



PETLINK-ALLOY

21/05/2025



21 DE MAYO DE 2025

MÉTODOS FORMALES EN INGENIERÍA DEL SOFTWARE

ADRIÁN MACHUCA MORILLA - JAVIER ARIAS FUENTES- ÁLVARO ORTA LÓPEZ

Índice

1	Introducción.....	5
2	Diagrama de Clases.....	7
2.1	Diagrama de Clases OCL.....	7
2.1.1	Especificación clases.....	8
2.2	Diagrama de Clases ALLOY.....	10
2.2.1	Nuevas especificaciones.....	10
3	Modelado en ALLOY.....	11
3.1	Necesidades del sistema.....	11
3.2	Signaturas.....	11
3.2.1	Usuario.....	11
3.2.2	Propietario.....	12
3.2.3	Mascota.....	12
3.2.4	Vacuna.....	13
3.2.5	Fecha.....	13
3.2.6	Fecha actual.....	13
3.2.7	Evento.....	14
3.2.8	Producto.....	14
3.2.9	Veterinario.....	15
3.2.10	Publicación.....	15
3.2.11	Publicación normal.....	15
3.2.12	Anuncio de adopción.....	16
3.2.13	Anuncio de producto.....	16
3.3	Facts.....	17
3.3.1	Fact “DueñosSimétricos”.....	17
3.3.2	Fact “AmistadBidireccional”.....	17
3.3.3	Fact “publicacionesBidireccionales”.....	18
3.3.4	Fact “UnAnuncioAdopcionPorMascota”.....	18
3.3.5	Fact “valorFechaActual”.....	18
3.3.6	Fact “fechaNoNegativa”.....	19
3.3.7	Fact “fechaPublicacionMenorQueFechaActual”.....	19
3.3.8	Fact “unAnuncioPorProducto”.....	19
3.3.9	Fact “CantidadMaximaDeMascotasPorPropietario”.....	20
3.3.10	Fact “AmistadBasadaEnGustos”.....	20
3.3.11	Fact “ProductoVerificadoPorVeterinarios”.....	21
3.3.12	Fact “ProductoRecomendadoPorMascota”.....	22

3.3.13	Fact “MascotaMaxTresPublicacionesPorFecha”	22
3.3.14	Fact “AnuncioAdopcionConVacunasValidas”	23
3.3.15	Fact “MascotaNoPuedeApuntarseDosVeces”	23
3.3.16	Fact “eventoSuspendido”	23
3.3.17	Fact “NombreUnico”	24
3.3.18	Fact “EventoNoAgresivo”	24
3.3.19	Fact “MascotaNoAmigos”	24
3.3.20	Fact “AnuncioAgresivo”	24
3.3.21	Fact “EventosNoPuedenTenerMismaFechaYUbicacion”	25
3.3.22	Fact “CantidadMinimaDeGustosPorMascota”	25
3.3.23	Fact “eventoConSemanaDeAnticipacion”	25
3.4	Predicados.....	26
3.4.1	Producto verificado por veterinario.....	26
3.4.2	Mascota en adopción sin vacunas.....	26
3.4.3	Un propietario con 5 mascotas	26
3.4.4	Cada mascota máximo 3 publicaciones por día	27
3.5	Asertos	28
3.5.1	Máximo de 3 publicaciones diarias.....	28
3.5.2	Anuncios destinados a distintas especies	28
4	Conclusión.....	29

1 Introducción

En esta práctica partimos del trabajo realizado en la primera entrega, donde propusimos un modelo conceptual para una red social pensada para propietarios y mascotas. El objetivo ahora es trasladar ese modelo al lenguaje Alloy, definiendo de forma formal las relaciones, restricciones y comportamientos que debe cumplir nuestro sistema.

