

## Trabajo Práctico 5

### Relaciones UML 1 a 1

**Alumno:** Airalde Milagros Abril

**Comisión:** 1

**Fecha:** 26/9/25

**GitHub:** <https://github.com/MilagrosAi/Programacion2.git>

#### **OBJETIVO GENERAL**

Modelar clases con relaciones 1 a 1 utilizando diagramas UML. Identificar correctamente el tipo de relación (asociación, agregación, composición, dependencia) y su dirección, y llevarlas a implementación en Java.

1) Pasaporte - Foto - Titular

a. Composición: Pasaporte → Foto

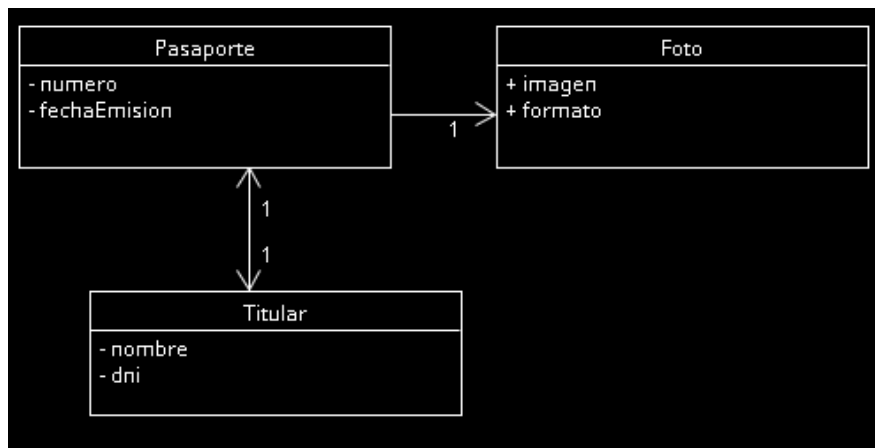
b. Asociación bidireccional: Pasaporte ↔ Titular

Clases y atributos:

i. Pasaporte: numero, fechaEmision

ii. Foto: imagen, formato

iii. Titular: nombre, dni



2) Celular - Batería - Usuario

a. Agregación: Celular → Batería

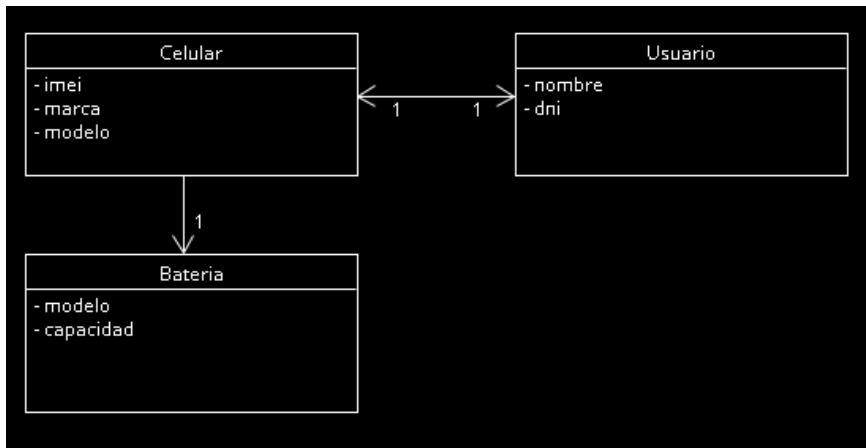
b. Asociación bidireccional: Celular ↔ Usuario

Clases y atributos:

i. Celular: imei, marca, modelo

ii. Batería: modelo, capacidad

iii. Usuario: nombre, dni



### 3) Libro - Autor - Editorial

a. Asociación unidireccional: Libro → Autor

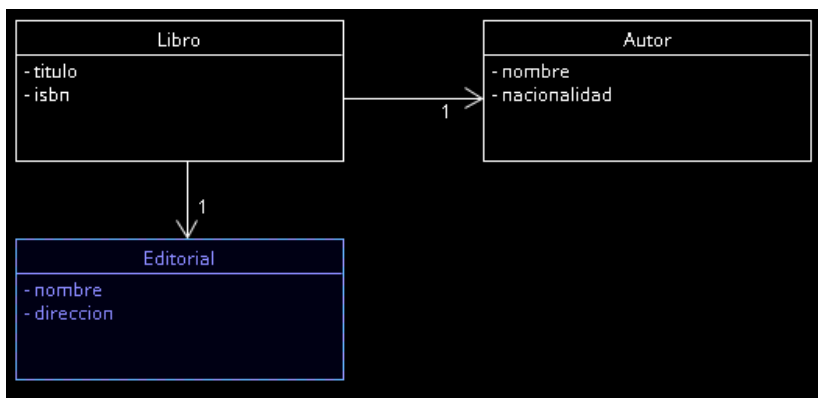
b. Agregación: Libro → Editorial

Clases y atributos:

i. Libro: titulo, isbn

ii. Autor: nombre, nacionalidad

iii. Editorial: nombre, direccion



### 4) TarjetaDeCrédito - Cliente - Banco

a. Asociación bidireccional: TarjetaDeCrédito ↔ Cliente

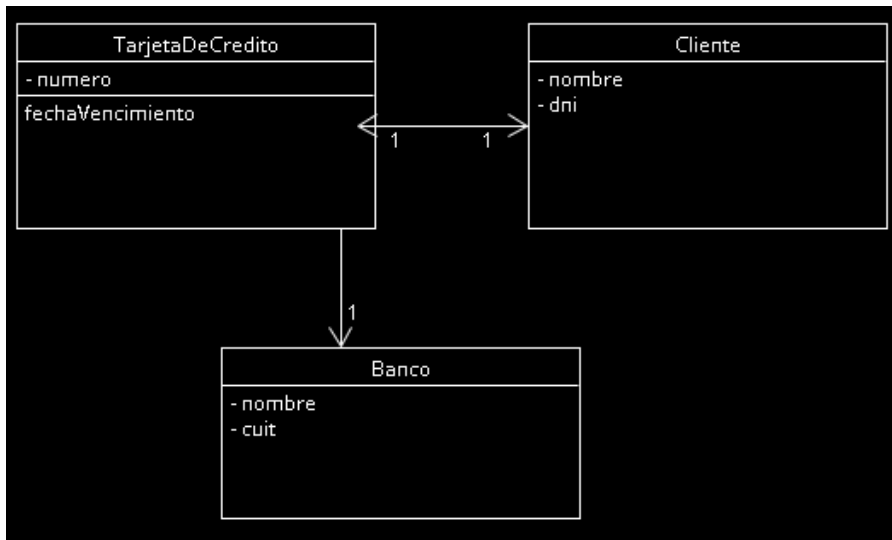
b. Agregación: TarjetaDeCrédito → Banco

Clases y atributos:

i. TarjetaDeCrédito: numero, fechaVencimiento

ii. Cliente: nombre, dni

iii. Banco: nombre, cuit



##### 5) Computadora - PlacaMadre - Propietario

a. Composición: Computadora → PlacaMadre

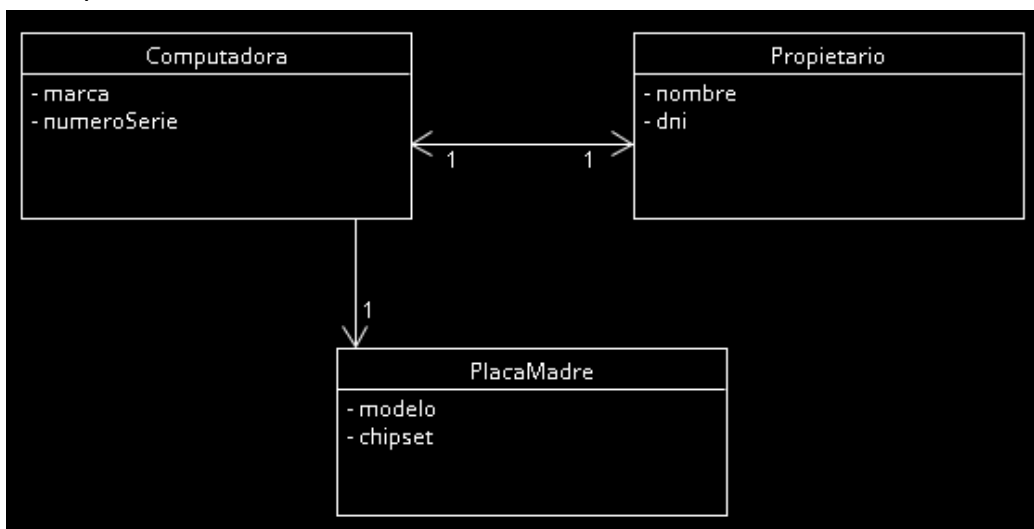
b. Asociación bidireccional: Computadora ↔ Propietario

