LENGUAJE UML. VISUAL PARADIGM

Entornos de desarrollo

Índice

Índice

1. ¿Por qué UML?

- **UML** son las siglas de "**Unified Modeling Language**" o "**Lenguaje Unificado de Modelado**".
- Se trata de un **estándar** que se ha adoptado a nivel internacional por numerosos organismos y empresas para **crear esquemas, diagramas** y **documentación** relativa a los **desarrollos de software** (programas informáticos).

1. ¿Por qué UML?

- Estos **diagramas** contenidos en UML son la forma más común y más utilizada de **modelado** de software.
- ¿Qué es **modelar**?
  - Modelar consiste en **hacer un diseño previo** de una aplicación antes de proceder a su desarrollo e implementación.
  - De forma similar que un arquitecto dibuja planos sobre la casa que va a construir, un analista de software (u otros perfiles) crea distintos diagramas UML que sirven de base para la posterior construcción/mantenimiento del sistema.
  - El modelado es la principal forma de visualizar el diseño de una aplicación con la finalidad de compararla con los requisitos antes de que el equipo de desarrollo comience a codificar

1. ¿Por qué UML?

- El modelado es **vital** en todo tipo de proyectos, pero cobra especialmente importancia a medida que el proyecto crece de tamaño.
- Para que una aplicación funcione correctamente, debe ser diseñada para permitir:
  - **La escalabilidad.**
  - **La seguridad.**
  - **La ejecución**.
- Utilizando diagramas UML se consigue **visualizar** y **verificar** los diseños de sus sistemas de software **antes** de que la implementación del código haga que los cambios sean **difíciles** y **demasiado costosos**.

1. ¿Por qué UML?

- Estos **diagramas de UML** son representaciones gráficas que muestran de forma parcial un sistema de información (bien esté siendo desarrollado o ya lo haya sido).
- Suelen estar acompañados de documentación que les sirve de apoyo, adoptando esta múltiples formas.
- Además, UML no excluye la posibilidad de mezclar diagramas, algo que, de hecho, suele ser bastante común.

1. ¿Por qué UML?

- Como **principal desventaja** ampliamente mencionada de UML podemos nombrar el hecho de que se trata de un lenguaje muy amplio, haciendo, en ocasiones, complicado utilizar todas las posibilidades que ofrece.
- No obstante, los analistas tienden a utilizar los diagramas de forma sencilla, consiguiendo que sean entendidos fácilmente por cualquier persona que accedan a ellos.

1. ¿Por qué UML?

- Los modelos o diagramas de UML nos ayudan a trabajar a un **mayor nivel de abstracción**.
- Permite **modelar cualquier tipo de aplicación** corriendo en cualquier combinación de hardware y software, sistema operativo, lenguaje de programación y red, es decir, UML es independiente de la plataforma hardware sobre la que actúa el software (**flexibilidad**).

1. ¿Por qué UML?

- UML ofrece ese modelado.
  - Utiliza diagrama.
  - Se denomina lenguaje por ser una forma común de expresarse por todos los analistas, desarrolladores y usuarios.
- Está desarrollado para **ayudar a todos estos** (y más) **perfiles** a:
  - Especificar.
  - Visualizar.
  - construir.
  - Documentar.

1. ¿Por qué UML?

- A pesar de que cada diagrama UML en particular aporta su visión particular al modelado, el lenguaje en su conjunto tiene **algunas características** que interesa resaltar:
  - Es muy **sencillo**.
    - Pese a que si es usado de forma completa puede llegar a complicarse, lo normal es que se simplifique.
  - Es capaz de modelar **todo tipo de sistemas**.
  - Es un lenguaje **universal**.
    - Hace que todos los miembros del equipo se relacionen a través de sus diagramas sean del ámbito que sean.

1. ¿Por qué UML?

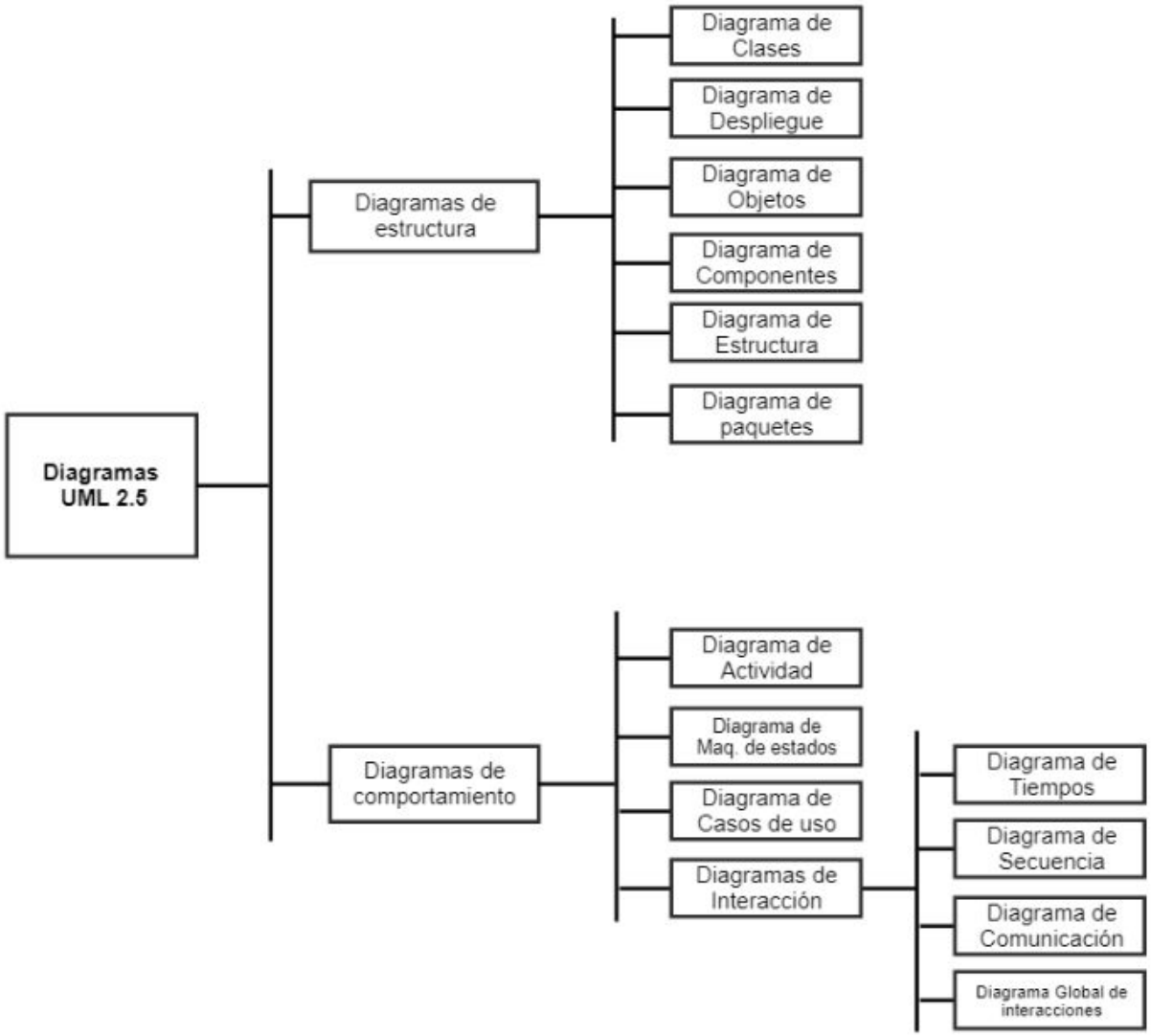
- Es fácilmente **extensible.**
  - Tiene mecanismos sencillos para especializar los conceptos fundamentales.
- Es **visual** y, por lo tanto, intuitivo.
- Es **independiente** del desarrollo, del lenguaje y de la plataforma.
- Bien ejecutado aporta un conjunto considerable de **buenas prácticas.**
- **No está completo.**
  - Utilizando los distintos diagramas no podemos estar seguros de comprender con totalidad el sistema que va a desarrollarse.
  - Los diagramas, para facilitar su comprensión pueden (y suelen) omitir información, pueden tener partes que se entienden de distintas maneras o, incluso, pueden tener conceptos que no pueden ser representados por ningún diagrama.

