Restricciones

# Restricciones básicas

* R1: Máx. 4 mascotas **activas**
* —--R2:Comprobar las compatibilidades
  + Si < 70% -> no pueden amigos Hacer un enum con todos los gustos de las mascotas y si coinciden en un 70% son besties
* R3:Anuncios certificados Productos recomendados con % de fiabilidad
* R4:Alimentación diferente para perros y gatos
* R5:Fotos diarias max 3
* R6:Comportamiento agresivo=suspensión
* R7:Para adopcion hay que tener todas las vacunas\*
* R8:Un usuario puede apuntarse a un evento si no está inscrito a otro en la misma fecha
* R9:Para crear eventos -> al menos 6 usuarios (propietarios)
* R10:Si el tiempo hasta que empiece el evento es menor a x y no hay suficientes usuarios interesados -> el evento no se confirma
* R11:Un usuario no puede tener más de una cuenta activa. -> dni
* R12:Si una mascota es suspendida por comportamiento agresivo, no puede ser inscrita en eventos hasta que se levante la suspensión.
* R13: Para que un producto este certificado debe haberla valorado mínimo 3 veterinarios
* R14: Una mascota no puede ser amigo de sí mismo
* R15 : Una mascota suspendida no puede estar en adopción
* R16:Los eventos no pueden solaparse si se realizan en la misma ubicación.
* R17:Una mascota debe tener como mínimo 3 gustos..
* R18: Un evento creado debe ser programado con una semana de antelación

..

javi

Adri machuca

adri

Alvaro

cambiar nombre calcularCompatibilidad

Restricciones chat gpt

* Los productos recomendados deben estar categorizados por especie (perros, gatos, etc.).
* Un usuario no puede marcar como “amiga” a su propia mascota.
* Los anuncios de adopción deben incluir al menos una foto de la mascota.
* Una mascota debe tener al menos un propietario registrado para poder estar activa en la plataforma.

**Pensamientos**

Las mascotas de un propietario son como una subpágina?? Si la mascota tiene varios propietarios ¿cómo hacerlo? id único para cada mascota?

* Para la recomendaciones de productos:
  + Crear clase veterinario con método que certifique el producto: prerrequisito para que producto exista -> certificarProducto
  + Atributo que indique si es certificado o no: ¿cómo indicamos que está certificado sin una clase veterinario?
* Propietario -> Publicaciones(Necesitaria un vector o algo que almacene todas las fotos asociadas a él? o sería ya la relación 1 a muchos entre publicación )
  + La relación podría ser 1 a muchos para un único usuario o que una publicación pueda tener más de un propietario asociado (por el tema etiquetas)

# Dudas

Anuncios añadiendo atributo “tipoAnuncio” y eliminamos dos clases?

* Para la amistad: el % de compatibilidad -> enum con x(10 por ejemplo) gustos y para poder ser amigos las dos mascotas deben tener un % de elementos iguales.

Array de los elementos de enum????

Puede tener varios valores de un enum?

codigo diagrama clases

[enlace diagrama](http://www.plantuml.com/plantuml/png/ZLN1RkCs4BthAuXSrjlk5kXLfxXsKlIG84ZRN8qnENQce3WgENA35lsNVaUFxO-L2asPOXHqSuYbuSpZl3UFr7KGy39qjeb_7zJ6GWWOcaQF3-hdC80dLdyqpJz_gQyLcLRzu0IFwCyHnztUOwj-54_ks3H_DaqkVl3S4yeh0Bu8ElCM47gqbWnF02bi7EM8Md6lYHtOJohYdb3z-xSZ3Hzp0xSi7BO4dY0KtHMDt47GB50tSI5rJfYgrQhQQObg2cBeKLCTzV0xLA43wcUu1qqTePD8U0k2vxK0TZ1LlkQkPo-GqjT7Zu5EiUmxPelWfkq533-WFv1T7C8JwC51CS0KgCU81inFt45ggnwZ6RIK9iYXPJFKn5duZMZhtlCU33-XO5mOXzTXnAhvQOZu75TlvZghLO_UyoTr18d_KVGtUKmzUnsXu1Uy9TYJfKZj4JL7V0EZarfrQnbaHlBIHycpY7uHLJ8Rz487qccZSlNpvqmff_WiuwgVzCoXZ_CH37jxnYowAw9LTybivNhfDZTqT_Zz40IMZyjcb4egAIx1UY_1bth0orkK8kpPU_0NqDKqlTHqjLHqTtEAyNoXrDAKXsB3xa2-I_mhMxa7y4AQUd22OL6zjCzCE8-QlG5XJnpU6IuuZHPygbuPWh2Es7O-fJcZjHkSfl-Xv2B4PBUbyhL9zsI6oV49ulHwyCFQRWacFPQGiWVVHNfhrpFC1_M85eJOXMVgGzEKdu2hhwwYJkFFT26tIWXTb3S7hhxDEVdYQfN78mNnl3EuInn6caND4YvqTEIm25pAswh-yeOnlrmiQeeZqIezlLG5HIVvEzSg5FfrG84gTwdhCZShMoQMbrHxkKSCxtgEtYqkkQBx-3DXHJ7771gJndabbwRKkkyXFOKTkQ0z2OUneZ9RiKj1eqjFEtGxE2TEbLClo_rbQ-tWDJsM9r_TusygcVJ0Fh4OgIQlgLZw3CvOLEah4RAZioJrrJcBryVVu4YudztWQlQrts8GSc2WEf0Bbfpjb0P0UnoJhsDqwEn_)

<https://www.plantuml.com/plantuml/uml>

@startuml

' Classes

abstract Usuario {

+ id: Integer

+ nombreUsuario: String

+ Sexo: String

}

class Propietario extends Usuario {

+ nombre: String

+ dni: String

+ nFotosDiarias: Integer

}

class Mascota extends Usuario{

+ nombre: String

+ especie: String

+ raza: String

+ fechaNacimiento: Date

+ salud: String

+ comportamientoAgresivo: Boolean

+ estadoPerfil: String

+ vacunasAlDia: Boolean

}

class Vacuna {

+ idVacuna: Integer

+ edadToma: Date

}

class Producto {

+ idProducto: Integer

+ nombre: String

+ foto: String

+ esAprobadoVeterinariamente: Boolean

+ categoriaEspecie: String (perro, gato, etc.)

