

# Diagrama de clases

---

Allyson Caro + Laura Farías + Víctor Anabalon + Hipólito Cayupi

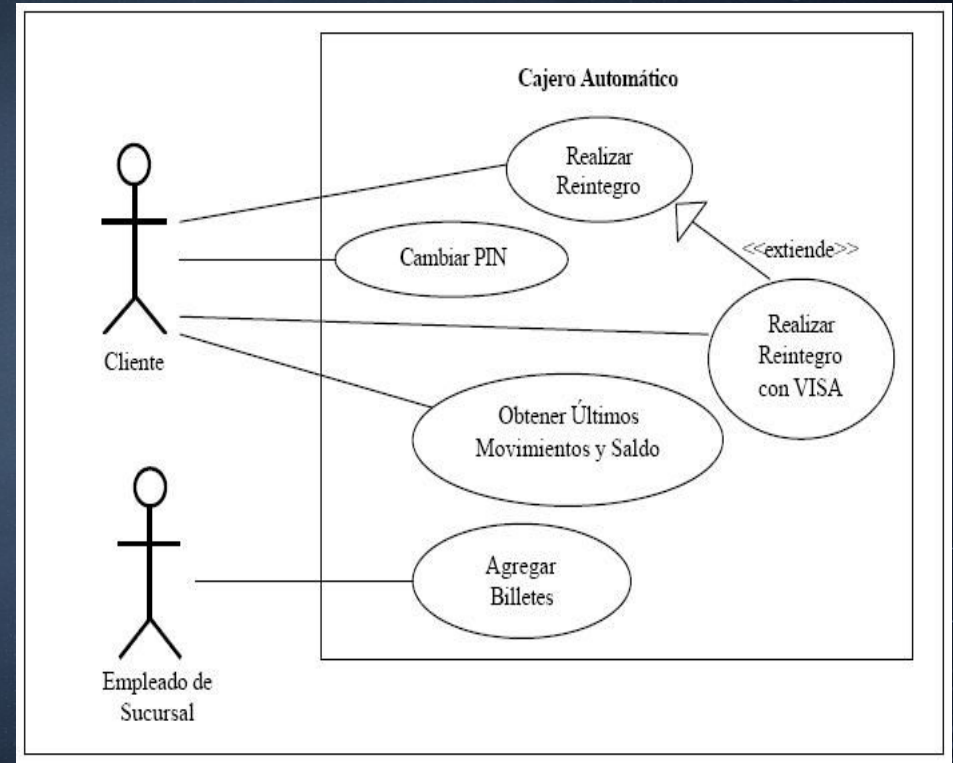


# ¿Qué es UML?

---

Lenguaje Unificado de Modelado

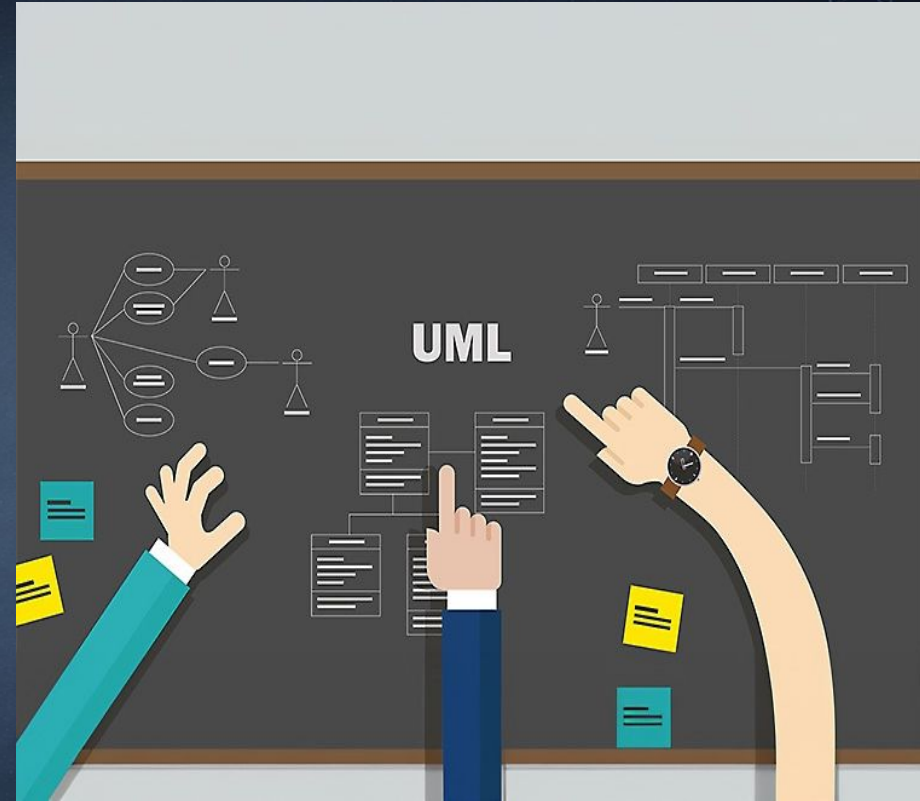
- UML es un lenguaje estandarizado de modelado.
- Permite visualizar la estructura y el comportamiento de sistemas y software.
- Los diagramas UML ayudan a representar gráficamente componentes del sistema.
- Facilita el análisis de relaciones y jerarquías entre clases y elementos del código.



• Casos de uso

# Ventajas de un diagrama UML

- Es muy sencillo
- Es capaz de modelar todo tipo de sistemas
- Es un lenguaje universal
- Es visual
- Es independiente
- Permite detección temprana de errores





# Diagramas UML más utilizados

The background is a dark blue gradient with various geometric shapes and lines. There are several hexagons, some with internal patterns, and some with dashed lines connecting them. There are also some light blue dots and curved lines, giving it a technical or network-like appearance.