Índice

2. Tipos de diagrama UML

- Existen dos clasificaciones de diagramas:
  - **Diagramas estructurales.**
  - **Diagramas de comportamiento.**

2. Tipos de diagrama UML



Índice

2.1. Diagramas estructurales

- Los **diagramas estructurales** muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción.
- Existen un total de siete tipos de diagramas de estructura:
  - **Diagrama de clases.**
  - Diagrama de componentes.
  - Diagrama de despliegue.
  - Diagrama de objetos.
  - Diagrama de paquetes.
  - Diagrama de perfiles.
  - Diagrama de estructura compuesta.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - Este diagrama no incluye la forma en la que se comportan a lo largo de la ejecución los distintos elementos, esa función puede ser representada a través de un diagrama de comportamiento.
  - Es un **diagrama puramente orientado al modelo de programación orientado a objetos**, ya que define las clases que se utilizarán cuando se pase a la fase de construcción y la manera en que se relacionan las mismas.
  - Se podría equiparar, salvando las distancias, al famoso diagrama de modelo Entidad-Relación (E/R).

2.1. Diagramas estructurales

- **Diagrama de clases**
  - Está formado por tres elementos.
    - Clases.
    - Relaciones.
    - Interfaces.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - Está formado por tres elementos.
    - **Clases.**
    - Relaciones.
    - Interfaces.

2.1. Diagramas estructurales

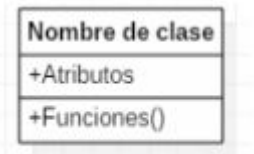
- **Diagrama de clases**
  - **Clases**
    - El **elemento principal del diagrama** y representa, como su nombre indica, una clase dentro del paradigma de la orientación a objetos.
    - Este tipo de elementos normalmente se utilizan para representar conceptos o entidades del "negocio".
    - Una clase define un **grupo de objetos** que comparten características, condiciones y significado.
    - La manera más rápida para encontrar clases sobre un enunciado, sobre una idea de negocio o, en general, sobre un tema concreto es buscar los **sustantivos** que aparecen en el mismo.
    - Por poner algún ejemplo, algunas clases podrían ser: Animal, Persona, Mensaje, Expediente… Es un concepto muy amplio y **resulta fundamental identificar de forma efectiva estas clases**, en caso de no hacerlo correctamente se obtendrán una serie de problemas en etapas posteriores, teniendo que volver a hacer el análisis y perdiendo parte o todo el trabajo que se ha hecho hasta ese momento.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Clases**
    - Bajando de nivel una clase está compuesta por tres elementos:
      - **Nombre de la clase**
      - **Atributos.**
      - **Funciones.**
    - **Estos elementos se incluyen en la representación** (o no, dependiendo del nivel de análisis).

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Clases**
    - Para representar la clase con estos elementos se utiliza una caja que es dividida en tres zonas utilizando para ello líneas horizontales:
      - **La primera de las zonas** se utiliza para el **nombre de la clase.** En caso de que la clase sea abstracta se utilizará su nombre en cursiva.
      - **La segunda de las zonas** se utiliza para escribir los **atributos** de la clase, uno por línea y utilizando el siguiente formato:
        - *visibilidad nombre_atributo : tipo = valor-inicial { propiedades }*
    - Aunque esta es **la forma "oficial"** de escribirlas, es común simplificando únicamente poniendo el nombre y el tipo o únicamente el nombre.
      - **La última** de las zonas incluye cada una de las **funciones** que ofrece la clase. De forma parecida a los atributos, sigue el siguiente formato:
        - *visibilidad nombre_funcion { parametros } : tipo-devuelto { propiedades }*
    - De la misma manera que con los atributos, se suele **simplificar** indicando únicamente el nombre de la función y, en ocasiones, el tipo devuelto.



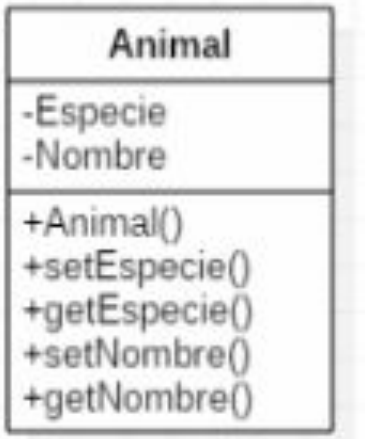
Notación de una clase

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Clases**
    - Tanto los atributos como las funciones incluyen al principio de su descripción la visibilidad que tendrá.
    - Esta visibilidad se identifica escribiendo un símbolo y podrá ser:
      - **(+) Pública.** Representa que se puede acceder al atributo o función desde cualquier lugar de la aplicación.
      - **(-) Privada.** Representa que se puede acceder al atributo o función únicamente desde la misma clase.
      - **(#) Protegida.** Representa que el atributo o función puede ser accedida únicamente desde la misma clase o desde las clases que hereden de ella (clases derivadas).
    - Estos tres tipos de visibilidad son los más comunes. No obstante, pueden incluirse otros en base al lenguaje de programación que se esté usando (no es muy común). Por ejemplo: (/) Derivado o (~) Paquete.
    - Un ejemplo de clase podría ser el siguiente:

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Clases**



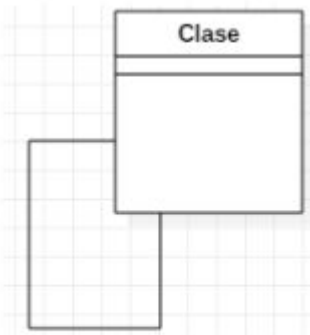
Ejemplo de una clase

2.1. Diagramas estructurales

- **Diagrama de clases**
  - Está formado por tres elementos.
    - Clases.
    - **Relaciones.**
    - Interfaces.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - Una relación i**dentifica una dependencia.**
    - Esta dependencia puede ser entre dos o más clases (más común) o una clase hacía sí misma (menos común, pero existen), este último tipo de dependencia se denomina *dependencia reflexiva*.
    - Las relaciones se representan con una línea que une las clases, esta línea variará dependiendo del tipo de relación.



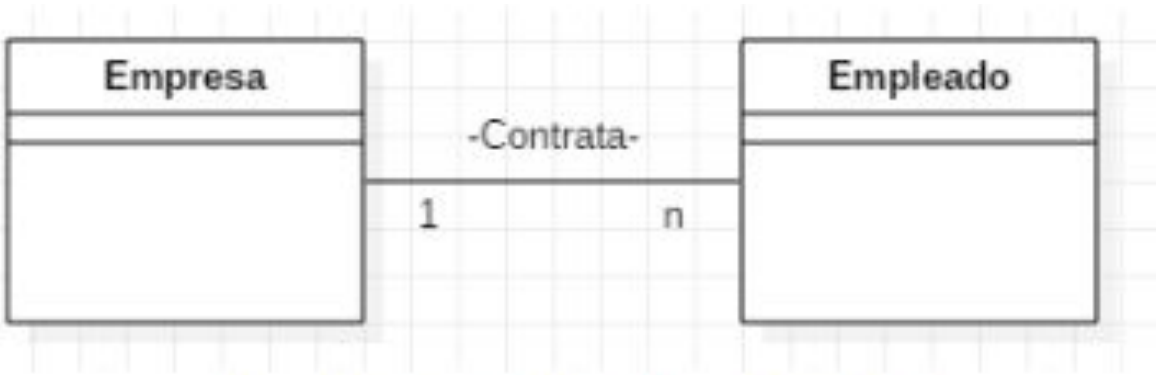
Relación reflexiva

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - Las relaciones en el diagrama de clases tienen varias **propiedades**, que dependiendo la profundidad que se quiera dar al diagrama se representarán o no.
    - Estas propiedades son las siguientes:
      - **Multiplicidad**. Es decir, el número de elementos de una clase que participan en una relación. Se puede indicar un número, un rango… Se utiliza *n* o * para identificar un número cualquiera.
      - **Nombre de la asociación.** En ocasiones se escriba una indicación de la asociación que ayuda a entender la relación que tienen dos clases. Suelen utilizarse verbos como por ejemplo: “Una empresa contrata a n empleados”

