Definir la arquitectura

Usando las mejores prácticas

Posible orden para crear una solución

- Pasos de pre-creación
- Definir requerimientos (analistas)
- Examinar el flujo de datos o la comunicación de procesos (definir la arquitectura - esqueleto) – Arquitectos
- Codificación (programadores)
- Pruebas (testers)
- Despliegue (instalación)
- Post-Creación (Capacitación, garantías, soporte, etc…)

Modelamiento

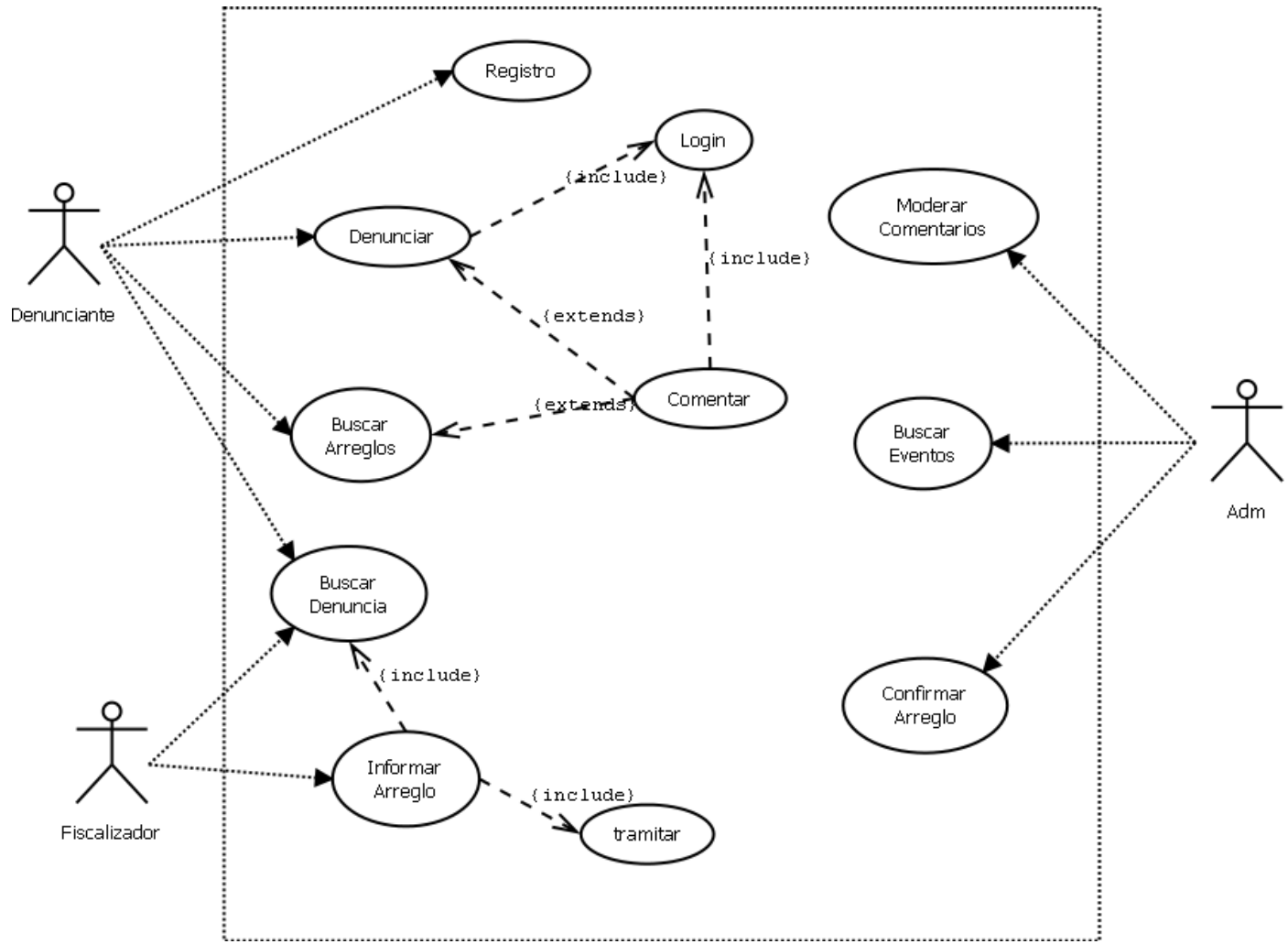
- Lenguaje para hacer modelamientos de SW (UML)
- UML (Unified Modeling Language) - Visual (Diagramas)
- Hay múltiples diagramas en el lenguaje, pero c/u expresa un punto de vista del modelamiento de sw.