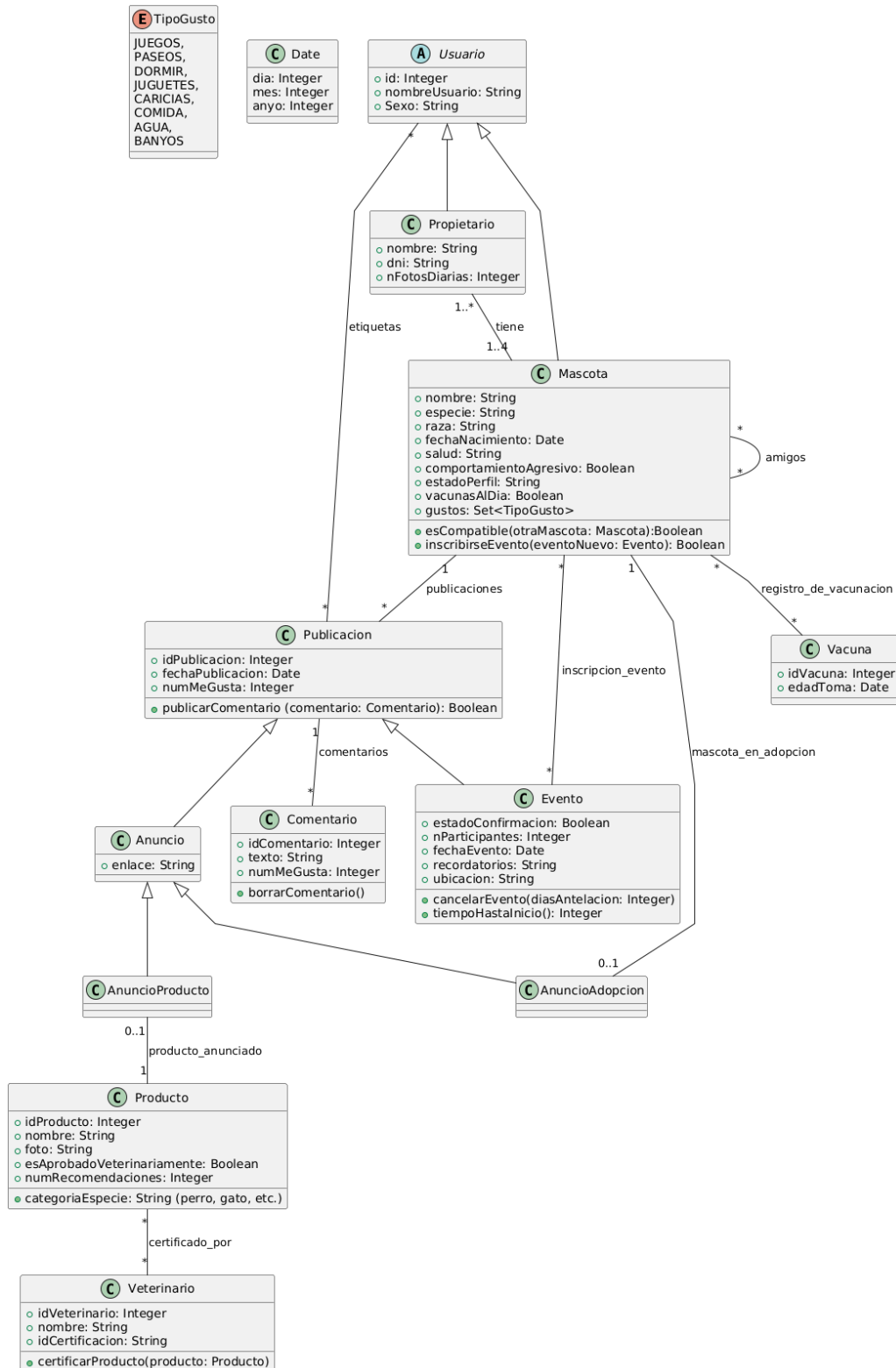
Para ello, especificaremos restricciones que garanticen que el modelo sea fiable, consistente y coherente, además de incluir funciones y predicados que recojan algunas de las operaciones o comprobaciones importantes que podría necesitar la aplicación. También añadiremos asertos que nos servirán para verificar que ciertos comportamientos clave se cumplen correctamente en todos los escenarios posibles que Alloy puede generar.

Con este ejercicio buscamos afianzar los conceptos de modelado formal vistos en clase y comprobar que nuestro modelo no solo cumple con el enunciado inicial, sino que además funciona correctamente bajo cualquier situación que se pueda dar en el sistema.

2 Diagrama de Clases

Vamos a empezar detallando de nuevo el diagrama de clases que modelamos en la práctica anterior para mostrar el punto de partida, aunque ya veremos que al intentar modelar todo esto en Alloy deberemos hacer algunos cambios.

2.1 Diagrama de Clases OCL



2.1.1 Especificación clases

Usuario

Representa a los individuos que utilizan la plataforma. Contiene atributos básicos como un identificador y un nombre de usuario. Esta clase es la base para diferentes tipos de usuarios dentro del sistema.

Propietario

Extiende la clase Usuario y representa a los dueños de mascotas. Además de los atributos heredados, incluye datos como nombre, DNI y un contador de fotos publicadas diariamente. Un propietario puede tener hasta 4 mascotas asociadas.

Mascota

Representa a los animales registrados por los propietarios. Cada mascota tiene atributos como nombre, especie, raza, fecha de nacimiento, estado de salud y comportamiento. También puede estar marcada como "perfil suspendido" en caso de comportamiento agresivo, que sería otro atributo de esta clase. Además, la clase almacena información sobre si las vacunas están al día y permite la interacción con otras mascotas mediante una relación de amistad.

Vacuna

Almacena información sobre las vacunas registradas en la plataforma. Incluye un identificador y la edad recomendada para su aplicación. Está asociada a la clase Mascota, permitiendo verificar si una mascota tiene sus vacunas actualizadas, lo que es un requisito obligatorio para publicar anuncios de adopción.

Producto

Representa los productos disponibles en la plataforma para ser recomendados a las mascotas. Contiene atributos como identificador, nombre, imagen, categoría según la especie de la mascota, certificación veterinaria y un porcentaje de fiabilidad en la recomendación.

Veterinario

Esta clase representa a los profesionales veterinarios registrados en el sistema. Incluye un identificador, nombre y un número de certificación. Los veterinarios tienen la capacidad de aprobar productos dentro de la plataforma, asegurando que las recomendaciones sean seguras y adecuadas para las mascotas.

Publicación

Representa cualquier tipo de contenido publicado en la plataforma. Contiene información sobre la fecha de publicación, el número de "me gusta" y los comentarios asociados. Esta clase se especializa en distintos tipos de publicaciones, como eventos y anuncios.

Comentario

Los comentarios son elementos clave para fomentar la interacción entre los usuarios. Cada comentario tiene un identificador, un texto y un contador de "me gusta". Se pueden asociar a publicaciones y contar con operaciones para su publicación y eliminación.