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**



Ejemplo de relación Empresa-Empleado

2.1. Diagramas estructurales

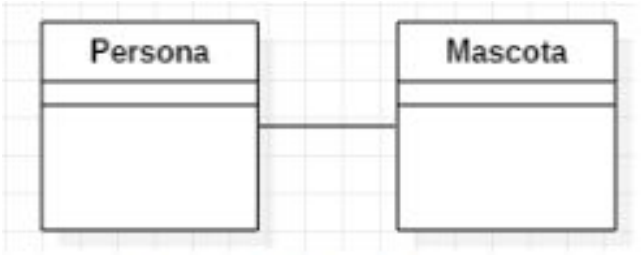
- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - Asociación.
      - Agregación.
      - Composición.
      - Dependencia.
      - Herencia.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Asociación.**
      - Agregación.
      - Composición.
      - Dependencia.
      - Herencia.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Asociación.**
        - Este tipo de relación es el más común y se utiliza para representar **dependencia semántica.**
        - Se representa con una simple línea continua que une las clases que están incluidas en la asociación.
        - Un ejemplo de asociación podría ser: “Una mascota pertenece a una persona”.



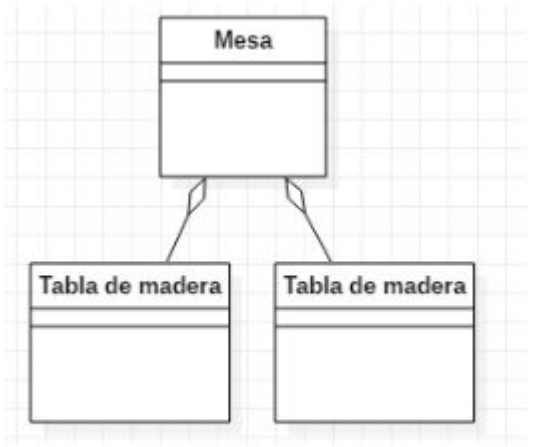
Ejemplo de asociación

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - Asociación.
      - **Agregación.**
      - Composición.
      - Dependencia.
      - Herencia.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Agregación.**
        - Representa relaciones en las que **un objeto es parte de otro,** pero aun así debe tener **existencia en sí mismo.**
        - Se representa con una línea que tiene un rombo en la parte de la clase que es una agregación de la otra clase (es decir, en la clase que contiene las otras).
        - Un ejemplo de esta relación podría ser: "Las mesas están formadas por tablas de madera y tornillos o, dicho de otra manera, los tornillos y las tablas forman parte de una mesa". Como ves, el tornillo podría formar parte de más objetos, por lo que interesa especialmente su abstracción en otra clase.



Ejemplo de agregación

2.1. Diagramas estructurales

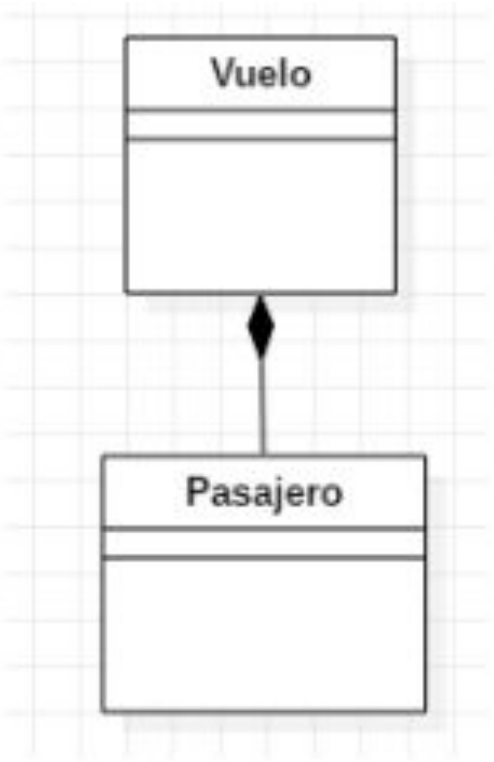
- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - Asociación.
      - Agregación.
      - **Composición.**
      - Dependencia.
      - Herencia.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Composición.**
        - Representa una **relación jerárquica entre un objeto y las partes que lo componen, pero de una forma más fuerte.**
        - En este caso, los elementos que forman parte no tienen sentido de existencia cuando el primero no existe. Es decir, cuando el elemento que contiene los otros desaparece, deben desaparecer todos ya que no tienen sentido por sí mismos sino que dependen del elemento que componen.
        - Además, suelen tener los mismos tiempo de vida. Los componentes no se comparten entre varios elementos, esta es otra de las diferencias con la agregación.
        - Se representa con una línea continua con un rombo relleno en la clase que es compuesta.
        - Un ejemplo de esta relación sería: “Un vuelo de una compañía aérea está compuesto por pasajeros, que es lo mismo que decir que un pasajero está asignado a un vuelo”

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Composición.**



Ejemplo de composición

2.1. Diagramas estructurales

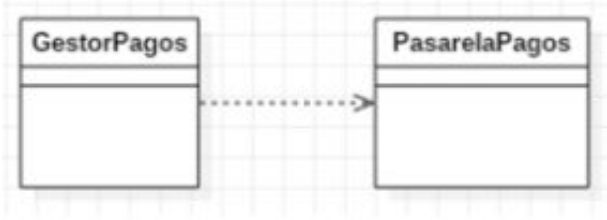
- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Composición.**
        - La diferencia entre agregación y composición es semántica, por lo que **a veces no está del todo definida.**
        - Ninguna de las dos tienen análogos en muchos lenguajes de programación (como por ejemplo Java).

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - Asociación.
      - Agregación.
      - Composición.
      - **Dependencia.**
      - Herencia.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Dependencia.**
        - Se utiliza este tipo de relación para **representar que una clase requiere de otra para ofrecer sus funcionalidades.**
        - Es muy sencilla y se representa con una flecha discontinua que va desde la clase que necesita la utilidad de la otra flecha hasta esta misma.
        - Un ejemplo de esta relación podría ser la siguiente:



Ejemplo de dependencia

2.1. Diagramas estructurales

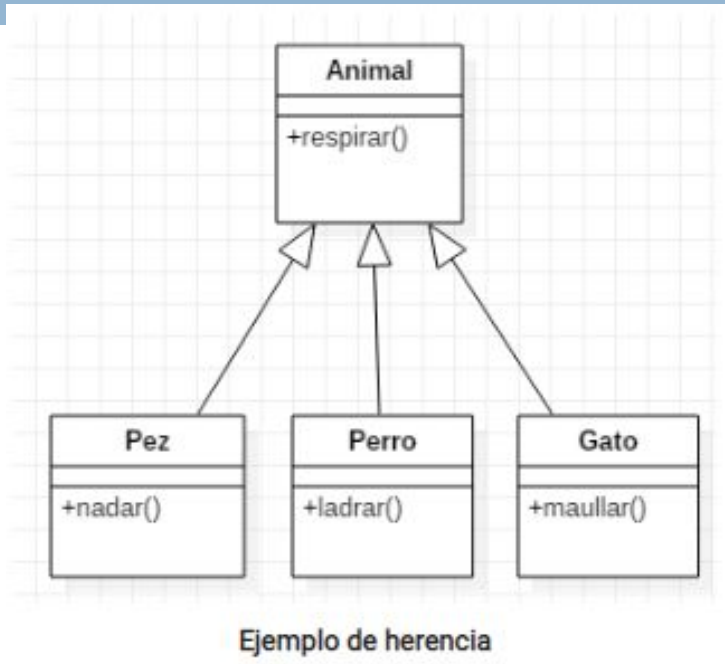
- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - Asociación.
      - Agregación.
      - Composición.
      - Dependencia.
      - **Herencia**.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Herencia.**
        - Otra relación muy común en el diagrama de clases es la herencia.
        - Este tipo de relaciones permiten que **una clase (clase hija o subclase) reciba los atributos y métodos de otra clase (clase padre o superclase).**
        - Estos atributos y métodos recibidos se suman a los que la clase tiene por sí misma.
        - Se utiliza en relaciones “es un”.
        - Un ejemplo de esta relación podría ser la siguiente: Un pez, un perro y un gato son animales.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Relaciones**
    - **Tipos de relaciones**
      - **Herencia.**