Puntos de vista principales en el desarrollo de SW:

- PV Funcional (atado a los requerimientos) – Diag. Casos de uso
- PV Estructural (atado al código) – Diag. de clases
- PV Dinámico (muestra la relación entre las clases para ejecutar un caso de uso) – Diag. De secuencias

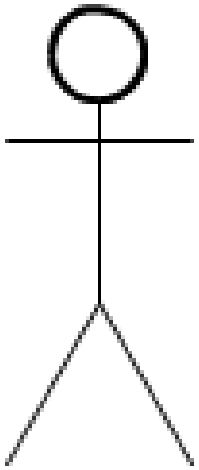
Punto de vista funcional – Casos de uso

Caso de uso	Informar arreglo	
Objetivo en el contexto	Un fiscalizador informa la reparación de un evento denunciado	
precondiciones	Debe existir un evento que esté en trámite de reparación	
Condición de éxito	Se guarda un registro de la reparación del evento para que el administrador pueda publicarla	
Condición de fracaso	El administrador no publica la reparación de un evento	
Actores primarios	Fiscalizador	
Actores secundarios	No existen	
Flujo principal	Paso	Acciones
	1	Un fiscalizador busca el evento para el cual desea informar su arreglo
	2	Agregar una descripción del evento reparado
	3	Adjuntar una fotografía del evento reparado.
	4	Registrar el informe de reparación
extensiones	Paso	Acciones
	1.1	El fiscalizador no encuentra el evento
	3.1	No se puede adjuntar una imagen
	3.2	No se puede informar la reparación del evento
	4.1	El informe no puede ser registrado



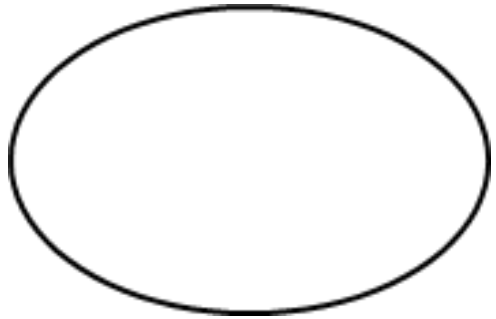
Elementos - actor

Un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.



Elementos – caso de uso

Es una operación o tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.



Elementos – relaciones

Nombre	Descripción	Símbolo
Asociación	(Comunica) Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso)	una flecha simple o una línea
Incluye	Un caso de uso contiene un comportamiento común para más de un caso de uso.	una flecha punteada que apunta al caso de uso común
Extiende	Un caso de uso distinto que maneja las excepciones u opciones no comunes al caso de uso básico	Una flecha punteada que apunta del caso de uso extendido al básico y colocando encima <<extends>>
Generaliza	Una “cosa” de UML es más general que otra “cosa”.	Una flecha continua de punta cerrada que apunta a la “cosa” general.

Punto de vista estructural – Diagrama de clases

- Describen la estructura estática de un sistema (PV estructural)

- Una clase es una categoría o grupo de cosas que tienen atributos (propiedades) y acciones similares (comportamientos).

Diagrama de clases - Clases

Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas: la superior contiene el nombre de la clase, la central contiene los atributos y la inferior las acciones.