Clases y atributos:

i. Computadora: marca, numeroSerie

ii. PlacaMadre: modelo, chipset

iii. Propietario: nombre, dni



##### 6) Reserva - Cliente - Mesa

a. Asociación unidireccional: Reserva → Cliente

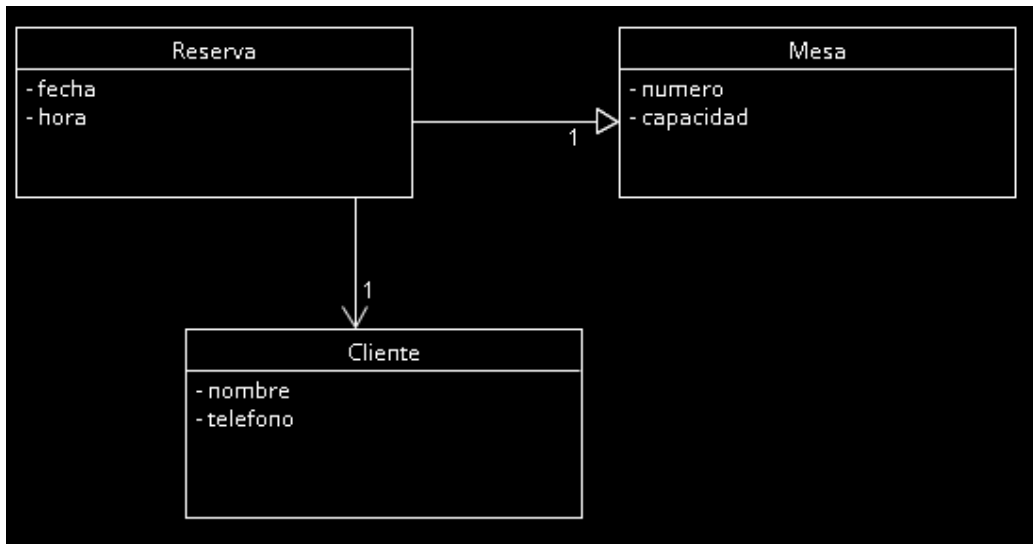
b. Agregación: Reserva → Mesa

Clases y atributos:

i. Reserva: fecha, hora

ii. Cliente: nombre, telefono

iii. Mesa: numero, capacidad



## 7) Vehículo - Motor - Conductor

a. Agregación: Vehículo → Motor

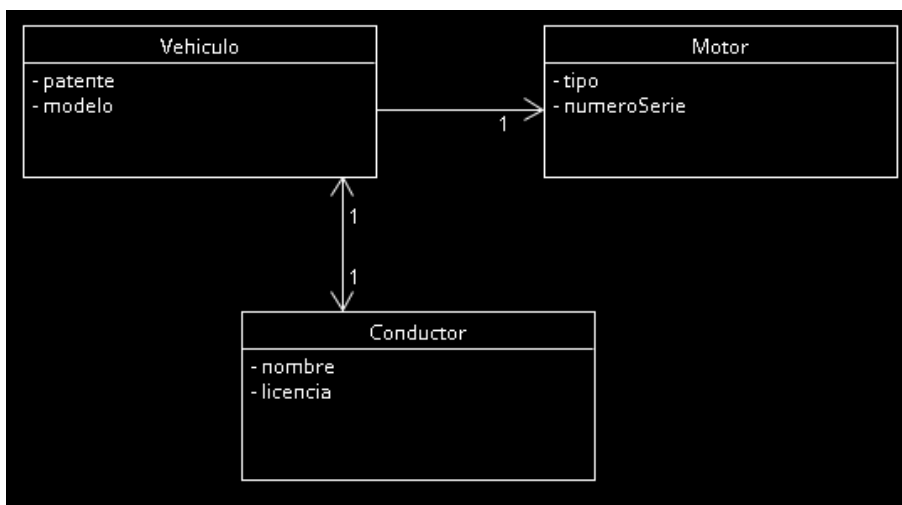
b. Asociación bidireccional: Vehículo ↔ Conductor