Ejemplo de herencia

- En este ejemplo, las tres clases (Pez, Perro, Gato) podrán utilizar la función respirar, ya que lo heredan de la clase animal, pero solamente la clase Pez podrá nadar, la clase Perro ladrar y la clase Gato maullar. La clase Animal podría plantearse ser definida abstracta, aunque no es necesario.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - Está formado por tres elementos.
    - Clases.
    - Relaciones.
    - **Interfaces.**

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Interfaces.**
    - Una interfaz es una entidad que declara una **serie de atributos, funciones y obligaciones.**
    - Es una especie de contrato donde toda instancia asociada a una interfaz debe de implementar los servicios que indica aquella interfaz.
    - Dado que únicamente son declaraciones **no pueden ser instanciadas**.
    - Las interfaces se asocian a clases. Una asociación entre una clase y una interfaz representa que esa clase cumple con el contrato que indica la interfaz, es decir, incluye aquellas funciones y atributos que indica la interfaz.
    - Su representación es similar a las clases, pero indicando arriba la palabra <<interface>>.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Interfaces.**

Notación de interfaz

2.1. Diagramas estructurales

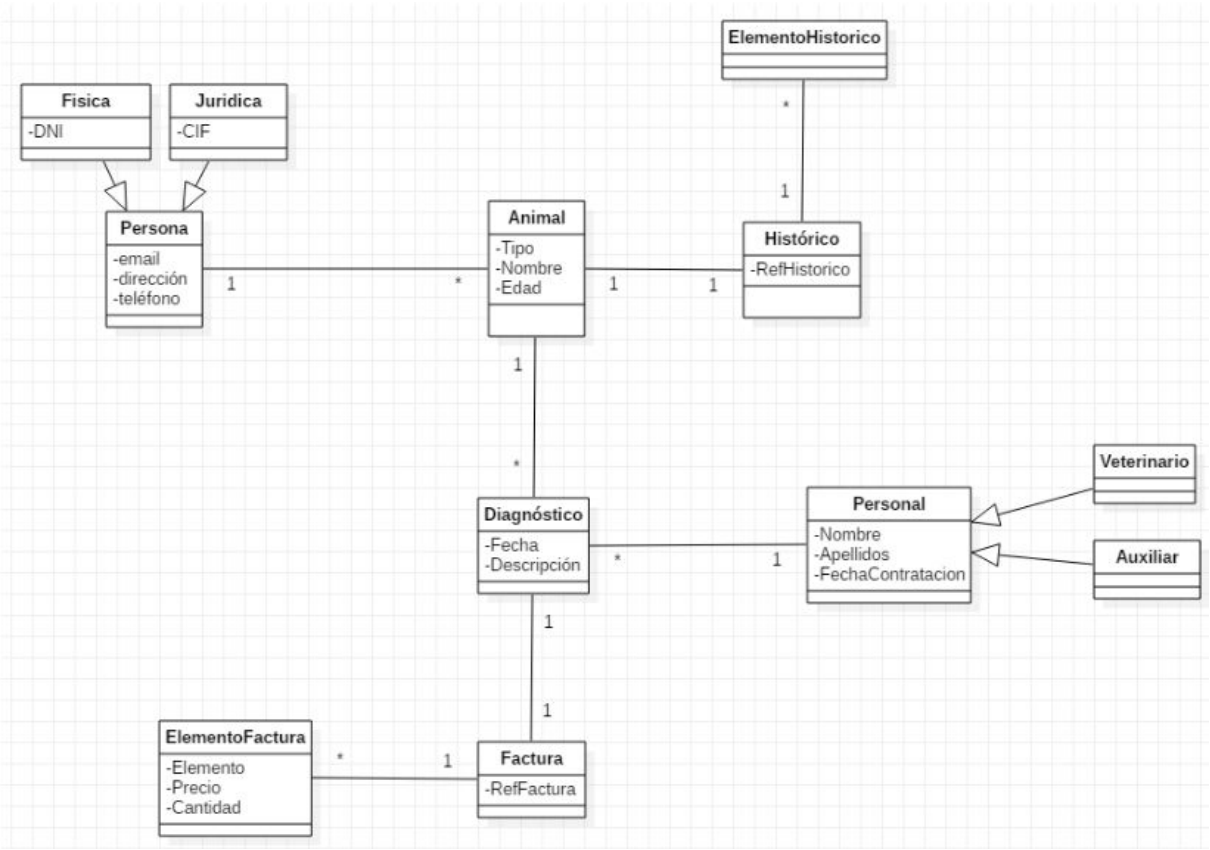
- **Diagrama de clases**
  - **Cómo dibujar un diagrama de clases**
    - Los diagramas de clase van de la mano con el diseño orientado a objetos. Por lo tanto, saber lo básico de este tipo de diseño es una parte clave para poder dibujar diagramas de clase eficaces.
    - Este tipo de diagramas son solicitados cuando se está describiendo la vista estática del sistema o sus funcionalidades. Unos pequeños pasos que puedes utilizar de guía para construir estos diagramas son los siguientes:
    - **Identifica** los nombres de las clase
      - El primer paso es identificar los objetos primarios del sistema. Las clases suelen corresponder a sustantivos dentro del dominio del problema.
    - **Distingue** las relaciones
      - El siguiente paso es determinar cómo cada una de las clases u objetos están relacionados entre sí. Busca los puntos en común y las abstracciones entre ellos; esto te ayudará a agruparlos al dibujar el diagrama de clase.
    - **Crea** la estructura
      - Primero, agrega los nombres de clase y vincúlalos con los conectores apropiados, prestando especial atención a la cardinalidad o las herencias. Deja los atributos y funciones para más tarde, una vez que esté la estructura del diagrama resuelta.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Buenas prácticas en la construcción del diagrama de clases**
    - Los diagramas de clase **pueden tender a volverse incoherentes** a medida que se expanden y crecen. Es mejor evitar la creación de diagramas grandes y **dividirlos** en otros más pequeños que se puedan vincular entre sí más adelante.
    - Usando la notación de clase simple, puedes crear rápidamente **una visión general de alto nivel** de su sistema. Se puede crear un diagrama detallado por separado según sea necesario, e incluso vincularlo al primero para una referencia fácil.
    - Cuantas más líneas se superpongan en sus diagramas de clase, más abarrotado se vuelve y, por tanto, más se complica utilizarlo. El lector se confundirá tratando de encontrar el camino. Asegúrate de que **no haya dos líneas cruzadas** entre sí, a no ser que no haya más remedio.
    - Usa **colores** para agrupar módulos comunes. Diferentes colores en diferentes clases ayudan al lector a diferenciar entre los diversos grupos.