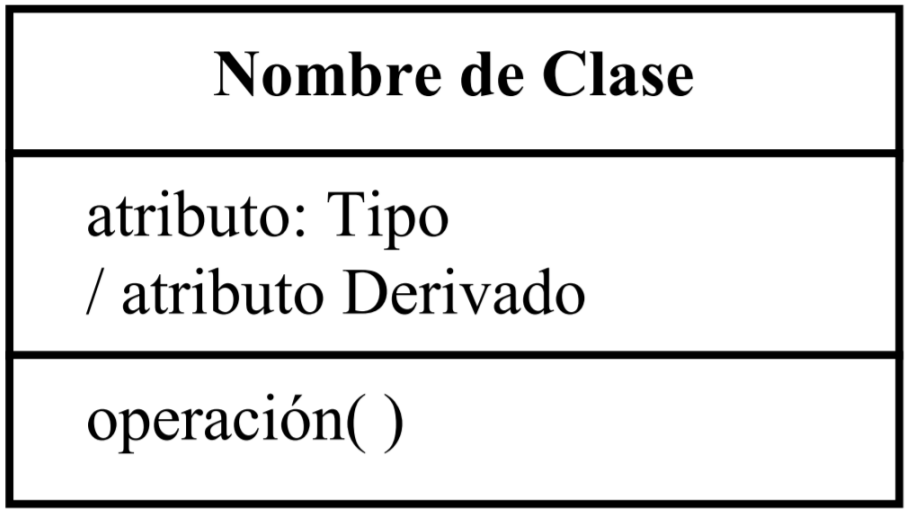


Diagrama de clases - Relaciones

Cuando diferentes objetos trabajan en función a una meta o propósito y se logra por medio de la colaboración entre las partes a través de las relaciones (conexión entre elementos).

Existen 4 relaciones en los diagramas de clases: dependencia, asociación, generalización y realización /implementación.

Diagrama de clases – dependencia

Es una relación de **uso** entre dos clases (una usa a la otra). En la práctica este tipo de relación se interpreta como que la *ClaseA* hace uso de la *ClaseB,* recibiéndola como parámetro de entrada en uno de sus métodos. Esta relación es la más básica de todas y de acoplamiento más débil. Ej:

- Conductor – Automóvil
- Escritor – Esfero
- Persona - Ropa

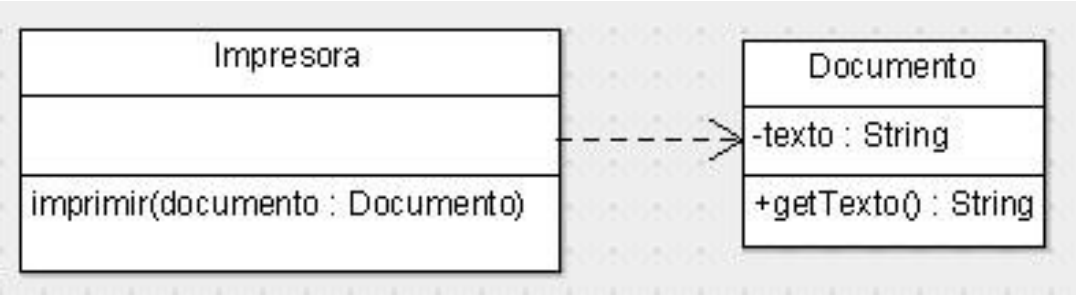


Diagrama de clases - asociación

Es una relación **estructural** que describe una conexión entre el todo y sus partes, donde se representa una relación del tipo “Tienen”. Ej:

- Ave – Pluma, Avión – Ala, Esfero – Tinta, Persona – Cabeza, Habitación – Puerta, Automóvil – Radio, Empresa – Empleado.

Multiplicidad: indica el número de instancias de una clase vinculadas a una de las instancias de la otra clase.

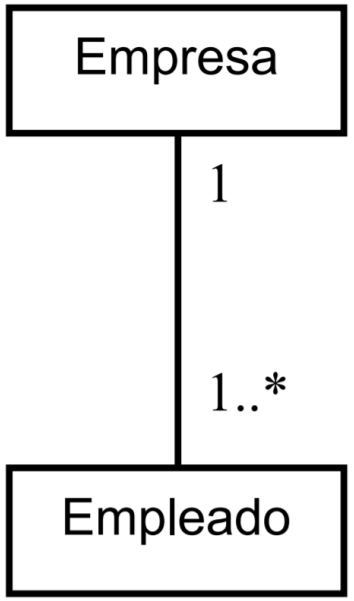


Diagrama de clases – composición

Es una relación de tipo **asociación** en donde la conexión denota una fuerte posesión de la Clase “Todo”, a la Clase “Parte”. Se grafica con un rombo relleno contra la clase que representa el todo. Ej:

- Avión – Ala
- Persona – Cabeza
- Habitación – Puerta

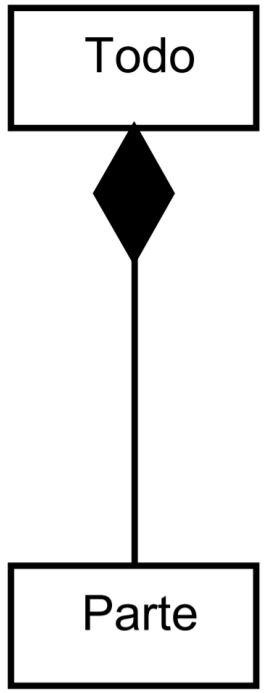


Diagrama de clases – agregación

Es una relación de tipo **asociación** en donde la conexión entre la Clase “Todo” juega un rol más importante que la Clase "Parte", pero las dos clases no son dependientes una de otra. Se grafica con un rombo vacío contra la Clase “Todo”. Ej:

- Automóvil – Radio
- Esfero – Tapa
- CPU – Unidad de DVD

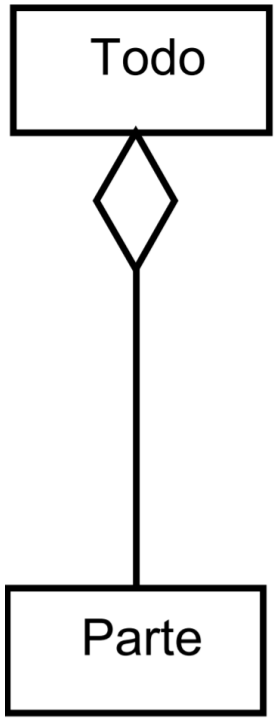


Diagrama de clases – generalización

Esta hace referencia a la relación “es-un”: de una súper clase o clase padre con una subclase o clase hija.

La generalización significa que los objetos hijos se pueden emplear en cualquier clase donde pueda aparecer el padre, pero no a la inversa. Es decir, el hijo puede sustituir al padre, pero el padre no puede sustituir al hijo.