+ porcentajeFiabilidadRecomendacion: Float

}

class Veterinario {

+ idVeterinario: Integer

+ nombre: String

+ idCertificacion: String

--

+ certificarProducto(producto: Producto)

}

class Publicacion {

+ idPublicacion: Integer

+ fechaPublicacion: Date

+ numMeGusta: Integer

}

class Comentario {

+ idComentario: Integer

+ texto: String

+ numMeGusta: Integer

--

+ borrarComentario()

+ publicarComentario()

}

class Evento extends Publicacion {

+ estadoConfirmacion: Boolean

+ nParticipantes: Integer

+ fechaEvento: Date

+ recordatorios: String

--

+ cancelarEvento(diasAntelacion: Integer)

}

class Anuncio extends Publicacion {

+ enlace: String

}

class AnuncioAdopcion extends Anuncio {

}

class AnuncioProducto extends Anuncio {

}

' Relationships

Propietario "1..\*" -- "1..4" Mascota : tiene

Mascota "\*" -- "\*" Vacuna : registro\_de\_vacunacion

Mascota "\*" -- "\*" Mascota : amigos

Mascota "1" -- "\*" Publicacion : publicaciones

Publicacion "1" -- "\*" Comentario : comentarios

Publicacion "\*" -- "\*" Usuario : etiquetas

Producto "\*" -- "\*" Veterinario : certificado\_por

Evento "\*" -- "\*" Usuario : usuarios\_inscritos

AnuncioAdopcion "1" -- "1" Mascota : mascota\_en\_adopcion

AnuncioProducto "1" -- "1" Producto : producto\_anunciado

' New class for recommendations to handle species

@enduml

Restricciones

R2 FALTANTE

class Mascota

attributes

idMascota : Integer

nombre : String

especie : String

raza : String

fechaNacimiento : Date

salud : String

comportamientoAgresivo : Boolean

estadoPerfil : String

operations

calcularCompatibilidad(otraMascota : Mascota) : Integer

end

association Amigos between

Mascota [0..\*] role mascota1

Mascota [0..\*] role mascota2

invariants

compatibilidadSuficiente: self.calcularCompatibilidad(mascota1, mascota2) >= 70

end

context Amigos

inv compatibilidadSuficiente:

self->forAll(a | a.mascota1.calcularCompatibilidad(a.mascota2) >= 70)

context **Investigador**::**crearProyecto**(nombre: String, descripcion: String, area:String, presupuesto:Real, confidencialidad:Integer)

pre nombreUnico: Proyecto.allInstances->exists(p: Proyecto| p.nombre = nombre)= false

pre descripcionGrande: descripcion.size > 50

pre areaInvestigador: area = self.area

pre presupuestoPositivo: presupuesto >= 0

pre confidencialidadValida: confidencialidad <= self.nivelAutorizacion

post post1: Proyecto.allInstances->select(p: Proyecto| p.nombre = nombre@pre)->size() = 1

context **Veterinario**::certificarProducto(producto: Producto)

pre Producto.idProducto: Producto.allInstances->exists(p: Producto| p.idProducto = producto.idProducto)= false

post post1: Producto

podría venir bien para definir el Estado del perfil

### **¿Qué es oclInState en OCL?**

oclInState es una función en OCL que se usa para comprobar si un objeto se encuentra en un estado específico dentro de una **máquina de estados** (State Machine).

Se usa en contextos donde una clase tiene estados definidos en un diagrama de estados UML y quieres verificar si una instancia está en un estado determinado.

### **Sintaxis de oclInState**

self.oclInState(NombreDelEstado)

* self → Se refiere a la instancia actual del objeto.
* NombreDelEstado → Es el nombre de un estado definido en una máquina de estados UML asociada a la clase.

### **Ejemplo 1: Mascota con Estados**

Supongamos que la clase Mascota tiene una máquina de estados con los siguientes estados:

* **Activo**
* **Suspendido**
* **EnAdopcion**

Si queremos verificar que una mascota está en estado **Suspendido**, podemos usar:

context Mascota

inv perfilSuspendido: self.oclInState(Suspendido) implies self.estadoPerfil = 'Suspendido'

Esto asegura que si la mascota está en el estado **Suspendido**, entonces su atributo estadoPerfil también refleja ese estado.

### **Ejemplo 2: Evento con Estados**

Si la clase Evento tiene los estados:

* **Pendiente**
* **Confirmado**
* **Cancelado**

Podemos definir una restricción para evitar que un evento cancelado tenga participantes:

context Evento

inv noParticipantesSiCancelado:

self.oclInState(Cancelado) implies self.nParticipantes = 0

Esto impide que eventos cancelados tengan participantes registrados.

### **¿Cuándo usar oclInState?**

* Cuando tu modelo incluye **máquinas de estado** UML para clases específicas.
* Si quieres imponer restricciones basadas en el estado actual de un objeto.

Si no has definido una máquina de estados en UML para una clase, **oclInState no funcionará**. En ese caso, usa un atributo como estado en la clase para representar el estado y comprobarlo con self.estado = 'Suspendido'.

Para definir estados en la herramienta **USE (UML-based Specification Environment)** y utilizarlos con OCL, puedes emplear **Protocol State Machines (PSM)** dentro de las definiciones de clases. Estas máquinas de estados permiten especificar los diferentes estados que una instancia de una clase puede tener y las transiciones entre ellos.

### **Definiendo Estados en USE**

En USE, los estados se definen dentro de la sección state machines en la definición de una clase. Aquí te muestro cómo hacerlo para la clase Mascota con los estados Activo, Suspendido y EnAdopcion:

class Mascota

attributes

nombre : String

estadoPerfil : String

-- otros atributos

state machines

psm EstadoMascota states

Activo,

Suspendido,

EnAdopcion

end

end

**Explicación:**

* psm EstadoMascota states: Define una Protocol State Machine llamada EstadoMascota.
* Activo, Suspendido, EnAdopcion: Lista de estados posibles para la clase Mascota.

### **Utilizando oclInState con Estados Definidos**

Una vez definidos los estados, puedes usar oclInState para verificar el estado actual de una instancia de Mascota. Por ejemplo, para asegurar que si una mascota está en estado Suspendido, su atributo estadoPerfil también refleje ese estado, puedes escribir:

context Mascota

inv ConsistenciaEstado: self.oclInState(Suspendido) implies self.estadoPerfil = 'Suspendido'

**Nota Importante:** Para que oclInState funcione correctamente, los estados deben estar definidos explícitamente en la máquina de estados de la clase, como se mostró anteriormente.

