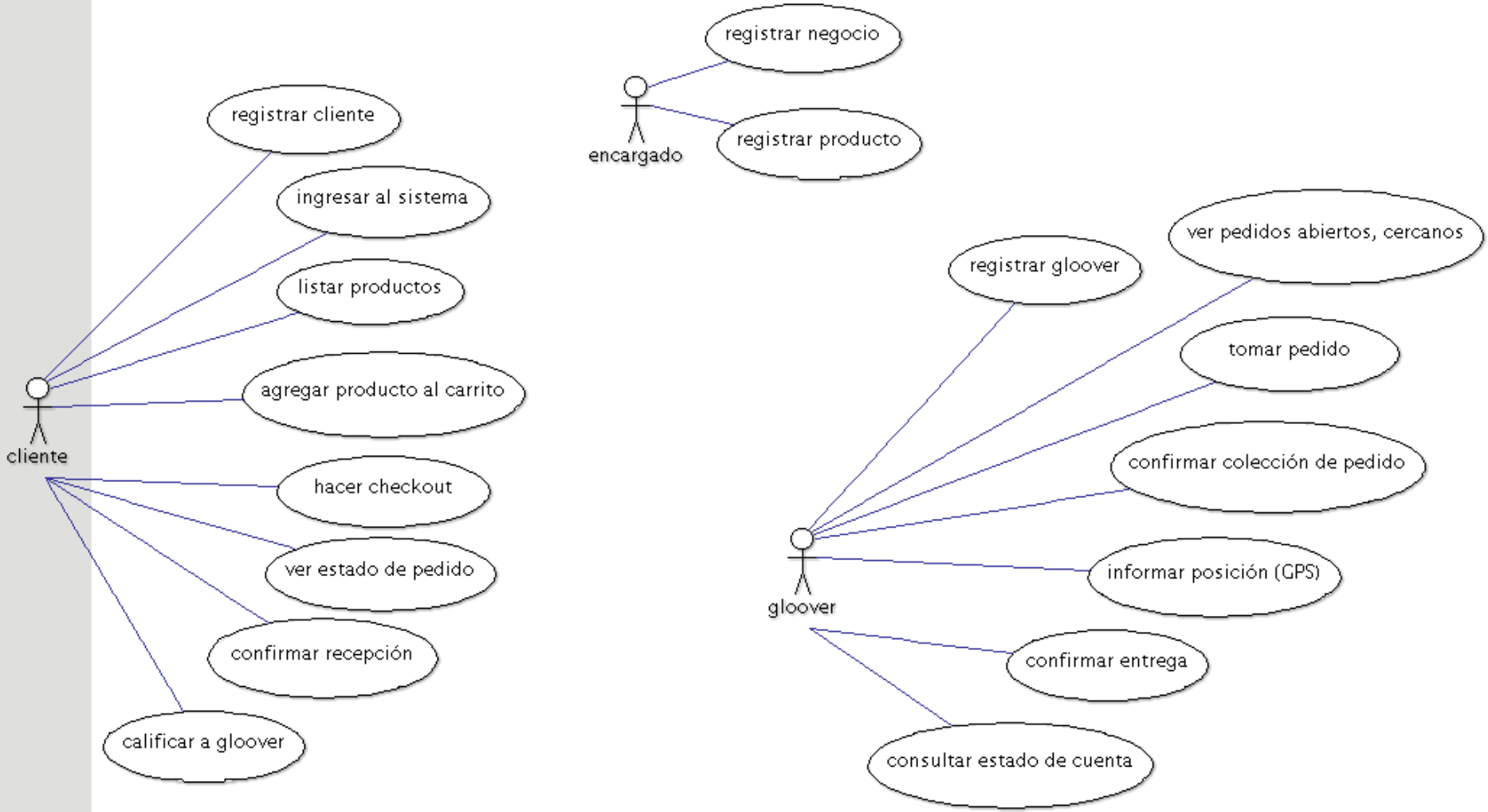
Casos de Uso del sistema Gloovo



El usuario compra….

- Para comprar el cliente debe:
  - Loguearse
  - Poner cosas en el carrito
  - Hacer el pedido: con todo lo que hay en el carrito

- Mensaje login (email, password) Enviado a gloovo

login (email, password)

cliente = clients.findAndValidate (email, password)

return (cliente)

Tenemos que ver como tratar el caso de error!!!