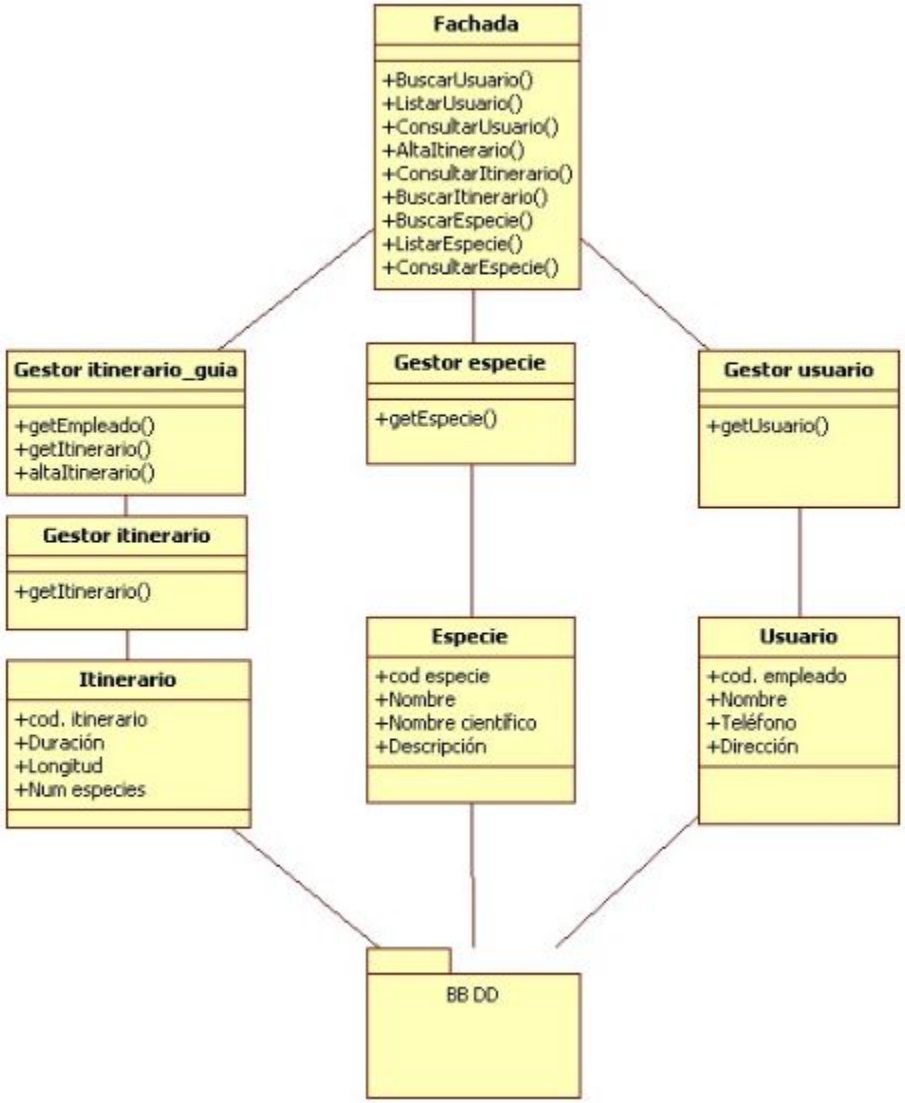
2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Ejemplo**



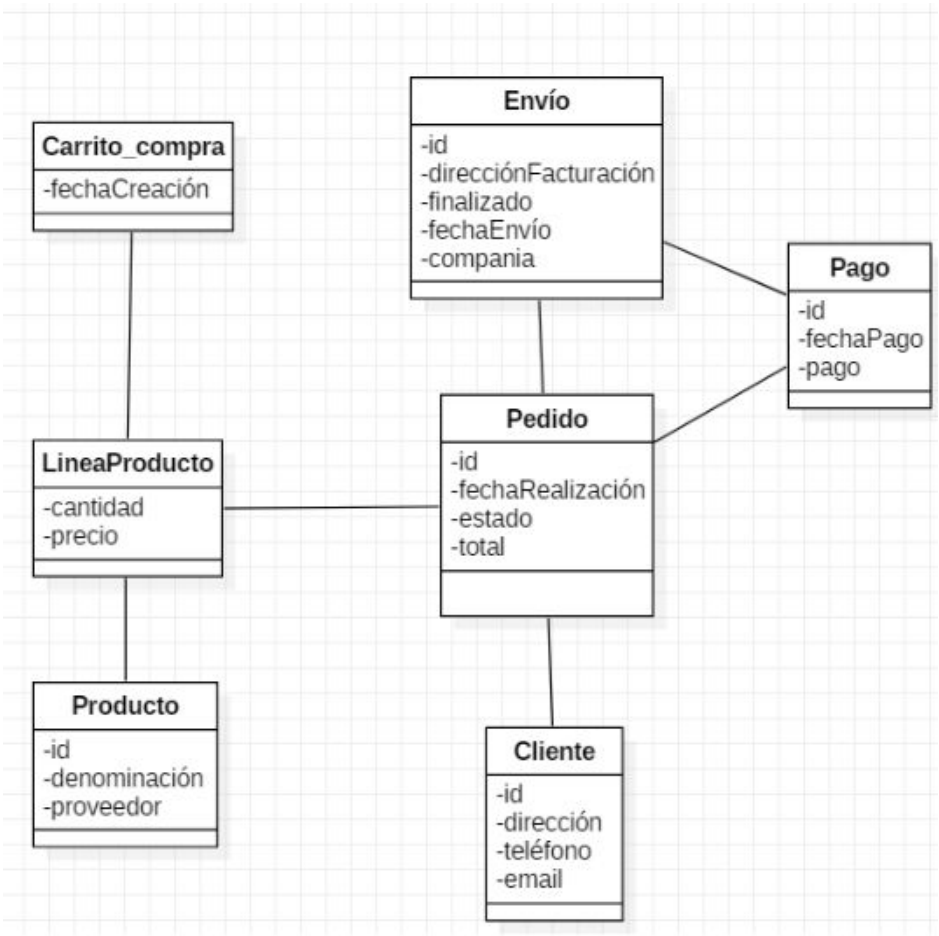
2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Ejemplo**



2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Ejemplo**



2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Actividad 0.-**
    - **Instalar Visual Paradigm.**

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Actividad 1.- Bibliotecas**
    - Una biblioteca tiene copias de libros. Estos últimos se caracterizan por su nombre, tipo (novela, teatro, poesía, ensayo), editorial, año y autor.
    - Los autores se caracterizan por su nombre, nacionalidad y fecha de nacimiento.
    - Cada copia tiene un identificador, y puede estar en la biblioteca, prestada, con retraso o en reparación.
    - Los lectores pueden tener un máximo de 3 libros en préstamo.
    - Cada libro se presta un máximo de 30 días, por cada día de retraso, se impone una "multa" de dos días sin posibilidad de coger un nuevo libro.
    - Realiza un diagrama de clases y añade los métodos necesarios para realizar el préstamo y devolución de libros.

2.1. Diagramas estructurales

- **Diagrama de clases**
  - **Actividad 2.- Empleados**
    - Una aplicación necesita almacenar información sobre empresas, sus empleados y sus clientes. Ambos se caracterizan por su nombre y edad.
    - Los empleados tienen un sueldo bruto, los empleados que son directivos tienen una categoría, así como un conjunto de empleados subordinados.
    - De los clientes además se necesita conocer su teléfono de contacto.
    - La aplicación necesita mostrar los datos de empleados y clientes.

Índice

2.2. Diagramas de comportamiento

- A diferencia de los diagramas estructurales, muestran como se comporta un sistema de información de forma dinámica. Es decir, describe los cambios que sufre un sistema a través del tiempo cuando está en ejecución.
- Hay un total de siete diagramas de comportamiento, clasificados de la siguiente forma:
  - Diagrama de casos de uso.
  - Diagrama de actividades.
  - Diagrama de máquina de estados.
  - Diagramas de interacción.
    - Diagrama de secuencia.
    - Diagrama de comunicación.
    - Diagrama de tiempos.
    - Diagrama global de interacciones.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - Es, con total seguridad, el diagrama **más conocido** y es utilizado para **representar los actores externos que interactúan con el sistema de información** y a través de que funcionalidades (casos de uso o requisitos funcionales) se relacionan.
  - Dicho de otra manera, muestra de manera **visual** las distintas **funciones** que puede realizar un **usuario** (más bien un tipo de usuario) **de un Sistema de Información**.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - Lo primero es saber cual es su finalidad:
    - **Representar los requisitos funcionales.**
    - **Representar los actores** que se comunican con el sistema.
      - Normalmente los actores del sistema son los usuarios y otros sistemas externos que se relacionan con el sistema. En el caso de los usuarios hay que entender el actor como un "perfil", pudiendo existir varios usuarios que actúan como el mismo actor.
    - **Representar las relaciones** entre requisitos funcionales y actores.
    - **Guiar el desarrollo** del sistema.
      - Crear un punto de partida sobre el que empezar a desarrollar el sistema.
    - **Comunicarse de forma precisa entre cliente y desarrollador.**
      - Simplifica la forma en que todos los participes del desarrollo, incluyendo el cliente, perciben como el sistema funcionará y ofrecerá una visión general común del mismo.

