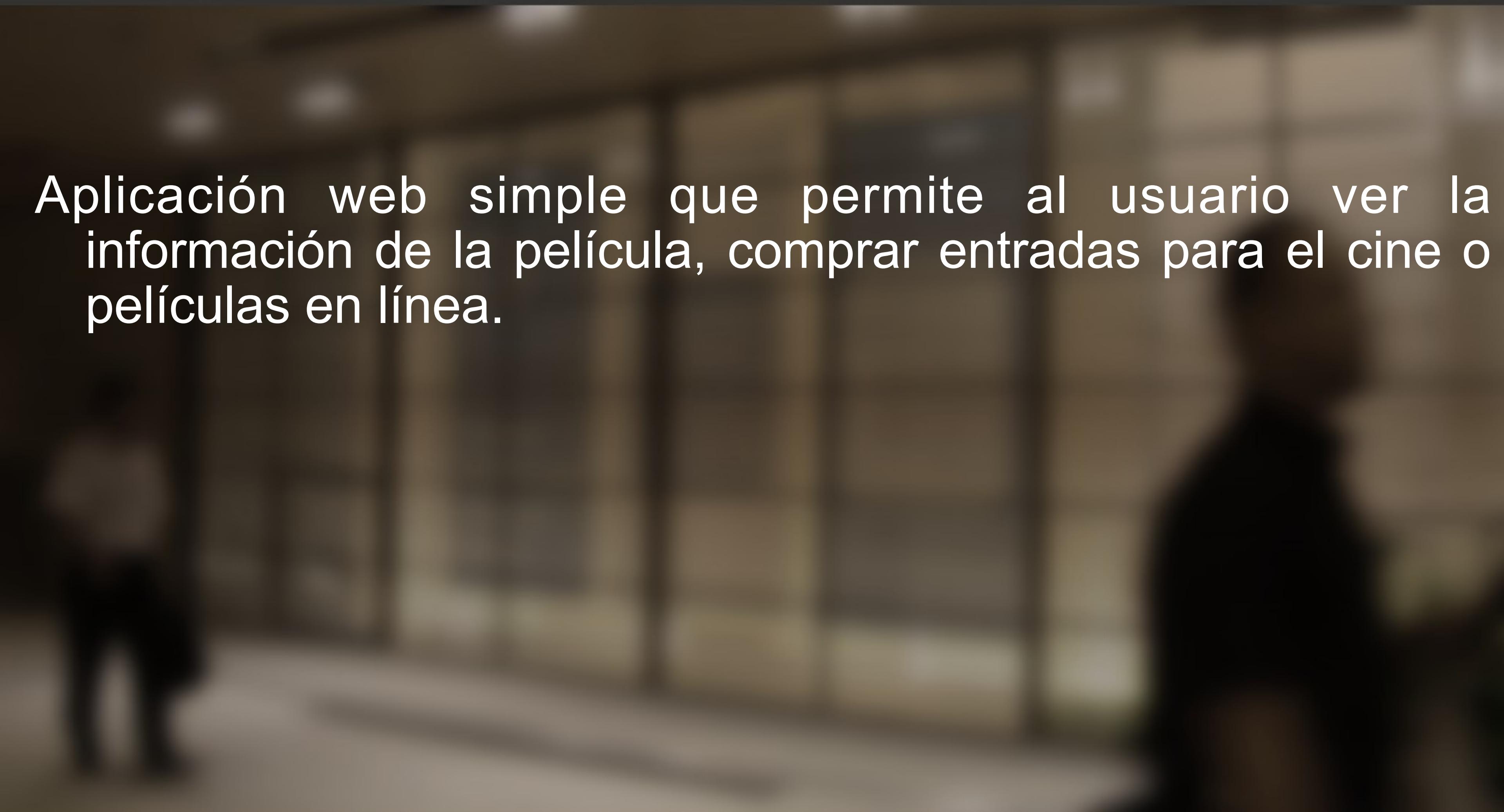
# Modelo de Presentación

## Presentation

- presentationAlternatives
- presentationGroup
- iteratedPresentationGroup
- inputForm
- presentationPage
- tab
- button
- anchor
- text
- image
- mediaObject
- selection
- fileUpload
- customComponent
- slider
- textField
- imageInput



# Base de Datos de Películas IMDB

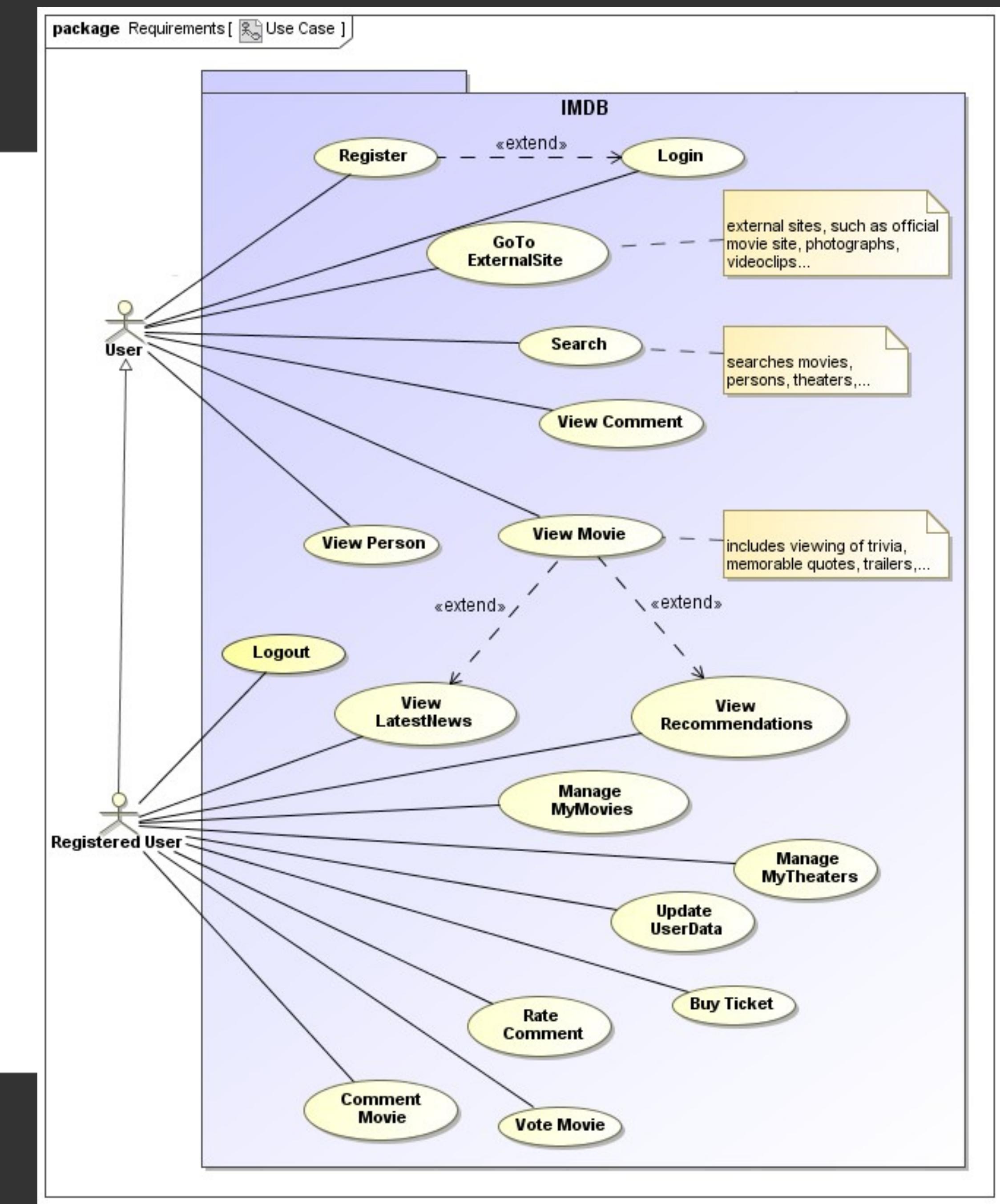


Aplicación web simple que permite al usuario ver la información de la película, comprar entradas para el cine o películas en línea.

