TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

Modalidad a Distancia

Materia: Programación II

Trabajo Práctico 5: Relaciones UML 1 a 1

Alumna: Marina Giselle Cordero

DNI: 31058096

Docente: Luciano Chiroli

UTN

Ciclo Lectivo: 2025

Comisión: 7

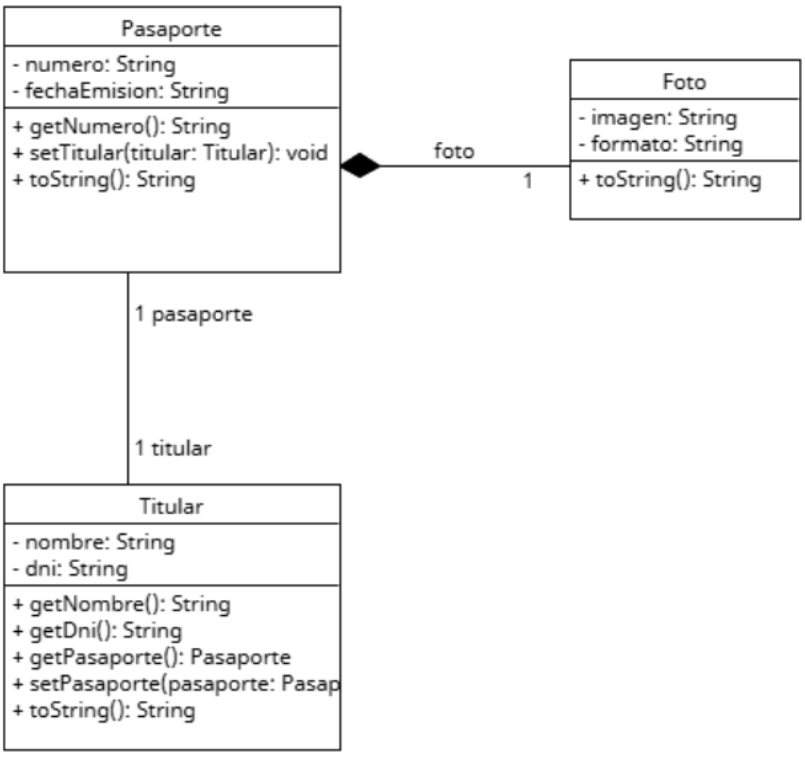
Objetivo del trabajo

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

Resolución de Ejercicios:

1. Pasaporte - Foto - Titular
    a. Composición: **Pasaporte → Foto**
    b. Asociación bidireccional: **Pasaporte ↔ Titular**

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio1/src/ejercicio1



Clase Foto:

```java
package ejercicio1;
// Clase Foto: Representa la foto asociada a un pasaporte
public class Foto {

    private String imagen;
    private String fomato;

    //Constructor
    public Foto(String imagen, String fomato) {
        this.imagen = imagen;
        this.fomato = fomato;
    }
    // Método toString para mostrar información de la foto
    @Override
    public String toString() {
        return "Foto{" + "imagen=" + imagen + ", fomato=" + fomato + '}';
    }
```

Clase Pasaporte:

```java
package ejercicio1;
public class Pasaporte {
    private String numero;
    private String fechaEmision;
    private Foto foto;
    private Titular titular;

    public Pasaporte(String numero, String fechaEmision, Foto foto) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = foto;
    }
    public String getNumero() {
        return numero;

    }
    public void setTitular(Titular titular) {
        this.titular = titular;
    }

    @Override
    public String toString() {
        String infoTitular= null;
        if( titular != null ){
            // mostramos solo nombre y dni, para eviatr recursion infinita.
            infoTitular = titular.getNombre() + ", Dni: " + titular.getDNI();
        }
        return "Pasaporte: " + numero + ", fecha de emisión: "
       + fechaEmision + " " + foto + " " + "titular: " + infoTitular;
    }
```

Clase Titular:

```java
package ejerciciol;
//Clase Titular: Representa la persona asociada al pasaporte
public class Titular {
    private String nombre;
    private String DNI;
    private Pasaporte pasaporte;// Relación bidireccional con Pasaporte

    //Constructor
    public Titular(String nombre, String DNI) {
        this.nombre = nombre;
        this.DNI = DNI;
    }
    // Getter y setter para la relación bidireccional con Pasaporte
    public Pasaporte getPasaporte() {
        return pasaporte;
    }
    // Establece la relación bidireccional con Pasaporte
    public void setPasaporte(Pasaporte pasaporte) {
        this.pasaporte = pasaporte;
    }

    public String getNombre() {
        return nombre;
    }

    public String getDNI() {
        return DNI;
    }

    // Método toString para mostrar información del titular
    // Incluye el número del pasaporte para reflejar la relación bidireccional
    @Override
    public String toString() {
        String infoPasaporte = null;
        if (pasaporte != null){
            // Mostramos solo el número del pasaporte para evitar recursión infinita
            infoPasaporte = "número de pasaporte: " + pasaporte.getNumero();
        }
        return "Titular: " + nombre + ", DNI=" + DNI + ", pasaporte: " + infoPasaporte;
    }
```

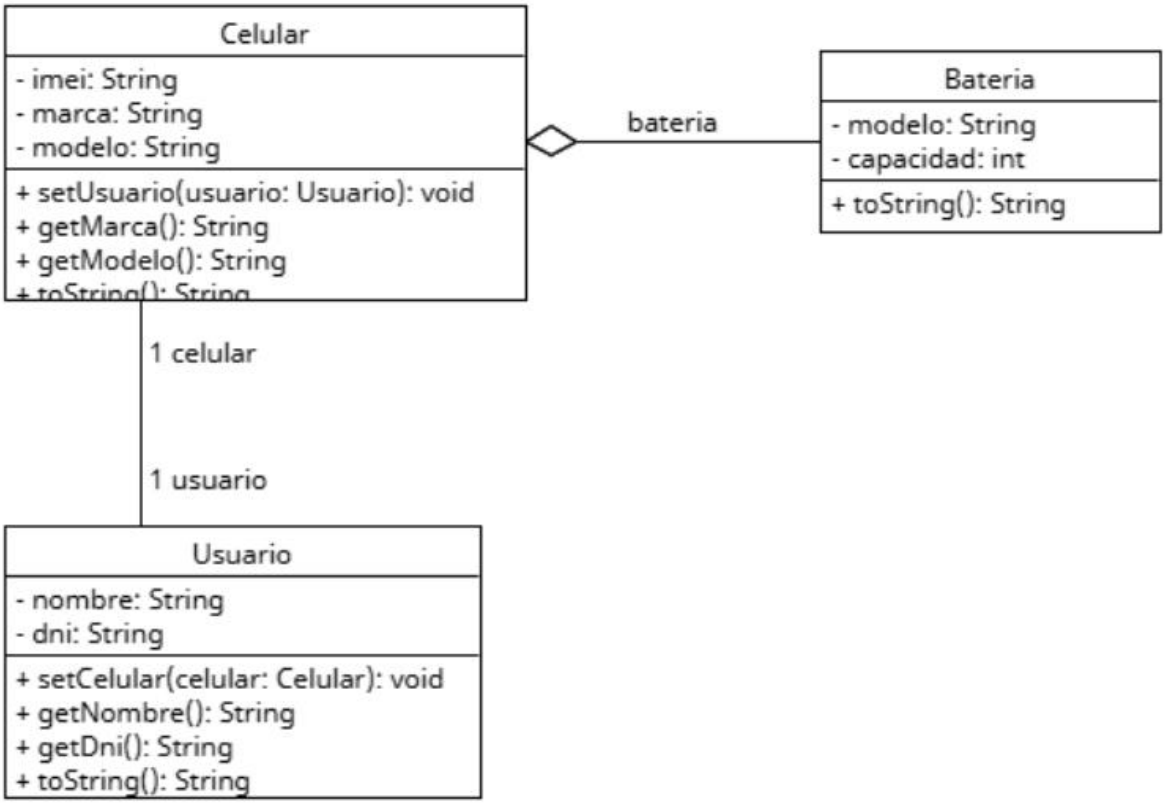
main:

```java
package ejercicio1;

public class Principal {
    public static void main(String[] args) {
        // Crear una foto
        Foto foto = new Foto(imagen: "imagen1245", fomato: "JPEG");

        // Crear un pasaporte con la foto (relación de composición)
        Pasaporte pasaporte = new Pasaporte(numero: "AB123486", fechaEmision: "12/02/2015", foto);

        // Crear un titular
        Titular titular = new Titular(nombre: "Marina Cordero", DNI: "31058096");

        // Establecer la relación bidireccional entre Pasaporte y Titular
        pasaporte.setTitular(titular);
        titular.setPasaporte(pasaporte);

        // Mostrar información usando toString
        System.out.println(x: pasaporte);
        System.out.println(x: titular);
        System.out.println(x: foto);
    }
```

```
run:
Pasaporte: AB123486, fecha de emisión: 12/02/2015 Foto{imagen=imagen1245, fomato=JPEG} titular: Marina Cordero, Dni: 31058096
Titular: Marina Cordero, DNI=31058096, pasaporte: número de pasaporte: AB123486
Foto{imagen=imagen1245, fomato=JPEG}
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Celular - Batería - Usuario
    a. Agregación: **Celular → Batería**
    b. Asociación bidireccional: **Celular ↔ Usuario**

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio2-UML/src/ejercicio2/uml



Clase Usuario:

```java
package ejercicio2.uml;
public class Usuario {

    private String nombre;
    private String DNI;
    private Celular celular;// Relación bidireccional con Celular

    public Usuario(String nombre, String DNI) {
        this.nombre = nombre;
        this.DNI = DNI;
    }

    public void setCelular(Celular celular) {
        this.celular = celular;
    }

    public String getNombre() {
        return nombre;
    }

    public String getDNI() {
        return DNI;
    }
    // Método toString para mostrar información del usuario
    // Incluye solo el modelo y marca del celular para reflejar la relación bidireccional
    // Evita recursión infinita
    @Override
    public String toString() {
        String infoCelular;
        if(celular != null){
            infoCelular = "marca: " +  celular.getMarca() + ", modelo: " + celular.getModelo();
        }
        else{
            infoCelular=null;
        }
        return "Usuario{" + "nombre=" + nombre + ", DNI=" + DNI + ", Celular: " + infoCelular + '}';
```

Clase Celular:

```java
package ejercicio2.uml;
public class Celular {

    private String marca;
    private String modelo;
    private String imei;
    private Bateria bateria;//Relación de agregación con Bateria
    private Usuario usuario;//Relación bidireccional con Usuario

    // El constructor incluye la batería como parámetro, reflejando la agregación
    public Celular(String marca, String modelo, String imei, Bateria bateria) {
        this.marca = marca;
        this.modelo = modelo;
        this.imei = imei;
        this.bateria = bateria;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }

    public String getMarca() {
        return marca;
    }

    public String getModelo() {
        return modelo;
    }
    @Override
    public String toString() {
        String infoUsuario;
        if (usuario != null){
            infoUsuario = "usuario: " + usuario.getNombre() + ", Dni: " + usuario.getDNI();
        }
        else{
            infoUsuario = null;
        }
        return "Celular{" + "marca=" + marca + ", modelo=" + modelo + ", imei="
```

Clase Bateria:

```java
package ejercicio2.uml;
public class Bateria {

    private String modelo;
    private int capacidad;

    public Bateria(String modelo, int capacidad) {
        this.modelo = modelo;
        this.capacidad = capacidad;
    }

    @Override
    public String toString() {
        return "Bateria{" + "modelo=" + modelo + ", capacidad=" + capacidad + '}';
    }
```

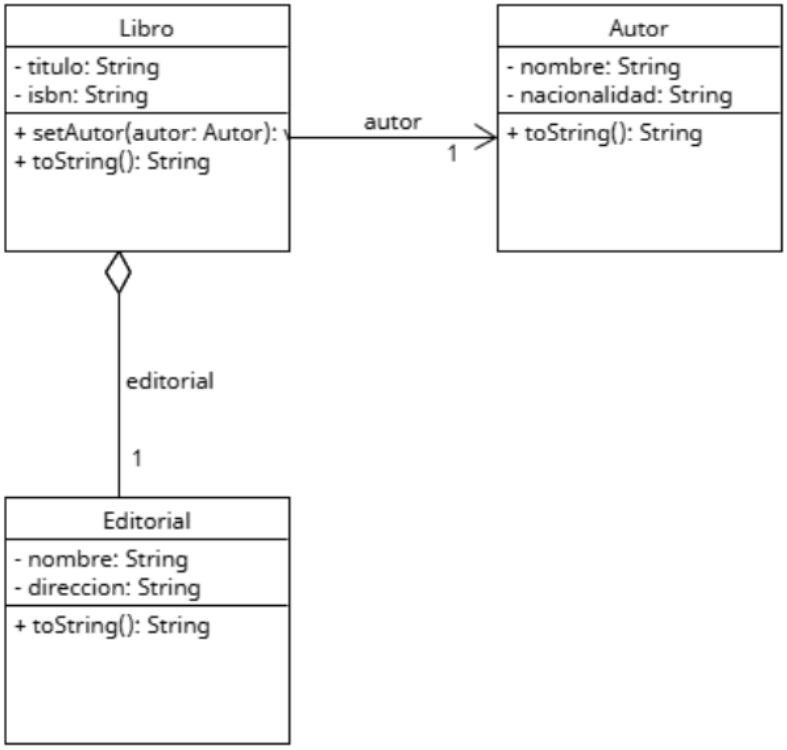
Main:

```java
package ejercicio2.uml;
public class Principal {
    public static void main(String[] args) {

        //Creamos una batería
        Bateria bateria = new Bateria(modelo:"BT-001", capacidad: 5000);

        //Creamos el celular con la batería (AGREGACION)
        Celular celular = new Celular(marca: "Samsung",modelo:"A 4",
                imei: "12345323454", bateria);

        //Creamos un usuario
        Usuario usuario = new Usuario(nombre:"Marina Cordero", DNI: "31058096");

        //Asociamos un usuario al celular
        celular.setUsuario(usuario);
        //Asociamos el celular al usuario para que la relación sea bidireccional.
        usuario.setCelular(celular);

        System.out.println(x: bateria);
        System.out.println(x: celular);
        System.out.println(x: usuario);
```

```
run:
Bateria{modelo=BT-001, capacidad=5000}
Celular{marca=Samsung, modelo=A 4, imei=12345323454, Bateria{modelo=BT-001, capacidad=5000}, usuario: Marina Cordero, Dni: 31058096}
Usuario{nombre=Marina Cordero, DNI=31058096, Celular: marca: Samsung, modelo: A 4}
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Libro - Autor - Editorial
    a. Asociación unidireccional: **Libro → Autor**
    b. Agregación: **Libro → Editorial**

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio3-UML/src/ejercicio3/uml



Clase Editorial:

```java
package ejercicio3.uml;

public class Editorial {
    private String nombre;
    private String direccion;

    public Editorial(String nombre, String direccion) {
        this.nombre = nombre;
        this.direccion = direccion;
    }

    @Override
    public String toString() {
        return "Editorial{" + "nombre=" + nombre + ", direccion=" + direccion + '}';
    }
```

Clase Libro:

```java
public class Libro {
    private String titulo;
    private String isbn;
    private Autor autor; // Asociación unidireccional con Autor (atributo en UML:
    private Editorial editorial; // Relación de agregación con Editorial.

    public Libro(String titulo, String isbn, Editorial editorial) {
        this.titulo = titulo;
        this.isbn = isbn;
        this.editorial = editorial;
    }

    public void setAutor(Autor autor) {
        this.autor = autor;
    }

    @Override
    public String toString() {
        return "Libro{" + "titulo: " + titulo + ", isbn=" + isbn
                + ", " + autor + ", " + editorial + '}';
    }
```

Clase Autor:

```java
package ejercicio3.uml;

public class Autor {
    private String nombre;
    private String nacionalidad;

    public Autor(String nombre, String nacionalidad) {
        this.nombre = nombre;
        this.nacionalidad = nacionalidad;
    }

    @Override
    public String toString() {
        return "Autor{" + "nombre=" + nombre + ", nacionalidad=" + nacionalidad + '}';
    }
```

Main:

```java
public class Ejercicio3UML {
    public static void main(String[] args) {
        // Crear una editorial
        Editorial editorial = new Editorial(nombre: "Bloomsbury", direccion:"Calle Falsa 123, Cordoba");

        // Crear un autor
        Autor autor = new Autor(nombre: "J. K. Rowling", nacionalidad: "británica");

        // Crear un libro con el autor y la editorial
        // Refleja la asociación unidireccional [Libro] --> [Autor] y la agregación [Libro] o--> [Editorial]
        Libro libro = new Libro(titulo: "Harry Poter", isbn:"456464845646", editorial);

        libro.setAutor(autor);// Asociacion a través de setter

        // Mostrar información usando toString
        System.out.println("Libro: " + libro);
        System.out.println("Autor: " + autor);
        System.out.println("Editorial: " + editorial);
    }
```