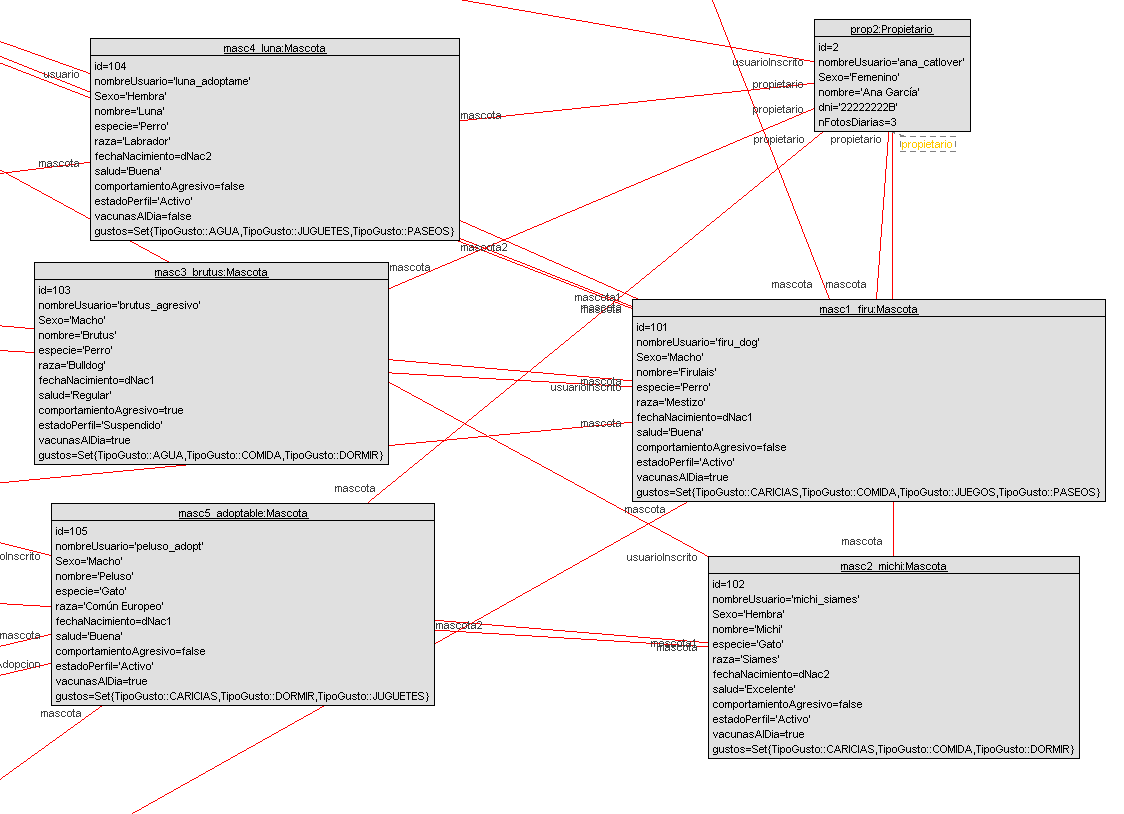
### **Recursos Adicionales**

Para más detalles sobre cómo definir y trabajar con máquinas de estados en USE, puedes consultar el manual oficial de USE disponible en su [repositorio de GitHub](https://github.com/useocl/use/blob/master/manual/main.md). Además, la documentación sobre [Protocol State Machines](https://perso.liris.cnrs.fr/pierre-edouard.portier/teaching_2014_2015/sq/ocl/GOGOLLA_2015_protocol_state_machines_documentation.pdf) proporciona ejemplos y explicaciones adicionales.

Al integrar estas definiciones en tu modelo USE, podrás aprovechar las capacidades de OCL para validar y verificar comportamientos específicos basados en los estados de tus objetos.

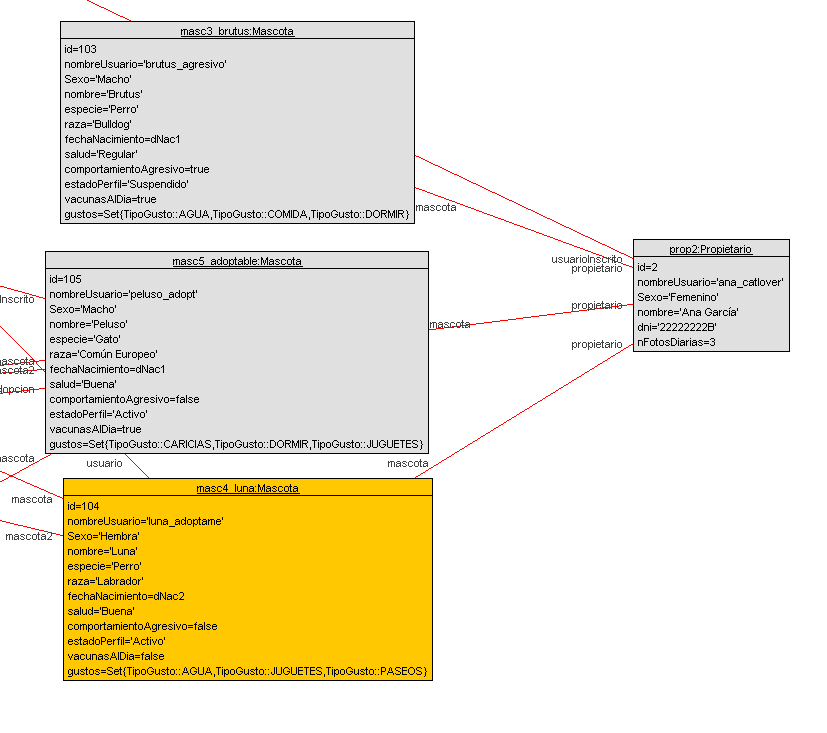
R1 FALSE El prop2 tiene 5 mascotas a su disposicion



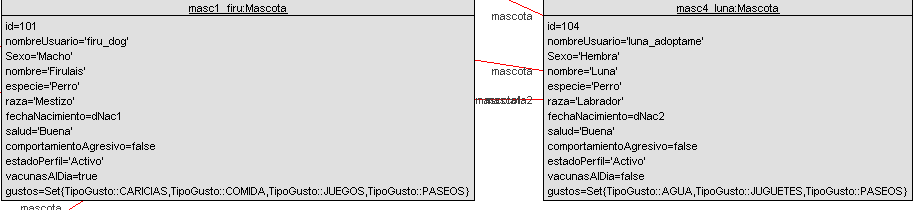


R1 TRUE:

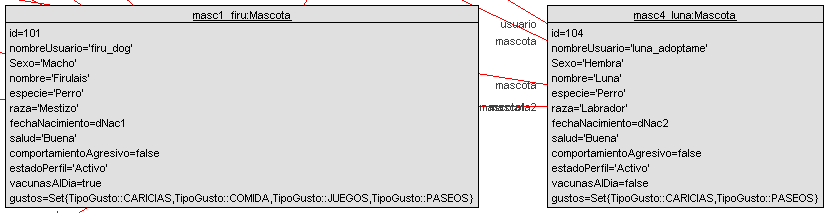




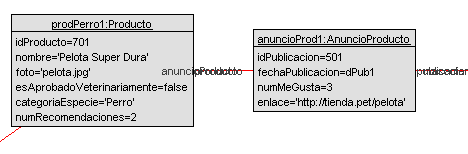
R2 FALSE



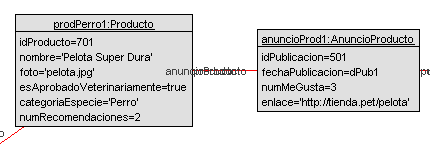
R2 TRUE



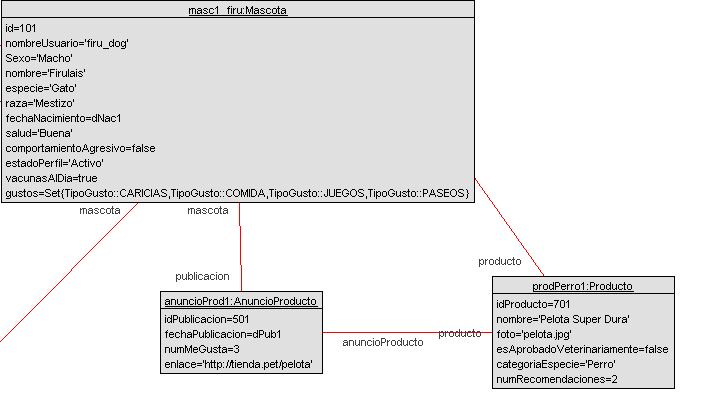
R3 FALSE



R3 TRUE

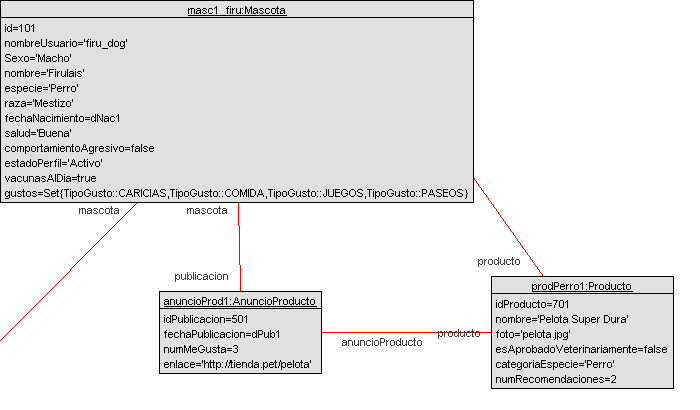


R4 FALSE



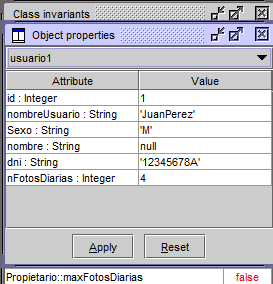


R4 TRUE

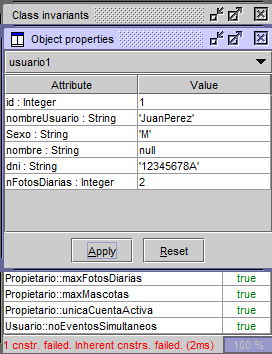




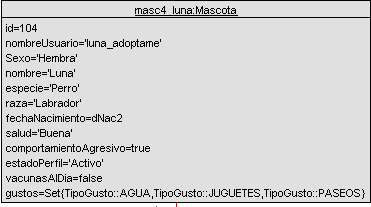
R5 FALSE



R5 TRUE

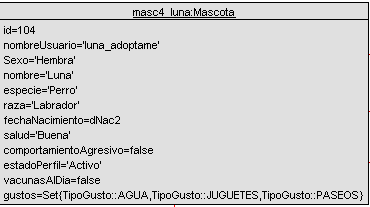


R6 FALSE



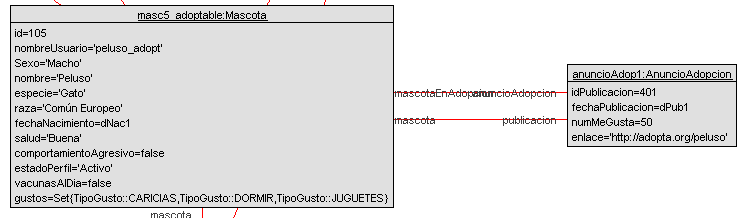


R6 TRUE

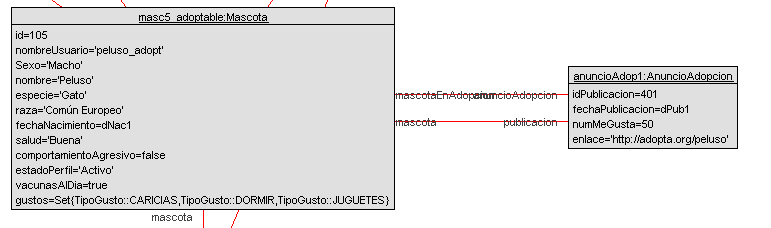




R7 FALSE

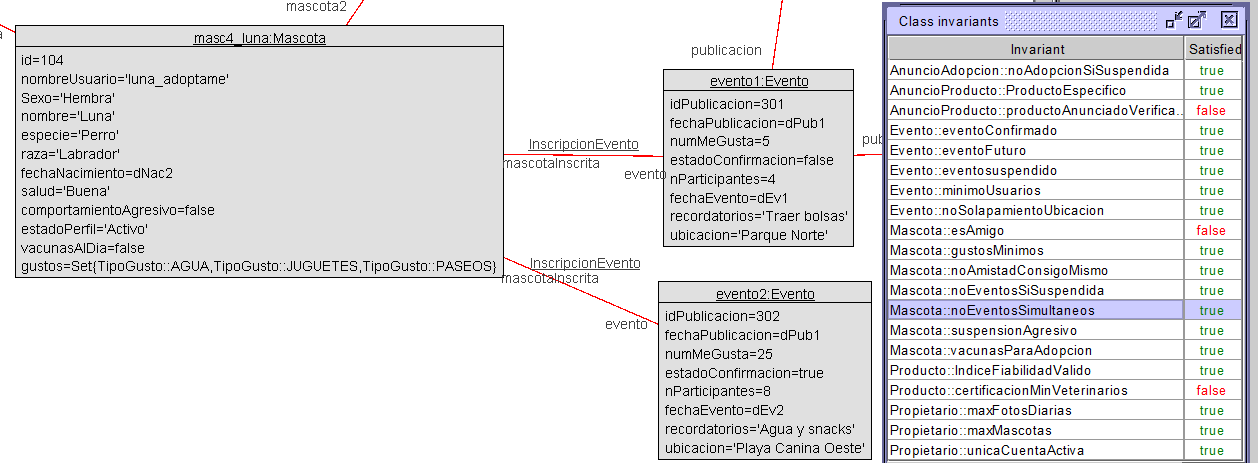


R7 TRUE

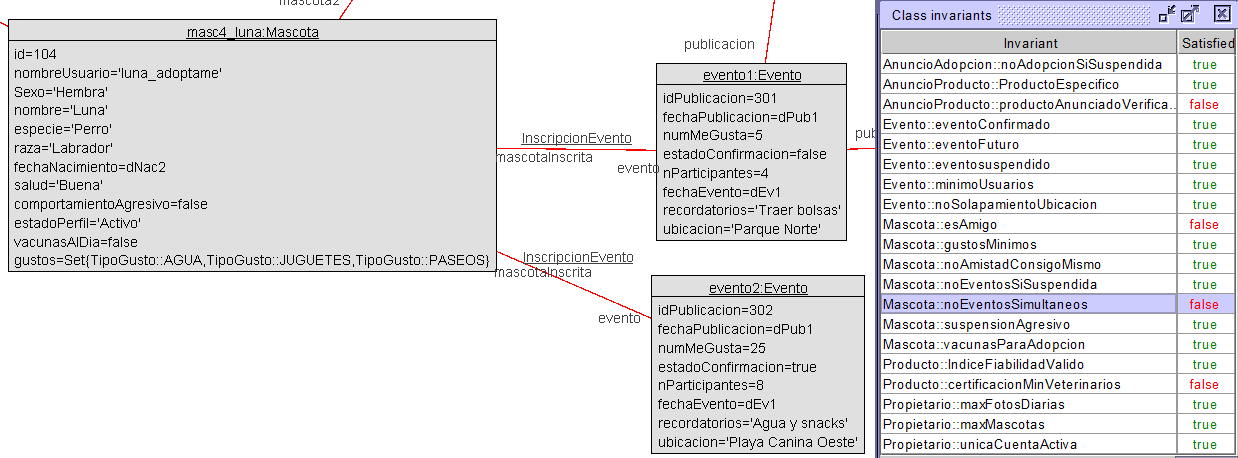