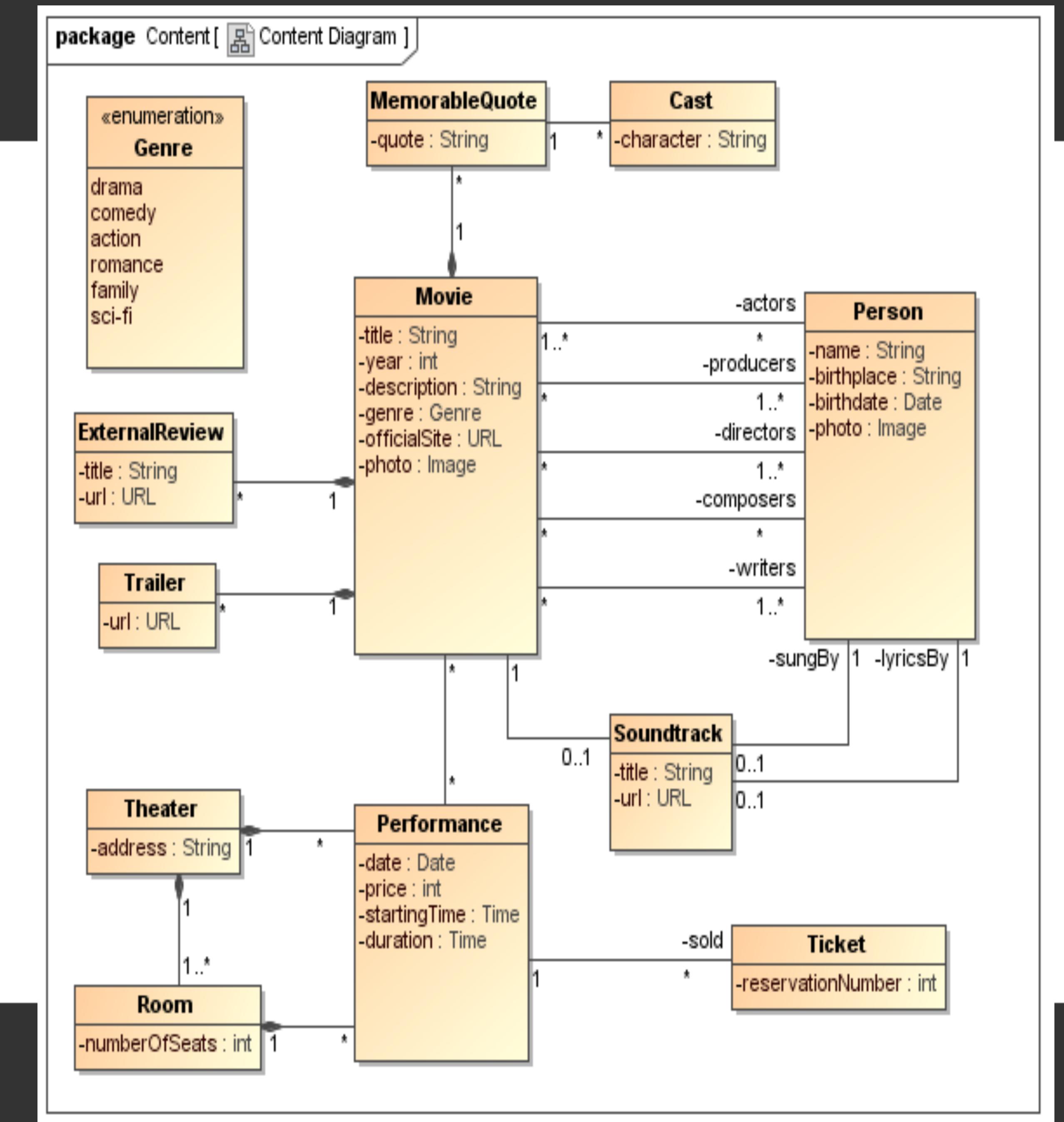
## Requisitos

- 1 Se hace una distinción entre los usuarios y los usuarios registrados. Sólo los usuarios registrados pueden comprar los billetes. Los usuarios no registrados pueden registrarse con un nombre de usuario que no ha sido utilizado por otro usuario, una dirección de correo electrónico y una contraseña elegida libremente.
- 1 Cada usuario puede buscar películas, actores, directores y otras personas y teatros.
- 1 Por cada película y la persona hay una página que muestra sus detalles. Los detalles de la película se compone de un título, el año de realización, una descripción, el género de la película y una foto. También hay listas de las personas que participaron en esta película, una banda sonora y otra información. Los detalles que una persona se limita a su nombre, fecha de nacimiento, lugar de nacimiento y una foto y los detalles de un teatro se compone de una dirección, salas y actuaciones. Los usuarios registrados pueden comentar y calificar las películas, almacenar sus películas favoritas y teatros en una lista personal y comprar los billetes.

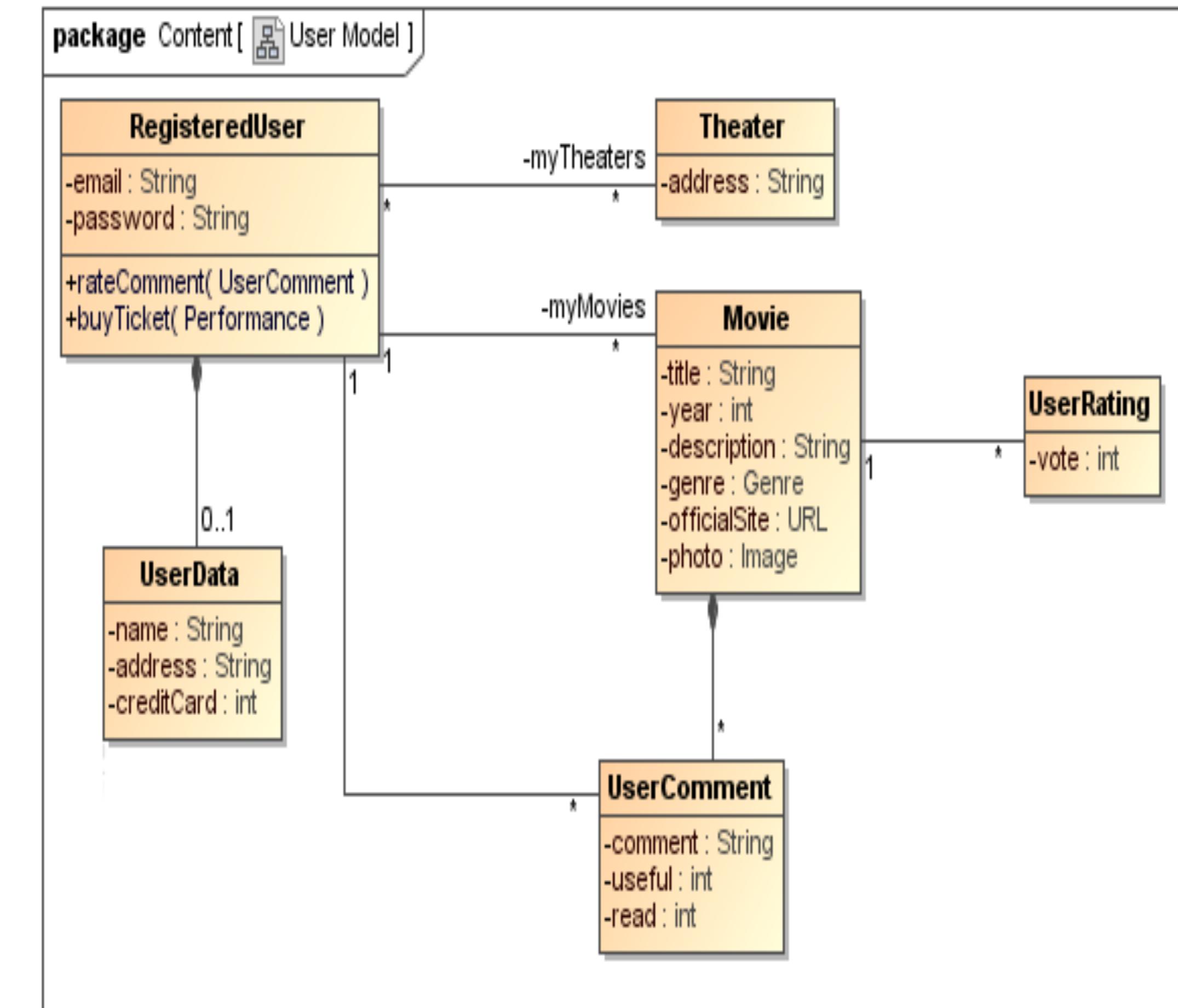
# Casos de Uso



# Diagrama de Contenido



# Diagrama de Contenido (modelo usuario)



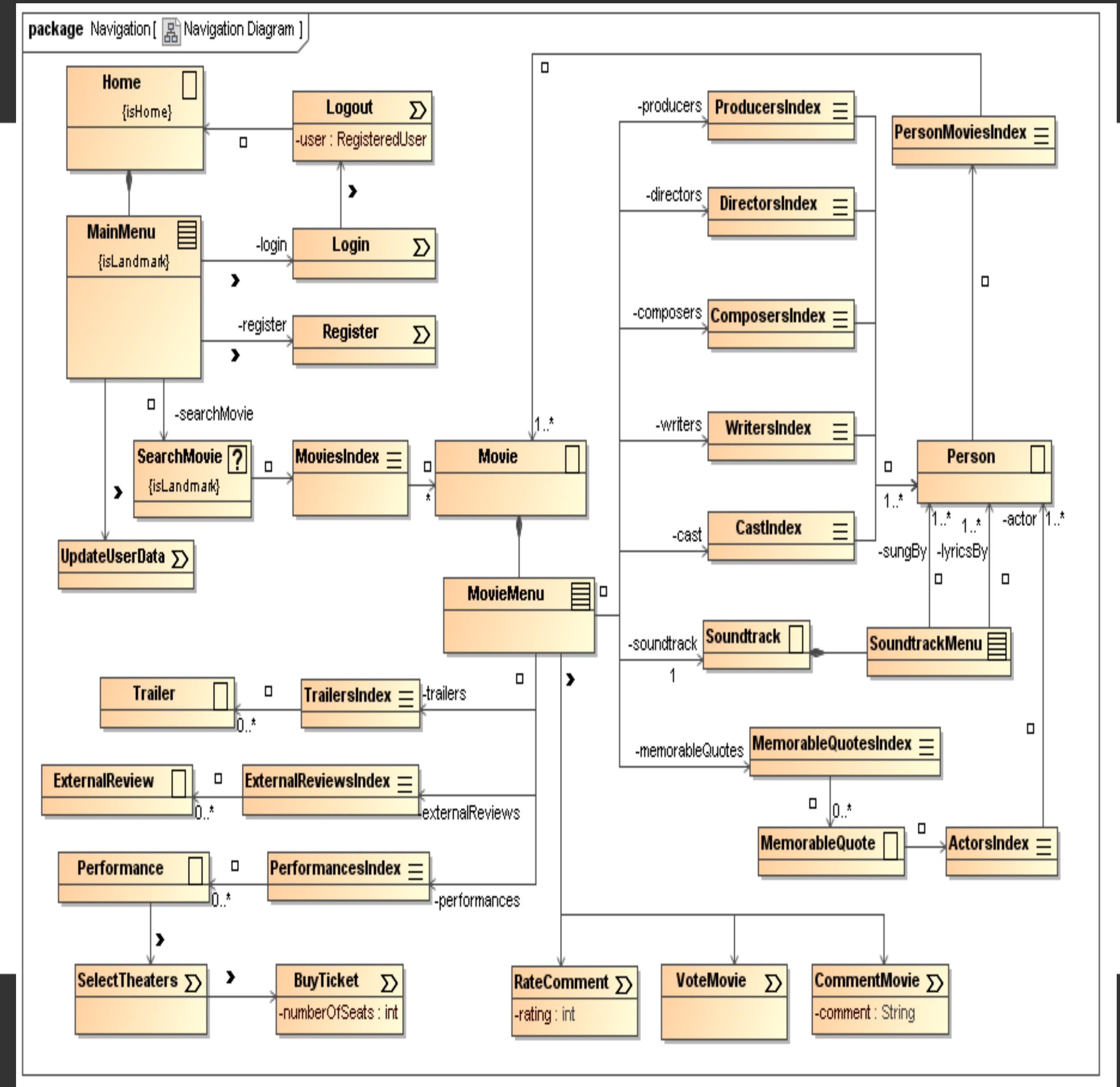
# Modelo Navegación

## Navigation

-  navigationClass
-  menu
-  externalNode
-  query
-  guidedTour
-  index
-  navigationLink