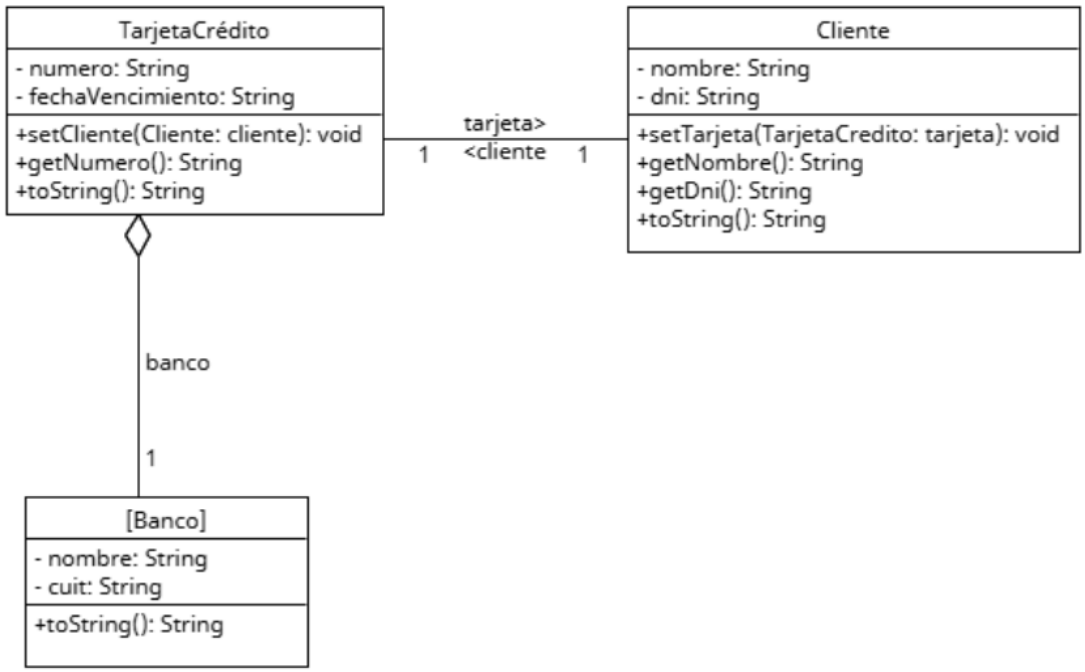
```
run:
Libro: Libro{titulo: Harry Poter, isbn=456464845646, Autor{nombre=J. K. Rowling, nacionalidad=británica}, Editorial{nombre=Bloomsbury, direccion=Calle Falsa 123, Cordoba}}
Autor: Autor{nombre=J. K. Rowling, nacionalidad=británica}
Editorial: Editorial{nombre=Bloomsbury, direccion=Calle Falsa 123, Cordoba}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejericio4-UML/src/ejericio4/uml



Main:

```java
package ejericio4.uml;
public class Principal {
    public static void main(String[] args) {
        // Crear un banco
        Banco banco = new Banco(nombre:"Santander", cuit: "30-12345678-9");

        // Crear un cliente
        Cliente cliente = new Cliente(nombre:"MArina Cordero", dni: "31058096");

        //Crear Tarjeta de  crédito
        TarjetaCredito tarjeta = new TarjetaCredito(nuemero: "4455555659788545", fechaVencimiento:"22/05/2028", banco);

        // Establecer la relación bidireccional entre TarjetaDeCrédito y Cliente usando setters
        tarjeta.setCliente(cliente);
        cliente.setTarjeta(tarjeta);

        // Mostrar información usando toString
        System.out.println( x: tarjeta);
        System.out.println( x: cliente);
        System.out.println( x: banco);
```

```
- Ejericio4-UML (run) ×
run:
TarjetaCredito{nuemero=4455555659788545, fechaVencimiento=22/05/2028, Banco{nombre=Santander, cuit=30-12345678-9}, Cliente: MArina Cordero, Dni: 31058096}
Cliente{nombre=MArina Cordero, cuit=31058096, tarjeta número: 4455555659788545}
Banco{nombre=Santander, cuit=30-12345678-9}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase TarjetaCredito:

```java
package ejercicio4.uml;
public class TarjetaCredito {
    private String numero;
    private String fechaVencimiento;
    private Cliente cliente;// Relación bidireccional con Cliente
    private Banco banco;// Relación de agregación con Banco

    // Constructor de TarjetaDeCrédito
    // Incluye solo banco como parámetro, reflejando la agregación
    public TarjetaCredito(String nuemero, String fechaVencimiento, Banco banco) {
        this.numero = nuemero;
        this.fechaVencimiento = fechaVencimiento;
        this.banco = banco;
    }

    // Getter y setter para la relación bidireccional con Cliente

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }

    public String getNumero() {
        return numero;
    }

    // Método toString para mostrar información de la tarjeta.
    //Incluye solo nombre y DNI del cliente para reflejar la relación bidireccional
    // Evita recursión infinita mostrando solo atributos específicos
    @Override
    public String toString() {
        String infoCliente;
        if(cliente != null){
            infoCliente = "Cliente: " + cliente.getNombre() + ", Dni: " + cliente.getDni();
        }else{
            infoCliente = null;
        }
        return "TarjetaCredito{" + "nuemero=" + numero + ", fechaVencimiento="
                + fechaVencimiento + ", " + banco + ", " + infoCliente + '}';
```

Clase Cliente:

```java
package ejericio4.uml;
public class Cliente {
    private String nombre;
    private String dni;
    private TarjetaCredito tarjeta;// Relación bidireccional con TarjetaDeCrédito

    public Cliente(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    // Establece la relación bidireccional con TarjetaDeCrédito usando un setter
    public void setTarjeta(TarjetaCredito tarjeta) {
        this.tarjeta = tarjeta;
    }

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

    // Método toString para mostrar información del cliente
    // Incluye el número de la tarjeta para reflejar la relación bidireccional
    // Evita recursión infinita mostrando solo el número
    @Override
    public String toString() {
        String infoTarjeta;
        if(tarjeta != null){
            infoTarjeta = "tarjeta número: " + tarjeta.getNumero();
        }else{
            infoTarjeta = null;
        }
        return "Cliente{" + "nombre=" + nombre + ", cuit=" + dni + ", " + infoTarjeta + '}';
    }
```

Clase Banco:

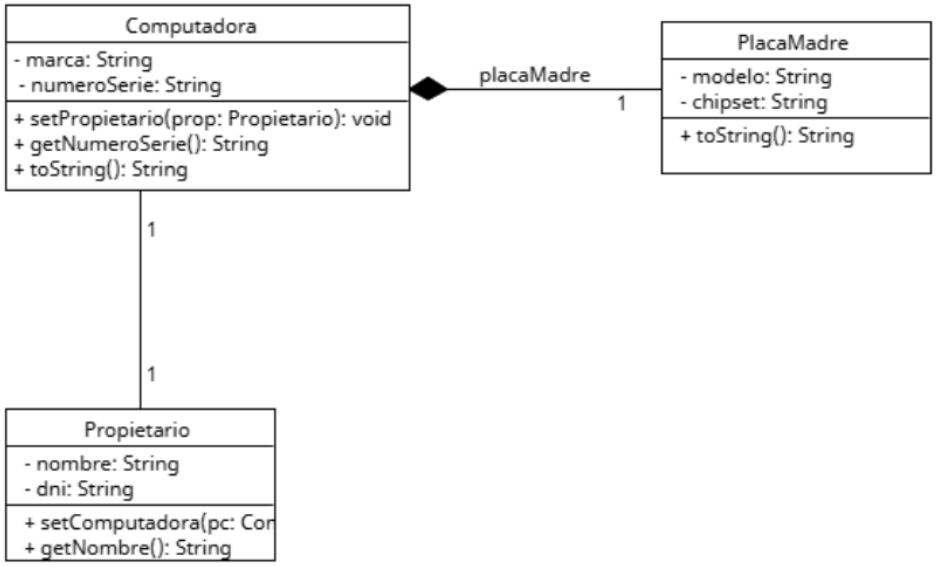
```java
package ejericio4.uml;
public class Banco {
    private String nombre;
    private String cuit;

    public Banco(String nombre, String cuit) {
        this.nombre = nombre;
        this.cuit = cuit;
    }

    @Override
    public String toString() {
        return "Banco{" + "nombre=" + nombre + ", cuit=" + 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

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio5-UML/src/ejercicio5/uml



Main:

```java
package ejercicio5.uml;
public class Principal {
    public static void main(String[] args) {
        // Composición: la PlacaMadre se crea junto con la Computadora
        PlacaMadre placa = new PlacaMadre(modelo: "ASUS ROG", chipset: "X570");
        Computadora pc = new Computadora(marca: "Dell", numeroSerie: "SN-2025-001", placaMadre: placa);

        // Propietario y asociación bidireccional
        Propietario propietario = new Propietario(nombre: "Marina Cordero", dni: "31058096");
        pc.setPropietario(propietario);
        propietario.setComputadora(computadora: pc);