# 1 Diagrama de Clases

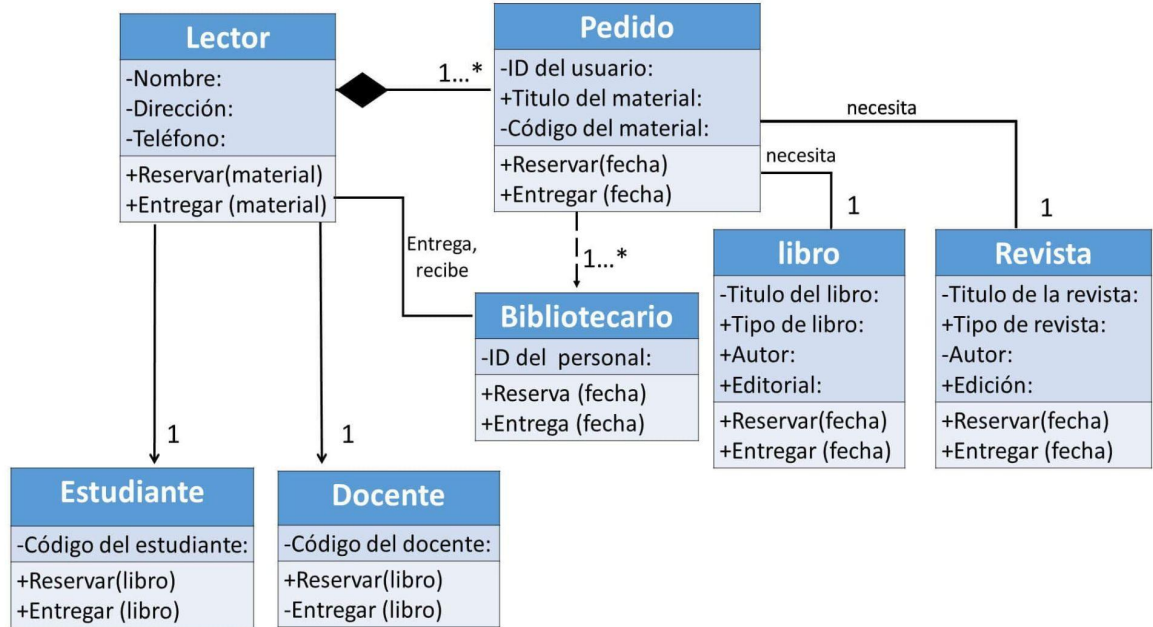
## ¿Qué muestra?

- Clases
- Atributos
- Métodos
- Relaciones entre clases

## ¿Para qué sirve?

- Modelar la estructura del sistema
- Base para programar en POO (Python, Java, etc.)

Diagrama de clases de un sistema de servicios bibliotecarios

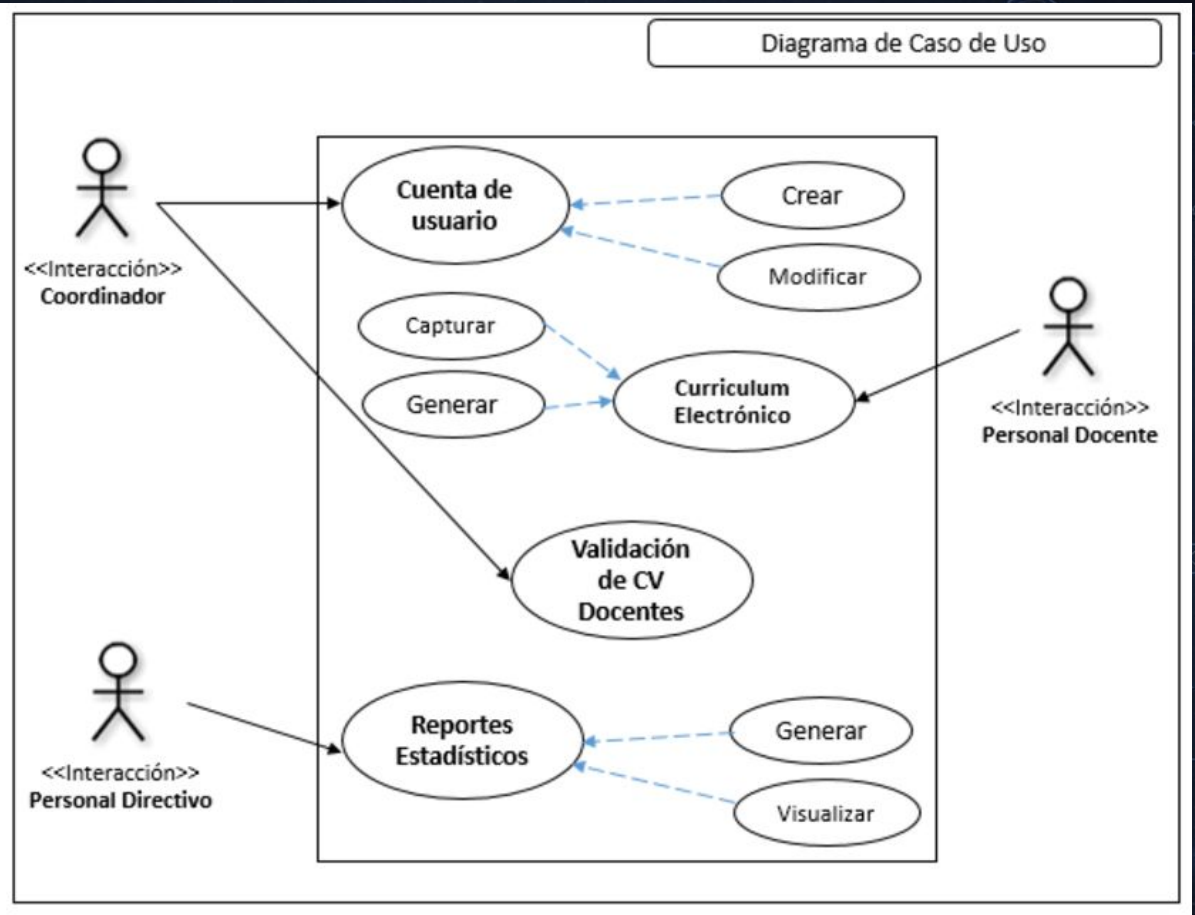


## 2 Diagrama de Casos de Uso

- Qué puede hacer el sistema
- Quién interactúa con él (actores)

### 📌 Ejemplo:

- Usuario → iniciar sesión
  - Cliente → realizar pedido
- 👉 No muestra código ni clases, solo **funcionalidades**.



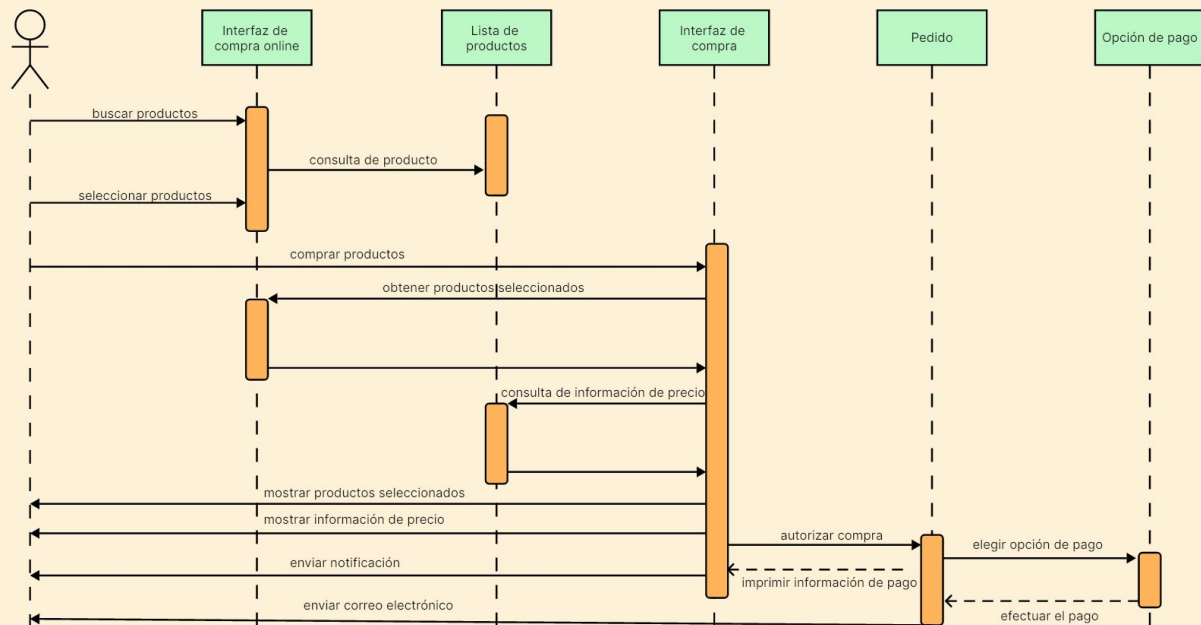
### 3 Diagrama de Secuencia

#### ¿Qué muestra?

- El orden en que los objetos se comunican
- Mensajes en el tiempo

#### 📌 Ideal para:

- Explicar flujos
- Procesos paso a paso (login, pago, etc.)




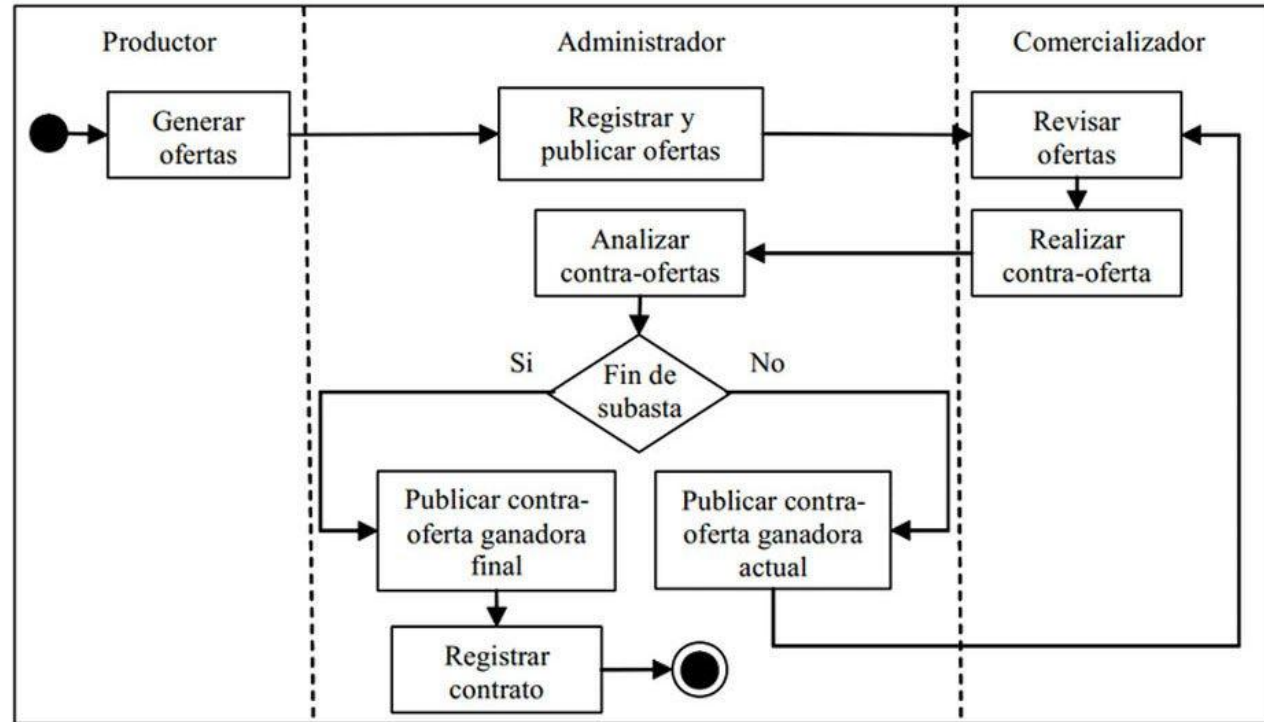


## 4 Diagrama de Actividades

### ¿Qué muestra?

- Flujo de acciones
- Decisiones (sí / no)

 Muy parecido a un **diagrama de flujo**  
Se usa para procesos de negocio o lógica compleja.



## 5 Diagrama de Estados

### ¿Qué muestra?

- Estados de un objeto
- Cómo cambia según eventos

### Ejemplo:

Pedido → *creado* → *pagado* →  
*enviado* → *entregado*

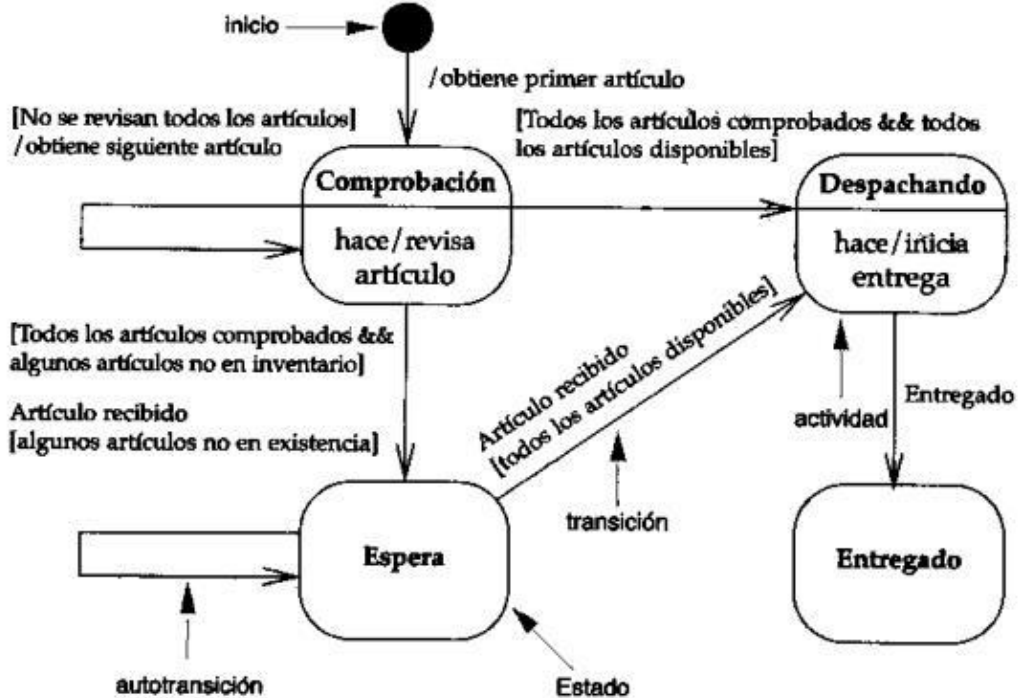
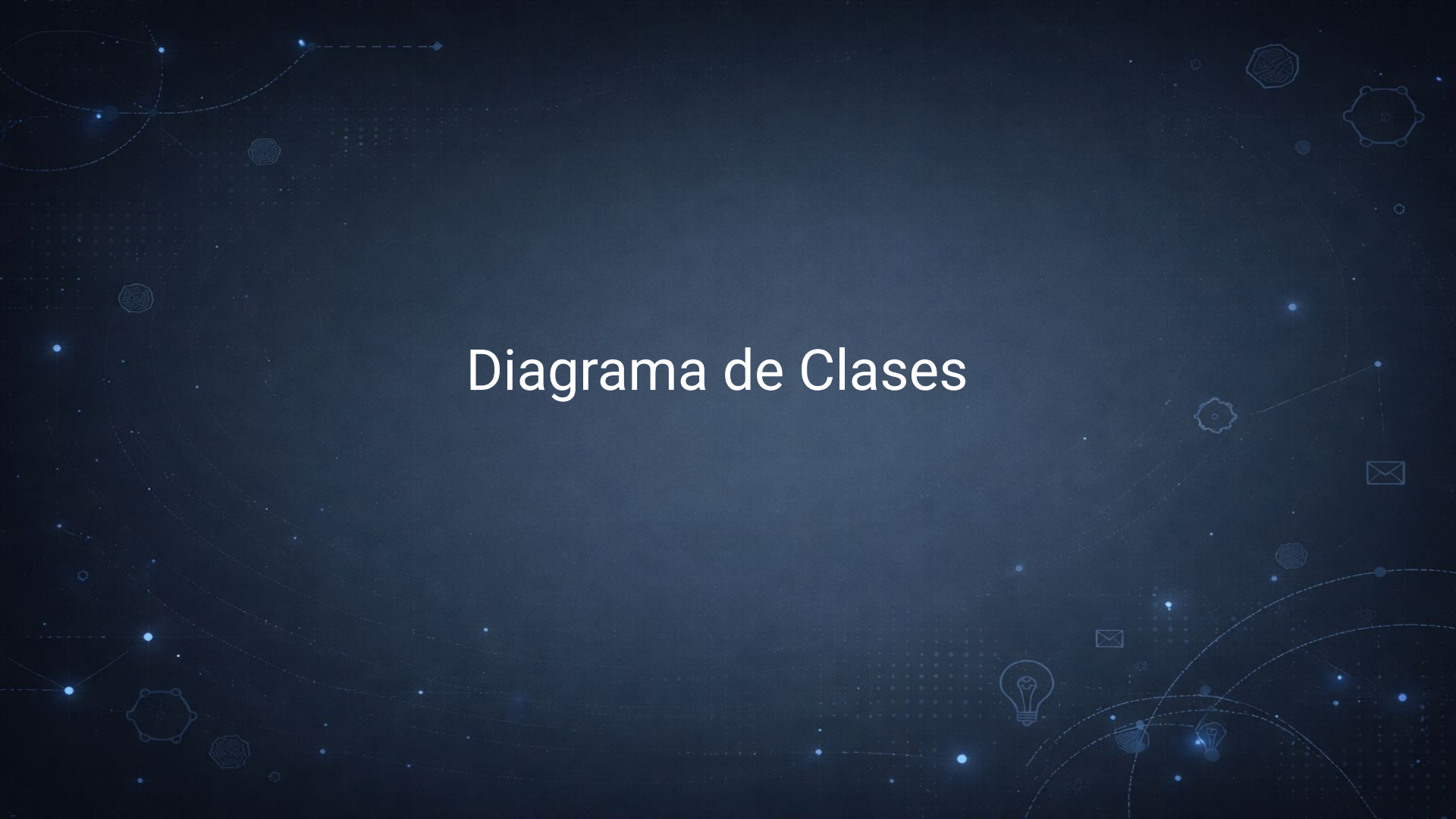


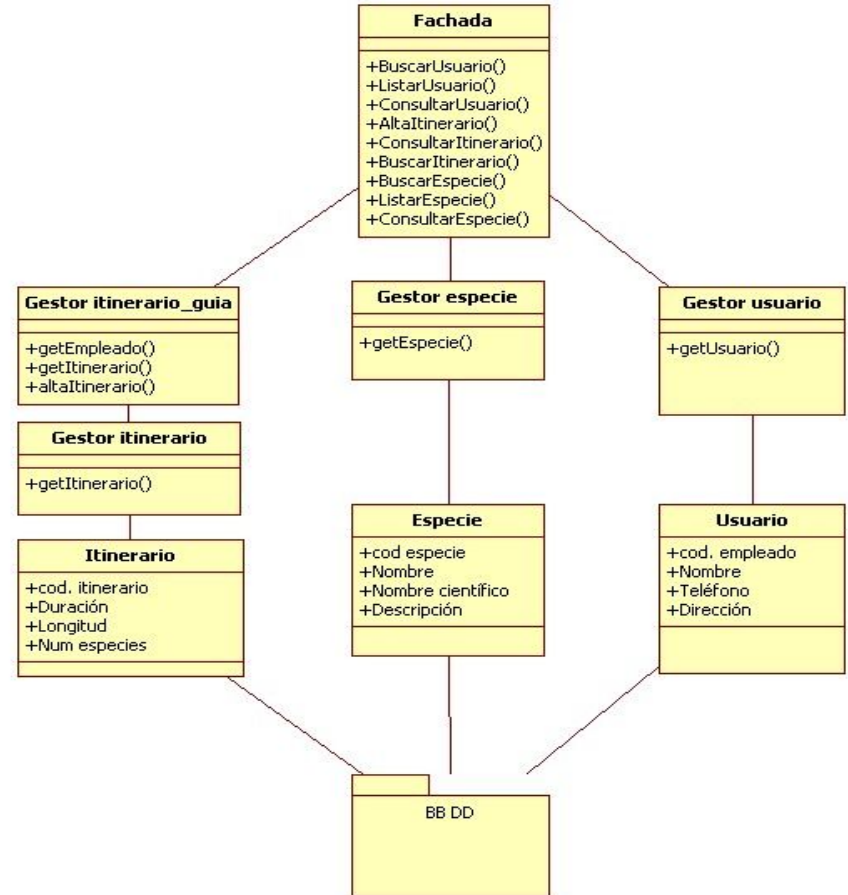
Figura 8-1: Diagrama de estados

# Diagrama de Clases

The background is a dark blue gradient with various abstract elements. There are faint, light blue geometric shapes like hexagons and octagons scattered across the frame. Thin, curved lines and dotted patterns are also visible, creating a sense of depth and complexity. Small, bright blue dots are placed at various points, some connected by thin lines, resembling a network or a constellation.

# ¿Qué es un Diagrama de Clases?

Un diagrama de clases es un tipo de diagrama UML que representa la estructura de un sistema orientado a objetos, mostrando sus clases, atributos, métodos y las relaciones entre ellas.



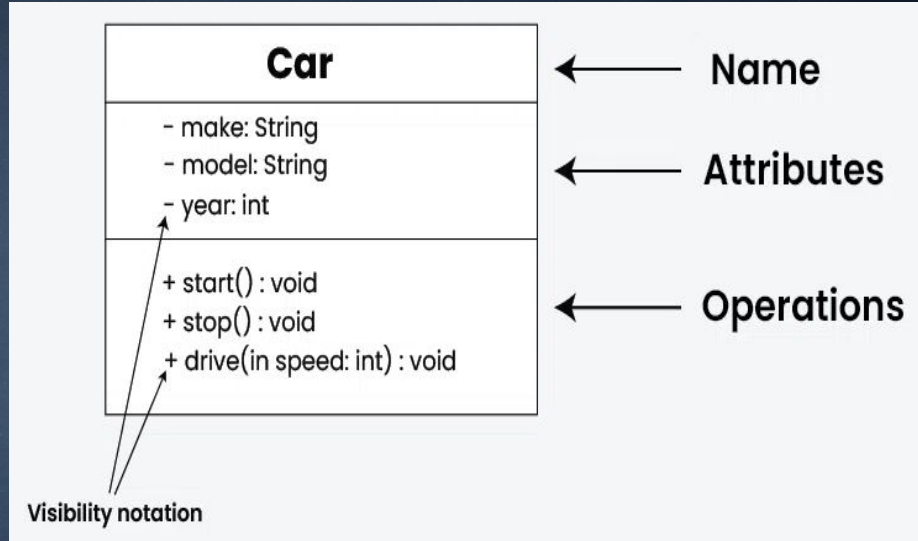


# Atributos y Métodos

- **Atributos:** representan el estado del objeto
  - Ejemplo: encendido, conectado, modelo
- **Métodos:** representan el comportamiento del objeto
  - Ejemplo: encender(), conectar(), estado()

## Simbología básica:

- + Público
- - Privado



# Estructura de un Diagrama de Clases







The background is a dark blue gradient with various abstract elements. There are faint, glowing lines forming arcs and paths. Scattered throughout are small, bright blue dots. Several geometric shapes, including hexagons and octagons, are visible, some with internal patterns. In the bottom right corner, there are icons of a lightbulb and an envelope, suggesting ideas and communication.

# Elementos principales

1. **Clases**
2. **Atributos**
3. **Métodos**
4. **Relaciones**
5. **Visibilidad**
6. **Multiplicidad**

Nombre de clase
Atributos
Métodos

Están representadas mediante un rectángulo con tres compartimientos que incluye nombre de la clase, atributos y métodos.

Pedido	
Sealed Clase	
Propiedades	
	Codigo
	Fecha
	Pendiente
	Proveedor
Métodos	
	SumaCantidadItems
	ValorPedido

Los atributos son los que la clase conoce como características del objeto.

Los métodos llamados tambien operaciones son los que indican como hacer las cosas.

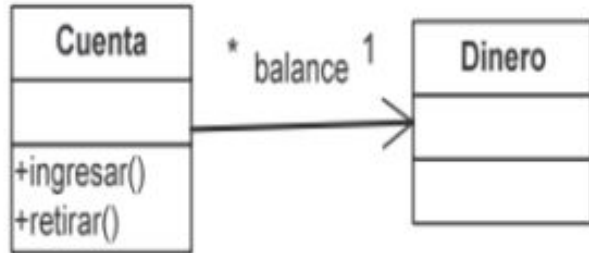


# Relaciones entre Clases

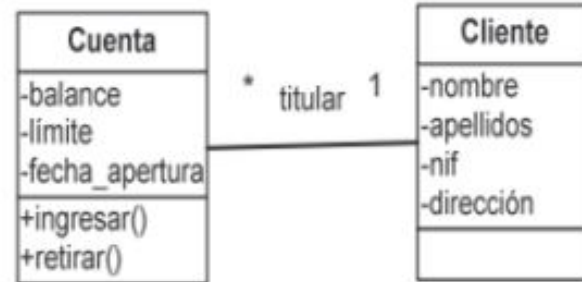
The background is a dark blue gradient with various abstract elements. There are faint, glowing lines forming arcs and paths. Scattered throughout are small, light blue dots and larger, more complex geometric shapes like hexagons and octagons, some of which are outlined in a lighter blue. The overall aesthetic is technical and futuristic.



# Relaciones entre clases: 1. Asociación

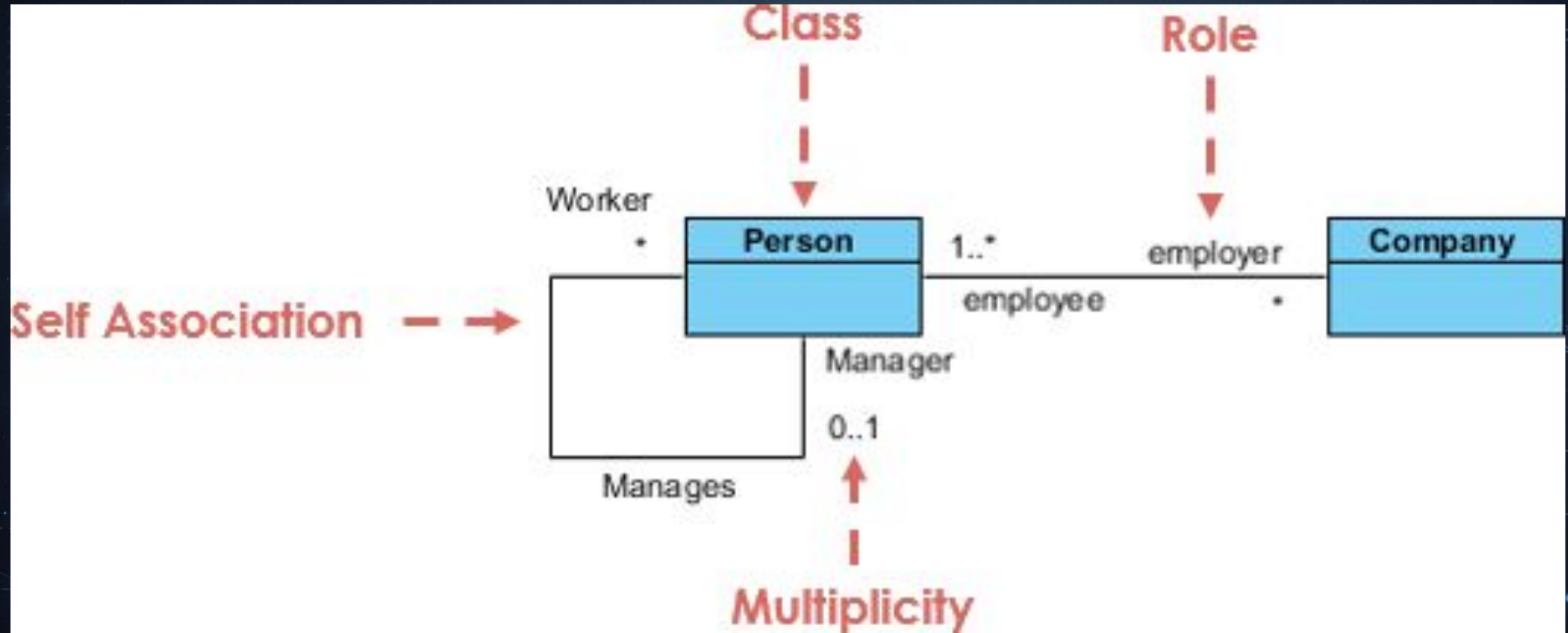


Asociación unidireccional

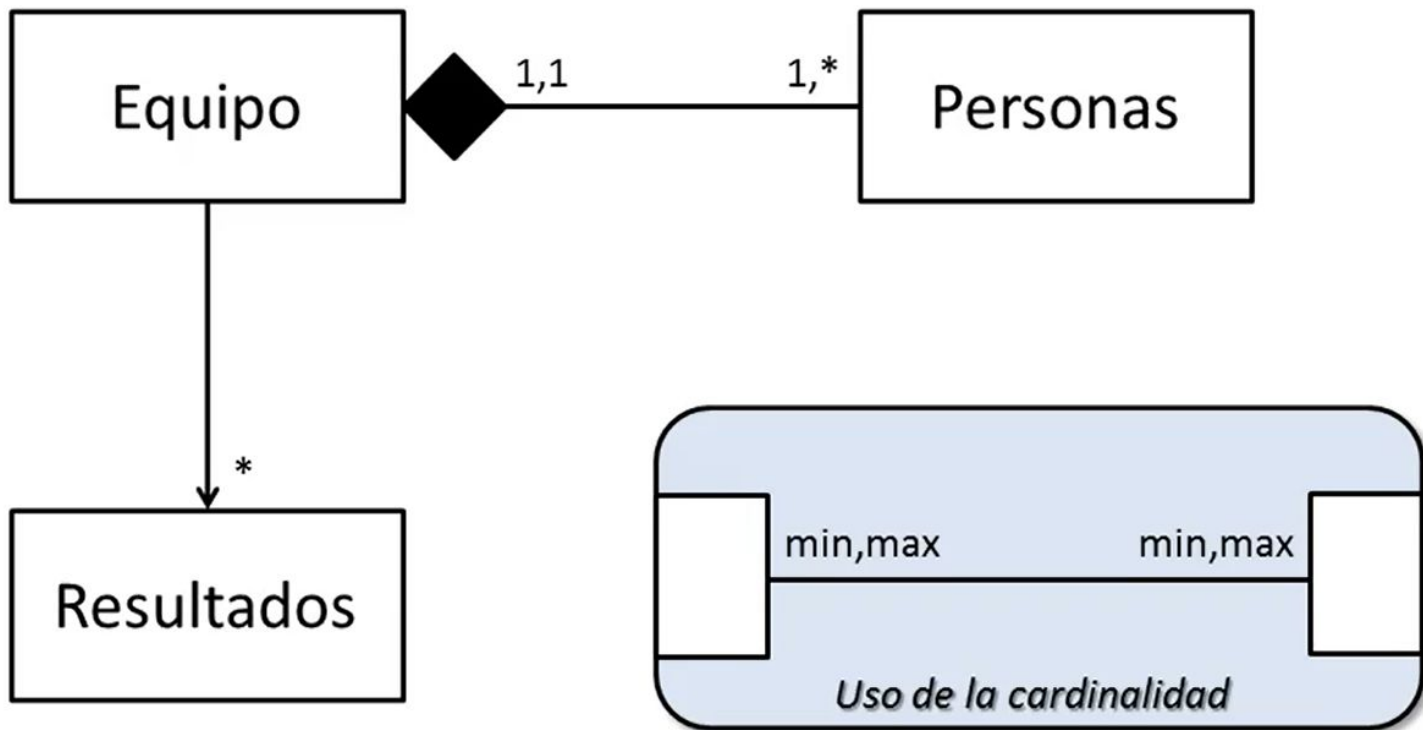


Asociación bidireccional

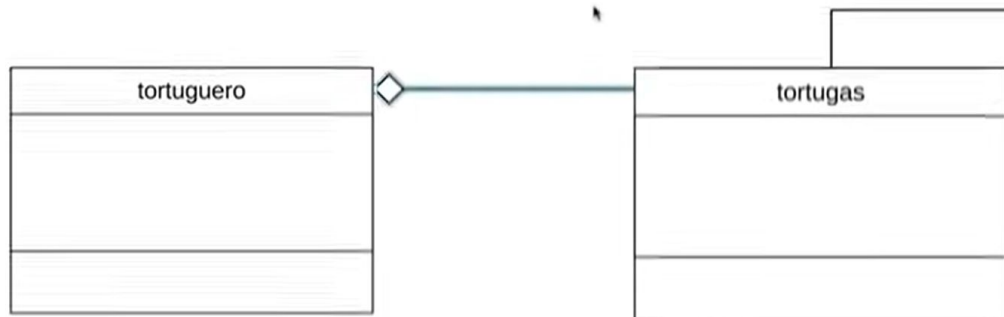
## Relaciones entre clases: 2. Autoasociación



# Asociaciones y Cardinalidad

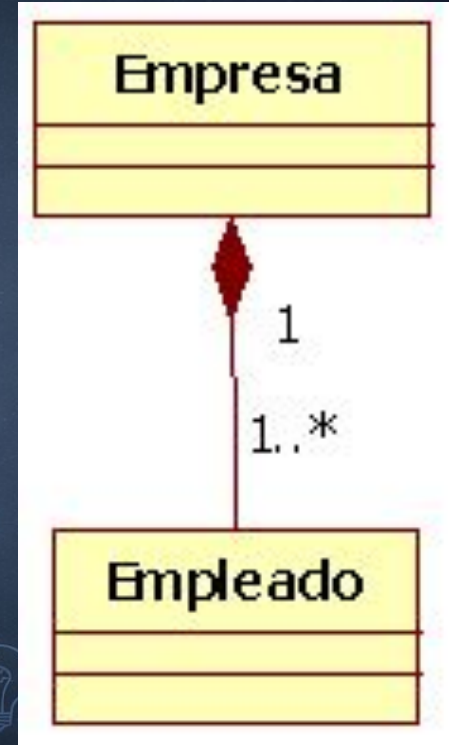
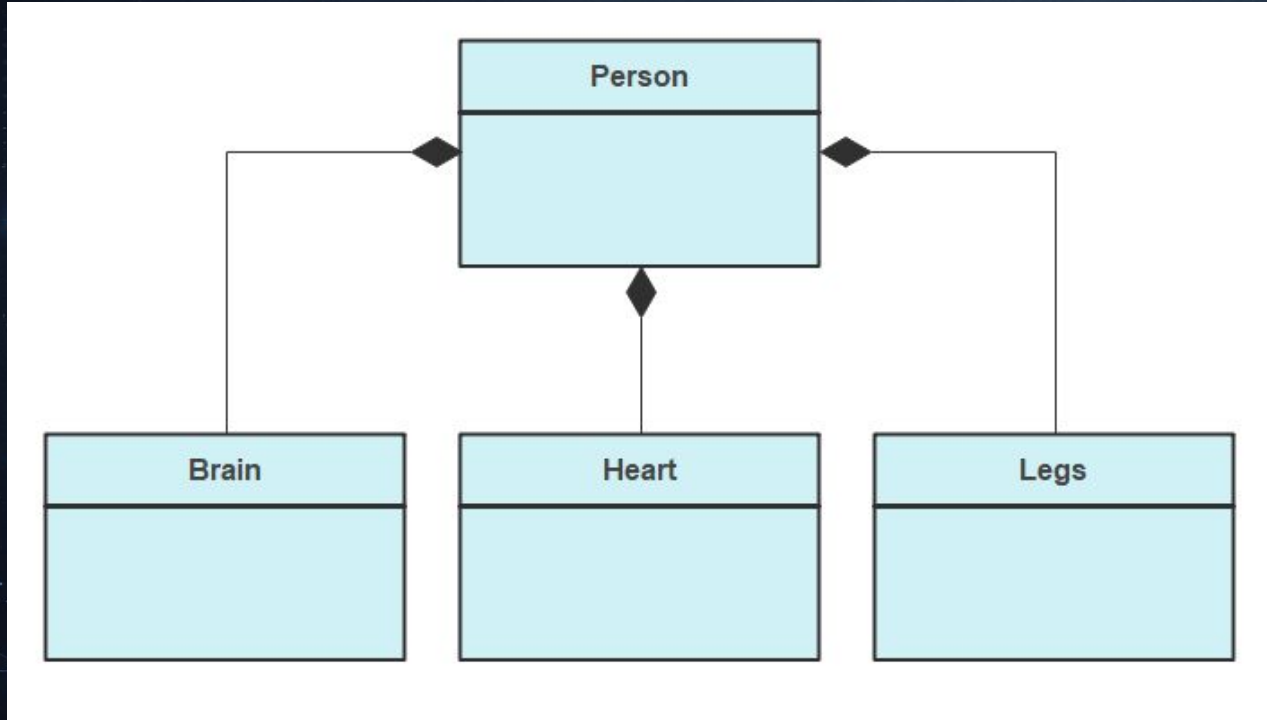


# Relaciones entre clases: 3. Agregación

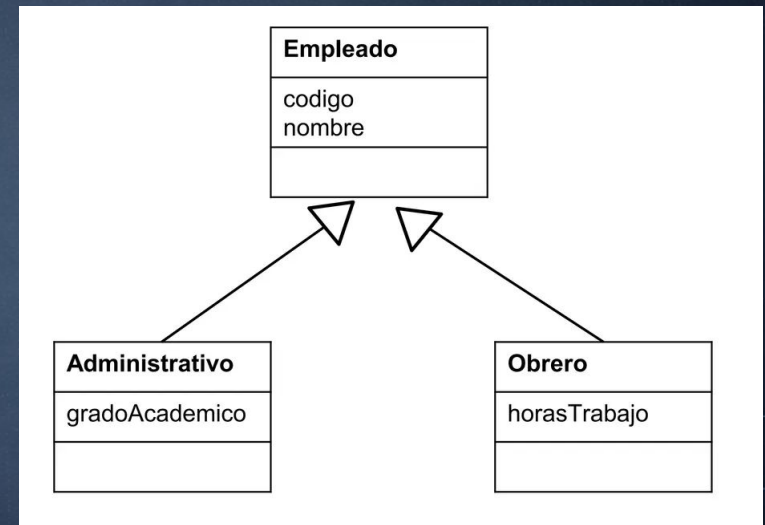
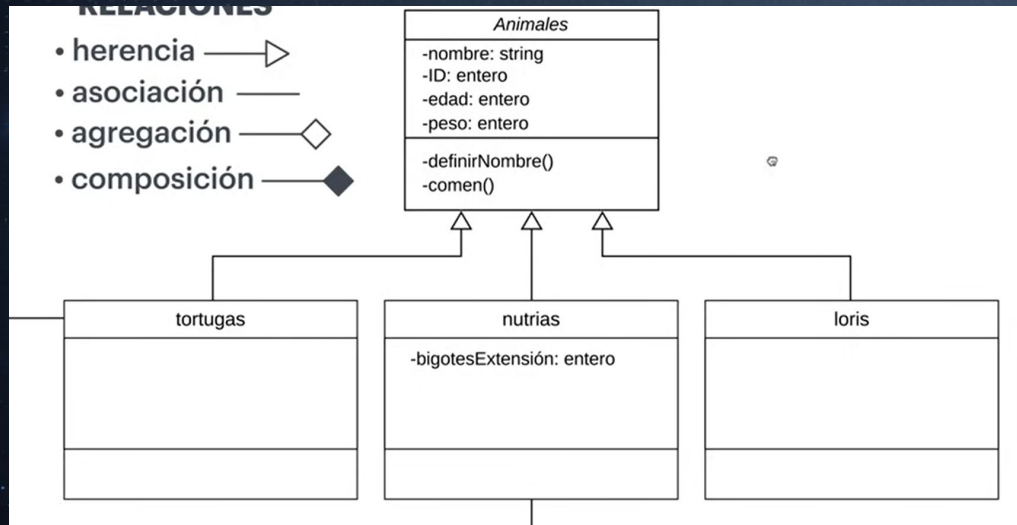




## Relaciones entre clases: 4. Composición



# Relaciones entre clases: 5. Generalización / Herencia



# Relaciones entre clases: 6. Dependencia

FacturaPrinter

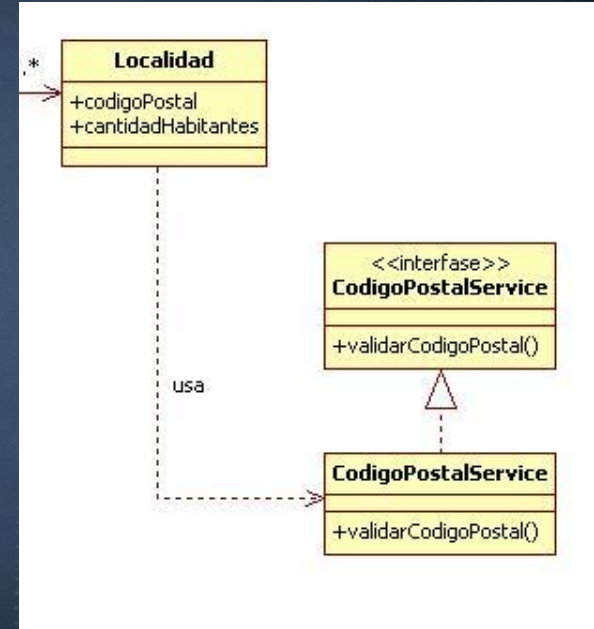
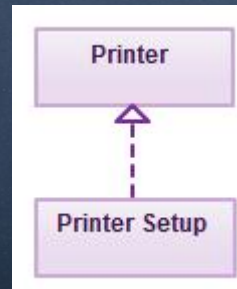
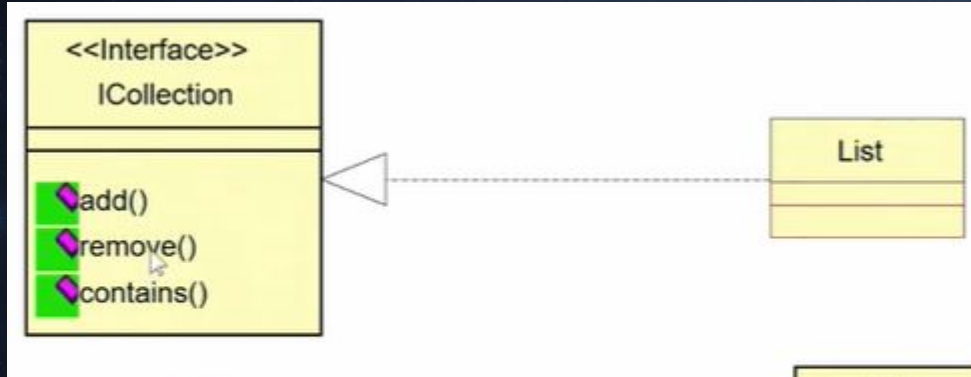
## Dependencia

Relación más débil que una asociación, donde el **cliente** solicita y el **servidor** provee un servicio.




FacturaPrinter -- - - - - > Factura

# Relaciones entre clases: 7. Realización (interfaces)





# Simbología e interpretación

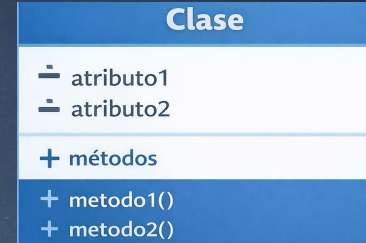
The background is a dark blue gradient with various abstract elements. There are faint, glowing blue lines forming arcs and paths. Scattered throughout are small, bright blue dots. Several geometric shapes, including hexagons and octagons, are visible, some with internal patterns. In the bottom right corner, there are icons of a lightbulb and an envelope, suggesting themes of ideas and communication.

# Simbología del diagrama de clases

Comprender la simbología es clave para interpretar y crear Diagramas de Clases efectivos.

Cada elemento tiene un propósito específico:

- **Rectángulo de Clase:** Dividido en 3 secciones para **Nombre**, **Atributos** y **Métodos**.
- **Visibilidad:**
  - + público: Accesible desde cualquier lugar.
  - - privado: Solo accesible desde dentro de la clase.
  - # protegido: Accesible desde la clase y sus subclases.
  - ~ paquete: Accesible dentro del mismo paquete.

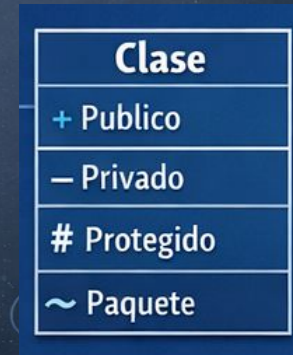


Relaciones y sus símbolos:

- **Herencia / Generalización:** flecha con triángulo vacío.
- **Agregación:** línea con diamante vacío.
- **Composición:** línea con diamante relleno.
- **Asociación:** línea continua simple.
- **Dependencia:** línea punteada con flecha.



- **Notas y Restricciones:** Se pueden añadir para aclarar detalles específicos o reglas de negocio.



# Interpretación de un Diagrama de Clases

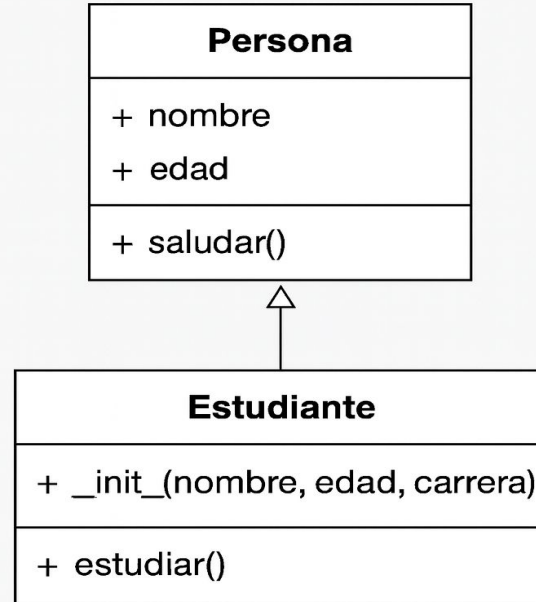
En este diagrama se muestran dos clases: **Persona** y **Estudiante**.

**Persona** es la **clase base o padre**. Contiene los **atributos** nombre y edad, y el **método** saludar().

**Estudiante** es la **subclase o hija**, que **hereda** todos los atributos y métodos de Persona.

Además, agrega su propio método estudiar().

La **flecha con triángulo hacia arriba (▲)** representa la **herencia**, indicando que Estudiante **proviene de** Persona.





# Representación de un diagrama de clases en código Python

Ejemplo.py Ejemplo.py...

```
1 # Clase base o padre
2 class Persona:
3     def __init__(self, nombre, edad):
4         self.nombre = nombre
5         self.edad = edad
6
7     def saludar(self):
8         print(f"Hola, soy {self.nombre} y tengo {self.edad} años.")
9
10
11 # Subclase o hija
12 class Estudiante(Persona):
13     def __init__(self, nombre, edad, carrera):
14         super().__init__(nombre, edad) # hereda atributos de Persona
15         self.carrera = carrera # nuevo atributo
16
17     def estudiar(self):
18         print(f"{self.nombre} está estudiando {self.carrera}.")
19
20 # * Ejemplo de ejecución
21
22 # Se crea un objeto de tipo Estudiante con los valores indicados.
23 alumno = Estudiante("Luis", 20, "Programación")
24 # Se llama al método saludar() heredado de la clase Persona.
25 alumno.saludar()
26 # Se llama al método estudiar() propio de la clase Estudiante.
27 alumno.estudiar()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Zerao@Vito MINGW64 ~/Documents/Diagrama de clases

```
$ python -u "c:\Users\Zerao\Documents\Diagrama de clases\Ejemplo.py"
```

Hola, soy Luis y tengo 20 años.

Luis está estudiando Programación.