## Process

-  processClass
-  userAction
-  systemAction
-  processLink



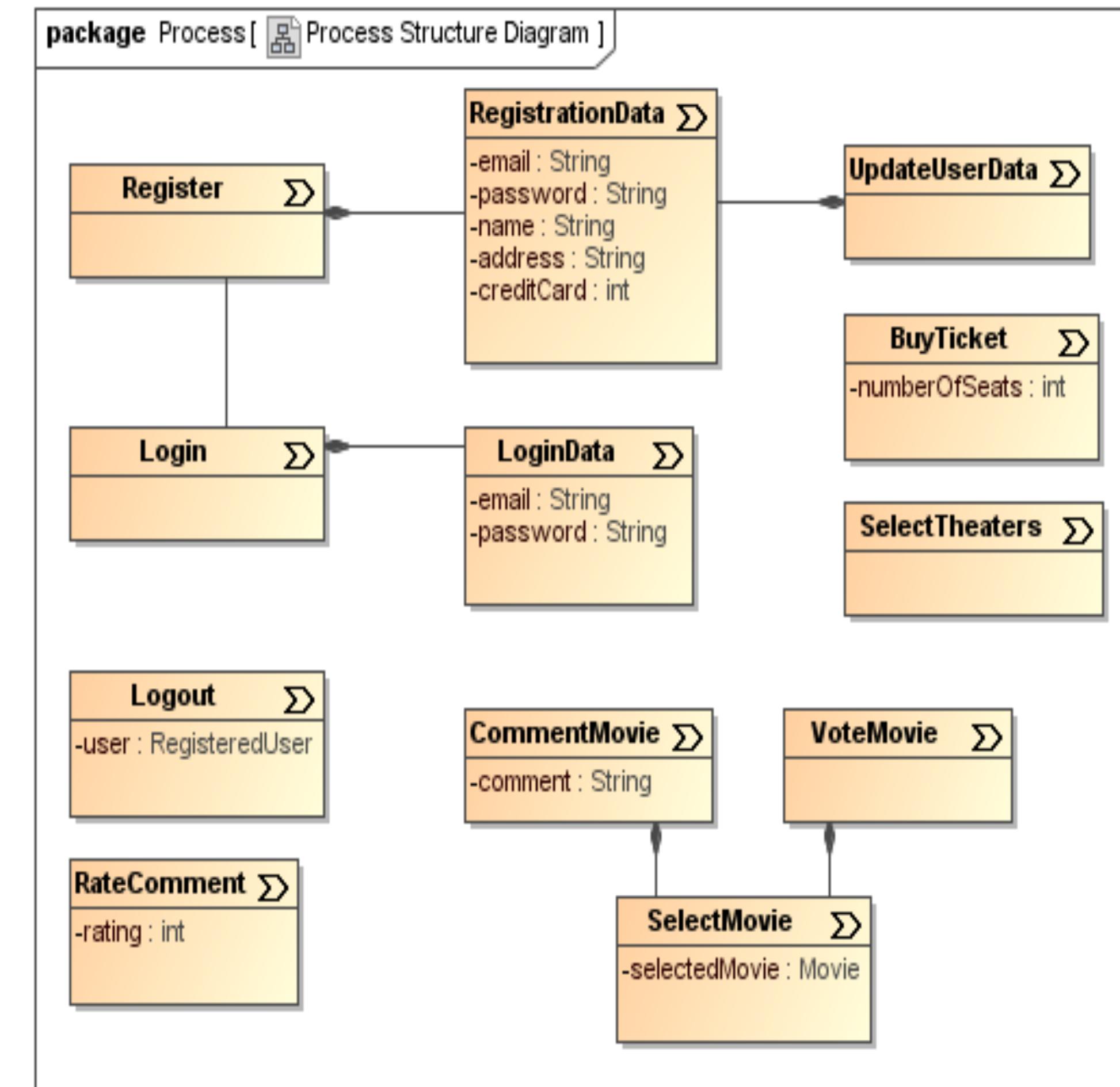
# Modelo Proceso

## Navigation

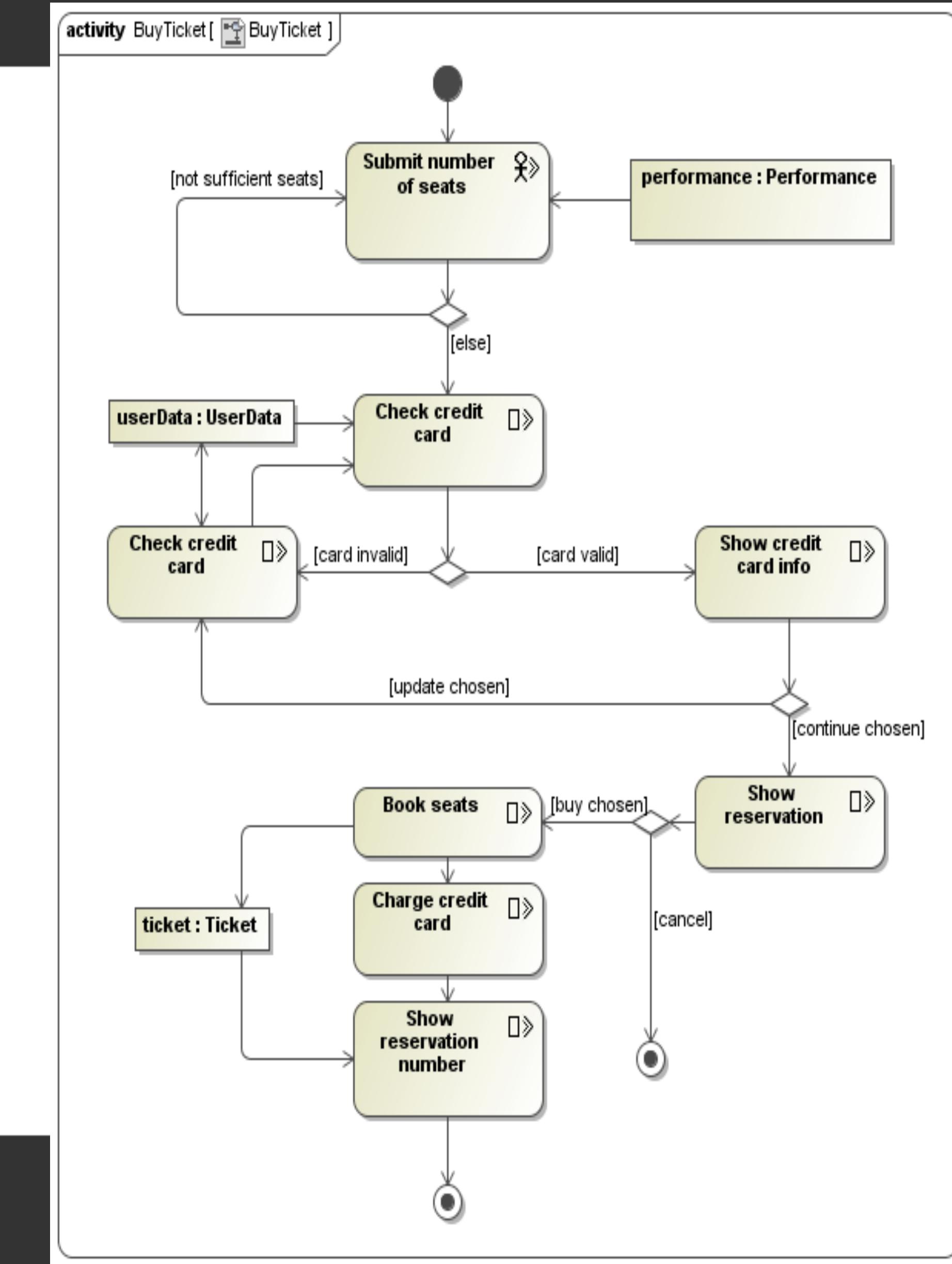
- navigationClass
- ☰ menu
- ↗ externalNode
- ?] query
- guidedTour
- ≡ index
- navigationLink