        // Mostrar información
        System.out.println(x: pc);
        System.out.println(x: propietario);
    }
```

Clase Computadora:

```java
package ejercicio5.uml;
public class Computadora {
    private String marca;
    private String numeroSerie;
    private PlacaMadre placaMadre;      // composición
    private Propietario propietario;    // asociación bidireccional

    // el constructor exige la PlacaMadre, reflejando la composición
    public Computadora(String marca, String numeroSerie, PlacaMadre placaMadre) {
        this.marca = marca;
        this.numeroSerie = numeroSerie;
        this.placaMadre = placaMadre;
    }

    public void setPropietario(Propietario propietario) {
        this.propietario = propietario;
    }

    public String getNumeroSerie() { return numeroSerie; }

    @Override
    public String toString() {
        String infoProp = (propietario != null)
                ? "Propietario: " + propietario.getNombre() + ", Dni: " + propietario.getDni()
                : "sin propietario";
        return "Computadora{" + "marca=" + marca + ", numeroSerie=" + numeroSerie +
                ", " + placaMadre + ", " + infoProp + '}';
    }
```

Clase PlacaMadre:

```java
package ejercicio5.uml;
public class PlacaMadre {
    private String modelo;
    private String chipset;

    public PlacaMadre(String modelo, String chipset) {
        this.modelo = modelo;
        this.chipset = chipset;
    }

    @Override
    public String toString() {
        return "PlacaMadre{" + "modelo=" + modelo + ", chipset=" + chipset + '}';
    }
```

Clase Propietario:

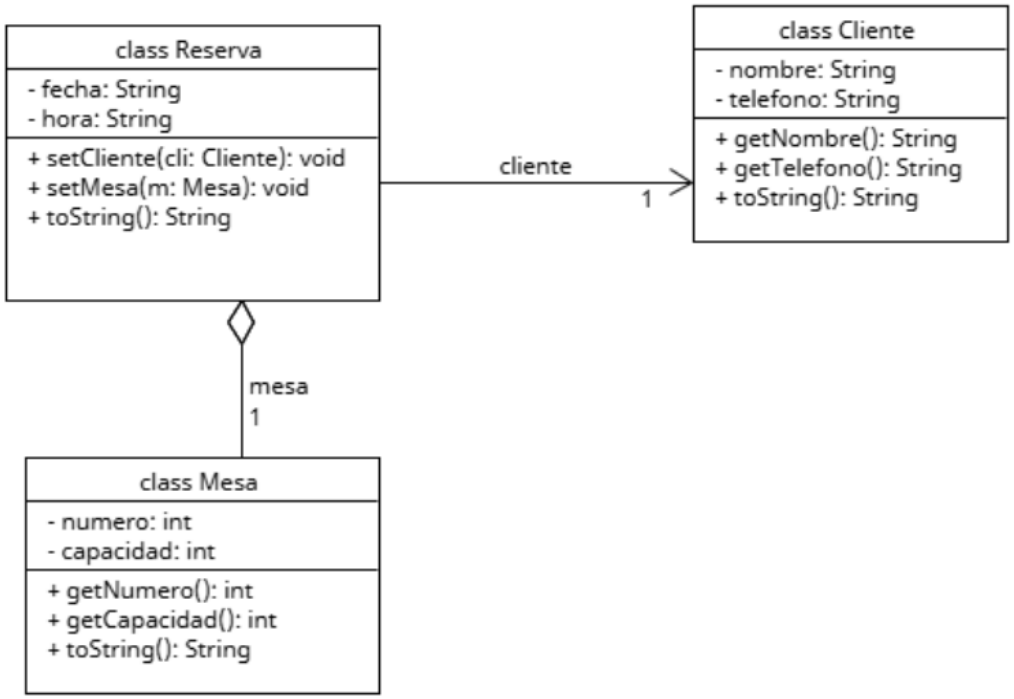
```java
package ejercicio5.uml;
public class Propietario {
    private String nombre;
    private String dni;
    private Computadora computadora;   // asociación bidireccional

    public Propietario(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }

    public void setComputadora(Computadora computadora) {
        this.computadora = computadora;
    }

    public String getNombre() { return nombre; }
    public String getDni() { return dni; }

    @Override
    public String toString() {
        String infoPC = (computadora != null)
                ? "Computadora serie: " + computadora.getNumeroSerie()
                : "sin computadora";
        return "Propietario{" + "nombre=" + nombre + ", dni=" + dni + ", " + infoPC + '}';
    }
```

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

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio6-UML/src/ejercicio6/uml



Main:

```java
package ejercicio6.uml;
public class Principal {
    public static void main(String[] args) {
        // Crear cliente y mesa independientes
        Cliente cliente = new Cliente(nombre: "Marina Cordero", telefono: "341-5551234");
        Mesa mesa = new Mesa(numero: 7, capacidad: 4);

        // Crear reserva asociando cliente y mesa
        Reserva reserva = new Reserva(fecha: "2025-09-14", hora: "20:30", cliente, mesa);

        // Mostrar la información completa
        System.out.println(x: reserva);
    }
```

```
Reserva{fecha=2025-09-14, hora=20:30, Cliente{nombre=Marina Cordero, telefono=341-5551234}, Mesa{numero=7, capacidad=4}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Cliente:

```java
package ejercicio6.uml;
public class Cliente {
    private String nombre;
    private String telefono;

    public Cliente(String nombre, String telefono) {
        this.nombre = nombre;
        this.telefono = telefono;
    }

    public String getNombre() {
        return nombre;
    }

    public String getTelefono() {
        return telefono;
    }

    @Override
    public String toString() {
        return "Cliente{" + "nombre=" + nombre + ", telefono=" + telefono + '}';
    }
```

Clase Mesa:

```java
package ejercicio6.uml;
public class Mesa {
    private int numero;
    private int capacidad;

    public Mesa(int numero, int capacidad) {
        this.numero = numero;
        this.capacidad = capacidad;
    }

    public int getNumero() {
        return numero;
    }

    public int getCapacidad() {
        return capacidad;
    }

    @Override
    public String toString() {
        return "Mesa{" + "numero=" + numero + ", capacidad=" + capacidad + '}';
    }
```

Clase Reserva:

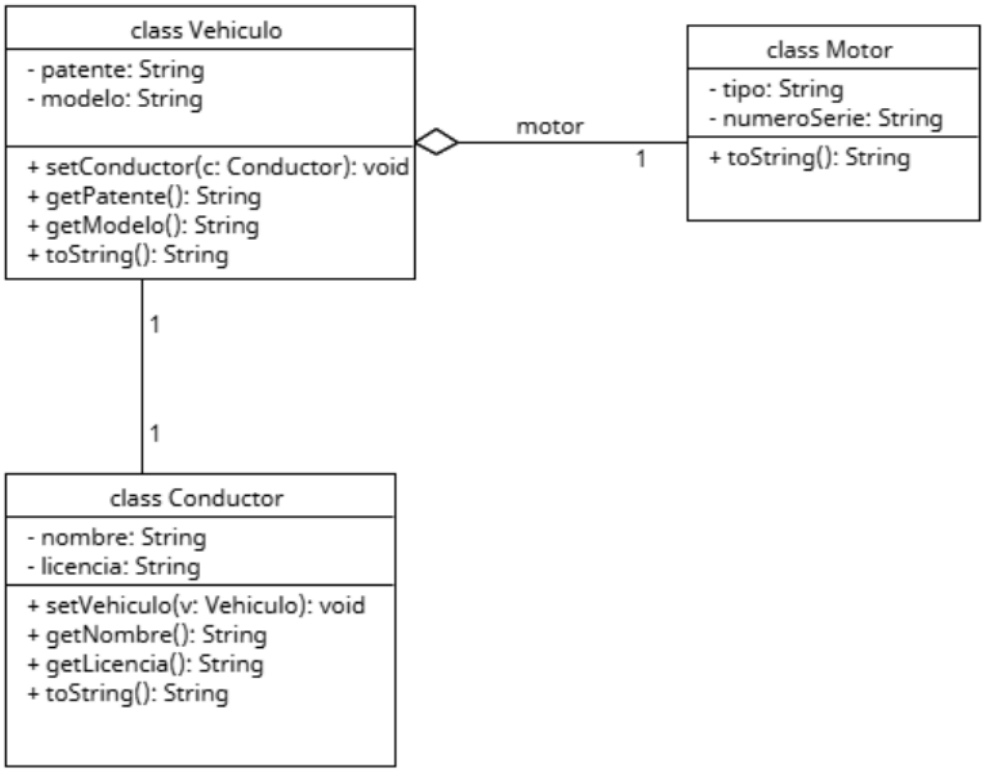
```java
package ejercicio6.uml;
public class Reserva {
    private String fecha;
    private String hora;
    private Cliente cliente; // Asociación unidireccional
    private Mesa mesa;       // Agregación

    public Reserva(String fecha, String hora, Cliente cliente, Mesa mesa) {
        this.fecha = fecha;
        this.hora = hora;
        this.cliente = cliente;
        this.mesa = mesa;
    }

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }

    public void setMesa(Mesa mesa) {
        this.mesa = mesa;
    }

    @Override
    public String toString() {
        return "Reserva{" + "fecha=" + fecha + ", hora=" + hora +
                ", " + cliente + ", " + mesa + '}';
    }
}
```