R8 FALSE

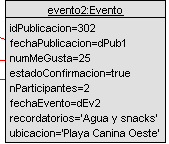
R8 TRUE



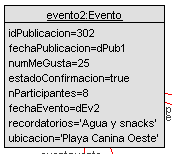
R8 FALSE



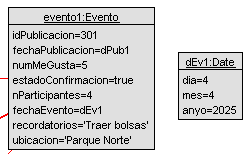
R9 FALSE



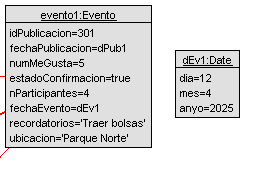
R9 TRUE



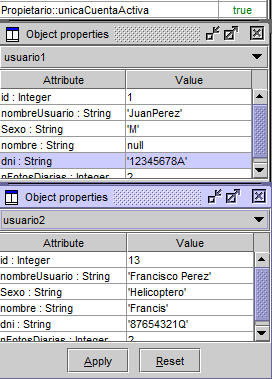
R10 FALSE



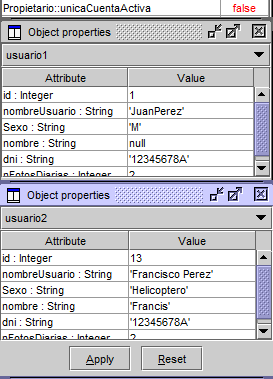
R10 TRUE



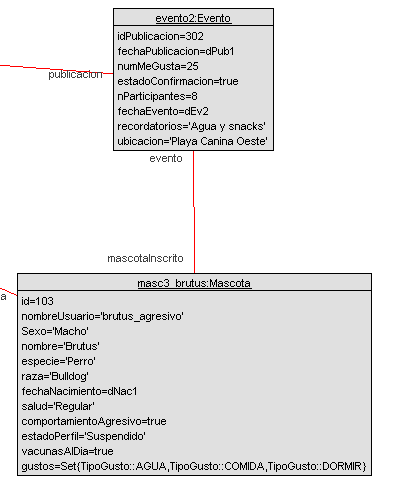
R11 TRUE



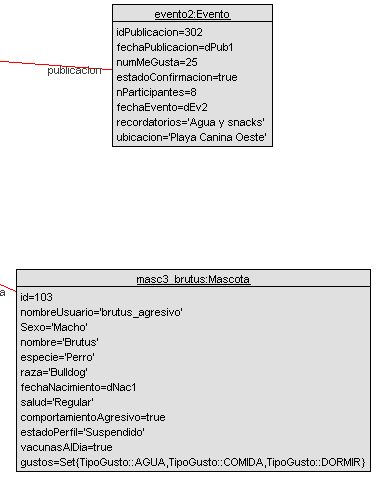
R11 FALSE



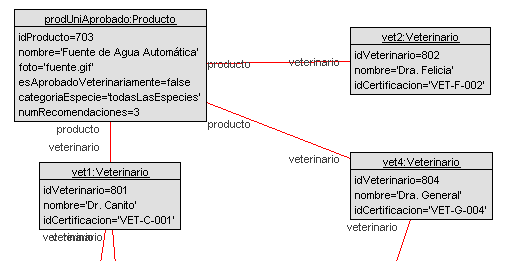
R12 FALSE



R12 TRUE



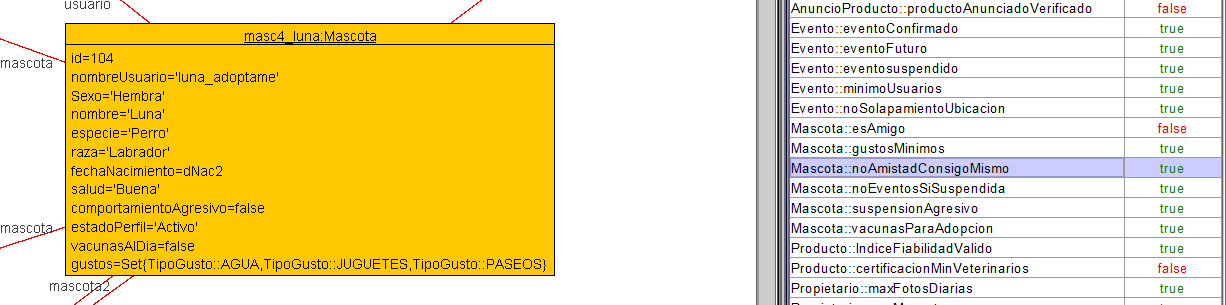
R13 FALSE



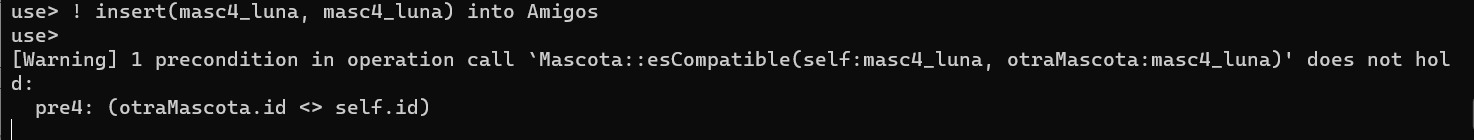


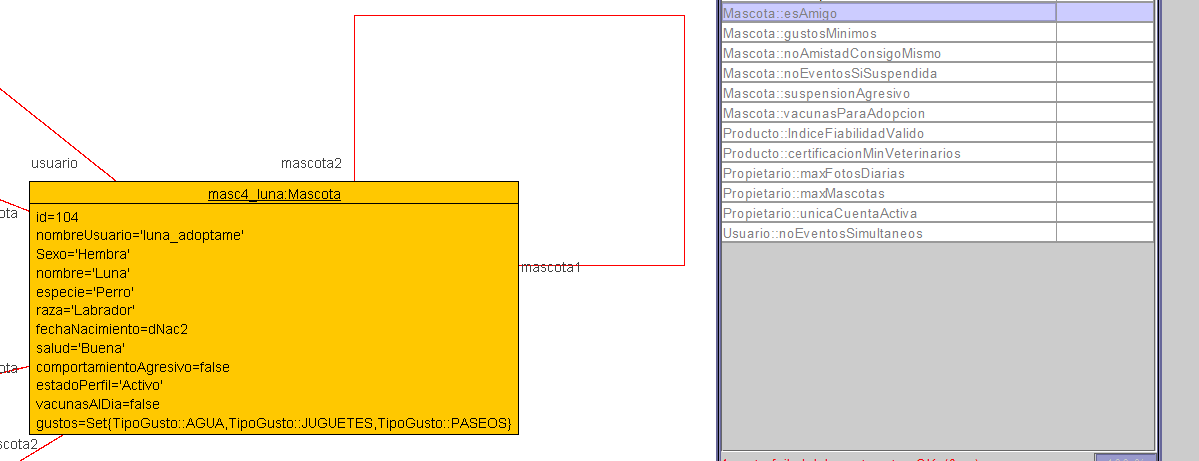
R13 TRUE

R14 TRUE