Clases y atributos:

i. Vehículo: patente, modelo

ii. Motor: tipo, numeroSerie

iii. Conductor: nombre, licencia



## 8) Documento - FirmaDigital - Usuario

a. Composición: Documento → FirmaDigital

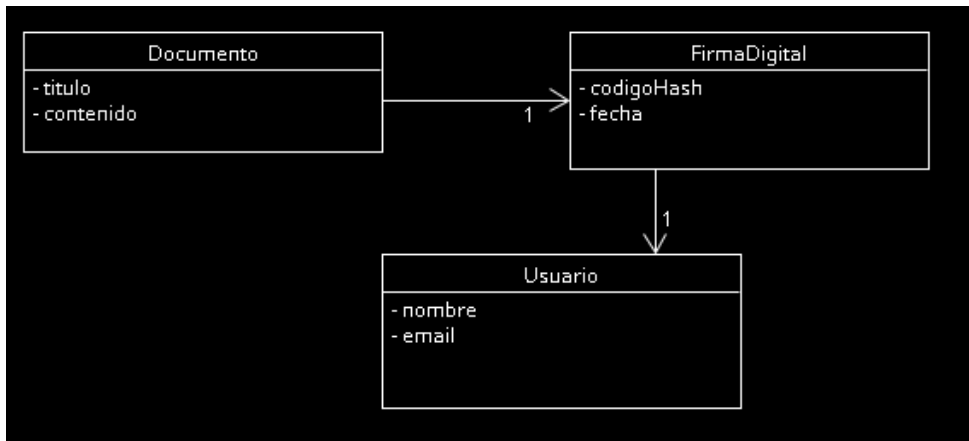
b. Agregación: FirmaDigital → Usuario

Clases y atributos:

i. Documento: titulo, contenido

ii. FirmaDigital: codigoHash, fecha

iii. Usuario: nombre, email



9)

CitaMédica - Paciente - Profesional

a. Asociación unidireccional: CitaMédica → Paciente,

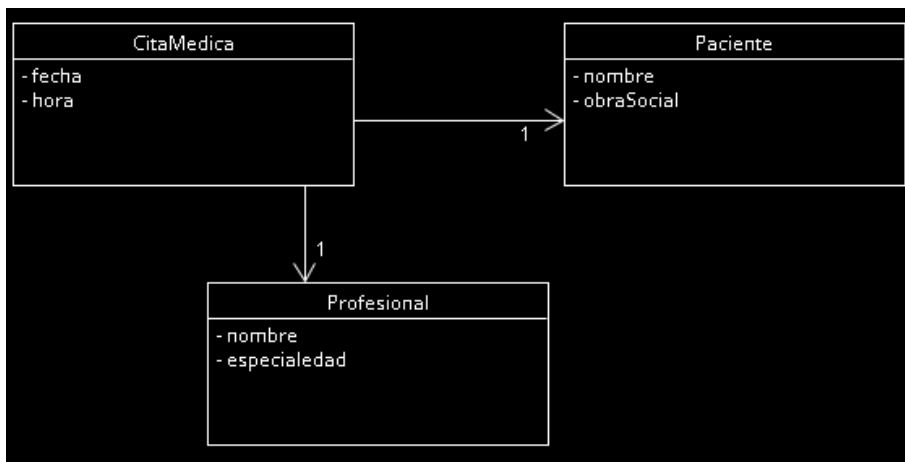
b. Asociación unidireccional: CitaMédica → Profesional

Clases y atributos:

i. CitaMédica: fecha, hora

ii. Paciente: nombre, obraSocial

iii. Profesional: nombre, especialidad



10)