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

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio7-UML/src/ejercicio7/uml



Main:

```java
package ejercicio7.uml;
public class Ejercicio7UML {
    public static void main(String[] args) {
       // Crear motor independiente (agregación)
        Motor motor = new Motor(tipo:"Nafta", numeroSerie: "MTR-98765");

        // Crear vehículo con el motor
        Vehiculo vehiculo = new Vehiculo(patente: "AB123CD", modelo: "Toyota Corolla", motor);

        // Crear conductor
        Conductor conductor = new Conductor(nombre: "Marina Cordero", licencia: "LIC-555888");

        // Establecer asociación bidireccional
        vehiculo.setConductor(conductor);
        conductor.setVehiculo(vehiculo);

        // Mostrar la información completa
        System.out.println(x: vehiculo);
        System.out.println(x: conductor);

    }
```

```
run:
Vehiculo{patente=AB123CD, modelo=Toyota Corolla, Motor{tipo=Nafta, numeroSerie=MTR-98765}, Conductor: Marina Cordero, Licencia: LIC-555888}
Conductor{nombre=Marina Cordero, licencia=LIC-555888, Vehiculo patente: AB123CD}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Motor:

```java
package ejercicio7.uml;
public class Motor {
    private String tipo;
    private String numeroSerie;

    public Motor(String tipo, String numeroSerie) {
        this.tipo = tipo;
        this.numeroSerie = numeroSerie;
    }

    @Override
    public String toString() {
        return "Motor{" + "tipo=" + tipo + ", numeroSerie=" + numeroSerie + '}';
    }
```

Clase Conductor:

```java
package ejercicio7.uml;
public class Conductor {
    private String nombre;
    private String licencia;
    private Vehiculo vehiculo;  // Asociación bidireccional

    public Conductor(String nombre, String licencia) {
        this.nombre = nombre;
        this.licencia = licencia;
    }

    public void setVehiculo(Vehiculo vehiculo) {
        this.vehiculo = vehiculo;
    }

    public String getNombre() { return nombre; }
    public String getLicencia() { return licencia; }

    @Override
    public String toString() {
        String infoVehiculo = (vehiculo != null)
                ? "Vehiculo patente: " + vehiculo.getPatente()
                : "sin vehiculo";
        return "Conductor{" + "nombre=" + nombre + ", licencia="
                + licencia + ", " + infoVehiculo + '}';
    }
```

Clase Vehiculo:

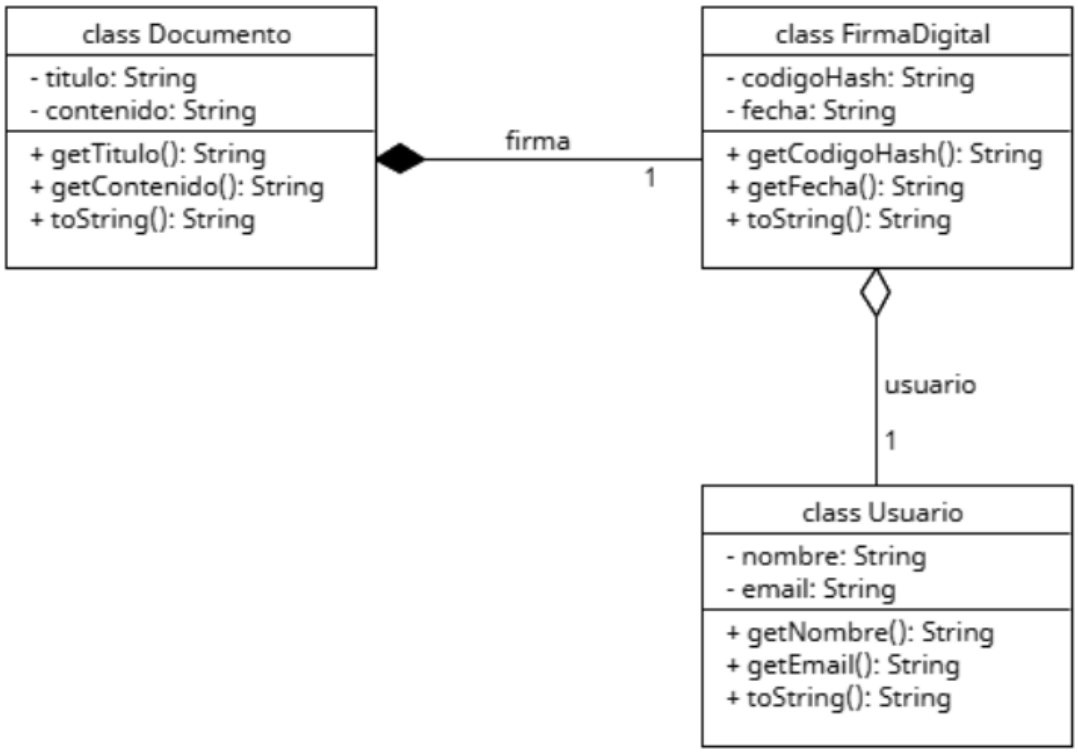
```java
package ejercicio7.uml;
public class Vehiculo {
    private String patente;
    private String modelo;
    private Motor motor;         // Agregación
    private Conductor conductor; // Asociación bidireccional

    public Vehiculo(String patente, String modelo, Motor motor) {
        this.patente = patente;
        this.modelo = modelo;
        this.motor = motor;
    }

    public void setConductor(Conductor conductor) {
        this.conductor = conductor;
    }

    public String getPatente() { return patente; }
    public String getModelo() { return modelo; }

    @Override
    public String toString() {
        String infoConductor = (conductor != null)
                ? "Conductor: " + conductor.getNombre() + ", Licencia: " + conductor.getLicencia()
                : "sin conductor";
        return "Vehiculo{" + "patente=" + patente + ", modelo=" + modelo + ", "
                + motor + ", " + infoConductor + '}';
    }
```

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

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio8-UML/src/ejercicio8/uml



Main:

```java
package ejercicio8.uml;
public class Ejercicio8UML {
    public static void main(String[] args) {
        // Crear usuario independiente (agregación)
        Usuario usuario = new Usuario(nombre: "Marina Cordero", email: "marina@example.com");

        // Crear firma digital asociada a ese usuario
        FirmaDigital firma = new FirmaDigital(codigoHash: "abc123hash", fecha: "2025-09-13", usuario);

        // Crear documento que contiene la firma (composición)
        Documento documento = new Documento(titulo: "Contrato de Servicio",
                contenido: "Contenido del contrato...", firmaDigital: firma);

        // Mostrar información
        System.out.println(x: documento);
    }
```

```
Documento{titulo=Contrato de Servicio, contenido=Contenido del contrato...
, FirmaDigital{codigoHash=abc123hash, fecha=2025-09-13, Usuario{nombre=Marina Cordero, email=marina@example.com}}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Usuario:

```java
package ejercicio8.uml;
public class Usuario {
    private String nombre;
    private String email;

    public Usuario(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }

    public String getNombre() { return nombre; }
    public String getEmail() { return email; }

    @Override
    public String toString() {
        return "Usuario{" + "nombre=" + nombre + ", email=" + email + '}';
    }
}
```

Clase FirmaDigital:

```java
package ejercicio8.uml;
public class FirmaDigital {
    private String codigoHash;
    private String fecha;
    private Usuario usuario; // Agregación

    public FirmaDigital(String codigoHash, String fecha, Usuario usuario) {
        this.codigoHash = codigoHash;
        this.fecha = fecha;
        this.usuario = usuario;
    }

    public String getCodigoHash() { return codigoHash; }
    public String getFecha() { return fecha; }

    @Override
    public String toString() {
        return "FirmaDigital{" +
                "codigoHash=" + codigoHash +
                ", fecha=" + fecha +
                ", " + usuario +
                '}';
    }
```

Lase Documento:

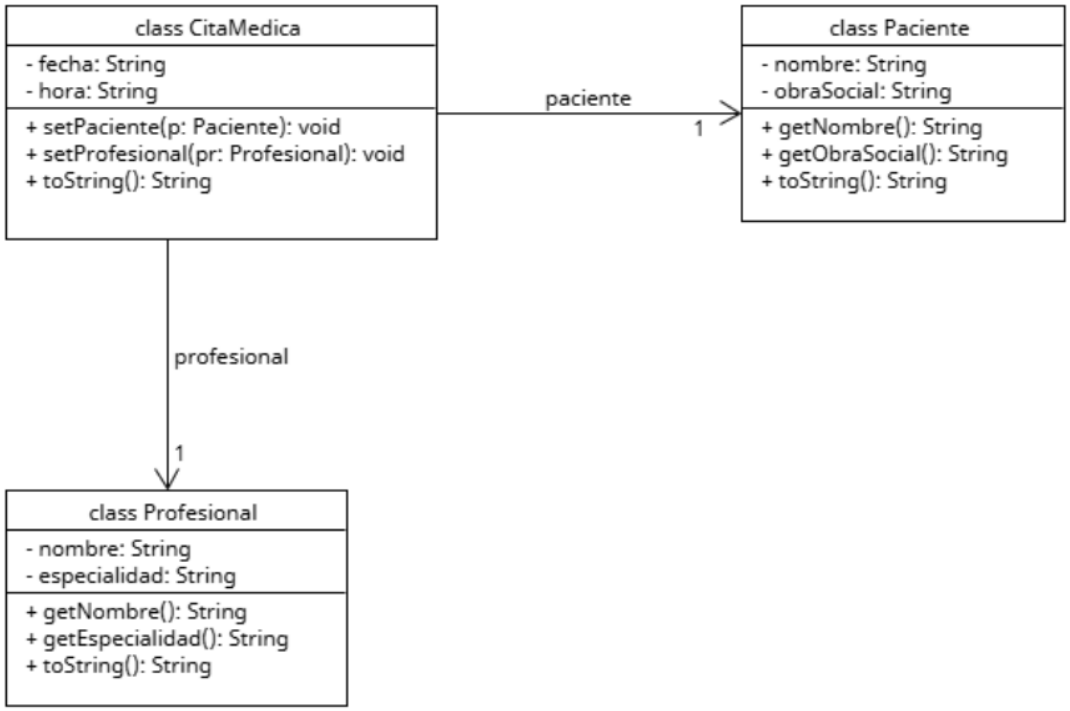
```java
package ejer
public clase
    private String titulo;
    private String contenido;
    private FirmaDigital firmaDigital; // Composición

    public Documento(String titulo, String contenido, FirmaDigital firmaDigital) {
        this.titulo = titulo;
        this.contenido = contenido;
        this.firmaDigital = firmaDigital;
    }

    public String getTitulo() { return titulo; }
    public String getContenido() { return contenido; }

    @Override
    public String toString() {
        return "Documento{" +
                "titulo=" + titulo +
                ", contenido=" + contenido +
                "\n, " + firmaDigital +
                '}';
    }
```

9. CitaMédica - Paciente - Profesional
    a. Asociación unidireccional: **CitaMédica → Paciente,**
    b. Asociación unidirecciona: **CitaMédica → Profesional**

    Clases y atributos:

        i. CitaMédica: fecha, hora
        ii. Paciente: nombre, obraSocial
        iii. Profesional: nombre, especialidad

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio9-UML/src/ejercicio9/uml



Main:

```
public static void main(String[] args) {
  // Crear paciente y profesional
    Paciente paciente = new Paciente(nombre: "Marina Cordero", obraSocial: "OSDE");
    Profesional profesional = new Profesional(nombre: "Dr. Juan Pérez", especialidad: "Cardiología");

    // Crear cita médica con relaciones unidireccionales
    CitaMedica cita = new CitaMedica(fecha: "2025-09-20", hora:"10:30", paciente, profesional);

    // Mostrar información de la cita
    System.out.println(x: cita);
}
```

```
CitaMedica{fecha=2025-09-20, hora=10:30, Paciente{nombre=Marina Cordero, obraSocial=OSDE}
, Profesional{nombre=Dr. Juan Pérez, especialidad=Cardiología}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Paciente:

```java
public class Paciente {
    private String nombre;
    private String obraSocial;

    public Paciente(String nombre, String obraSocial) {
        this.nombre = nombre;
        this.obraSocial = obraSocial;
    }

    public String getNombre() { return nombre; }
    public String getObraSocial() { return obraSocial; }

    @Override
    public String toString() {
        return "Paciente{" + "nombre=" + nombre + ", obraSocial=" + obraSocial + '}';
    }
}
```

Clase Profesional:

```java
public class Profesional {
    private String nombre;
    private String especialidad;

    public Profesional(String nombre, String especialidad) {
        this.nombre = nombre;
        this.especialidad = especialidad;
    }

    public String getNombre() { return nombre; }
    public String getEspecialidad() { return especialidad; }

    @Override
    public String toString() {
        return "Profesional{" + "nombre=" + nombre + ", especialidad=" + especialidad + '}';
    }
```

Clase CitaMedica:

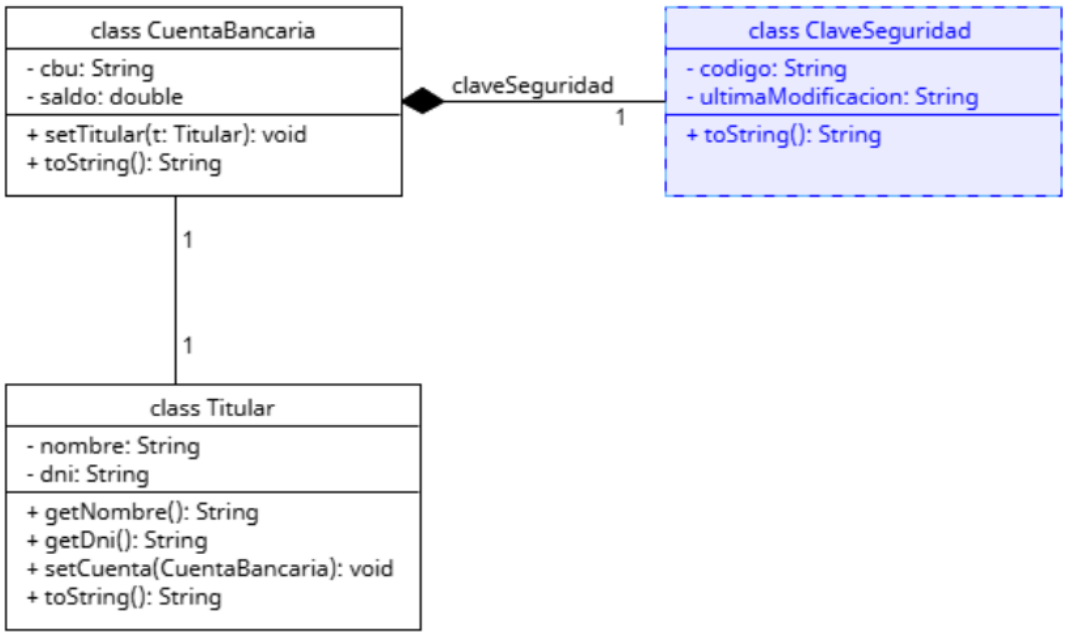
```java
public class CitaMedica {
    private String fecha;
    private String hora;
    private Paciente paciente;       // asociación unidireccional
    private Profesional profesional; // asociación unidireccional

    public CitaMedica(String fecha, String hora, Paciente paciente, Profesional profesional) {
        this.fecha = fecha;
        this.hora = hora;
        this.paciente = paciente;
        this.profesional = profesional;
    }

    public void setPaciente(Paciente paciente) { this.paciente = paciente; }
    public void setProfesional(Profesional profesional) { this.profesional = profesional; }

    @Override
    public String toString() {
        return "CitaMedica{" +
                "fecha=" + fecha +
                ", hora=" + hora +
                ", " + paciente +
                "\n, " + profesional +
                '}';
```

10. CuentaBancaria - ClaveSeguridad - Titular
    a. Composición: **CuentaBancaria → ClaveSeguridad**
    b. Asociación bidireccional: **CuentaBancaria ↔ Titular**

Clases y atributos:

i. CuentaBancaria: cbu, saldo
ii. ClaveSeguridad: codigo, ultimaModificacion
iii. Titular: nombre, dni.

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio10-UML/src/ejercicio10/uml



Main:

```
public static void main(String[] args) {
    // Crear titular
    Titular titular = new Titular(nombre: "Marina Cordero", dni: "31058096");

    // Crear cuenta bancaria (la clave de seguridad se crea dentro)
    CuentaBancaria cuenta = new CuentaBancaria(
            cbu: "1234567890123456789012", saldo: 25000.50,
            codigoClave: "ABC123", ultimaModificacion: "2025-09-13"
    );