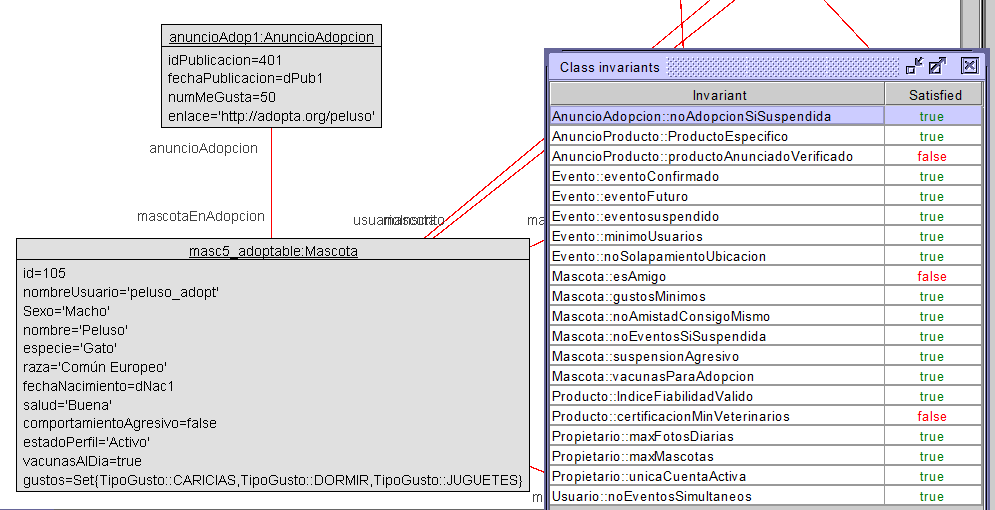


R14 FALSE

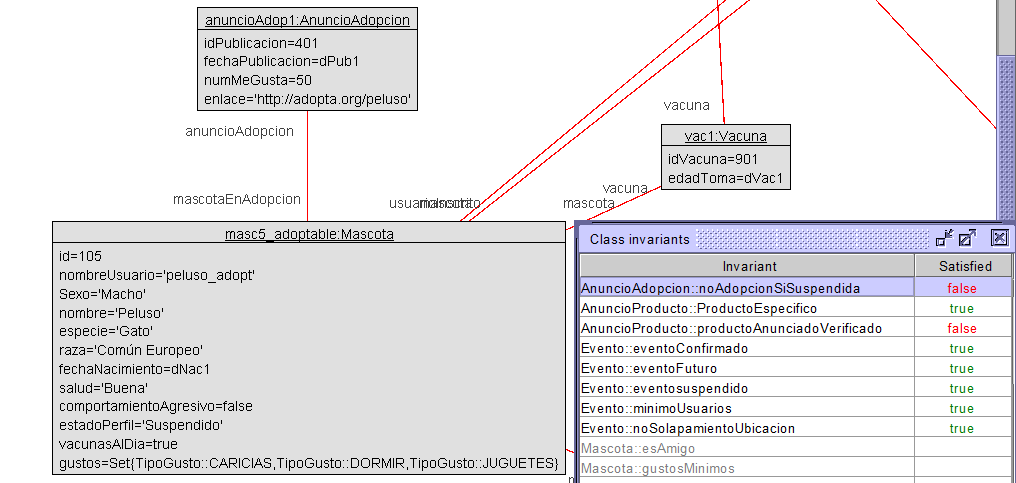




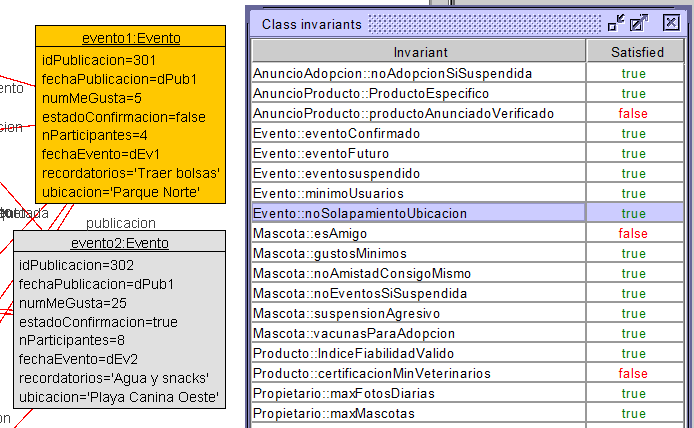
R15: TRUE



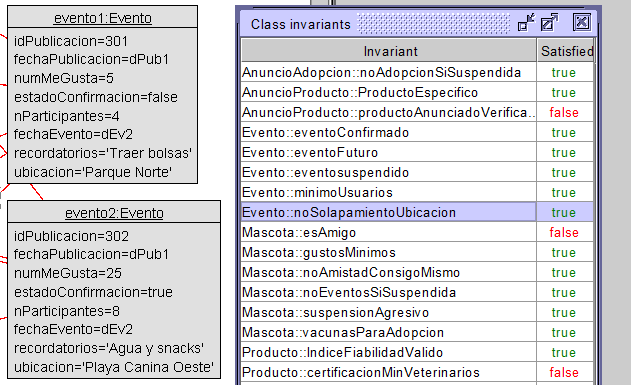
R15: FALSE



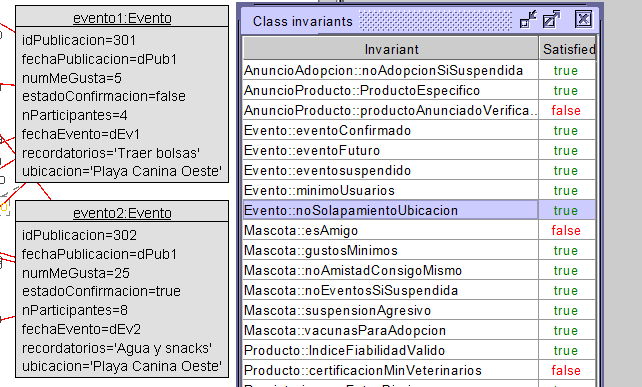
R16: TRUE fecha distintal | ubicación distinta



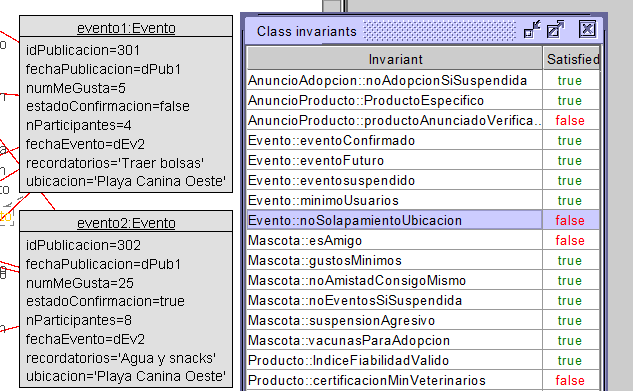
R16: TRUE fecha igual | ubicación distinta



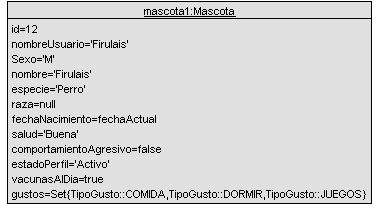
R16: TRUE fecha distinta | ubicación igual



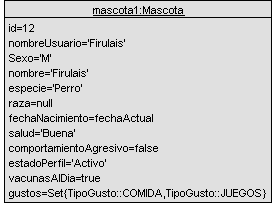
R16: FALSE



R17: TRUE

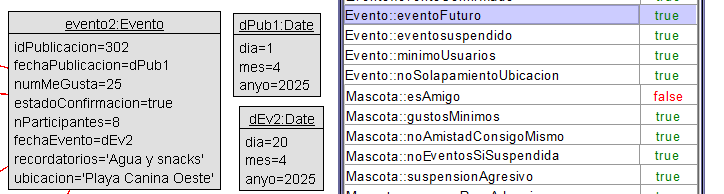


R17: FALSE

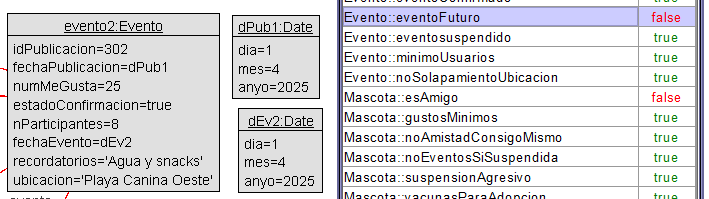




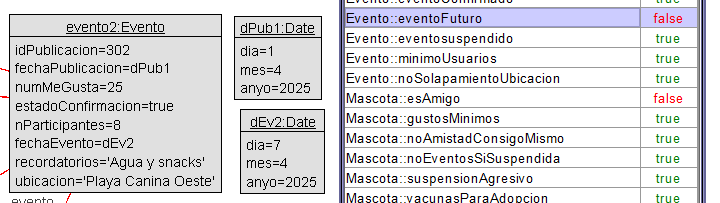