## Process

- Σ processClass
- ❖ userAction
- systemAction
- processLink



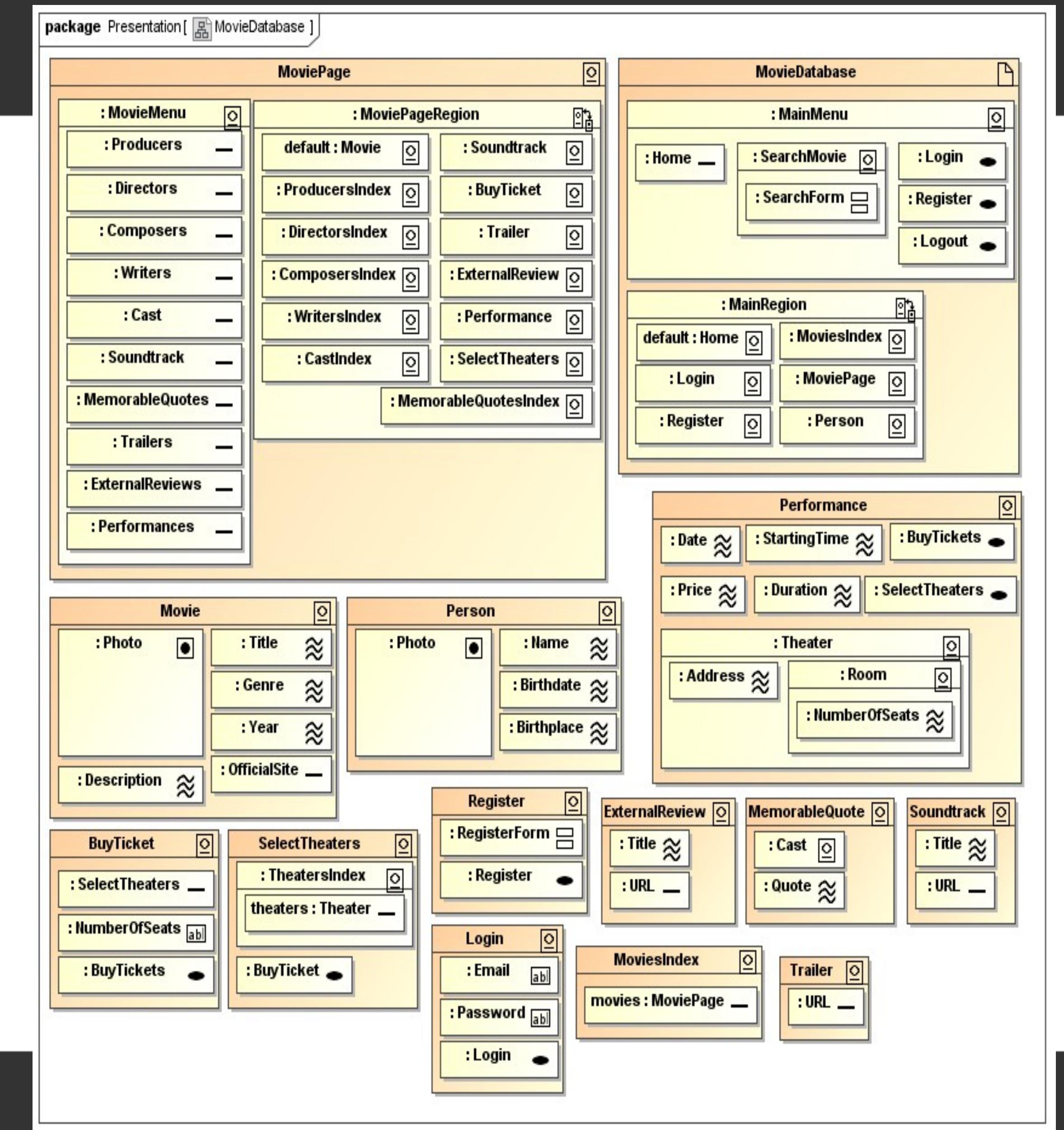
# Proceso (actividad)



# Diagrama de Presentación

## Presentation

- ⌚ presentationAlternatives
- ⌚ presentationGroup
- ⌚ iteratedPresentationGroup
- ⌚ inputForm
- ⌚ presentationPage
- ⌚ tab
- button
- anchor
- ~~ text
- image
- ⌚ mediaObject
- ⌚ selection
- ⌚ fileUpload
- ⌚ customComponent
- ⌚ slider
- ⌚ textInput
- ⌚ imageInput



# PVS

Sistema para la gestión de la publicación. Esta aplicación web está diseñado para la gestión de las publicaciones científicas. Por lo tanto, las publicaciones se conservan en una base de datos que se puede acceder a través de una interfaz para leer o escribir. La caracterización de una publicación en el PVS es aproximadamente equivalente a la caracterización de una publicación por el formato BibTeX.

# Requisitos

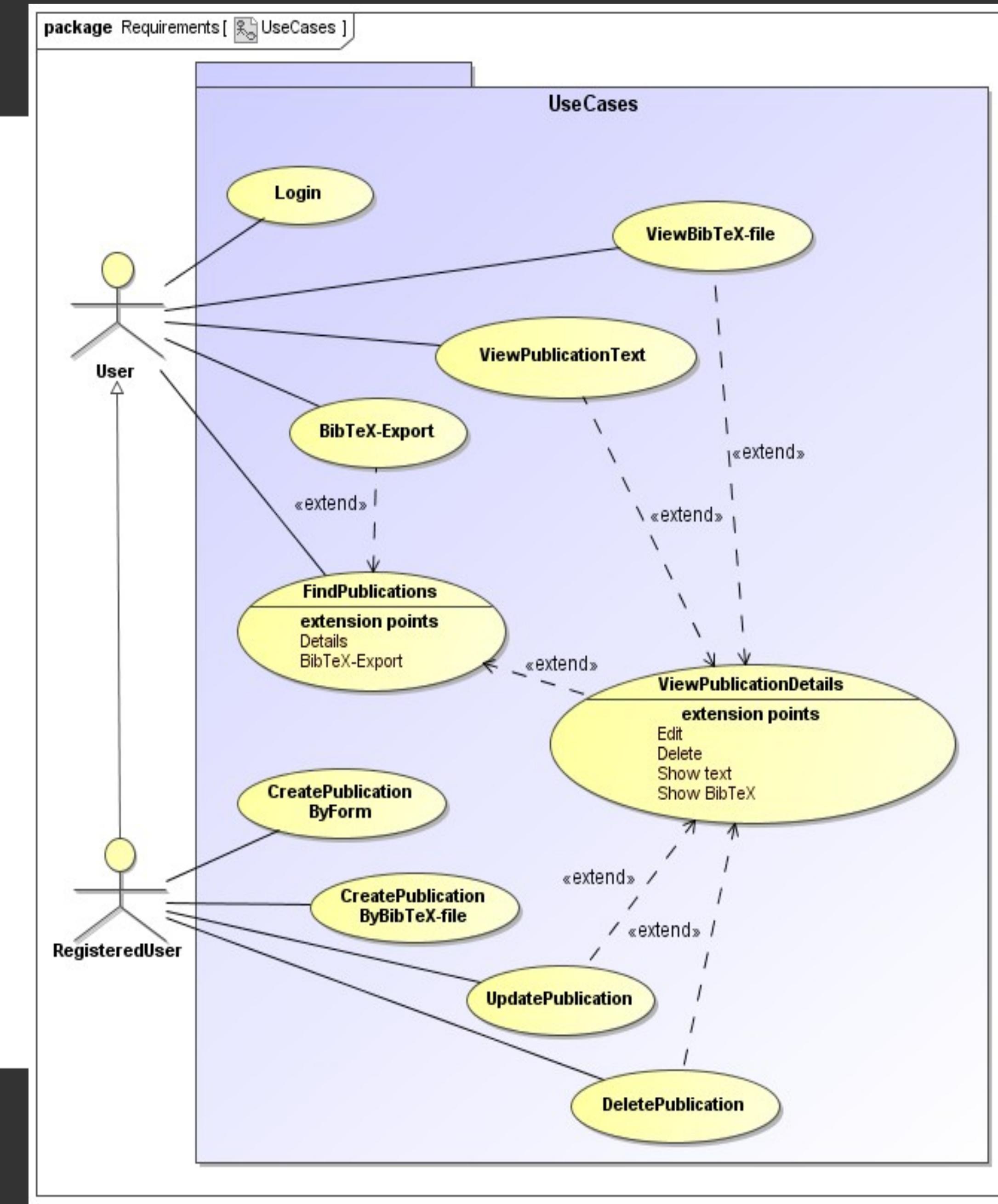
## Descripción informal de las funcionalidades de esta aplicación web:

- 1 PVS distingue dos grupos de usuarios. Los usuarios registrados son el principal grupo objetivo y pueden usar la completa funcionalidad de la aplicación. Además, la aplicación está disponible públicamente en limitado grado
- 1 Todos los usuarios deben ser capaces de consultar a las publicaciones a través de dos modos diferentes. La búsqueda simple con cuatro parámetros fijos y el modo avanzado, donde el usuario puede establecer sus propios parámetros de búsqueda.
- 1 Una lista de las publicaciones que se han generado a través de la función de búsqueda se pueden exportar como un archivo de BibTeX.
- 1 El texto de la publicación puede ser recuperado si está disponible. Hay dos maneras de lograr esto. El texto puede ser descargado del servidor o puede ser puesto a disposición a través de la especificación de una URL externa.
- 1 Para cada publicación que se incluye en la base de datos se pueden examinar sus detalles. Dentro de este punto de vista toda la información almacenada para su publicación y los autores y editores son mostrados.
- 1 Los usuarios registrados pueden utilizar un formulario Web para agregar una nueva publicación de la base de datos o modificar los datos de una publicación ya existente. También puede eliminar una publicación a partir de la base de datos.
- 1 Las publicaciones descritas en el formato BibTex se pueden importar en la base de datos de PVS. El usuario tiene la opción de subir un archivo BibTex o para introducir el contenido de un archivo en un cuadro de texto designada del formulario de importación.
- 1