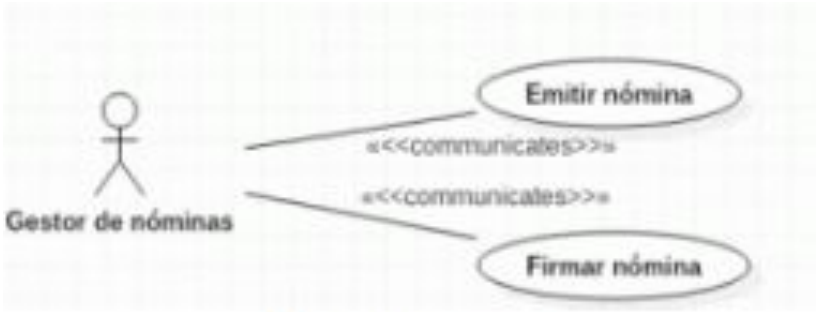
2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Actores**
      - Un actor es **algo** o **alguien** externo al sistema que interactúa de forma directa con el sistema.
      - Cuando decimos que interactúa nos referimos a que **aporta** información, **recibe** información, **inicia** una acción…

Representación de un actor

2.2. Diagramas de comportamiento

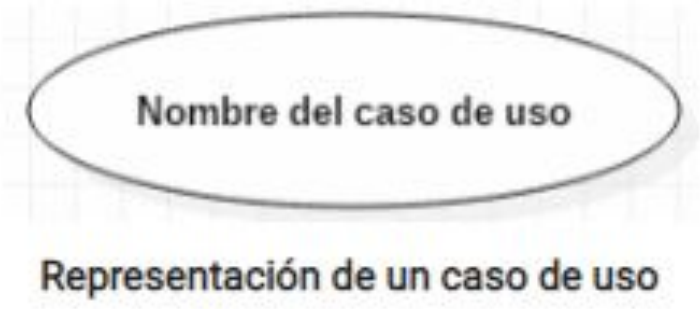
- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Actores**
      - **No** hay que entender los usuarios como **personas singulares,** sino como "perfiles o **roles**" que identifican a un tipo de usuario, pero no al usuario en sí.



Ejemplo de actor

2.2. Diagramas de comportamiento

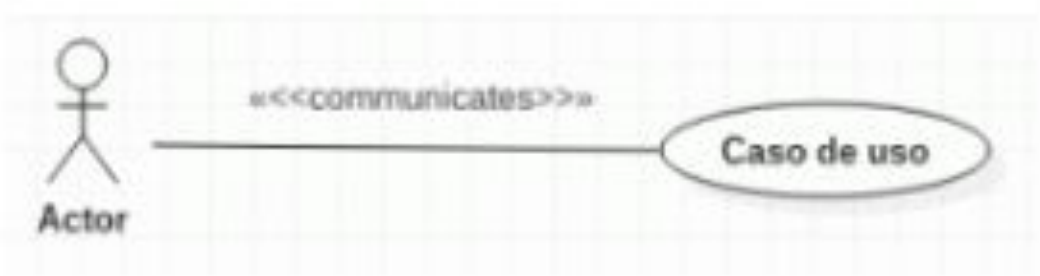
- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Casos de uso**
      - Existen muchos ejemplos de casos de uso. Algunos podrían ser: Crear pedido, Listar productos, Enviar correo. Cualquier acción que realice la aplicación.



Representación de un caso de uso

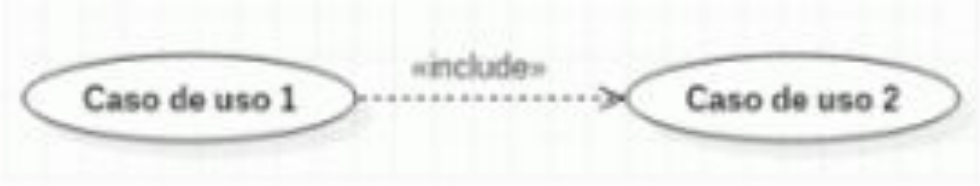
2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**
      - Las relaciones **conectan los casos de uso** con los actores o los casos de uso entre sí.



2.2. Diagramas de comportamiento

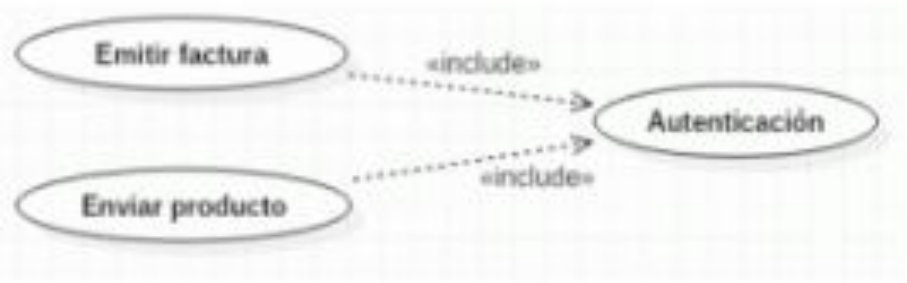
- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**
      - **<<include>>:** Se utiliza para representar que un caso de uso **utiliza siempre** a otro caso de uso.
      - Es decir, un caso de uso se ejecutará obligatoriamente (lo incluye, lo usa).
      - Se representa con una flecha discontinua que va desde el caso de uso de origen al caso de uso que se incluye.



Relación include entre dos casos de uso

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**
      - Un uso típico de este tipo de relaciones se produce cuando dos casos de uso **comparten una funcionalidad.**



Ejemplo de uso de include

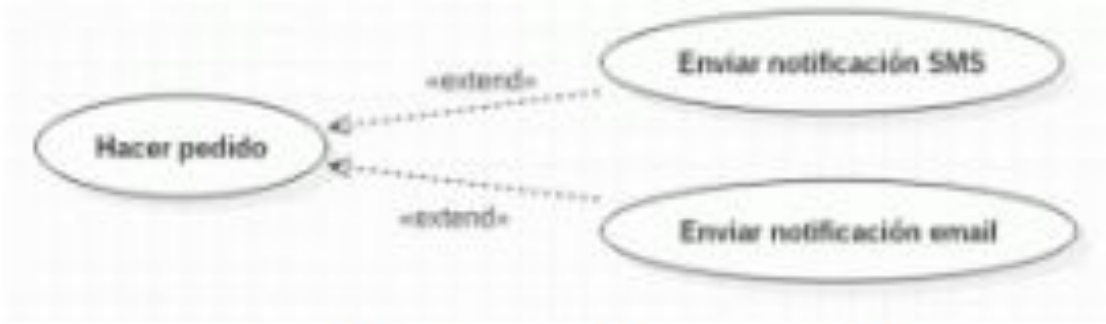
2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**
      - **<<extend>>:** Este tipo de relaciones se utilizan cuando un caso de uso tiene un comportamiento **opcional**, reflejado en otro caso de uso.
      - Es decir, un caso de uso puede ejecutar, normalmente dependiendo de alguna **condición** o **flujo** del programa, otro caso de uso.
      - Se representa con una flecha discontinua que va desde el caso de uso opcional al original.