R18: TRUE



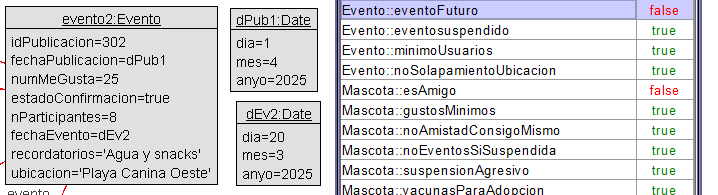
R18: FALSE mismo dia



R18: FALSE <7 dias



R18: FALSE fecha publicacion posterior a fecha evento

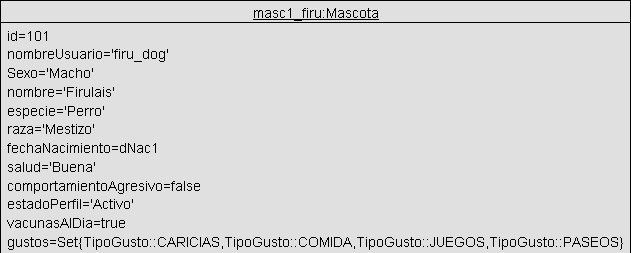
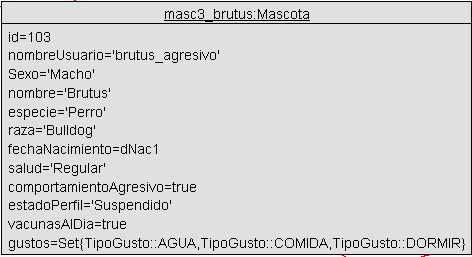
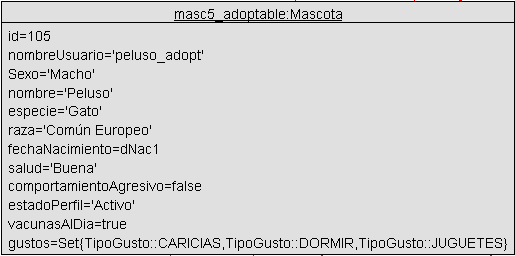


# Contratos

En este mñetodo al tener asignada una expresión ocl, esta operación no puede ser evaluada con !openter para precondiciones y !opexit para las postcondiciones. En su lugar usamos ‘?’ llamando a la operación a través de una instancia concreta (al igual que openter) y vemos el resultado que devuelve y los warnings indicando las precondiciones que no se cumplen.