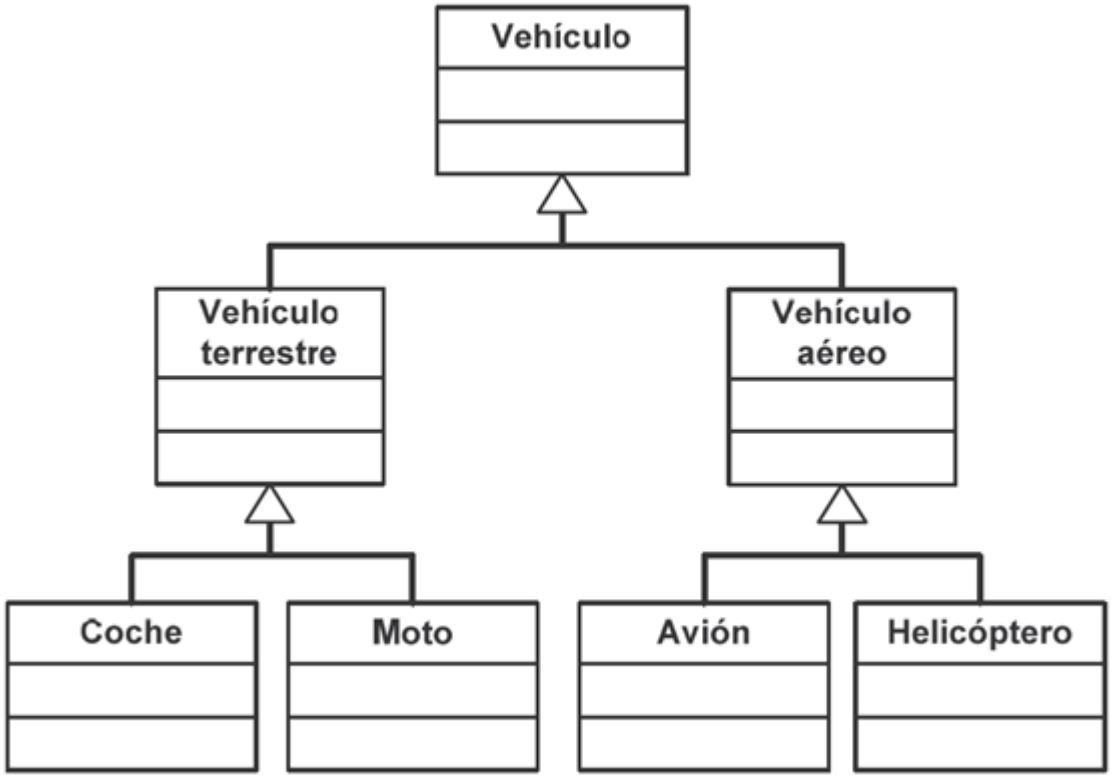
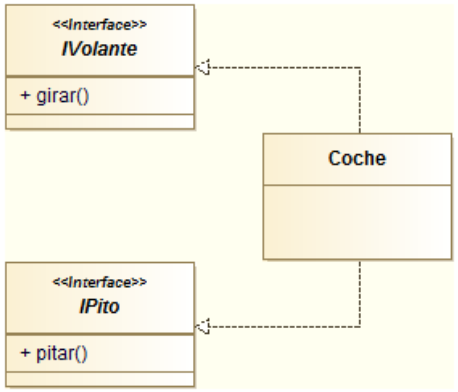
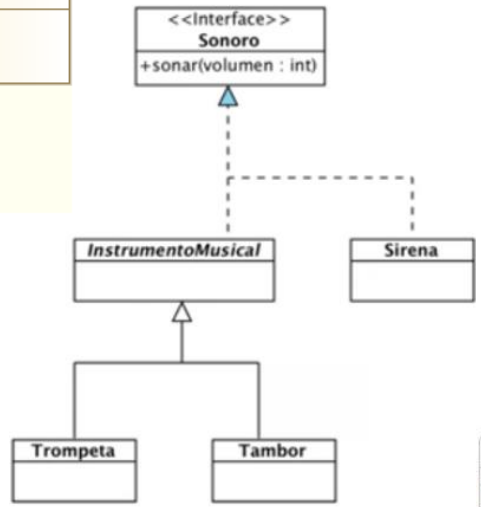
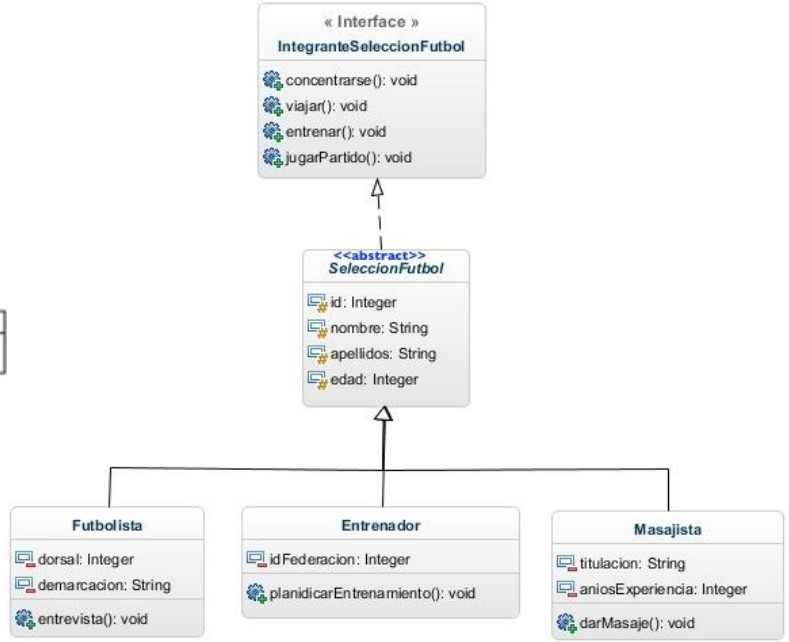


Diagrama de clases – realización/implementación

Es una relación semántica entre un clasificador (clases abstractas o interfaces) donde éste especifica unas normas o un reglamento a otro clasificador que garantiza que se cumplirá.