Relación extend entre dos casos de uso

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**
      - Un ejemplo de esta relación podría ser la siguiente:



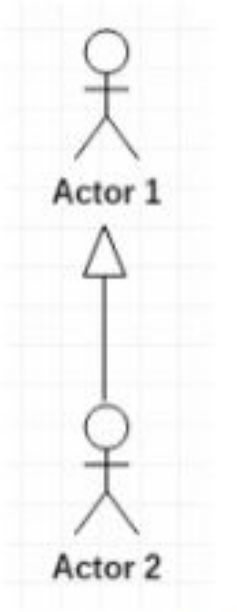
Ejemplo de relaciones extend

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**
      - Existe, además, otra relación denominada **generalización** que consiste en hacer que un elemento **herede** el comportamiento de otro.
      - Aunque se puede utilizar entre **casos de uso**, es **más común** utilizarlo **entre actores**, haciendo que uno de los actores tenga acceso a las **funcionalidades** de otro.
      - Se representa con una flecha con la punta hueca que va desde el elemento que hereda al elemento heredado:

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Elementos de un diagrama de casos de uso**
    - **Relaciones**



Generalización entre dos actores

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Descripción de requisitos funcionales y no funcionales**
    - Es común en este tipo de diagramas describir cada caso de uso junto con la secuencia de **pasos necesaria para completarlo** y las posibles **excepciones** hasta definir todas las situaciones posibles.
    - Esta descripción servirá de guía para el desarrollo, la profundidad de las situaciones que se traten dependerá de cada fase del proyecto o de cada situación en particular.
    - Existen dos tipos de requisitos:
      - **Requisitos funcionales**
      - **Requisitos no funcionales**

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Descripción de requisitos funcionales y no funcionales**
    - Ejemplos requisito funcional:

RF-01	Acceso Aplicación
Versión	Versión 1.0
Autores	Alonso Quijano
Objetivos Asociados	OBJ-01: Acceso Controlado a la Aplicación Software.
Requisitos asociados	RI-01: Información de los Usuarios.
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un usuario decida acceder a la aplicación.
Precondición	El usuario tiene que disponer de un nombre de usuario y una contraseña para poder acceder y deberá tener el acceso habilitado.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Descripción de requisitos funcionales y no funcionales**

Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema entrar en la aplicación.
	2	El sistema solicita al usuario que introduzca el nombre de usuario y su contraseña.
	3	El usuario introduce su nombre y su contraseña.
	4	El sistema comprueba los datos introducidos.
	5	Si los datos son correctos el sistema muestra la página de inicio de la aplicación.
Excepciones	**Paso**	**Acción**
	5	Si el nombre de usuario no es correcto. El sistema muestra un mensaje. Ir al paso 2.
	5	Si la contraseña no es correcta. El sistema muestra un mensaje. Ir al paso 2.
	5	Si el usuario no tiene el acceso habilitado a la aplicación. El sistema muestra un mensaje. Ir al paso 2.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Descripción de requisitos funcionales y no funcionales**

Postcondición	Si el nombre de usuario y la contraseña son correctos accede a la pantalla de inicio de la aplicación.
Importancia	Vital.
Urgencia	Inmediatamente.
Comentarios	Ninguno.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Descripción de requisitos funcionales y no funcionales**
    - Ejemplo requisito no funcional:

RNF-01	Entorno de Explotación
Versión	Versión 1.0
Autores	Alonso Quijano
Objetivos asociados	OBJ-05: Funcionamiento óptimo por usuario estándar
Requisitos asociados	
Descripción	El sistema deberá funcionar sin ningún tipo de limitación en equipos con: Pentium IV a 2,4 GHz, con 1 GBytes de RAM y al menos 6 GBytes de disco duro.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Alta.
Comentarios	Ninguno.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - **Actividad 1.- Resuelve este diagrama de casos de uso**
    - La clínica veterinaria almacena datos de contacto de todos sus clientes como pueden ser: Nombre, Apellidos, DNI, Fecha de nacimiento, Teléfono o Email. Estos datos son introducidos y gestionados por los auxiliares, que ejercen las funciones administrativas.

    - Además se almacena información de cada uno de las mascotas de las que es dueño cada cliente. Obviamente, cada cliente puede tener más de una mascota, pero cada mascota solo puede pertenecer a un único cliente. Se permite, además, cambiar el dueño de una mascota por otro.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - Al dar de alta un nuevo animal, se comprobará en el registro del REIAC (Red Española de Identificación de Animales de Compañía) si el animal está correctamente dado de alta. Este proceso únicamente se hará en animales que tengan la obligación de estar identificados.

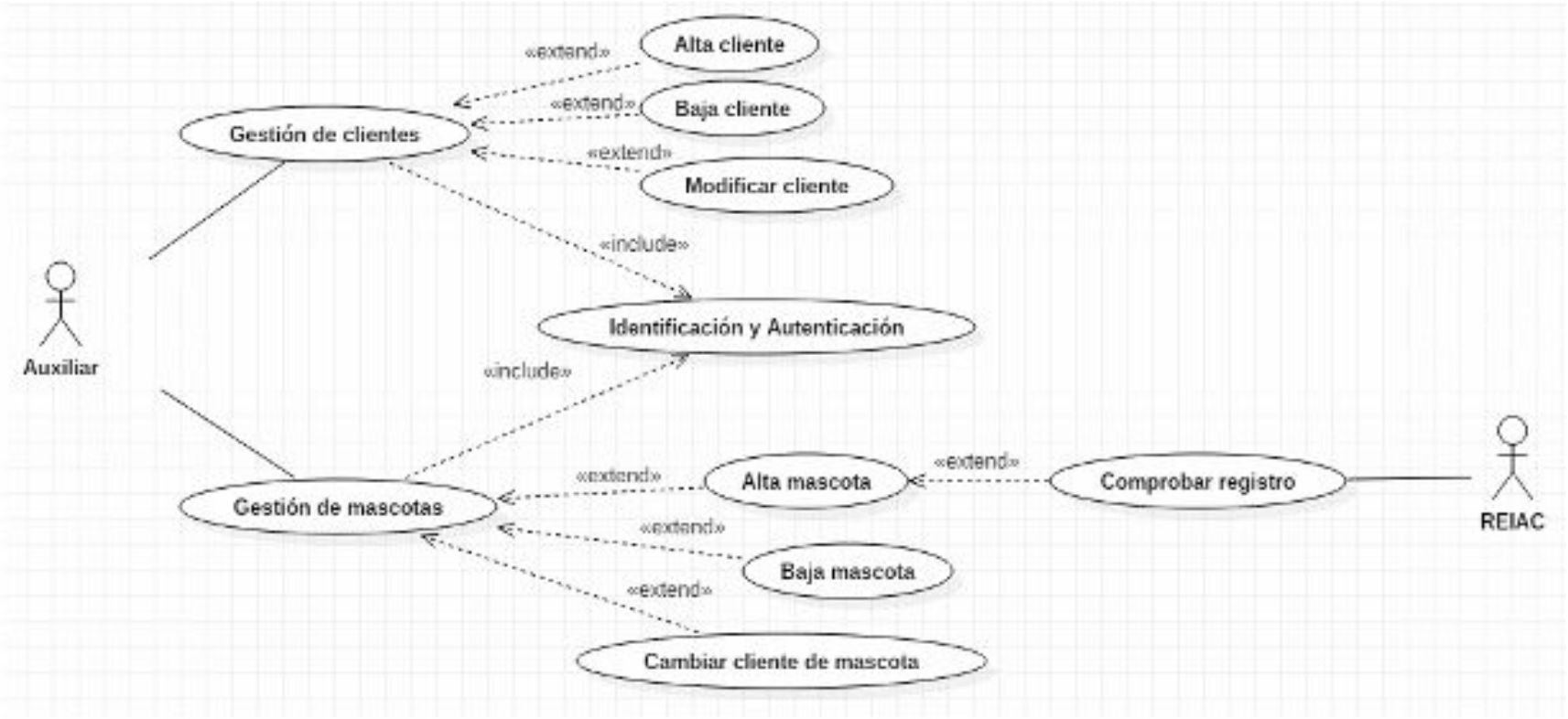
  - Cada vez que un veterinario realiza una consulta sobre un animal, esta queda almacenada incluyendo datos básicos como: Tiempo de consulta, Identificación de la persona que lo ha tratado, Animal tratado, Importe total, Resolución, Recetas… Para calcular el tiempo de la consulta el veterinario tendrá un botón en la aplicación donde pueda pulsar cuando comienza la consulta para calcular el tiempo a modo de cronómetro y otro botón para finalizar.