CuentaBancaria - ClaveSeguridad - Titular

a. Composición: CuentaBancaria → ClaveSeguridad

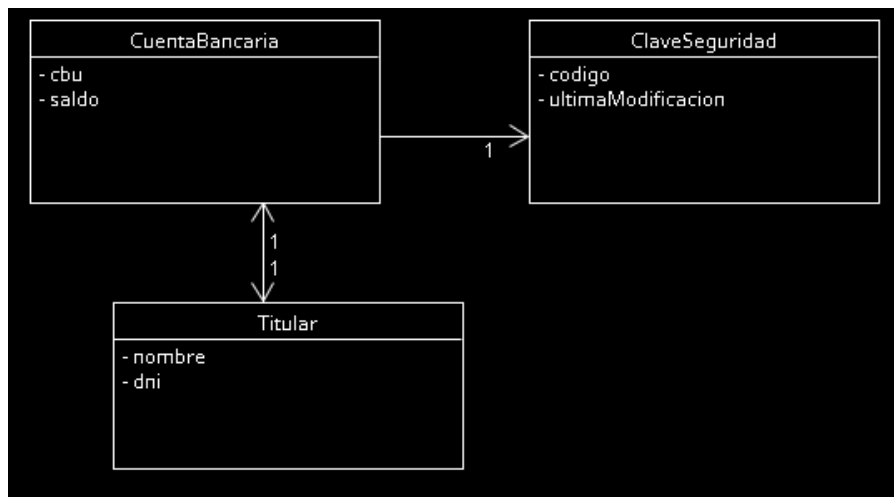
b. Asociación bidireccional: CuentaBancaria ↔ Titular

Clases y atributos:

i. CuentaBancaria: cbu, saldo

ii. ClaveSeguridad: codigo,

iii. Titular: nombre, dni.



### 11) Reproductor - Canción - Artista

a. Asociación unidireccional: Canción → Artista

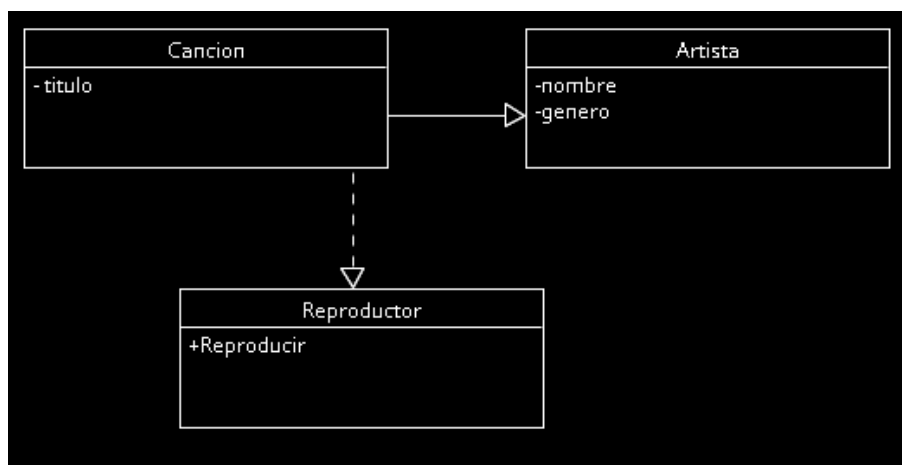
b. Dependencia de uso: Reproductor.reproducir(Cancion)

Clases y atributos:

i. Canción: titulo.

ii. Artista: nombre, genero.

iii. Reproductor->método: void reproducir(Cancion cancion)