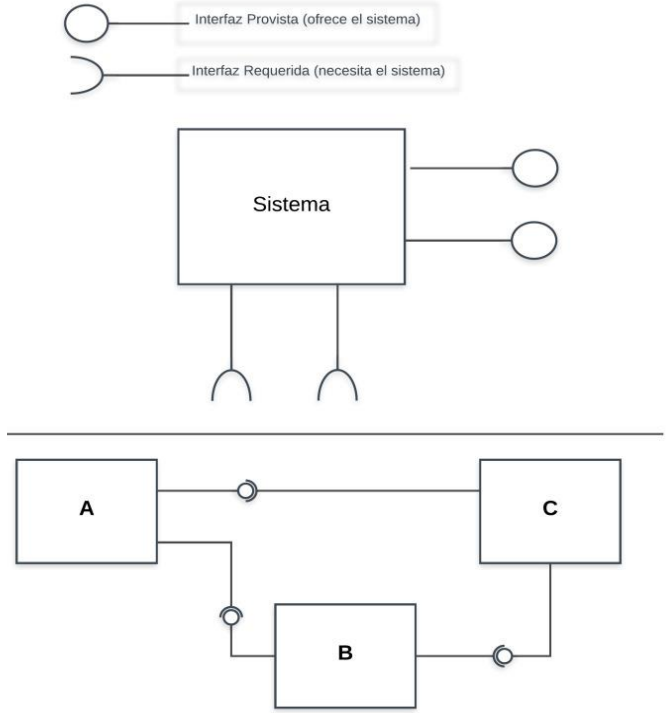
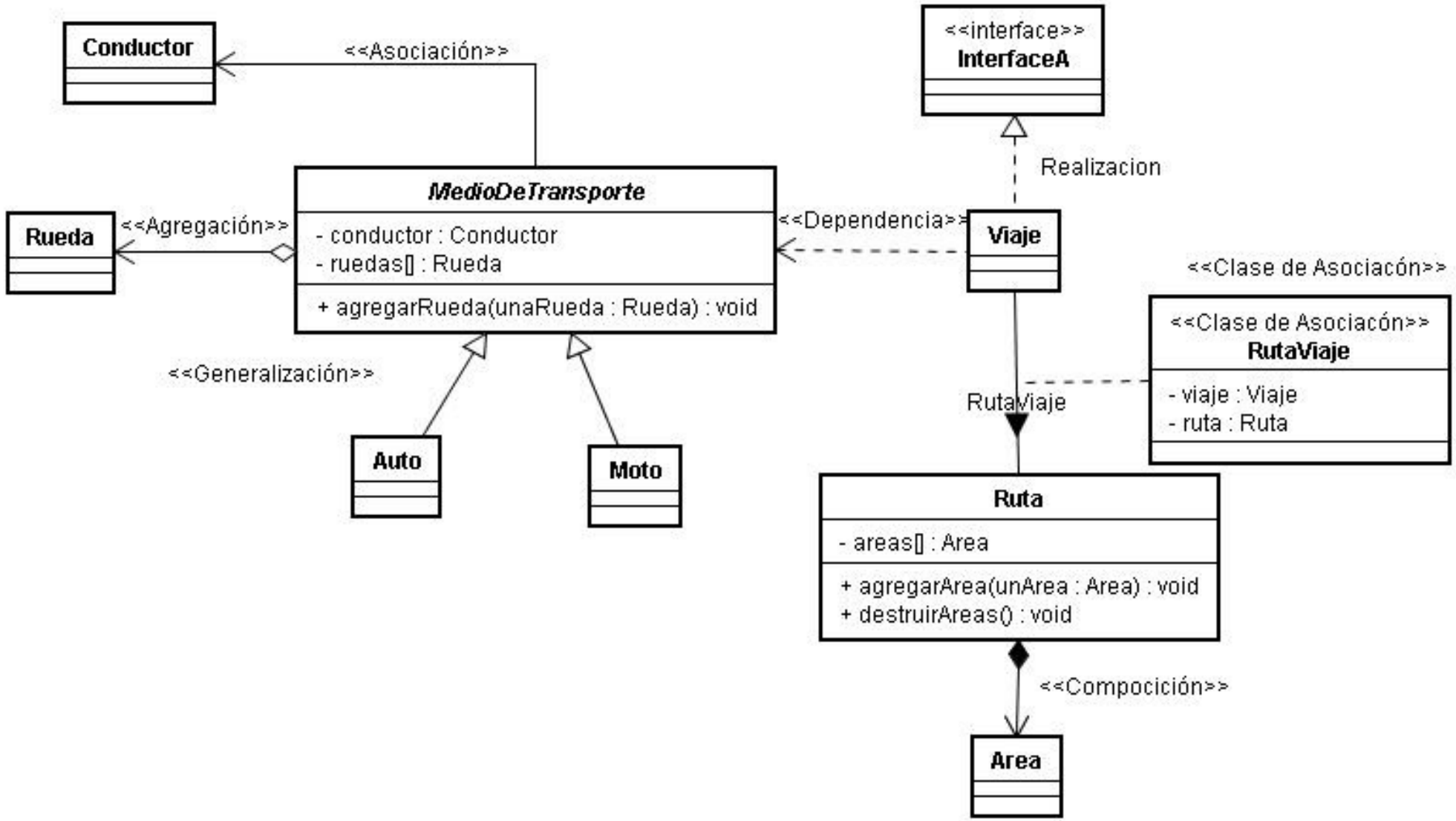