Manejar el carrito de compras

- Necesitamos una clase Carrito. Un objeto carrito (por ahora) tiene una coleccion de productos (en esta version un producto tiene un negocio asociado).
- Quien conoce al carrito? (**ver inicializaciones**)

- Mensaje agregarProducto (p) en Clase Cliente

agregarProducto (p)

carrito.agregar (p)

Constructor Cliente corregido

Cliente (nomb, dir, mail, pass)

nombre= nomb

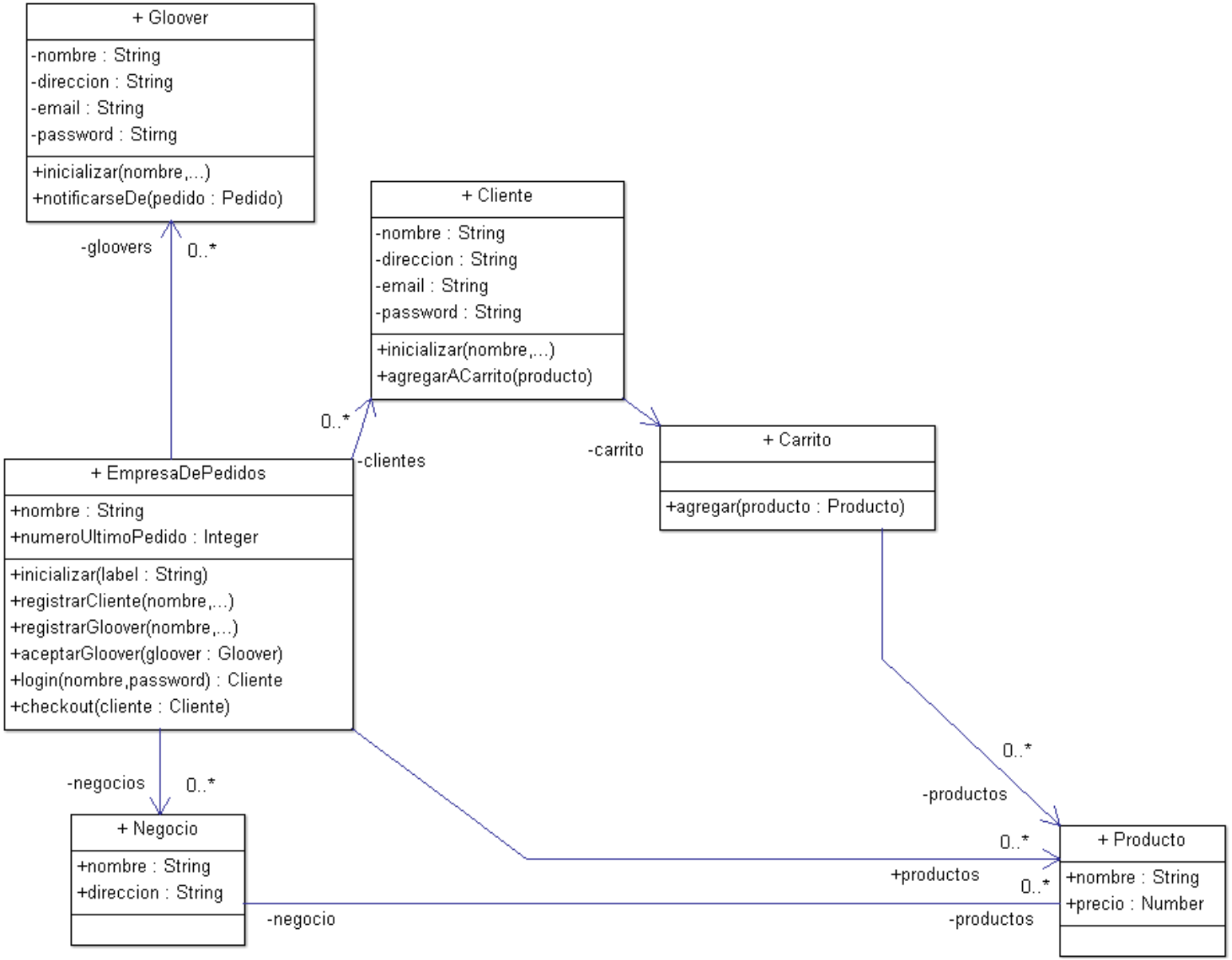
direccion= dir

email = mail

password= pass

carrito= new Carrito

ClienteConCarrito



Generar un pedido

- El cliente decide “cerrar” su pedido y comprar
- Mensaje checkout (cliente) en objeto gloovo
- Por que en gloovo y no en cliente? (ver proceso checkout)

checkout (cliente, direccionEnvio,medioDePago)