    // Establecer relación bidireccional
    cuenta.setTitular(titular);
    titular.setCuenta(cuenta);

    // Mostrar información
    System.out.println(x: cuenta);
```

```
run:
CuentaBancaria{cbu=12345678901234567890l2, saldo=25000.5,
 ClaveSeguridad{codigo=ABC123, ultimaModificacion=2025-09-13}, titular=Marina Cordero DNI: 31058096}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase ClaveDeSeguridad:

```
public class ClaveSeguridad {
    private String codigo;
    private String ultimaModificacion;

    public ClaveSeguridad(String codigo, String ultimaModificacion) {
        this.codigo = codigo;
        this.ultimaModificacion = ultimaModificacion;
    }

    @Override
    public String toString() {
        return "ClaveSeguridad{" +
                "codigo=" + codigo +
                ", ultimaModificacion=" + ultimaModificacion + '}';
    }
}
```

Clase Titular:

```java
public class Titular {
    private String nombre;
    private String dni;
    private CuentaBancaria cuenta; // relación bidireccional

    public Titular(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }

    public String getNombre() { return nombre; }
    public String getDni() { return dni; }

    public void setCuenta(CuentaBancaria cuenta) {
        this.cuenta = cuenta;
    }

    @Override
    public String toString() {
        return "Titular{" + "nombre=" + nombre + ", dni=" + dni + '}';
    }
```

Clase CuentaBancaria:

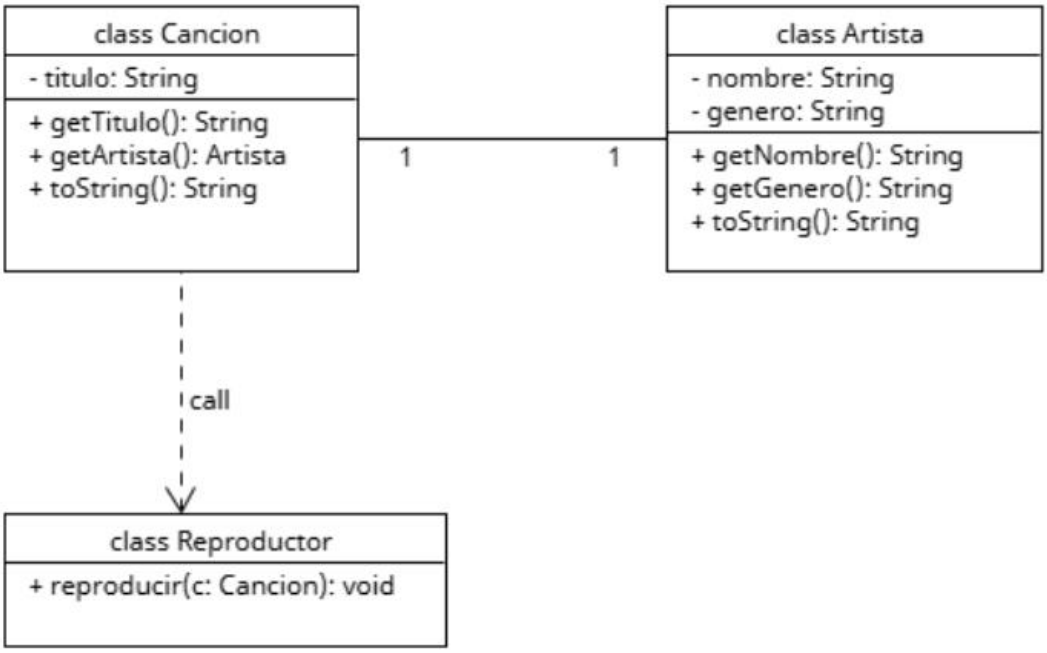
```java
public class CuentaBancaria {
    private String cbu;
    private double saldo;
    private ClaveSeguridad claveSeguridad; // composición
    private Titular titular;               // relación bidireccional

    public CuentaBancaria(String cbu, double saldo, String codigoClave, String ultimaModificacion)
        this.cbu = cbu;
        this.saldo = saldo;
        // Composición: la clave se crea junto con la cuenta
        this.claveSeguridad = new ClaveSeguridad(codigo: codigoClave, ultimaModificacion);
    }

    public void setTitular(Titular titular) {
        this.titular = titular;
    }

    @Override
    public String toString() {
        return "CuentaBancaria{" +
                "cbu=" + cbu +
                ", saldo=" + saldo +
                ",\n " + claveSeguridad +
                ", titular=" + (titular != null ? titular.getNombre() + " DNI: " +
                titular.getDni() : "sin titular") + '}';
    }
```

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)

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio11-UML/src/ejercicio11/uml



Main:

```java
public static void main(String[] args) {
    // Crear artista
    Artista artista = new Artista(nombre: "Marina Cordero", genero: "Pop");

    // Crear canción asociada al artista
    Cancion cancion = new Cancion(titulo: "Mi Canción Favorita", artista);

    // Crear reproductor y usar dependencia de uso
    Reproductor reproductor = new Reproductor();
    reproductor.reproducir(cancion); // usa Cancion solo en el método
```

```
Reproduciendo: Mi Canci▯n Favorita - Artista: Marina Cordero
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Artista:

```java
private String nombre;
private String genero;

public Artista(String nombre, String genero) {
    this.nombre = nombre;
    this.genero = genero;
}

public String getNombre() { return nombre; }
public String getGenero() { return genero; }

@Override
public String toString() {
    return "Artista{" + "nombre=" + nombre + ", genero=" + genero + '}';
}
```

Clase Cancion:

```java
public class Cancion {
    private String titulo;
    private Artista artista; // Asociación unidireccional

    public Cancion(String titulo, Artista artista) {
        this.titulo = titulo;
        this.artista = artista;
    }

    public String getTitulo() { return titulo; }
    public Artista getArtista() { return artista; }

    @Override
    public String toString() {
        return "Cancion{" + "titulo=" + titulo + ", " + artista + '}';
    }
```

Clase Reproductor:

```java
public class Reproductor {
    // Dependencia de uso: usa Cancion como parámetro, no la guarda como atributo
    public void reproducir(Cancion cancion) {
        System.out.println("Reproduciendo: " + cancion.getTitulo() +
                " - Artista: " + cancion.getArtista().getNombre());
    }
```

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)

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio2-UML/src/ejercicio2/uml

Main:

```
public static void main(String[] args) {
    Contribuyente contrib = new Contribuyente(nombre: "Marina Cordero", cuil:"20-31058096-4");

    // Crear impuesto y establecer relación con setter
    Impuesto impuesto = new Impuesto(monto: 15000.0);
    impuesto.setContribuyente(contribuyente: contrib);

    Calculadora calc = new Calculadora();
    calc.calcular(impuesto);
}
```

```
Contribuyente: Marina Cordero
Monto base: 15000.0
Total con inter�s 10%: 16500.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Contribuyente:

```java
public class Contribuyente {
    private String nombre;
    private String cuil;

    public Contribuyente(String nombre, String cuil) {
        this.nombre = nombre;
        this.cuil = cuil;
    }

    public String getNombre() { return nombre; }
    public String getCuil()   { return cuil; }

    @Override
    public String toString() {
        return "Contribuyente{" + "nombre=" + nombre + ", cuil=" + cuil + '}';
    }
```

Clase Impuesto:

```java
public class Impuesto {
    private double monto;
    private Contribuyente contribuyente; // asociación unidireccional

    public Impuesto(double monto) {
        this.monto = monto;
    }

    // Asociación mediante setter
    public void setContribuyente(Contribuyente contribuyente) {
        this.contribuyente = contribuyente;
    }

    public double getMonto() { return monto; }
    public Contribuyente getContribuyente() { return contribuyente; }

    @Override
    public String toString() {
        return "Impuesto{" +
                "monto=" + monto +
                ", contribuyente=" +
                (contribuyente != null ? contribuyente.getNombre() : "sin asignar") +
                '}';
    }
```

Clase Calculadora:

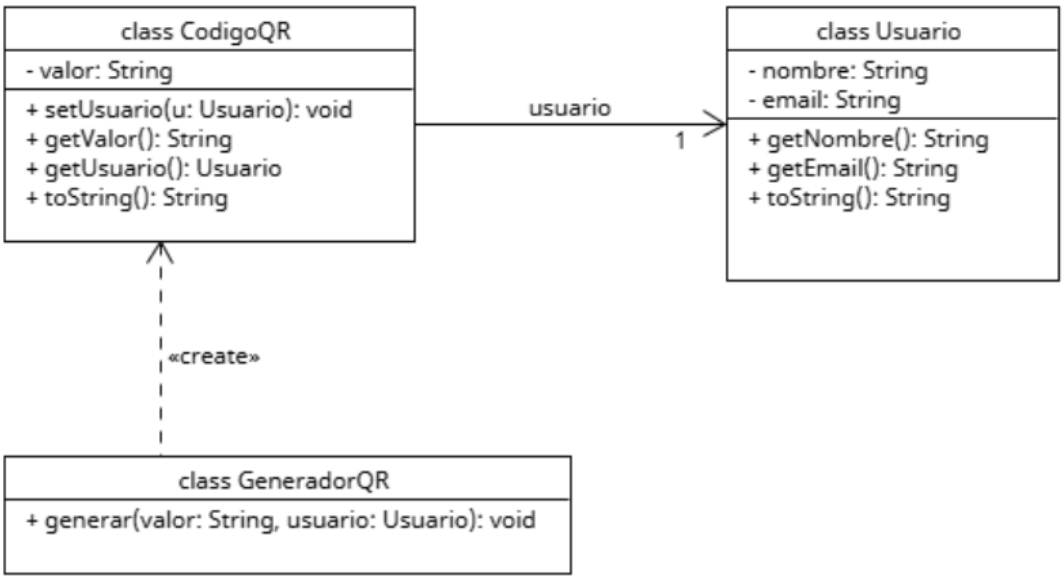
```java
public class Calculadora {
    // Dependencia de uso: solo usa Impuesto como parámetro
    public void calcular(Impuesto impuesto) {
        double interes = impuesto.getMonto() * 0.10;
        double total = impuesto.getMonto() + interes;

        System.out.println("Contribuyente: " +
                (impuesto.getContribuyente() != null
                 ? impuesto.getContribuyente().getNombre()
                 : "No asignado"));
        System.out.println("Monto base: " + impuesto.getMonto());
        System.out.println("Total con interés 10%: " + total);
    }
```

13. GeneradorQR - Usuario - CódigoQR
    a. Asociación unidireccional: **CódigoQR → Usuario**
    b. Dependencia de creación: **GeneradorQR.generar(String, Usuario)**

Clases y atributos:

i. CodigoQR: valor.
ii. Usuario: nombre, email.
iii. GeneradorQR->método: void generar(String valor, Usuario usuario)

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio13-UML/src/ejercicio13/uml



Main:

```
public class Principal {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Usuario usuario = new Usuario(nombre: "Marina Cordero", email: "marina@example.com");

        GeneradorQR generador = new GeneradorQR();
        // Dependencia de creación: crea y asocia el CódigoQR dentro del método
        generador.generar(valor: "ABC-123-XYZ", usuario);
    }
```

```
run:
QR generado: CodigoQR{valor=ABC-123-XYZ, usuario=Marina Cordero}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Usuario:

```java
 */
public class Usuario {
    private String nombre;
    private String email;

    public Usuario(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }

    public String getNombre() { return nombre; }
    public String getEmail()  { return email; }

    @Override
    public String toString() {
        return "Usuario{" + "nombre=" + nombre + ", email=" + email + '}';
    }
```

Clase CodigoQR:

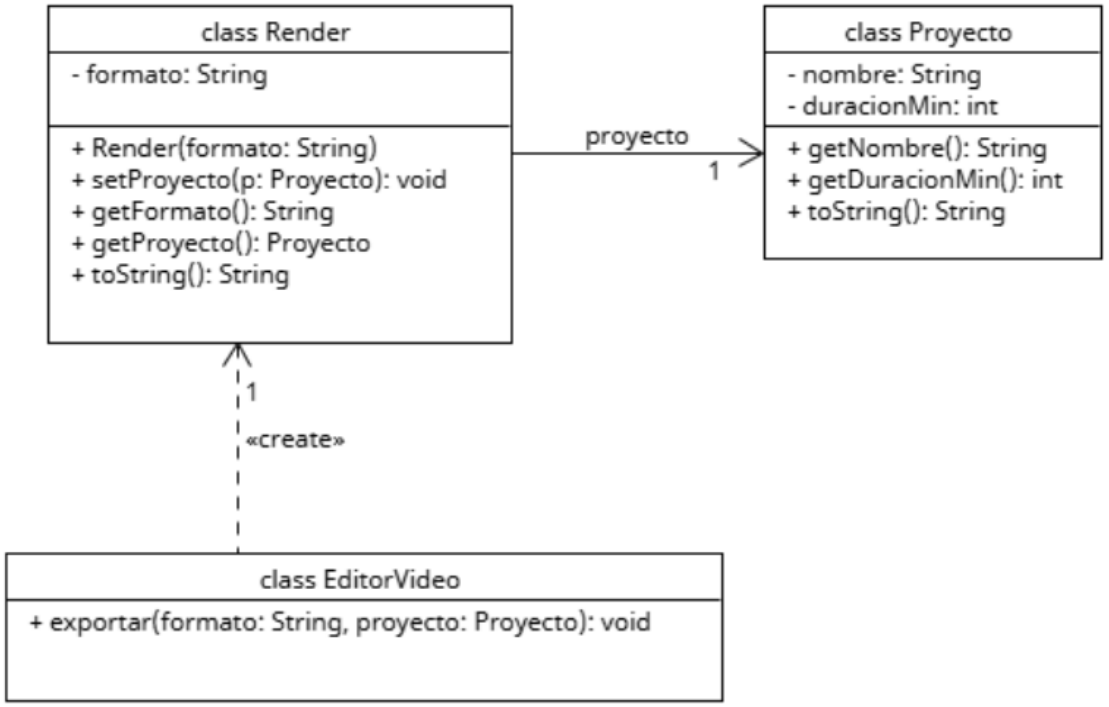
```java
public class CodigoQR {
    private String valor;
    private Usuario usuario; // Asociación unidireccional (por setter)

    public CodigoQR(String valor) {
        this.valor = valor;
    }

    // Establece la asociación con Usuario
    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }

    public String getValor()   { return valor; }
    public Usuario getUsuario(){ return usuario; }

    @Override
    public String toString() {
        return "CodigoQR{" +
                "valor=" + valor +
                ", usuario=" +
                (usuario != null ? usuario.getNombre() : "sin asignar") +
                '}';
    }
```

Clase GeneradorQR:

```java
public class GeneradorQR {
    // Dependencia de creación: crea un CódigoQR y le asigna el Usuario mediante setter
    public void generar(String valor, Usuario usuario) {
        CodigoQR qr = new CodigoQR(valor);
        qr.setUsuario(usuario);   // asociación establecida con setter
        System.out.println("QR generado: " + qr);
    }
```

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)

Enlace Git Hub: https://github.com/Marigi84/TP5-Relaciones-POO-UML/tree/master/Ejercicio14-UML/src/ejercicio14/uml



Main:

```java
public class Principal {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Proyecto proyecto = new Proyecto(nombre: "Video Promocional", duracionMin: 12);

        EditorVideo editor = new EditorVideo();
        // Aquí se ve la dependencia de creación y la asociación por setter
        editor.exportar(formato: "MP4", proyecto);
    }
```

```
Exportaci▯n realizada: Render{formato=MP4, proyecto=Video Promocional}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Proyecto:

```java
*/
public class Proyecto {
    private String nombre;
    private int duracionMin;

    public Proyecto(String nombre, int duracionMin) {
        this.nombre = nombre;
        this.duracionMin = duracionMin;
    }

    public String getNombre() { return nombre; }
    public int getDuracionMin() { return duracionMin; }

    @Override
    public String toString() {
        return "Proyecto{" + "nombre=" + nombre + ", duracionMin=" + duracionMin + '}';
    }
```

Clase Render:

```
public void setProyecto(Proyecto proyecto) {
    this.proyecto = proyecto;
}

public String getFormato() { return formato; }
public Proyecto getProyecto() { return proyecto; }

@Override
public String toString() {
    return "Render{" +
           "formato=" + formato +
           ", proyecto=" + (proyecto != null ? proyecto.getNombre() : "sin asignar") +
           '}';
}
```

Clase EditorDeVideo:

```
public class EditorVideo {
    // Dependencia de creación: crea un Render en este método
    public void exportar(String formato, Proyecto proyecto) {
        Render render = new Render(formato); // crea el objeto
        render.setProyecto(proyecto);        // establece la asociación con setter
        System.out.println("Exportación realizada: " + render);
    }
```

Conclusión

A través del desarrollo de este trabajo práctico pude afianzar mis conocimientos sobre las relaciones 1 a 1 en el modelado orientado a objetos. La identificación de asociaciones, agregaciones, composiciones y dependencias, así como su correcta implementación en diagramas UML y en Java, me permitió comprender la importancia de diseñar clases con vínculos claros y bien definidos. Este proceso fortaleció mi capacidad de análisis y abstracción, habilidades esenciales para el desarrollo de software de calidad.