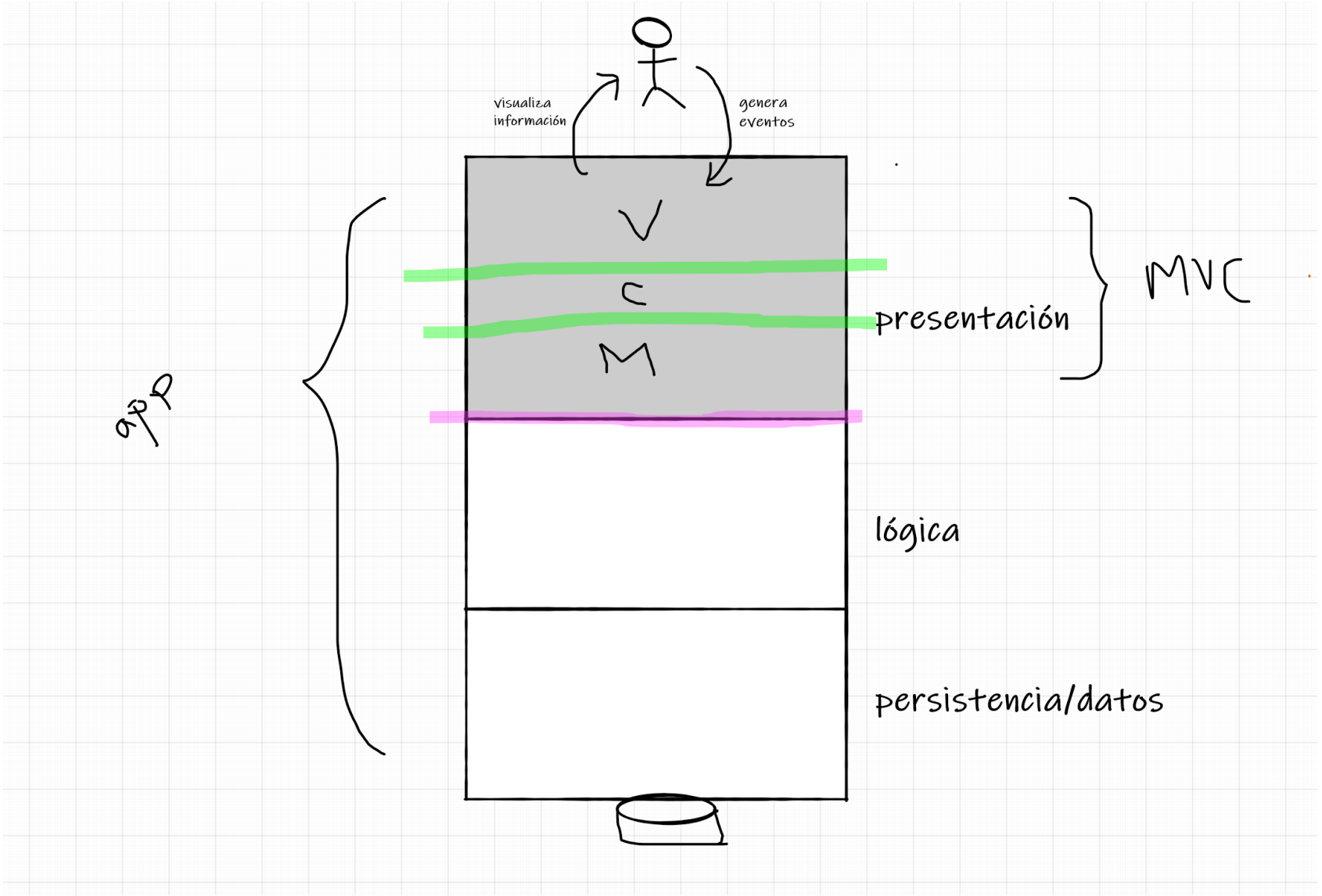


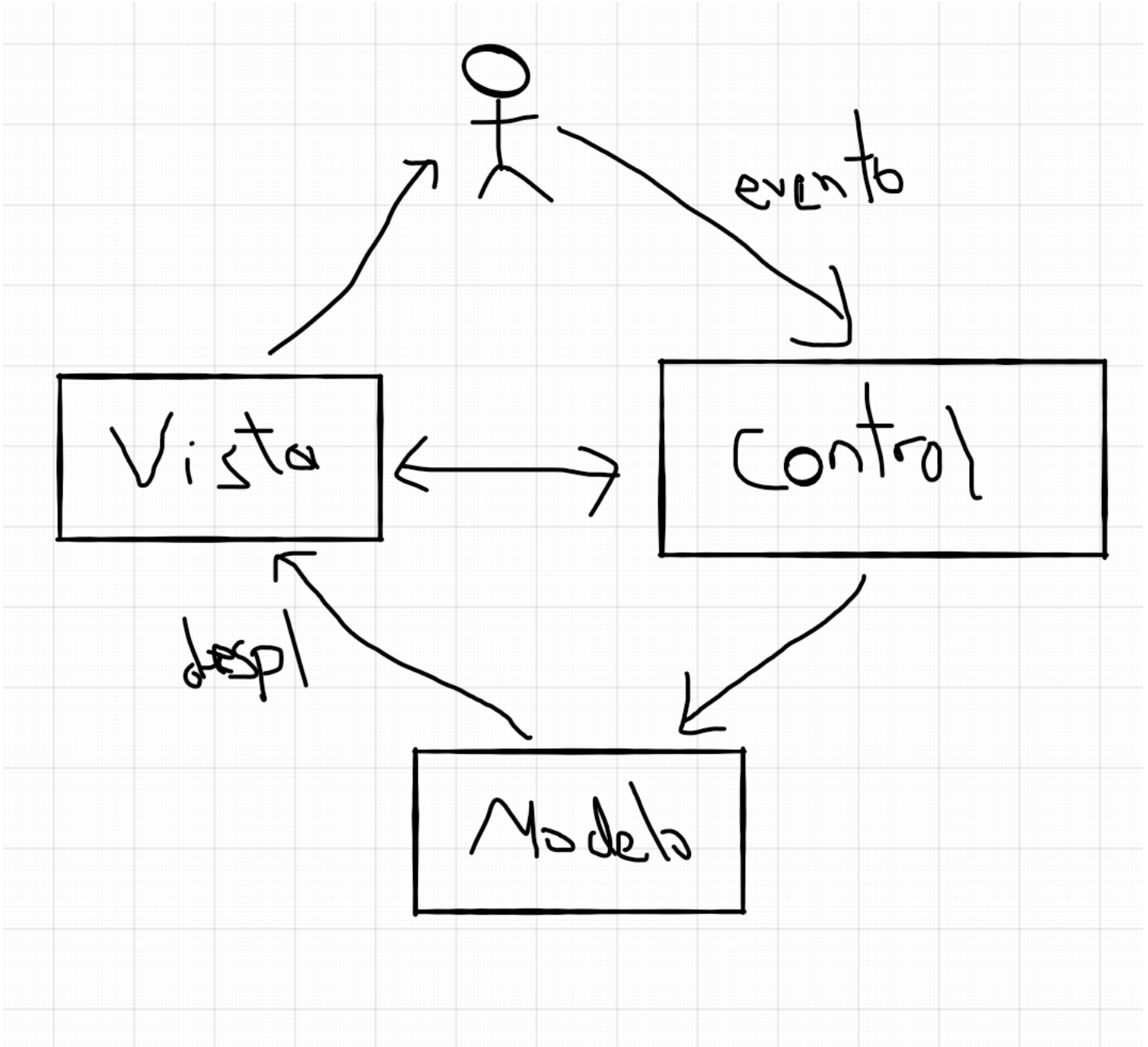
Diagrama de clases - Relaciones



Arquitectura de 3 capas



MVC



Estructura de app con MVC