## Bibtex

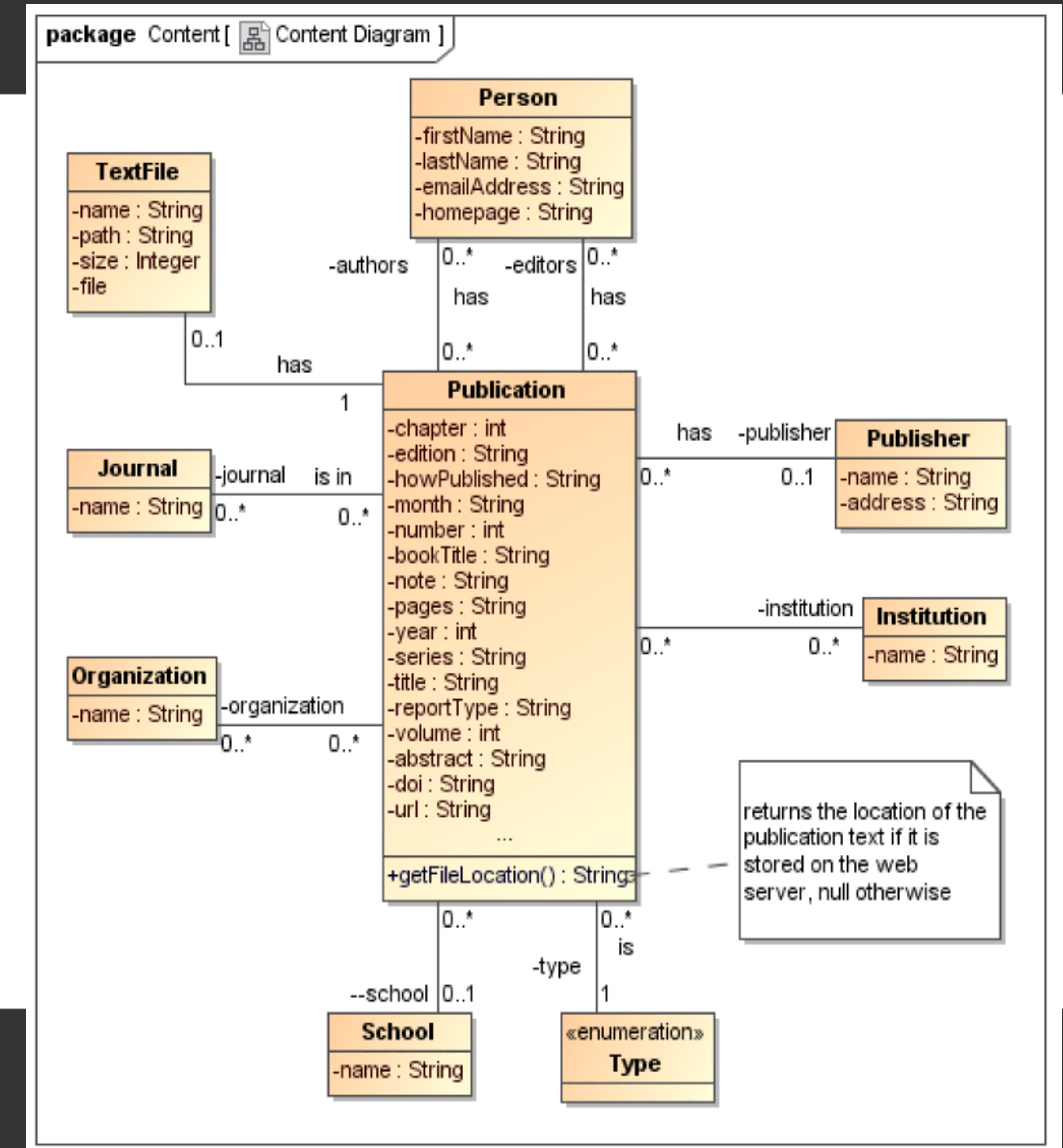
```
@Book{abramowitz+stegun,  
author = "Milton Abramowitz and Irene A. Stegun",  
title = "Handbook of Mathematical Functions with  
Formulas,  
Graphs, and Mathematical Tables",  
publisher = "Dover",  
year = 1964,  
address = "New York",  
edition = "ninth Dover printing, tenth GPO printing",  
isbn = "0-486-61272-4"  
}
```

# Casos de USO



# Contenido

Críticas a la  
solución ?



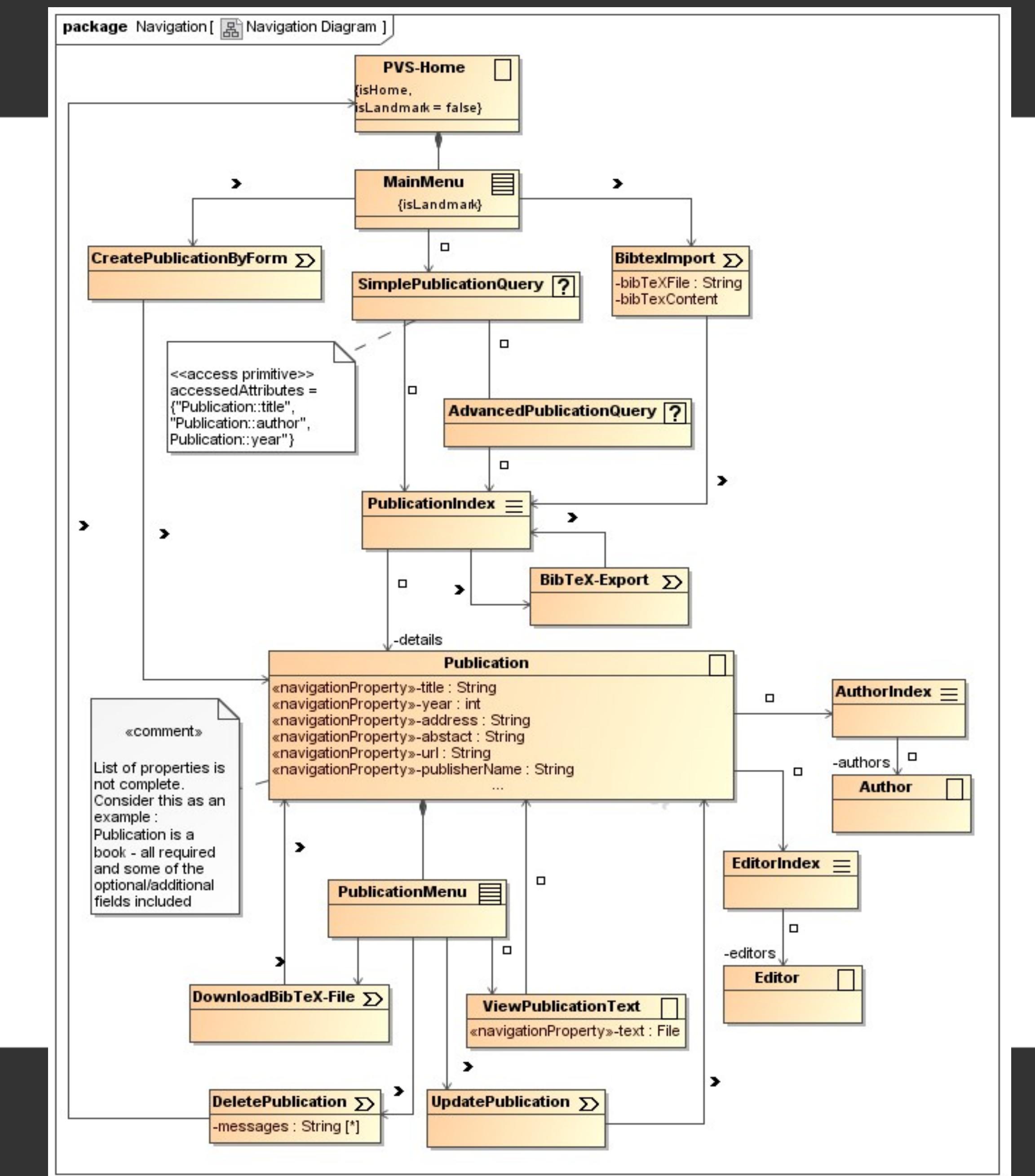
# Navegación

## Navigation

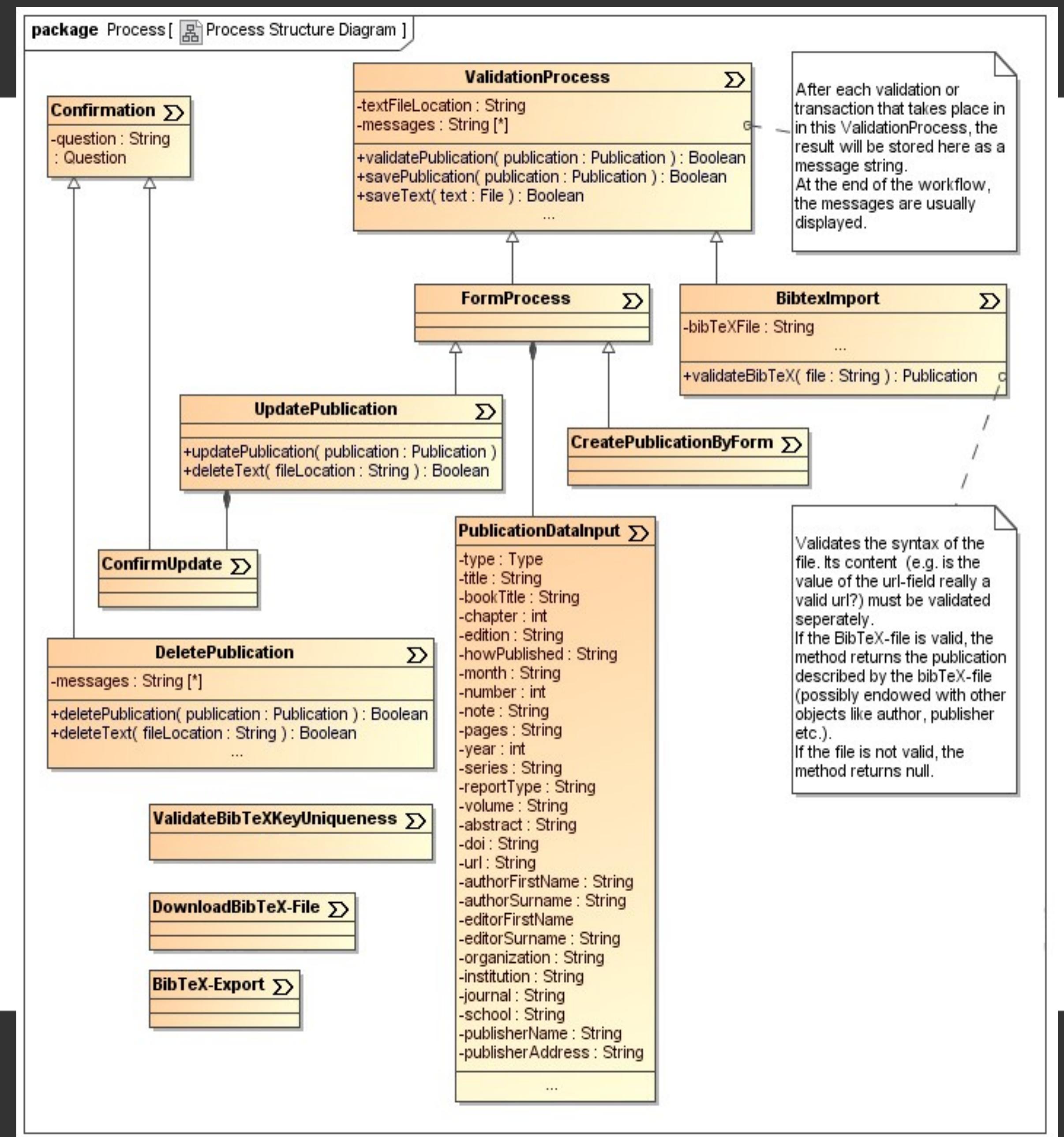
- navigationClass
- ☰ menu
- ↗ externalNode
- ? query
- guidedTour
- ☰ index
- navigationLink