### 12) Impuesto - Contribuyente - Calculadora

a. Asociación unidireccional: Impuesto → Contribuyente

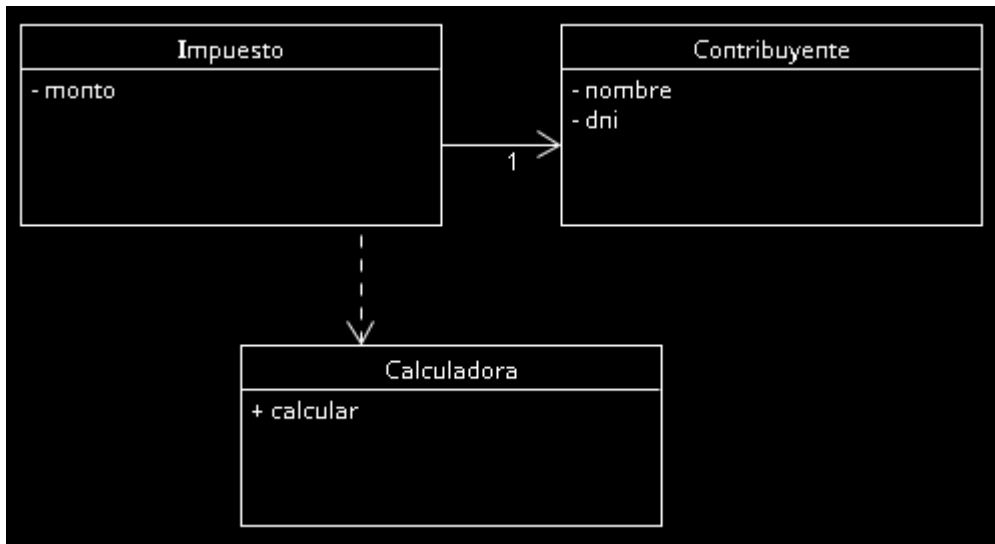
b. Dependencia de uso: Calculadora.calcular(Impuesto)

Clases y atributos:

i. Impuesto: monto.

ii. Contribuyente: nombre, cuil.

iii. Calculadora->método: void calcular(Impuesto impuesto)



### 13)13. GeneradorQR - Usuario - CódigoQR

a. Asociación unidireccional: CódigoQR → Usuario

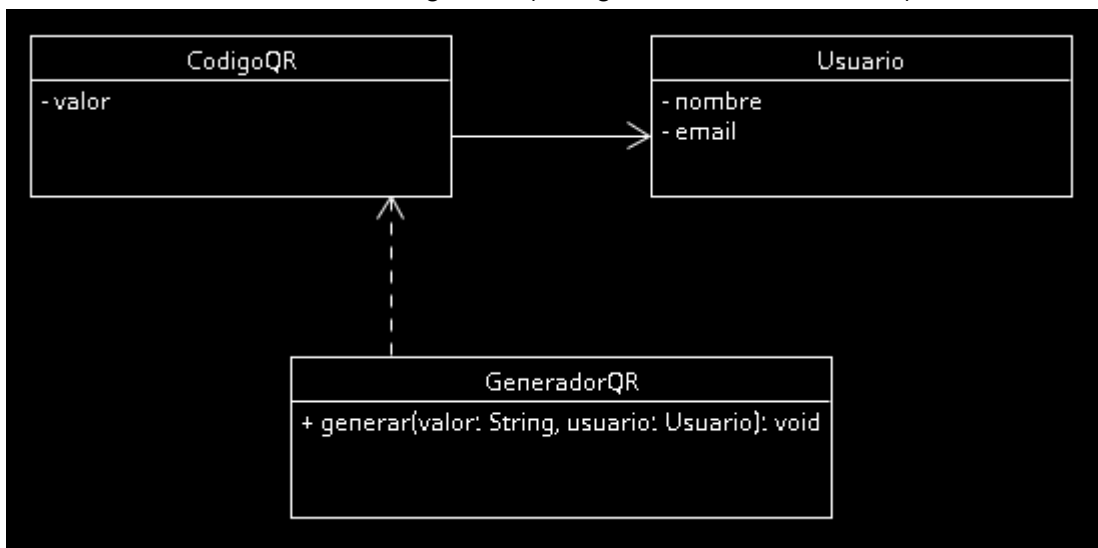
b. Dependencia de creación: `GeneradorQR.generar(String, Usuario)`

Clases y atributos:

i. CódigoQR: valor.

ii. Usuario: nombre, email.

iii. GeneradorQR->método: `void generar(String valor, Usuario usuario)`



### 14) EditorVideo - Proyecto - Render

a. Asociación unidireccional: Render → Proyecto

b. Dependencia de creación: `EditorVideo.exportar(String, Proyecto)` c.

Clases y atributos:

i. Render: formato.

ii. Proyecto: nombre, duracionMin.

iii. EditorVideo->método: `void exportar(String formato, Proyecto proyecto)`