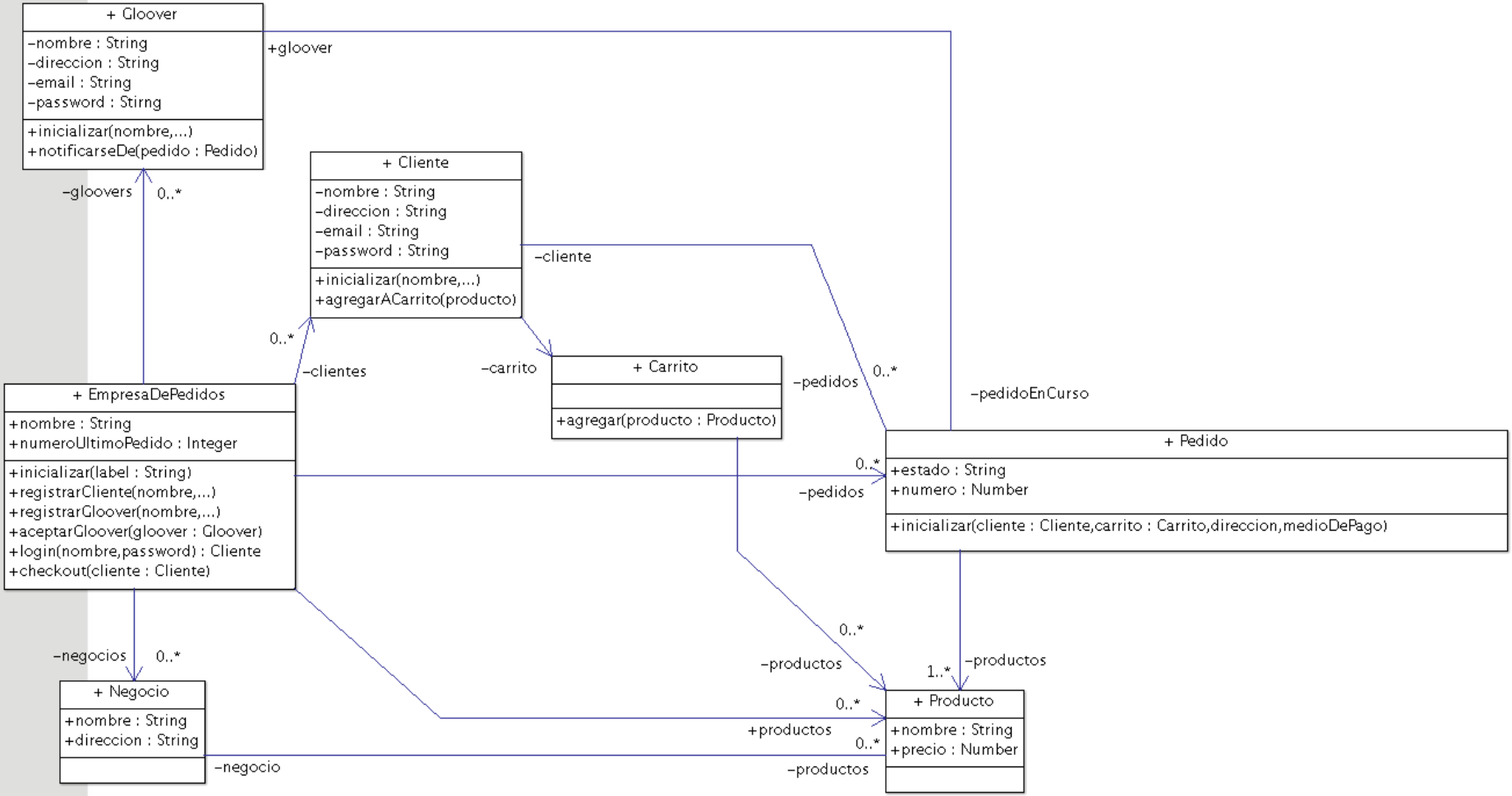
p= new Pedido (cliente, cliente.carrito, direccionEnvio, medioDePago)

gloovers.notificarseDe (p) ***(Como escribimos esto?)***

pedidos.add (p)

Observar relacion Cliente-Pedido, Metodo carrito en Cliente?