Evento

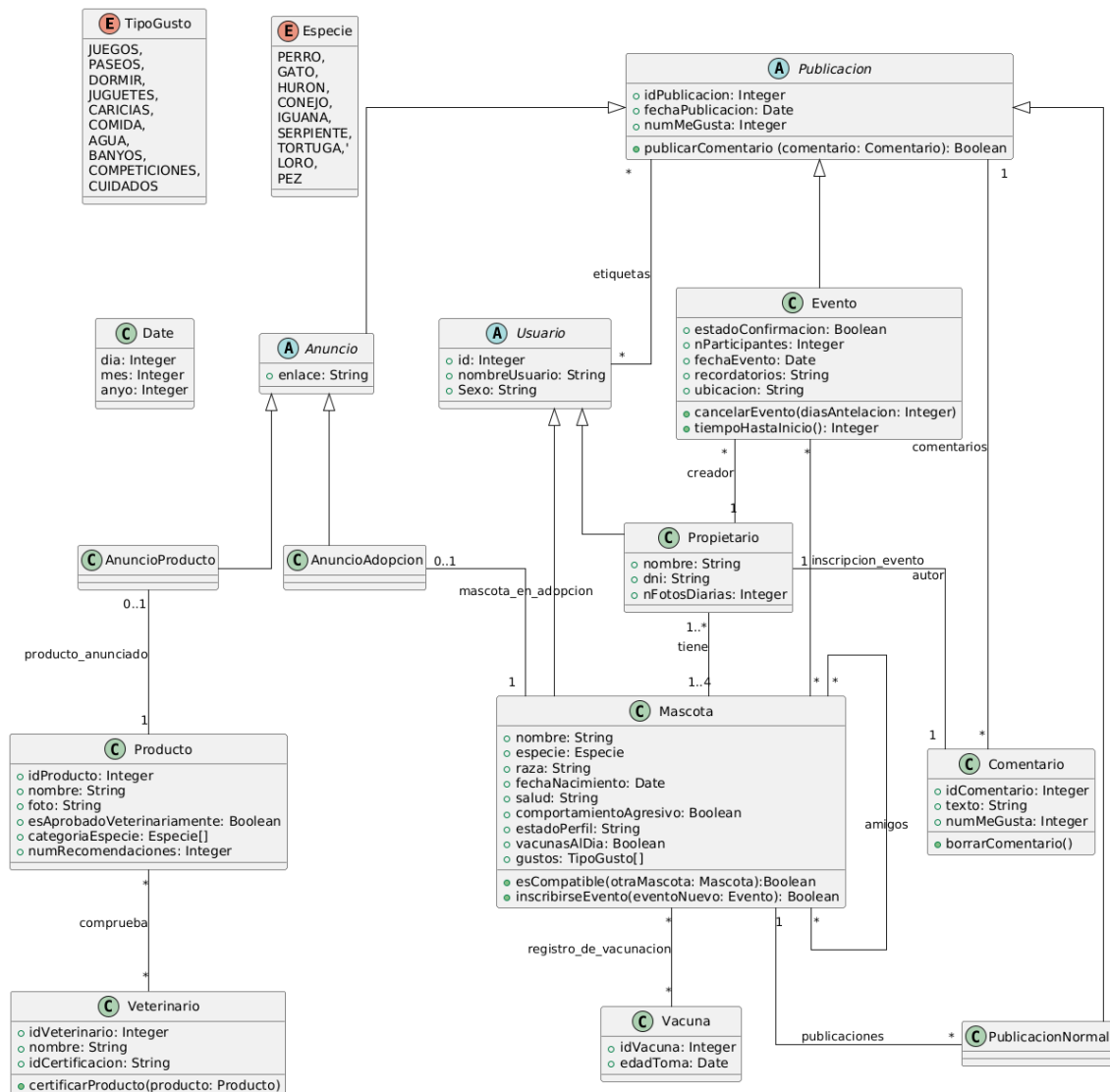
Extiende la clase Publicación y permite la organización de encuentros entre usuarios y mascotas. Un evento solo se puede confirmar si hay al menos seis usuarios interesados y debe cancelarse si faltan menos de siete días para su realización y no hay suficientes participantes. Además, cuenta con atributos como estado de confirmación, número de participantes y recordatorios.

Anuncio

También extiende la clase Publicación y representa los anuncios dentro de la plataforma. Se divide en dos tipos:

- **Anuncio de Adopción:** Relacionado con la adopción de mascotas. Solo se puede publicar si la mascota en cuestión tiene todas sus vacunas al día.
- **Anuncio de Producto:** Publicidad de productos específicos. Los productos anunciados deben ser compatibles con la especie de la mascota, asegurando que un perro no reciba anuncios de comida para gatos.

2.2 Diagrama de Clases ALLOY



2.2.1 Nuevas especificaciones

2.2.1.1 *PublicacionNormal*

Encontramos una inconsistencia al usar la clase publicación como clase padre de anuncios y eventos. Por tanto, la clase publicación pasa a ser publicación normal, que hereda de una clase abstracta que mantiene los atributos y relaciones comunes entre todos los tipos de publicación.

En el diagrama anterior, un anuncio producto tenía la relación con Mascota de “publicaciones”.

2.2.1.2 *Nueva relación entre Comentario y Propietario*


Esta relación indica al autor del comentario.

2.2.1.3 *Nueva relación entre Evento y Propietario*


Esta relación indica al creador del evento.

3 Modelado en ALLOY

3.1 Necesidades del sistema




```
1 enum Especie {
2   PERRO,
3   GATO,
4   HURON,
5   CONEJO,
6   IGUANA,
7   SERPIENTE,
8   TORTUGA,
9   LORO,
10  PEZ
11 }
```



```
1 enum TipoGusto {
2   JUEGOS,
3   PASEOS,
4   DORMIR,
5   JUGUETES,
6   CARICIAS,
7   COMIDA,
8   AGUA,
9   BANYOS,
10  COMPETICIONES,
11  CUIDADOS
12 }
```


3.2 Signaturas

3.2.1 Usuario



```
1 abstract sig Usuario {
2   nombre : one Nombre,
3 }
```

Representa la clase abstracta de la que heredan tanto los propietarios como las mascotas. Contiene únicamente un atributo: nombre, que corresponde a una instancia de la signatura Nombre.



```
1 sig Nombre{
2   // representa el nombre del usuario
3   // necesaria para fact NombreUnico
4 }
```

Representa los nombres de usuarios y mascotas. Se utiliza para mantener restricciones de unicidad.

3.2.2 Propietario



```
1 sig Propietario extends Usuario {  
2   dueño : some Mascota  
3 }
```

Modela a los usuarios propietarios de mascotas. Tiene una relación dueño con al menos una mascota (some Mascota), indicando que todo propietario debe tener al menos una.

3.2.3 Mascota



```
1 sig Mascota extends Usuario {  
2   mascota : some Propietario,  
3   registroVacunacion : set Vacuna,  
4   amigos : set Mascota,  
5   gustos : some TipoGusto,  
6   vacunasTodas : lone vacunasAlDia,  
7   comportamiento : lone ComportamientoAgresivo,  
8   especieMascota : one Especie,  
9   publicacionesPropias : set PublicacionNormal,  
10 }
```

Representa a las mascotas registradas en la red social. Relaciona:

- mascota: los propietarios de la mascota.
- registroVacunacion: vacunas registradas para la mascota.
- amigos: otras mascotas amigas.
- gustos: los gustos de la mascota.
- vacunasTodas: indica si la mascota tiene todas las vacunas al día. Si existe la relación con esta signatura, entonces están al día; de lo contrario, no las tendría.
- comportamiento: indica si tiene comportamiento agresivo. Mismo funcionamiento que la signatura anterior: si existe la relación, sus comportamiento es agresivo.
- especieMascota: la especie de la mascota.
- publicacionesPropias: publicaciones normales realizadas por la mascota.



```
1 sig vacunasAlDia{  
2   // representa el atributo booleano "vacunasAlDia" de la mascota  
3 }
```

Atributo booleano representado por su relación con la mascota:

- Si existe relación → todas las vacunas al día.
- Si no hay relación → le falta alguna vacuna.



```
1 one sig ComportamientoAgresivo{
2   // Representa el comportamiento agresivo de una mascota
3   // necesaria para restricciones de comportamiento en mascotas
4 }
```

Indica si una mascota tiene comportamiento agresivo. Solo puede haber una instancia activa en el sistema para modelar este comportamiento.

3.2.4 Vacuna



```
1 sig Vacuna {
2   // representa objeto vacuna
3 }
```

Modela las vacunas que pueden recibir las mascotas.

3.2.5 Fecha



```
1 sig Fecha{
2   // Representamos las fechas como un número entero
3   ValorDias:one Int
4 }
```

Representa una fecha como un número entero (ValorDias), facilitando así las comparaciones y operaciones con fechas.

3.2.6 Fecha actual



```
1 one sig fechaActual {
2   // Representamos la fecha actual para poder
3   // realizar comparaciones con otras fechas
4   ValorDias: one Int
5 }
```

Contiene la fecha actual del sistema para permitir comprobar fechas de publicaciones, eventos o adopciones.

3.2.7 Evento



```
1 sig Evento extends Publicacion {
2   participantes : set Mascota,
3   creador : one Propietario,
4   ubicacion : one Ubicacion,
5   fechaEvento : one Fecha,
6   estadoEvento : lone EventoConfirmado,
7 }
```

Publicación que representa un evento creado en la red social, incluyendo:

- participantes: mascotas inscritas.
- creador: propietario que organiza el evento.
- ubicacion: ubicación del evento.
- fechaEvento: día del evento.
- estadoEvento: confirmación del evento.



```
1 one sig EventoConfirmado{
2   // Representa el estado de confirmación de un evento
3 }
```

Modelo para indicar que un evento ha sido confirmado. Solo puede existir una instancia.



```
1 sig Ubicacion{
2   // Representa la ubicación de un evento
3   // necesaria para la restricción de eventos
4 }
```

Representa la ubicación física de un evento.

3.2.8 Producto



```
1 sig Producto {
2   Especiedestinada : some Especie,
3 }
```

Representa los productos que se pueden anunciar. Se asocia a una o varias especies (some Especie).

3.2.9 Veterinario



```
1 sig Veterinario {  
2   comprueba : set Producto  
3 }
```

Modela a los veterinarios que pueden certificar productos, manteniendo una relación comprueba con los productos que han evaluado.

3.2.10 Publicación



```
1 abstract sig Publicacion {  
2   etiquetado : set Usuario,  
3   fechaPublicada : one Fecha,  
4 }
```

Clase abstracta que representa una publicación en la red social, con:

- etiquetado: usuarios etiquetados en la publicación.
- fechaPublicada: fecha de publicación.



```
1 sig Comentario {  
2   comentario : one Publicacion,  
3   autor : one Propietario  
4 }
```

Representa un comentario de un propietario sobre una publicación, asociando:

- comentario: la publicación comentada.
- autor: el propietario que comenta.

3.2.11 Publicación normal



```
1 sig PublicacionNormal extends Publicacion {  
2   publicador : one Mascota,  
3 }
```

Publicación convencional realizada por una mascota (publicador).

3.2.12 Anuncio de adopción



```
1 sig AnuncioAdopcion extends Publicacion {  
2   mascotaAdopcion : one Mascota  
3 }
```

Publicación que ofrece una mascota en adopción, asociándose a una mascota.

3.2.13 Anuncio de producto



```
1 sig AnuncioProducto extends Publicacion {  
2   productoAnunciado : one Producto,  
3   mascotasRecomendadas : some Mascota,  
4 }
```

Publicación que anuncia un producto, indicando:

- productoAnunciado: el producto ofrecido.
- mascotasRecomendadas: mascotas a las que se recomienda el producto.

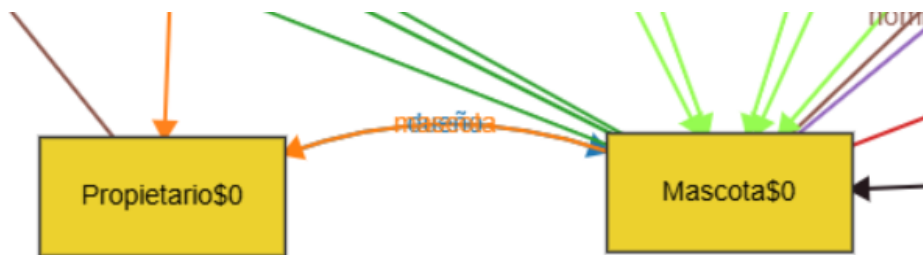
3.3 Facts

3.3.1 Fact “DueñosSimétricos”



```
1 fact DueñosSimétricos {  
2   all m: Mascota | all p: m.mascota | m in p.dueño  
3 }
```

Esta restricción asegura la coherencia en la relación de propiedad entre propietarios y mascotas. Se establece que, si una mascota pertenece a un propietario, dicha relación debe reflejarse también desde el punto de vista del propietario, de modo que este la tenga incluida entre sus mascotas. Esto garantiza la bidireccionalidad y evita inconsistencias en la representación de esta relación. Este fact se utiliza para definir correctamente la cardinalidad.

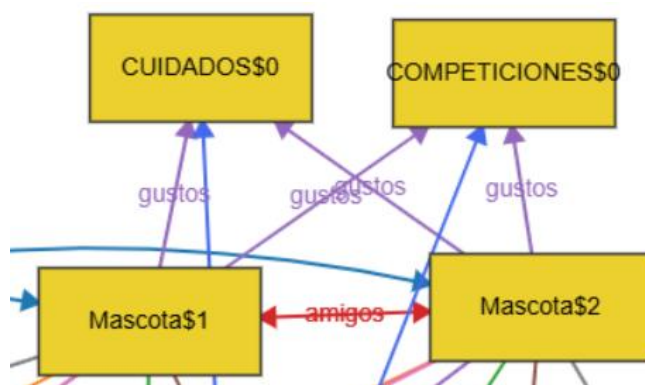


3.3.2 Fact “AmistadBidireccional”



```
1 fact AmistadBidireccional {  
2   // Para que dos mascotas sean amigas, deben tener una relación de amistad mutua  
3   all m1, m2: Mascota | m1 in m2.amigos implies m2 in m1.amigos  
4 }
```

Define la reciprocidad obligatoria en las relaciones de amistad entre mascotas. Si una mascota considera amiga a otra, esta debe a su vez considerarla amiga. Este hecho preserva la lógica social del modelo, donde las amistades no pueden ser unilaterales. Este fact se utiliza para definir correctamente la cardinalidad.

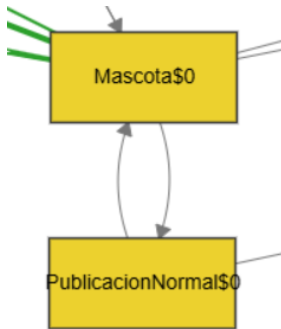


3.3.3 Fact “publicacionesBidireccionales”



```
1 fact publicacionesBidireccionales{
2   all p: PublicacionNormal | p in p.publicador.publicacionesPropias
3 }
```

Esta restricción establece que toda publicación debe pertenecer al publicador que la ha generado. De este modo, se garantiza que ninguna publicación quede huérfana o se asocie incorrectamente a otro publicador distinto al original, manteniendo la integridad de la autoría de las publicaciones. Este fact se utiliza para definir correctamente la cardinalidad.



3.3.4 Fact “UnAnuncioAdopcionPorMascota”



```
1 fact UnAnuncioAdopcionPorMascota {
2   all m: Mascota | lone a: AnuncioAdopcion | a.mascotaAdopcion = m
3 }
```

Limita a una sola la cantidad de anuncios de adopción posibles para cada mascota. Así, una mascota no puede estar anunciada en adopción en más de un anuncio al mismo tiempo, evitando situaciones ambiguas o conflictivas en los procesos de adopción. Este fact se utiliza para definir correctamente la cardinalidad.

3.3.5 Fact “valorFechaActual”



```
1 fact valorFechaActual{
2   fechaActual.ValorDias > 4
3 }
```

Con este hecho asignamos un valor a la instancia única de fechaActual para dar juego a generar fechas anteriores y posteriores a esta fecha. Ya que, como vamos a ver en el siguiente hecho, todas las fechas deben ser mayores que 0.

3.3.6 Fact “fechaNoNegativa”



```
1 fact fechaNoNegativa{
2   all f : Fecha | f.ValorDias > 0
3 }
```

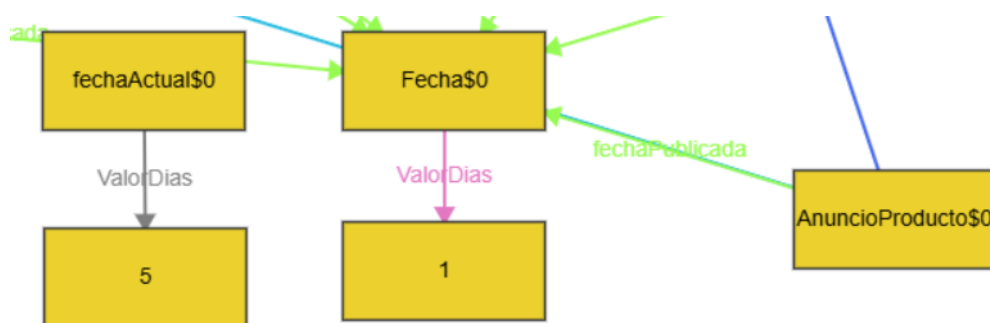
Con este hecho, forzamos a que todas las fechas deban ser positivas



3.3.7 Fact “fechaPublicacionMenorQueFechaActual”



```
1 fact fechaPublicacionMenorQueFechaActual{
2   all p: Publicacion | p.fechaPublicada.ValorDias ≤ fechaActual.ValorDias
3 }
```

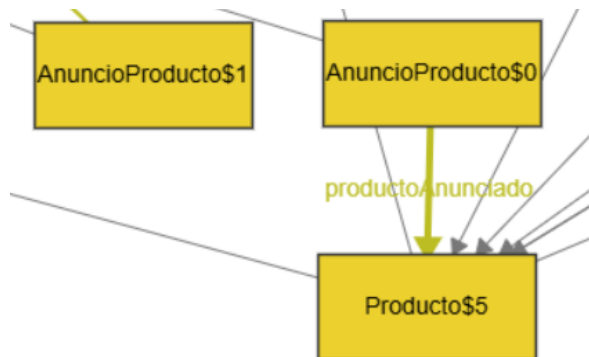


3.3.8 Fact “unAnuncioPorProducto”



```
1 fact unAnuncioPorProducto{
2   all p: Producto | lone aP: AnuncioProducto | p in aP.productoAnunciado
3 }
```

Restringe la cantidad de anuncios asociados a cada producto, permitiendo que un producto solo esté incluido en un único anuncio de producto. Esto garantiza una trazabilidad clara y única en la relación entre productos y anuncios.



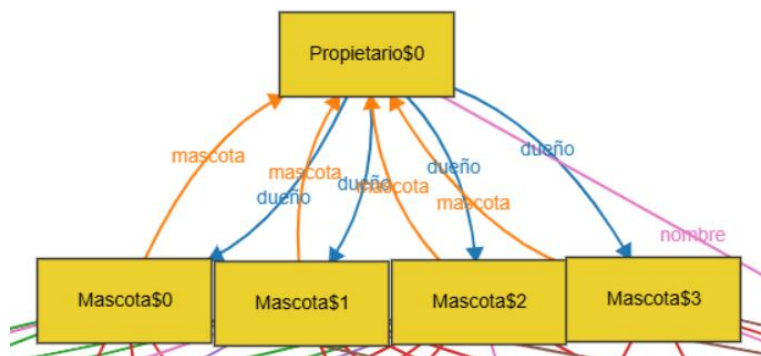
3.3.9 Fact “CantidadMaximaDeMascotasPorPropietario”



```
1 fact CantidadMaximaDeMascotasPorPropietario {
2   all p: Propietario | #p.dueño ≤ 4
3 }
```

Limita a cuatro el número máximo de mascotas que puede tener cada propietario en el sistema. De este modo se evita una acumulación excesiva de mascotas por cada usuario y tenemos algo más de control sobre ellas.

Restricción 1 OCL



3.3.10 Fact “AmistadBasadaEnGustos”



```
1 fact AmistadBasadaEnGustos {
2   all m1, m2: Mascota | m2 in m1.amigos implies gustosCoincidentes[m1, m2] ≥ 2
3 }
```

Establece que para que dos mascotas puedan entablar amistad deben tener al menos dos gustos en común. **Restricción 2 OCL**



```
1 fun gustosCoincidentes[m1, m2: Mascota]: Int {  
2   #(m1.gustos & m2.gustos)  
3 }
```

Función necesaria para el hecho que calcula la cantidad de gustos compartidos entre dos mascotas, funcionando como criterio base para permitir o no la relación de amistad.

3.3.11 Fact “ProductoVerificadoPorVeterinarios”



```
1 fact ProductoVerificadoPorVeterinarios {  
2   all aP: AnuncioProducto | productoVerificadoPorVeterinario[aP.productoAnunciado] > 0  
3 }
```

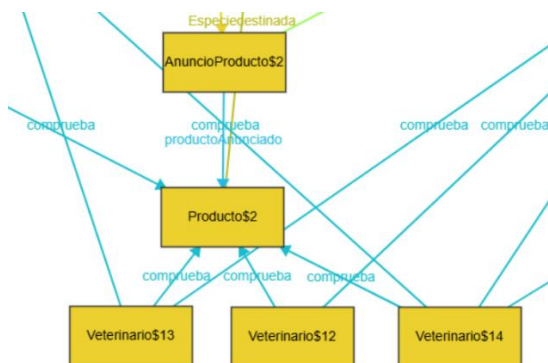
Determina que un producto solo se considera verificado cuando ha sido aprobado por al menos cuatro veterinarios distintos. Esto incrementa la fiabilidad de los productos que se promocionan.

Por simplificar el modelo en alloy, consideramos que un producto es verificado si ha sido revisado por al menos cuatro veterinarios → **Restricción 3 y restricción 13 OCL**



```
1 fun productoVerificadoPorVeterinario[prod: Producto]: Int {  
2   #{ v: Veterinario | prod in v.comprueba } > 3  
3   implies 1  
4   else 0  
5 }
```

Función necesaria para el hecho que emula la verificación de un producto: Si un producto ha sido comprobado por más de 3 veterinarios hace que sea un producto verificado.