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**
  - En caso de que el animal se quede ingresado en la clínica, el cliente debe ser capaz de acceder al estado en tiempo real del animal. Además podrá comunicarse con una cámara que tendrá el animal colocada, donde podrá ver su situación actual. La gestión de estas cámaras no corresponde al sistema, sino que se utilizará una aplicación ya presente en el veterinario.
  - Las recetas y otros documentos relacionados con el servicio se incluirán en un gestor de contenidos que ya está en funcionamiento en la clínica veterinaria.
  - Una vez terminado el servicio, el cliente no tiene porque realizar inmediatamente el pago, sino que puede identificarse posteriormente en la aplicación vía web y realizar el pago. Si el cliente tarda más de una semana se efectuará un recargo sobre el precio inicial.
  - Además, el cliente debe ser capaz de obtener un histórico de todas las consultas que ha recibido cualquiera de sus mascotas.

2.2. Diagramas de comportamiento

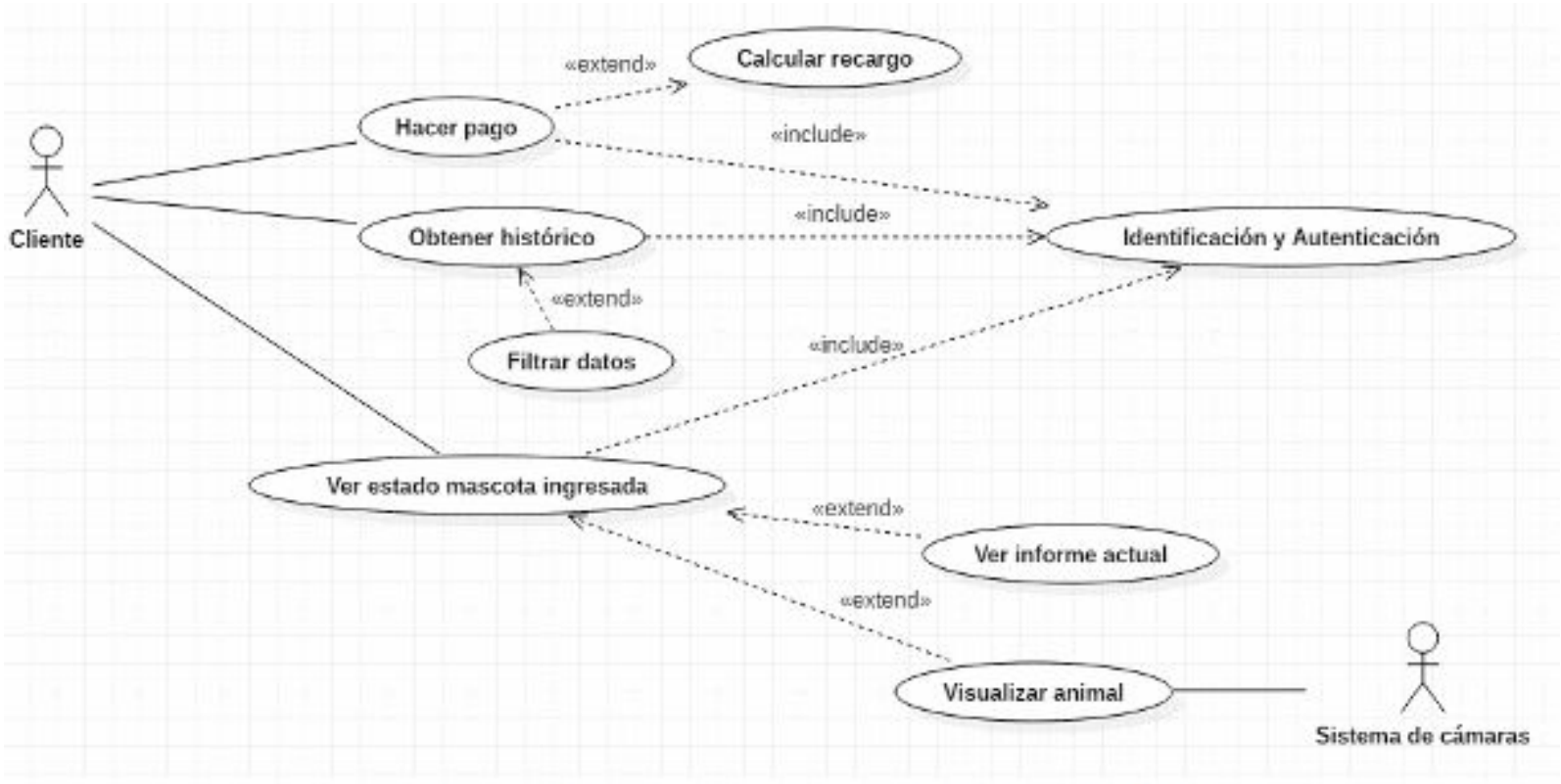
- **Diagrama de casos de uso.**



Ejemplo Diagrama de casos de uso del actor "auxiliar"

2.2. Diagramas de comportamiento

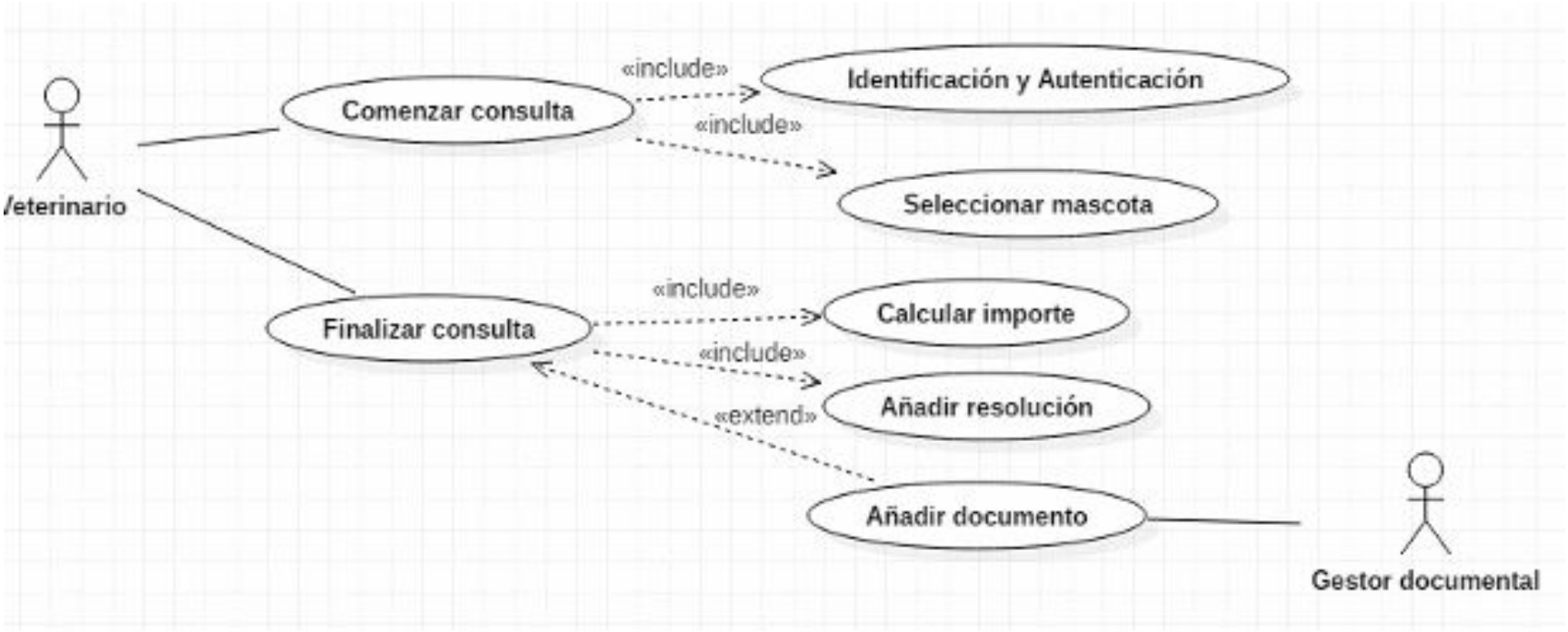
- **Diagrama de casos de uso.**



Ejemplo Diagrama de casos de uso del actor "Cliente"

2.2. Diagramas de comportamiento

- **Diagrama de casos de uso.**



Ejemplo Diagrama de casos de uso del actor "veterinario"