Diseño actual



Notificacion de un pedido nuevo

- En la clase Gloover tenemos un metodo

  notificarseDe (p)

- Supongamos que la notificacion es via Whatsapp.

- En la clase Gloover tendriamos codigo especifico necesario para enviar un Whatsapp desde nuestro sistema.

- No parece ser una incumbencia de esa clase.

- Una solucion mejor es delegar esta tarea en un objeto de una clase especifica (ComunicacionWhatsapp)

Notificacion…

- Entonces en Clase Gloover

```
notificarseDePedido (p)
    c= new ConexionConWhatsapp
    c enviarMensaje.(numeroWhatsapp, p)
```

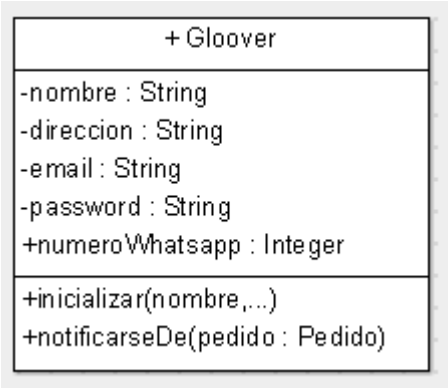
Entonces:

un gloover debe conocer su numeroWhatsapp

Que pasa con p (el pedido actual)?

Que mandamos por whatsapp? UN TEXTO

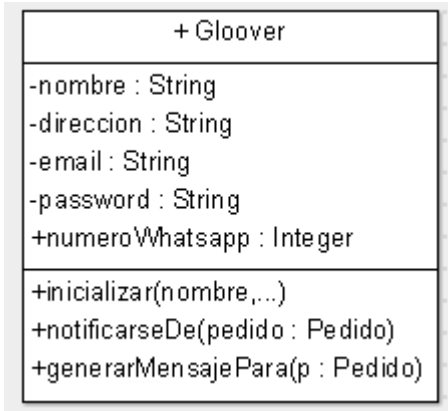
Puede recibir el objeto c un pedido?



Notificacion….

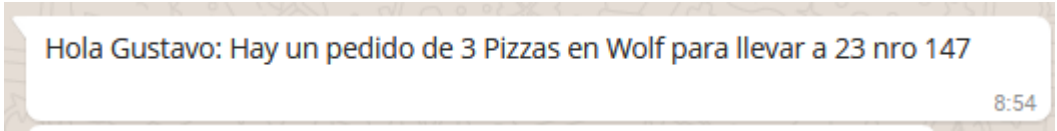
notificarseDePedido (p)

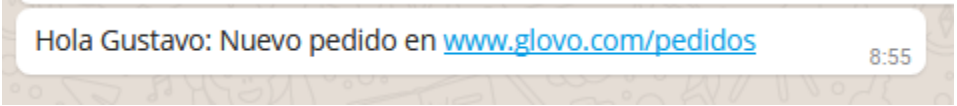
```
c = new ConexionConWhatsapp
texto= this.generarMensajePara (p)
c.enviarMensaje (numeroWhatsapp, texto)
```



generarMensajePara (p)

-que necesitamos aca para retornar un mensaje util para el gloover?





Un gloover se postula…

- El gloover recibe una notificacion de pedido y se postula
- Donde esta el metodo correspondiente?
- Quien lo “dispara”?

- Opciones:
  - En la clase Gloover

    postularsePara (p) Y Tendria que avisarle a gloovo (lo conoce?)

  - En la clase EmpresaDePedidos

    postularGlover (gloover, pedido)

Un gloover se postula….y se asigna

- Esté donde esté el “primer” metodo (dependera de la configuracion de la interfaz entre otras cosas), en EmpresaDePedidos se procesa el pedido

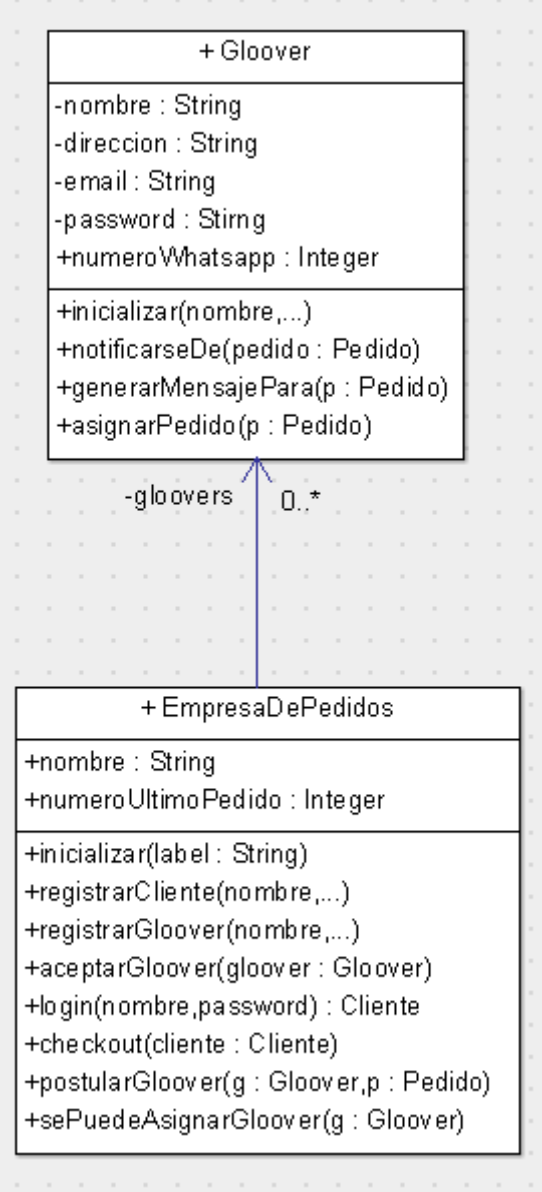
postularGloover (gloover, pedido)

this.sePuedeAsignarGloover (gloover) Mirar si el tipo tiene denuncias, etc

gloover.asignarPedido (p)

LE TENEMOS QUE AVISAR AL CLIENTE? Como hacemos?

EmpresaDePedidos y Gloover modificados



Asignar un pedido a un gloover, avisarle al usuario

- En la Clase Gloover tenemos el metodo

asignarPedido (pedido)

pedidoEnCurso =pedido

pedido.asignarGloover (self)

- En la clase Pedido

asignarGloover (g)

gloover= g

estado = “glooverAsignado”

CUAL ERA EL ESTADO INICIAL?

Temas en Gloovo que dan origen a jerarquias

- Variacion en las formas de notificar a los Gloovers
- Variacion en como decidir a que Gloovers notificar
- Medios de pago del cliente
- Distintos tipos de pedidos….

Notificacion a los Gloovers

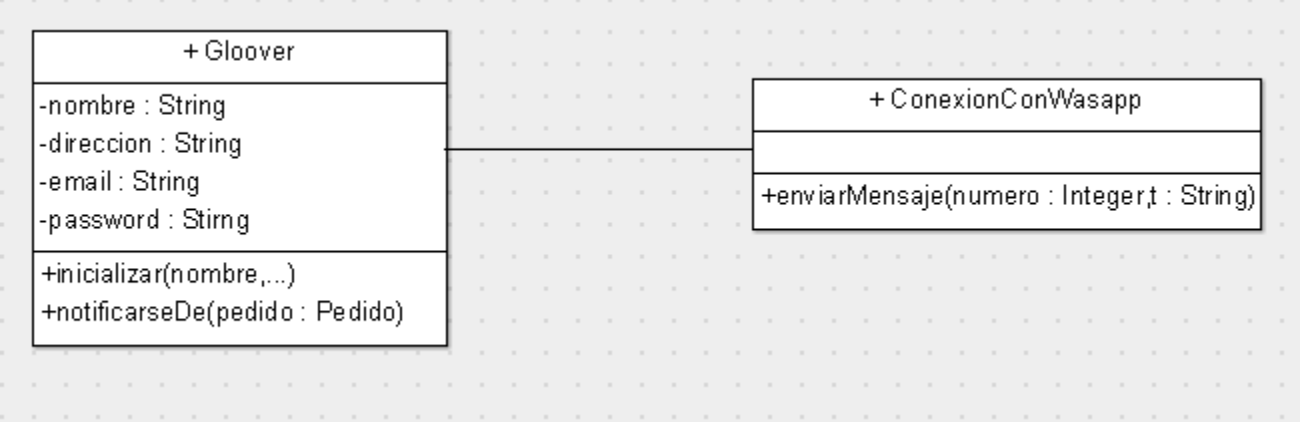
- En la Clase Gloover teniamos

notificarseDePedido (p)

c=new ConexionConWhatsapp

texto = this.generarMensajePara (p)

c.enviarMensaje (numeroWhatsapp, texto)



Que hacemos si queremos darle al Gloover la posibilidad de elegir como notificarse: Telegram, Facebook, Whatsapp, etc….