## Process

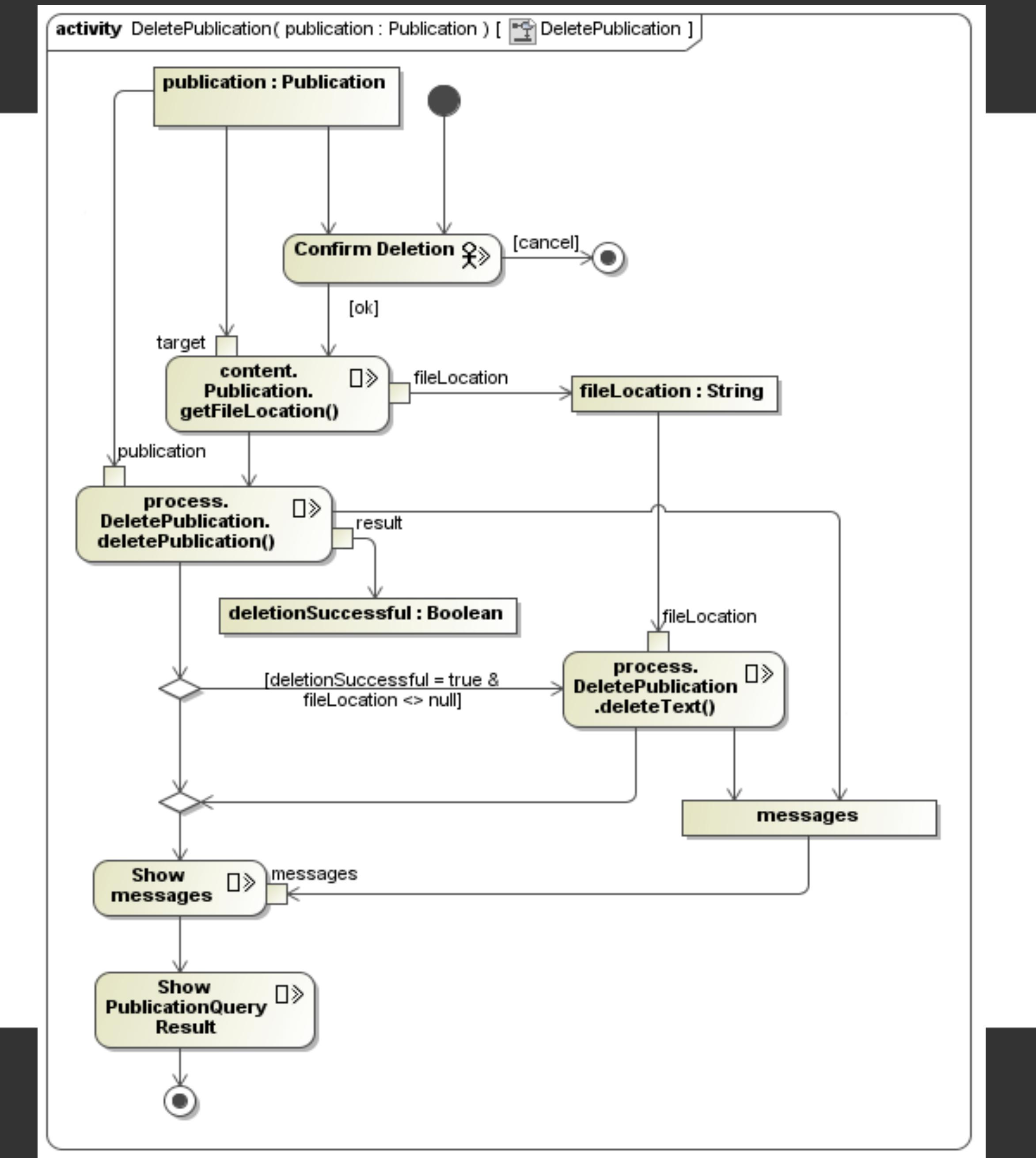
- Σ processClass
- ⌚» userAction
- » systemAction
- » processLink



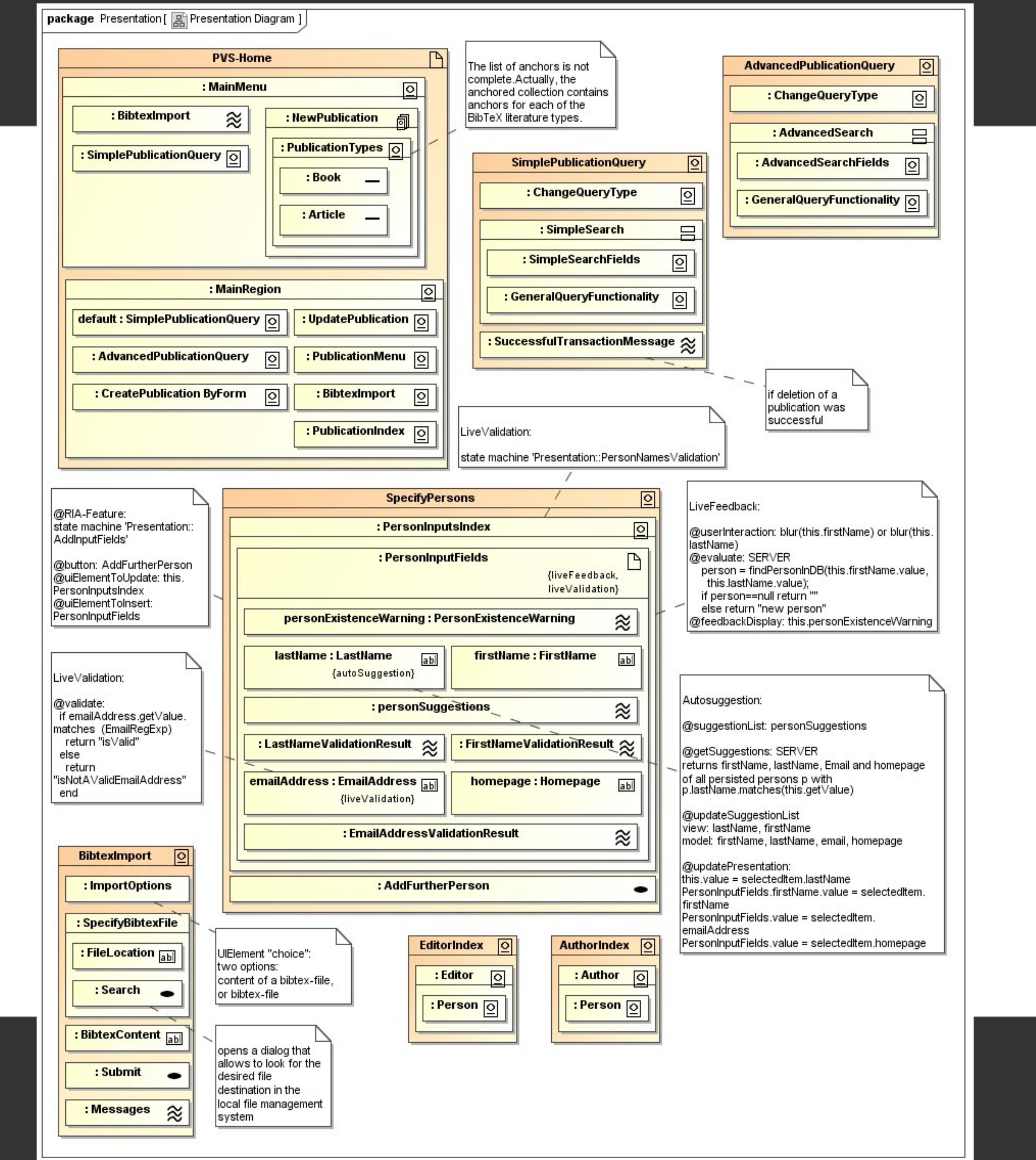
# Proceso



# Proceso



# Presentación



# Linkbook (Philoponella)

Aplicación RIA (adaptive rich internet application) de una red social que permite a los usuarios administrar las direcciones web interesantes e información relacionada, para encontrar nuevos sitios web interesantes y chatear con otros usuarios. El RIA observa el comportamiento del usuario y utiliza esta información para adaptar aspectos de navegación y presentación de la aplicación web de forma dinámica al perfil del usuario. Características como sugerencias en vivo, validación, arrastrar y soltar y auto aumenta la usabilidad de la aplicación.

# Requisitos. Funcionalidad

- | Linkbook distingue dos tipos de usuarios: los visitantes anónimos de la aplicación y el usuario registrado. Ambos tipos de usuarios tienen la posibilidad de ver toda la información en el sistema, excepto los datos privados de usuario. Los usuarios registrados tienen también la opción de ampliar la base de datos de la aplicación con diferentes métodos, descritos más adelante.
- | Comentarios sobre los sitios web constan de un nombre, la dirección, la categoría de sitios web, algunas descripciones en varios idiomas, algunas imágenes de vista previa y la lista de comentarios del usuario.
- | Las categorías de los sitios web pueden anidarse dentro de otras categorías.
- | Los usuarios registrados pueden administrar sitios web interesantes en su propia lista de favoritos y otros usuarios de una lista de amigos.
- | Las entradas pueden ser buscadas por los usuarios de esta aplicación seleccionando una categoría de sitios web o que contengan una o más palabras clave. Los usuarios registrados también pueden limitar los resultados de búsqueda con respecto a otros usuarios de su lista de amigos.
- | Los usuarios pueden ver la parte pública de los favoritos de otros usuarios y amigos. Los usuarios registrados pueden enviar mensajes de otro usuario registrado.
- |

## Requisitos. Características RIA

- λ Validación de entradas, como nombres de usuario, direcciones de correo electrónico o direcciones de páginas web
- λ Los detalles de la lista de información se puede intercambiar y/o realizar un collage animado.
- λ Las entradas de las listas se pueden ordenar en categorías mediante arrastrar
- λ La vista previa de imágenes se pueden consultar en una ventana separada en su tamaño completo
- λ Al usuario se le presenta una breve lista de palabras clave que coinciden con el contenido del campo de búsqueda