## esCompatible(Mascota otraMascota)

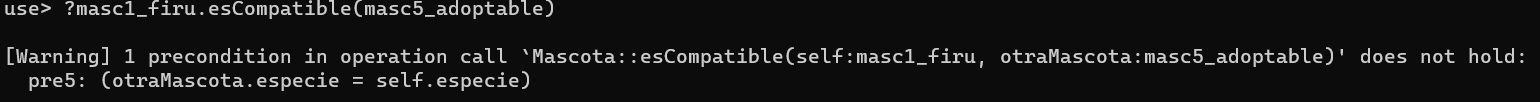
(instancias para que vea los valores que tienen)



* pre: otraMascota.id <> self.id



* pre: otraMascota.especie = self.especie

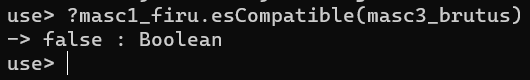


* pre: otraMascota.estadoPerfil <> 'Suspendido' and self.estadoPerfil <> 'Suspendido'



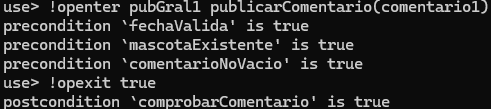
* Todo bien: (quitando suspensión de brutus por comportamiento)

No salta ningún warning indicando que las precondiciones se cumplen. El false únicamente indica que no son compatibles

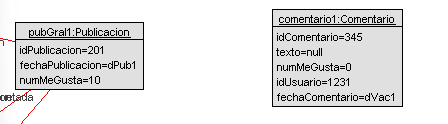


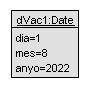
## publicarComentario(comentario:Comentario)

TODO BIEN:



Falla precondicion y postCondicion

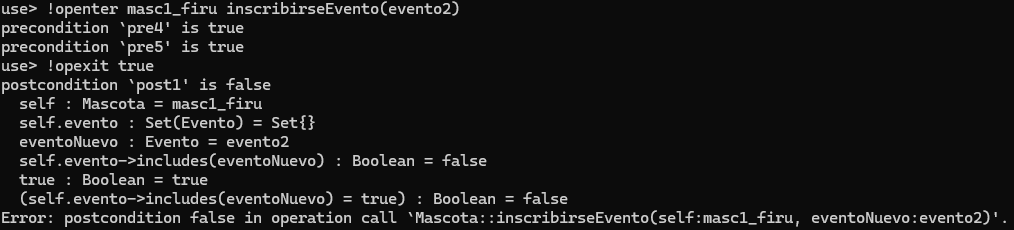


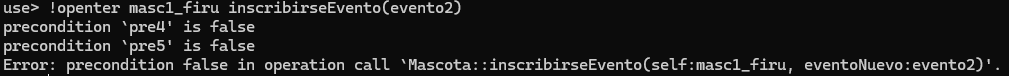
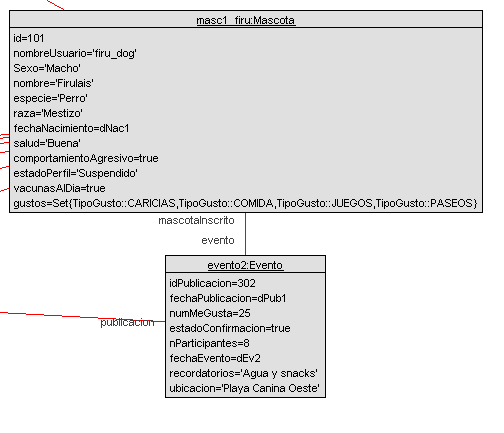




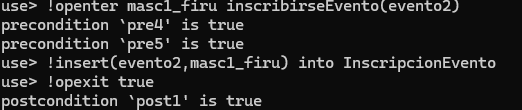
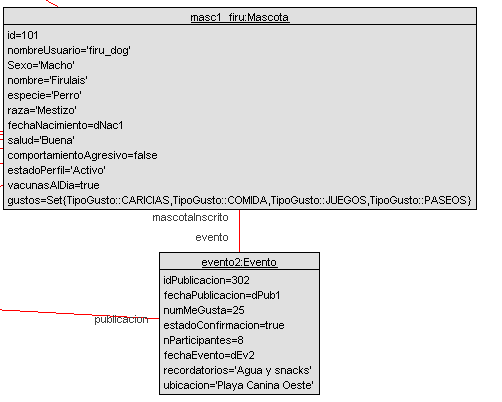
## INSCRIBIRSEVENTO

FALLA LA POST



FALLA LAS PRE 

FUNCIONA COMPLETO

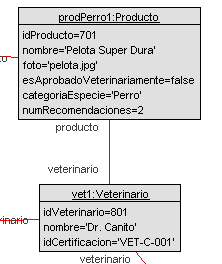


## CertificarProducto

pre a true

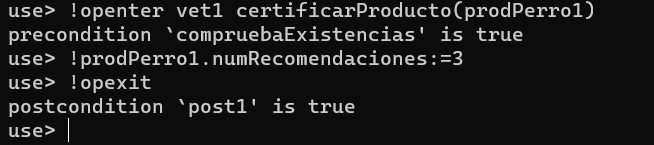
pre compruebaExistencias: self.producto->exists(p : Producto | p.idProducto = producto.idProducto) = true





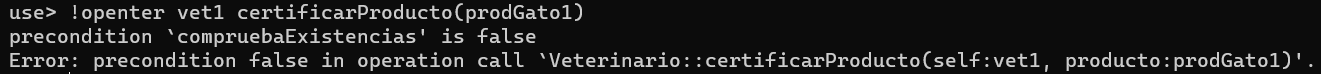
post true

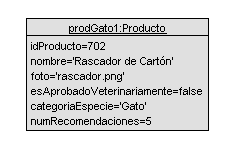
post post1: producto.numRecomendaciones = producto.numRecomendaciones@pre + 1



FALSE

No tiene ninguna relacion con ningun veterinario por tanto no puede ser certificado





post:

