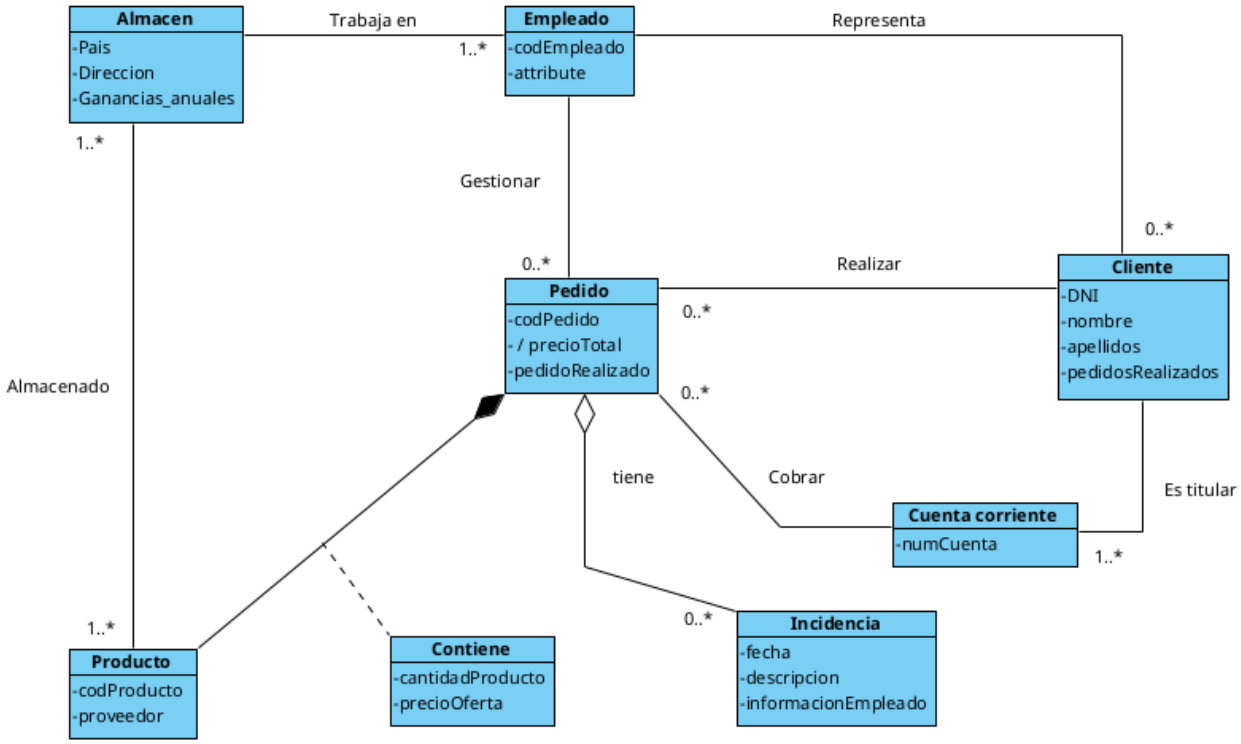
Control 1

Fundamentos de Ingeniería del Software

Lucas Hidalgo Herrera

---

1. Diagrama de conceptos



2. Justificaciones

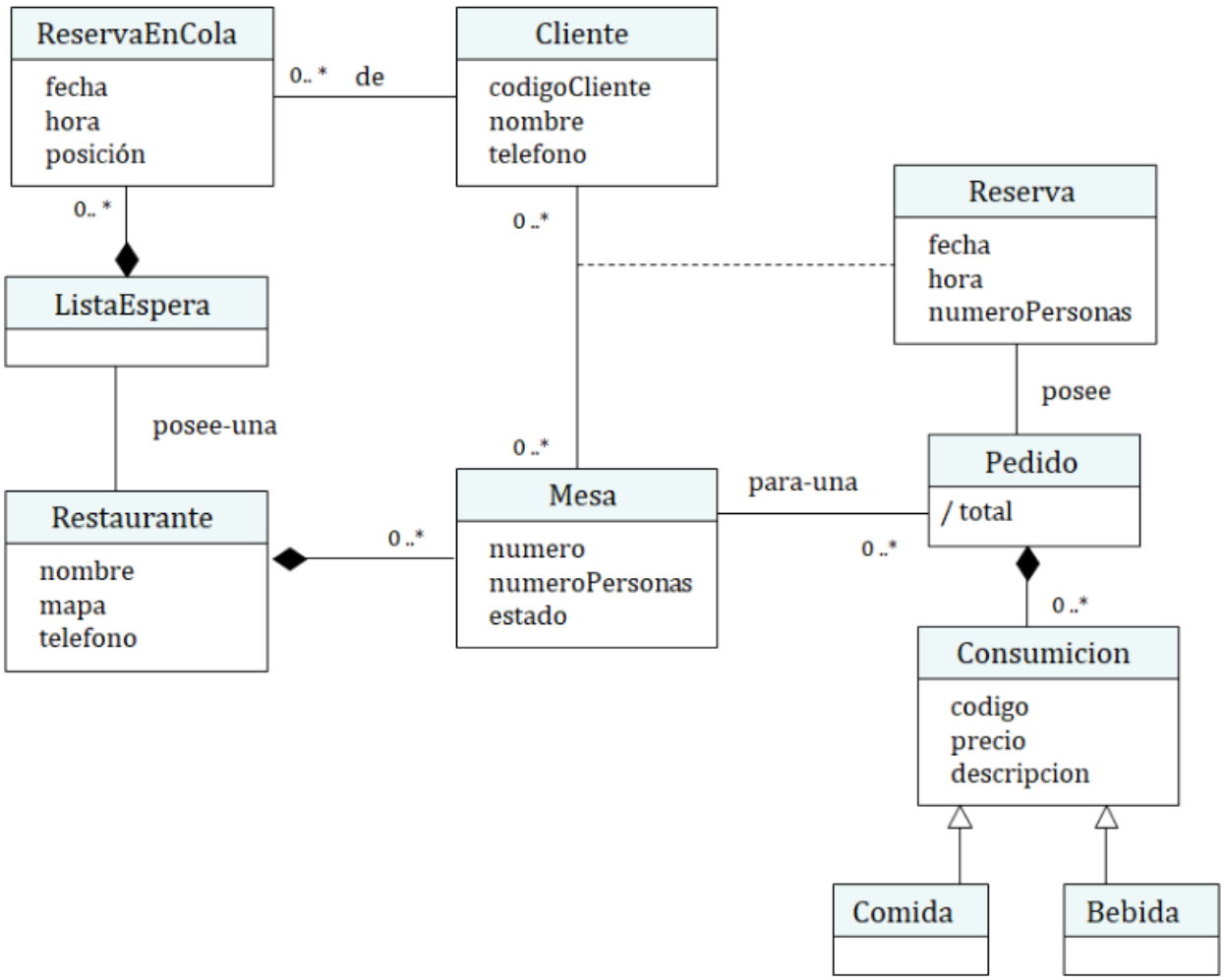
Con respecto a las justificaciones más importantes:

- **Composición producto-pedido:** he considerado una composición porque un *Producto* forma parte de un *Pedido* y sin los productos no tiene sentido el pedido. Por tanto, es una relación parte-todo de carácter fuerte.
- **Agregación incidencia-pedido:** he considerado la relación de agregación debido a que, en este caso, no es una relación parte-todo de carácter fuerte sino débil. Esto es así ya que no siempre habrá una *Incidencia* en un *Pedido* y no tiene sentido que un *Pedido* deje de existir cuando no tenga una *Incidencia*.
- **Relación Contiene:** Debido a que la oferta se aplica dependiendo de la cantidad de producto comprado de un determinado producto, he decidido imponer que la cantidad de *Producto* sea un atributo de la relación; de la misma manera ocurre con el precio de la oferta.

Puede parecer un error pues no asemeja que la oferta se aplique cuando se supera una cantidad de producto comprado, pero esto podría ser objetivo de una operación como *aplicarOfertaSi(cantidadumbral,oferta)* que forma parte del Diagrama de Secuencia del Sistema.

- **Relación representa:** En un principio, puede parecer claro realizar una generalización, es decir, que el concepto *Representante de ventas* sea un generalizado de *Empleado* pues un representante es un *Empleado*, pero realmente no hay una función que los distinga entre ellos salvo representar.
  Por tanto, he considerado mejor opción realizar la asociación; además, con esto ya se refleja la relación entre algunos empleados y los clientes.
- **Concepto Proveedor:** He decidido no crear un concepto *Proveedor* sino añadir un atributo al producto. En el enunciado no aparece el *Proveedor* como un concepto que tenga relevancia en el problema; por tanto, en el caso de que este diagrama de conceptos refleje una base de datos no considero que se deba guardar mucha información sobre este concepto.

3. Modelo conceptual



4. Contratos

Nombre	añadirListaEspera(idRestaurante,idCliente,fecha,hora)
Responsabilidad	Crear una reserva en cola del cliente idCliente para la lista de espera del restaurante *idRestaurante*.
Tipo	Sistema de Gestión de Reservas de Restaurantes
Notas	Como valores correctos de *fecha* y *hora* se entiende que es un valor posterior al momento de la realización de la reserva. No es necesario devolver nada, pero por cuestiones de sencillez a la hora de encajar en el sistema devolvemos si ha habido éxito o no, podrá permitir la realización de comprobaciones.
Excepciones	• Si el *idRestaurante* no existe. • Si el *idCliente* no existe. • Si idRestaurante e idCliente no disponen de un formato correcto. • Si *fecha* y *hora* no disponen de valores correctos o no tienen el formato correcto.
Salida	Booleano que refleja si la operación ha tenido éxito o no.
Precondiciones	• El restaurante *idRestaurante* no tiene ninguna reserva prevista anteriormente en la *fecha* y *hora* determinadas. • El cliente *idCliente* no ha reservado previamente en el restaurante *idRestaurante* en un horario que solape al deseado.

Postcondiciones	• Se creó un objeto *ReservaEnCola*, **res** con atributos:
	  ◦ **fecha:** toma el valor *fecha* de la llamada a la operación.
	  ◦ **hora:** toma el valor *hora* de la llamada a la operación.
	• Se añadió el objeto **res** a la lista de espera del restaurante con *idRestaurante*.

Nombre	añadirPedido(idRestaurante, NumeroMesa, listaIdConsumición)
Responsabilidad	Añade un pedido a la mesa *NumeroMesa* del restaurante *idRestaurante* con los datos de la consumición *listaIdConsumición*.
Tipo	Sistema de Gestión de Reservas de Restaurantes
Notas	No es necesario devolver nada, pero por cuestiones de sencillez a la hora de encajar en el sistema devolvemos si ha habido éxito o no; podrá permitir la realización de comprobaciones.
Excepciones	• Si el *idRestaurante* no existe o tiene un formato incorrecto. • Si el restaurante *idRestaurante* no dispone de una mesa con el número de mesa pasado como argumento. • Si hay algún identificador de consumición no válido en la lista *listaIdConsumición*.
Salida	Booleano que refleja si la operación ha tenido éxito o no.
Precondiciones	• *listaIdConsumición* debe contener, al menos, un elemento.
Postcondiciones	• Se creó un objeto *Pedido*,**ped**, conformado por los objetos *Consumición*, **cons**, de cada uno de los identificadores de la lista de consumiciones *listaIdConsumición.* • Se asoció el objeto **ped** a la mesa con número de mesa *NumeroMesa* del restaurante *idRestaurante*.