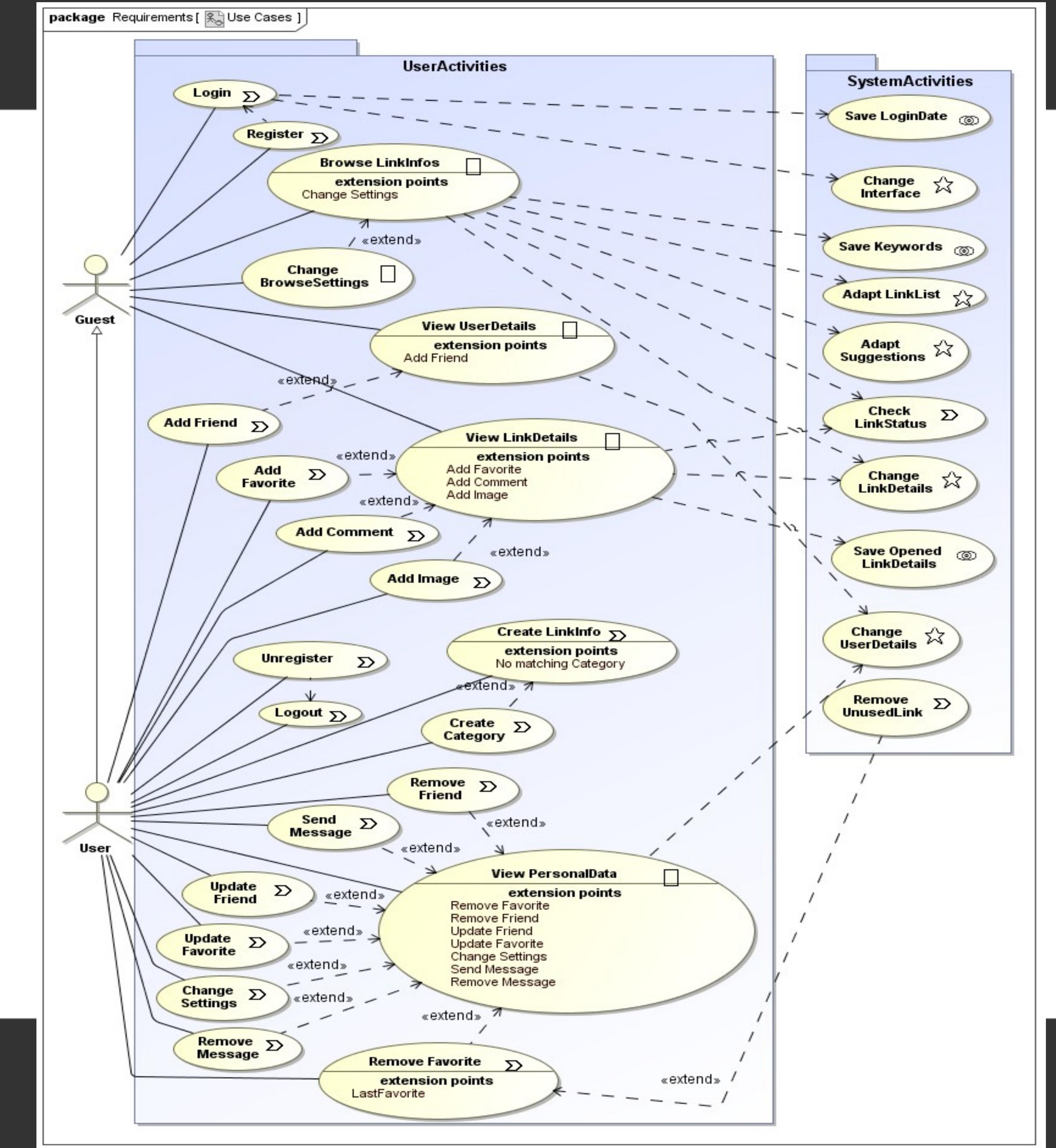
## Requisitos. Características RIA

- λ Validación de entradas, como nombres de usuario, direcciones de correo electrónico o direcciones de páginas web
- λ Los detalles de la lista de información se puede intercambiar y/o realizar un collage animado.
- λ Las entradas de las listas se pueden ordenar en categorías mediante arrastrar
- λ La vista previa de imágenes se pueden consultar en una ventana separada en su tamaño completo
- λ Al usuario se le presenta una breve lista de palabras clave que coinciden con el contenido del campo de búsqueda

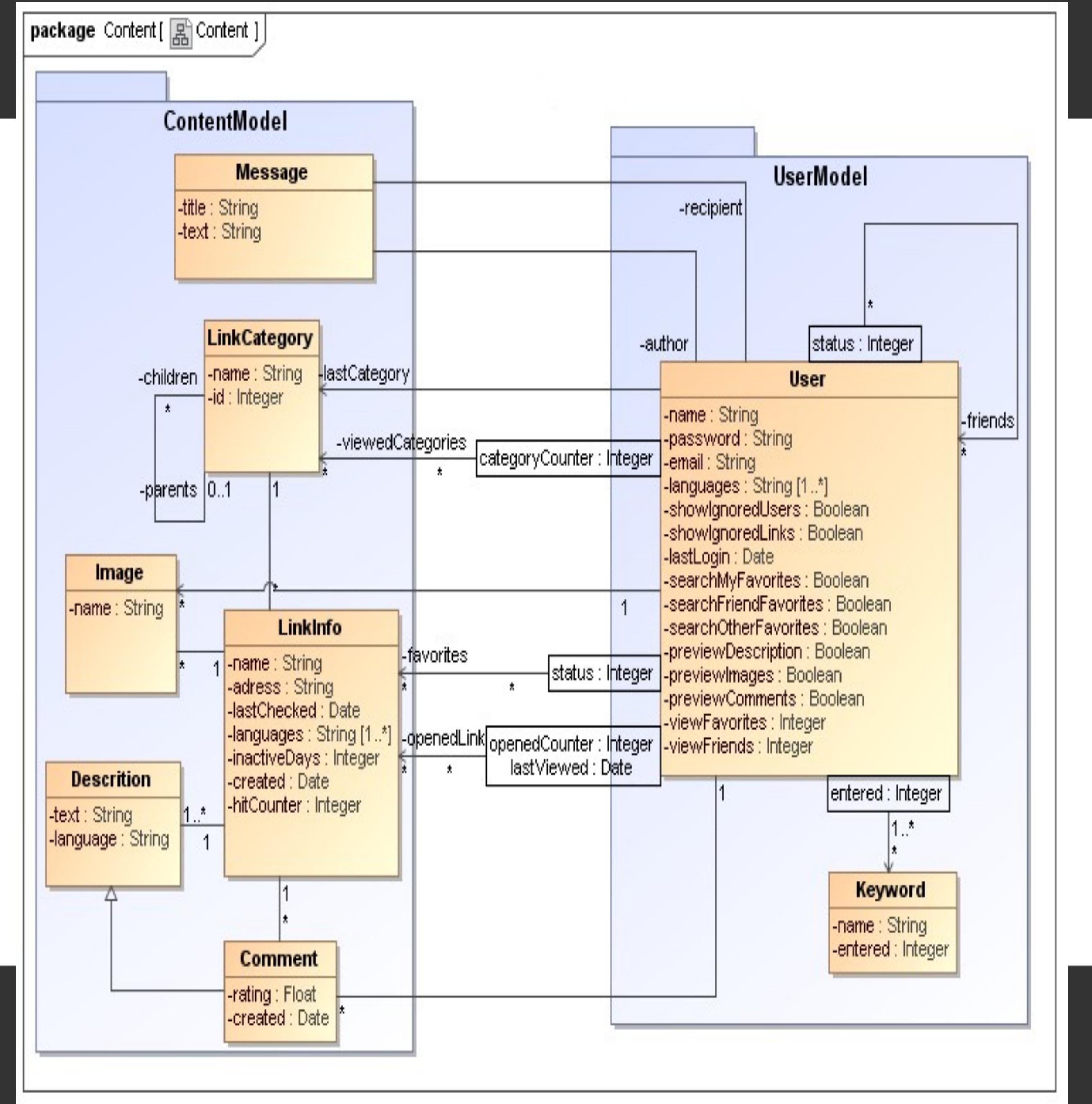
## Requisitos. Adaptabilidad

- λ Ordenar las diferentes listas, como la lista de búsqueda, etc.
- λ Resaltar elementos de lista individuales para mostrar sus cambios o la actividad
- λ Eliminar las páginas escritas en un idioma que el usuario no es capaz de entender
- λ Ajuste de la configuración de la interfaz de usuario según las actividades anteriores del usuario