3.3.12 Fact “ProductoRecomendadoPorMascota”

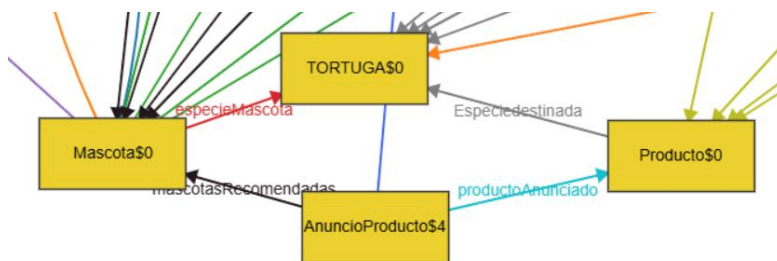


```

1 fact ProductoRecomendadoPorMascota {
2   all aP: AnuncioProducto | {
3     all m: aP.mascotasRecomendadas | {
4       #(aP.productoAnunciado.EspecieDestinada &
5         m.especieMascota)=1
6     }
7   }
8 }

```

Restringe la recomendación de productos a mascotas cuya especie esté incluida entre las especies destinatarias de dicho producto. Evita así situaciones que no tienen sentido como recomendar productos para gatos a perros. **Restricción 4 OCL**



3.3.13 Fact “MascotaMaxTresPublicacionesPorFecha”



```

1 fact MascotaMaxTresPublicacionesPorFecha {
2   all m: Mascota | all f: Fecha |
3     #{ p: PublicacionNormal | p in m.publicacionesPropias and p.fechaPublicada = f } ≤ 3
4 }

```

Limita a un máximo de tres el número de publicaciones que puede realizar una mascota en un mismo día. Esto previene saturaciones de contenido. **Restricción 5 OCL**



3.3.14 Fact “AnuncioAdopcionConVacunasValidas”



```
1 fact AnuncioAdopcionConVacunasValidas {  
2   all aa: AnuncioAdopcion | one aa.mascotaAdopcion.vacunasTodas  
3 }
```

Obliga a que cualquier mascota puesta en adopción tenga todas sus vacunas al día. Con esto garantizamos que las mascotas en adopción estén en condiciones sanitarias óptimas.

Restricción 7 OCL

3.3.15 Fact “MascotaNoPuedeApuntarseDosVeces”



```
1 fact MascotaNoPuedeApuntarseDosVeces {  
2   all m: Mascota, e1, e2: Evento |  
3     e1 ≠ e2 and m in e1.participantes and m in e2.participantes implies  
4     e1.fechaEvento.ValorDias ≠ e2.fechaEvento.ValorDias  
5 }
```

Prohíbe que una misma mascota se apunte a más de un evento que se celebre en la misma fecha, evitando solapamientos de participación. **Restricción 8 OCL**

3.3.16 Fact “eventoSuspendido”



```
1 fact eventoSuspendido{  
2   all e: Evento | {  
3     (diasRestantes[e.fechaEvento] ≤ 7 and #{e.participantes} ≤ 6)  
4     implies #{e.estadoEvento} > 0  
5   }  
6 }
```

Si a menos de siete días para la fecha de un evento no se han inscrito al menos seis mascotas, el evento se considera suspendido. Esta restricción vela por la viabilidad y el interés de los eventos. **Restricción 10 OCL**



```
1 fun diasRestantes[f1: Fecha]: Int {  
2   f1.ValorDias.minus[fechaActual.ValorDias]  
3 }
```

Función necesaria para el hecho que calcula los días restantes de la fecha pasada por parámetro

3.3.17 Fact “NombreUnico”



```
1 fact NombreUnico {  
2   all u1, u2: Usuario | u1 ≠ u2 implies u1.nombre ≠ u2.nombre  
3 }
```

Prohíbe la existencia de usuarios con nombres idénticos en el sistema, garantizando la unicidad de identificación por nombre. **Restricción 11 OCL**

3.3.18 Fact “EventoNoAgresivo”



```
1 fact EventoNoAgresivo {  
2   all e:Evento | no e.participantes.comportamiento  
3 }
```

Impide que mascotas catalogadas como agresivas participen en eventos, protegiendo así la seguridad de los participantes. **Restricción 12 OCL**

3.3.19 Fact “MascotaNoAmigos”



```
1 fact MascotaNoAmigos {  
2   all m: Mascota | m !in m.amigos  
3 }
```

Impide que una mascota se añada a si mismo a su propia lista de amigos, evitando relaciones erróneas. **Restricción 14 OCL**

3.3.20 Fact “AnuncioAgresivo”



```
1 fact AnuncioAgresivo {  
2   all a:AnuncioAdopcion | no a.mascotaAdopcion.comportamiento  
3 }
```

Prohíbe que mascotas con comportamiento agresivo aparezcan en anuncios de adopción, ya que podrían suponer un riesgo para futuros adoptantes. **Restricción 15 OCL**

3.3.21 Fact “EventosNoPuedenTenerMismaFechaYUbicacion”



```
1 fact EventosNoPuedenTenerMismaFechaYUbicacion {  
2   all e1, e2: Evento | e1 ≠ e2 implies  
3     (e1.fechaEvento ≠ e2.fechaEvento or e1.ubicacion ≠ e2.ubicacion)  
4 }
```

Impide que dos eventos se celebren en la misma fecha si comparten ubicación. Solo se permite coincidencia de fechas si los eventos ocurren en lugares diferentes. **Restricción 16 OCL**

3.3.22 Fact “CantidadMinimaDeGustosPorMascota”



```
1 fact CantidadMinimaDeGustosPorMascota {  
2   all m: Mascota | #m.gustos ≥ 3  
3 }
```

Obliga a que cada mascota tenga, al menos, tres gustos asignados en su perfil. Esto permite mejorar las sugerencias de amistad y la personalización del contenido en la plataforma. **Restricción 17 OCL**

3.3.23 Fact “eventoConSemanaDeAnticipacion”



```
1 fact eventoConSemanaDeAnticipacion {  
2   all e: Evento | diferenciaFechas[e.fechaEvento, e.fechaPublicada] > 7  
3 }
```

Establece que la fecha de celebración de un evento debe situarse 7 días después de la fecha de publicación. **Restricción 18 OCL**



```
1 fun diferenciaFechas[f1: Fecha, f2: Fecha]: Int {  
2   f1.ValorDias.minus[f2.ValorDias]  
3 }
```

Función

necesaria para el hecho que calcula la diferencia de días entre las dos fechas pasadas por parámetro.

3.4 Predicados

3.4.1 Producto verificado por veterinario



```
1 pred ProductoVerificadoPorVeterinariosPred {
2   #AnuncioProducto > 4
3   some a :AnuncioProducto | productoVerificadoPorVeterinario[a.productoAnunciado] > 0
4   some a2 :AnuncioProducto | productoVerificadoPorVeterinario[a2.productoAnunciado] = 0
5 }
6 //run ProductoVerificadoPorVeterinariosPred for 15
```

Creamos una instancia en la que existan al menos 4 anuncios de productos en los cuales uno de ellos tenga el producto verificado y otro anuncio no tenga el producto verificado. Nuestro predicado haría que el modelo quedara inconsistente.

3.4.2 Mascota en adopción sin vacunas



```
1 pred mascotaEnAdopcionSinVacunas {
2   // Debe existir al menos un anuncio de adopción
3   some aa: AnuncioAdopcion |
4     // La mascota en adopción NO tiene sus vacunas al día
5     #(aa.mascotaAdopcion.vacunasTodas)=0
6
7 }
8
```

Creamos una instancia en la que existe un anuncio de adopción que tenga una mascota sin la vacunación completa. Nuestro predicado haría que el modelo quedara inconsistente.

3.4.3 Un propietario con 5 mascotas



```
1 pred cincoMascotasUnPropietario {
2   -- Hay exactamente un propietario
3   one p: Propietario | {
4     -- Hay exactamente 5 mascotas
5     #Mascota = 5
6
7     -- Cada mascota pertenece al propietario
8     all m: Mascota | m in p.dueño and p in m.mascota
9   }
10 }
```

Creamos una instancia en la que exista un único propietario y existan 5 mascotas, de esta forma con este propietario tenga 5 mascotas. Nuestro predicado haría que el modelo quedara inconsistente.

3.4.4 Cada mascota máximo 3 publicaciones por día



```
1 pred MascotaMaxTresPublicacionesPorFechaPred{
2   one m: Mascota | {
3     // La mascota tiene exactamente 4 publicaciones en un día
4     one f: Fecha |
5       #{ p: PublicacionNormal | p in m.publicacionesPropias and p.fechaPublicada = f } = 4
6   }
7 }
8 //run MascotaMaxTresPublicacionesPorFechaPred for 15
9
```

Creamos una instancia en la que exista una única mascota y esta tenga 4 publicaciones normales. Nuestro predicado haría que el modelo quedara inconsistente.

3.5 Asertos

3.5.1 Máximo de 3 publicaciones diarias



```
1  assert MascotaMaxTresPublicacionesPorFechaAssert {
2      #PublicacionNormal =4
3
4      //fechaActual.ValorDias = 100
5      all m: Mascota | all f: Fecha |
6          #{ p: PublicacionNormal | p in m.publicacionesPropias and p.fechaPublicada = f } <= 3
7  }
8  check MascotaMaxTresPublicacionesPorFechaAssert for 15 but 8 Int
9
```

Verifica que las mascotas como máximo tenga 3 publicaciones en un solo día.

3.5.2 Anuncios destinados a distintas especies



```
1  assert anuncios{
2      // Para cada anuncio de adopción, la mascota en adopción debe tener todas sus vacunas al día
3      // La mascota en adopción tiene todas sus vacunas al día
4      #AnuncioProducto > 4
5      some aa: AnuncioProducto | #aa.productoAnunciado.Especiedestinada = 1
6
7      // Para cada anuncio de adopción, el propietario del anuncio debe ser el dueño de la mascota en adopción
8      some ap: AnuncioProducto | #ap.productoAnunciado.Especiedestinada > 1
9  }
10
11  check anuncios for 15
```

Creamos una instancia con cuatro anuncios de productos en el que alguno esté destinado a una especie y otro a más de una especie.

4 Conclusión

Realizando el proyecto hemos tenido que modificar numerosos aspectos del modelo anterior en OCL. Partiendo de la base de que en Alloy no usamos atributos para hacer las restricciones y por tanto hemos creado numerosas signaturas para solucionar esta carencia.

En Alloy hemos solucionado el problema de las fechas que tuvimos en el anterior modelo en el cual no podíamos crear una fecha actual al sistema. En el nuevo modelo conseguimos solucionar el problema teniendo una fecha actual del sistema. Para simplificar las comparaciones entre fechas, hemos emulado la fecha como un único valor entero (solo los días). Como diferencia entre OCL y Alloy tenemos que algunas restricciones como la número 6 y la número 9 en las que en Alloy no hemos incluido, pero están contempladas de otras formas.

Alloy nos ha ofrecido una oportunidad de modelar un proyecto de forma más abstracta y por tanto preocuparnos más de las relaciones entre clases.