# Casos Uso



# Modelo Contenido



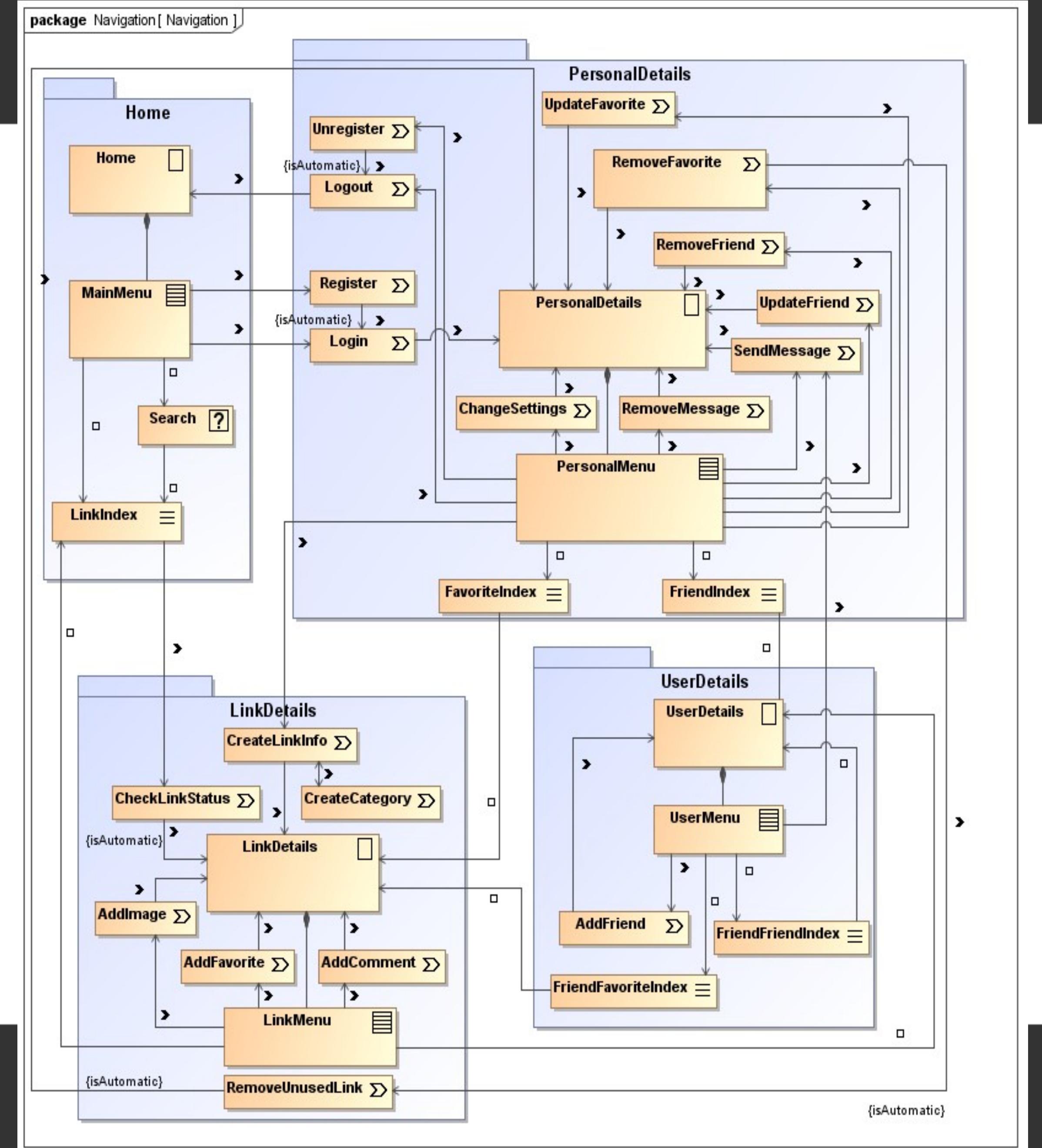
# Modelo Navegación

## Navigation

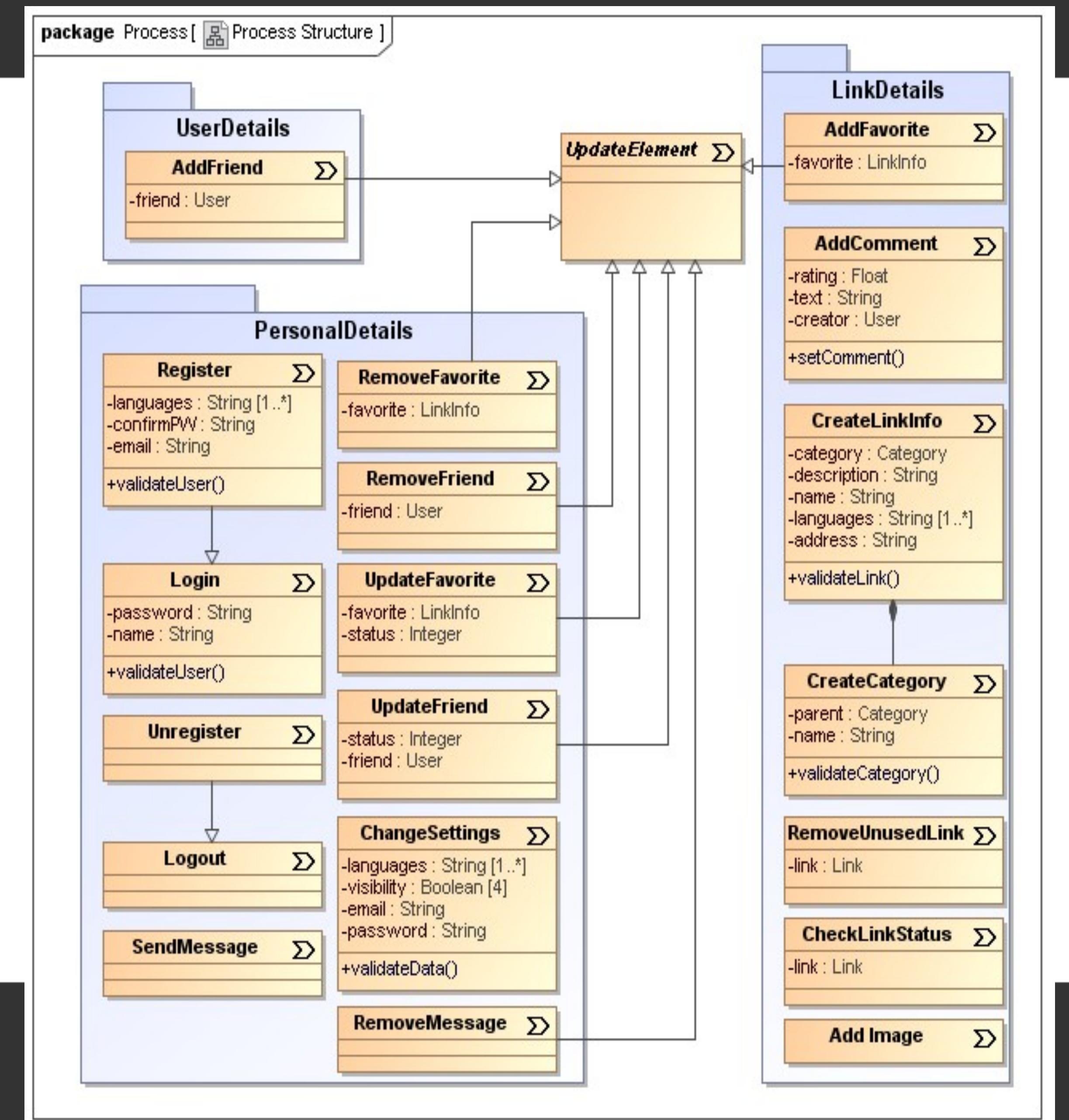
-  navigationClass
-  menu
-  externalNode
-  query
-  guidedTour
-  index
-  navigationLink

## Process

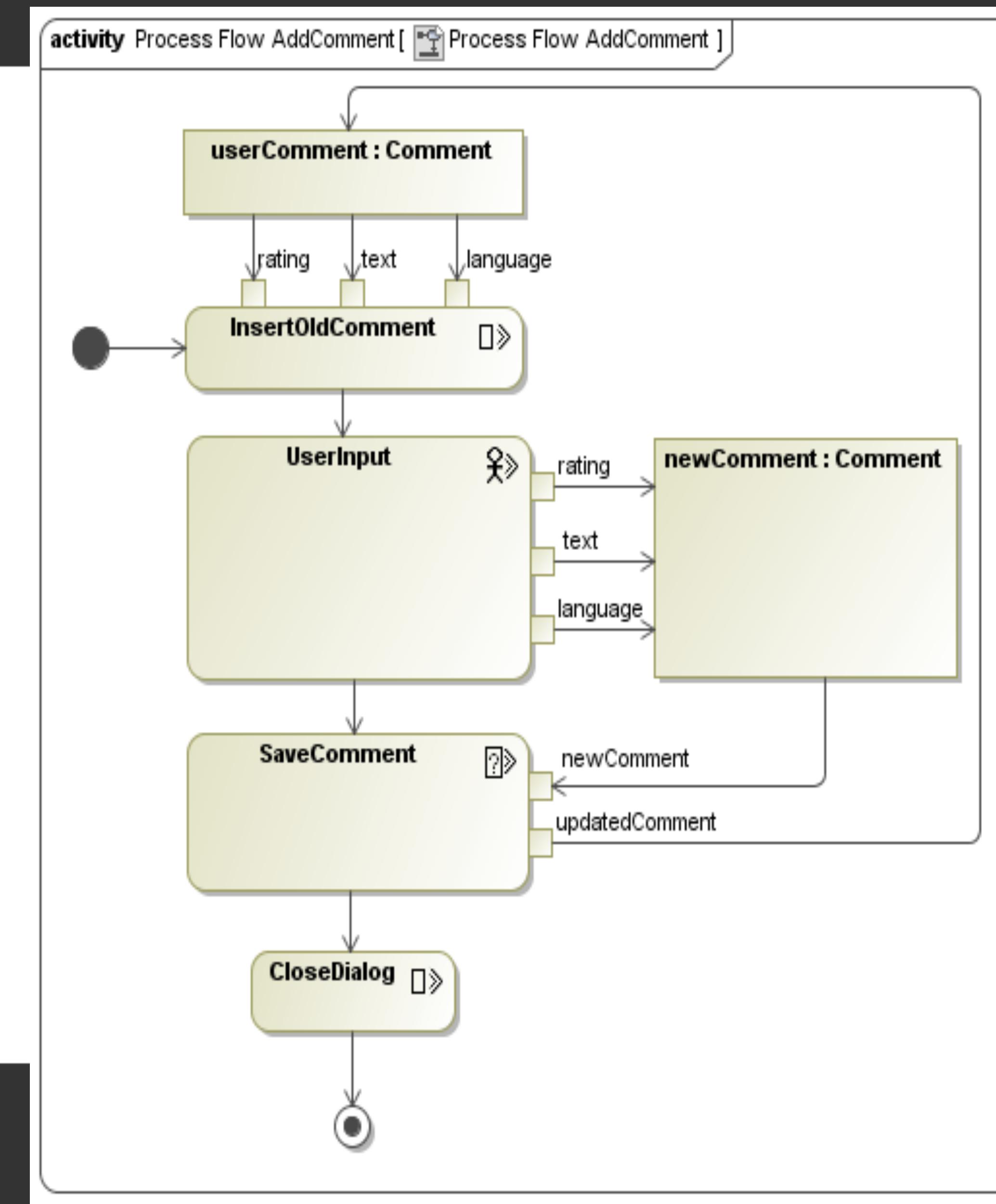
-  processClass
-  userAction
-  systemAction
-  processLink



# Diagrama de Proceso



# Diagrama de Proceso. Actividad



# Diagrama de Presentación

## Presentation

- presentationAlternatives
- presentationGroup
- iteratedPresentationGroup
- inputForm
- presentationPage
- tab
- button
- anchor
- text
- image
- mediaObject
- selection
- fileUpload
- customComponent
- slider
- textInput
- imageInput