Opciones….

- Ponerle un “tipoDeNotificacion” como variable y luego el if correspondiente.

- Problemas con esto? Que info tenemos en las variables de instancia? El numero de cada posible cuenta?

Solucion: Sacar el problema de Gloover

- Tenemos un objeto notificador

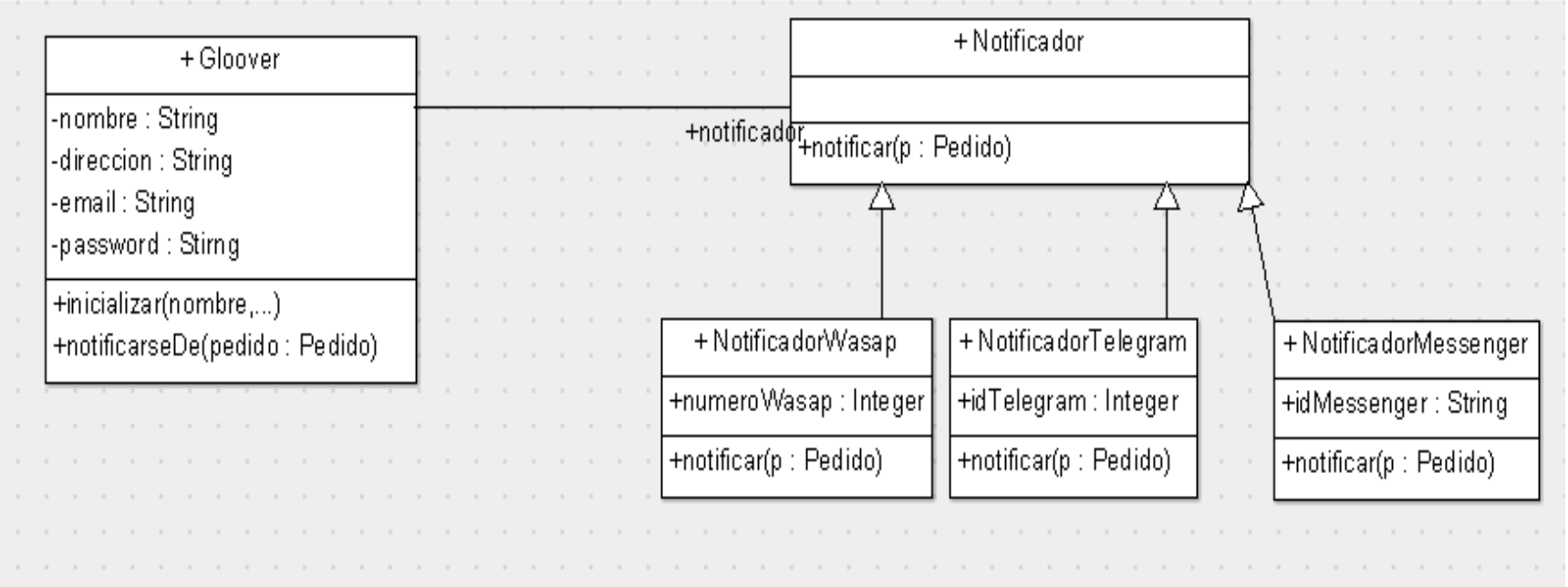
- Entonces el gloover DELEGA en ese objeto y en vez de:

```
notificarseDePedido (p)
    c:=new ConexionConWhatsapp
    texto= this.generarMensajePara (p)
    c enviarMensaje (numeroWhatsapp, texto)
.....
  notificarseDePedido (p)
   notificador.notificar (p)
```

- Que tiene que saber hacer el notificador?
- Como lo configuramos? Cuando?

Jerarquia de Notificadores

- Como manejamos la variabilidad de formas de notificacion



Configuracion de los Notificadores

- Cada gloover conoce un objeto notificador al que le delega la tarea de notificarlo (con los datos que tiene el notificador)
- Cuando inicializamos el gloover, tenemos que crear un Notificador (de la clase correspondiente) e inicializarlo
- Como seria este proceso? Donde lo hacemos? Cuando? De donde sacamos los datos?

Inicializacion Notificadores

- En Clase Gloover

inicializarNotificadorWasap (numeroWasap)

notificador= new NotificadorWasap (numeroWasap)

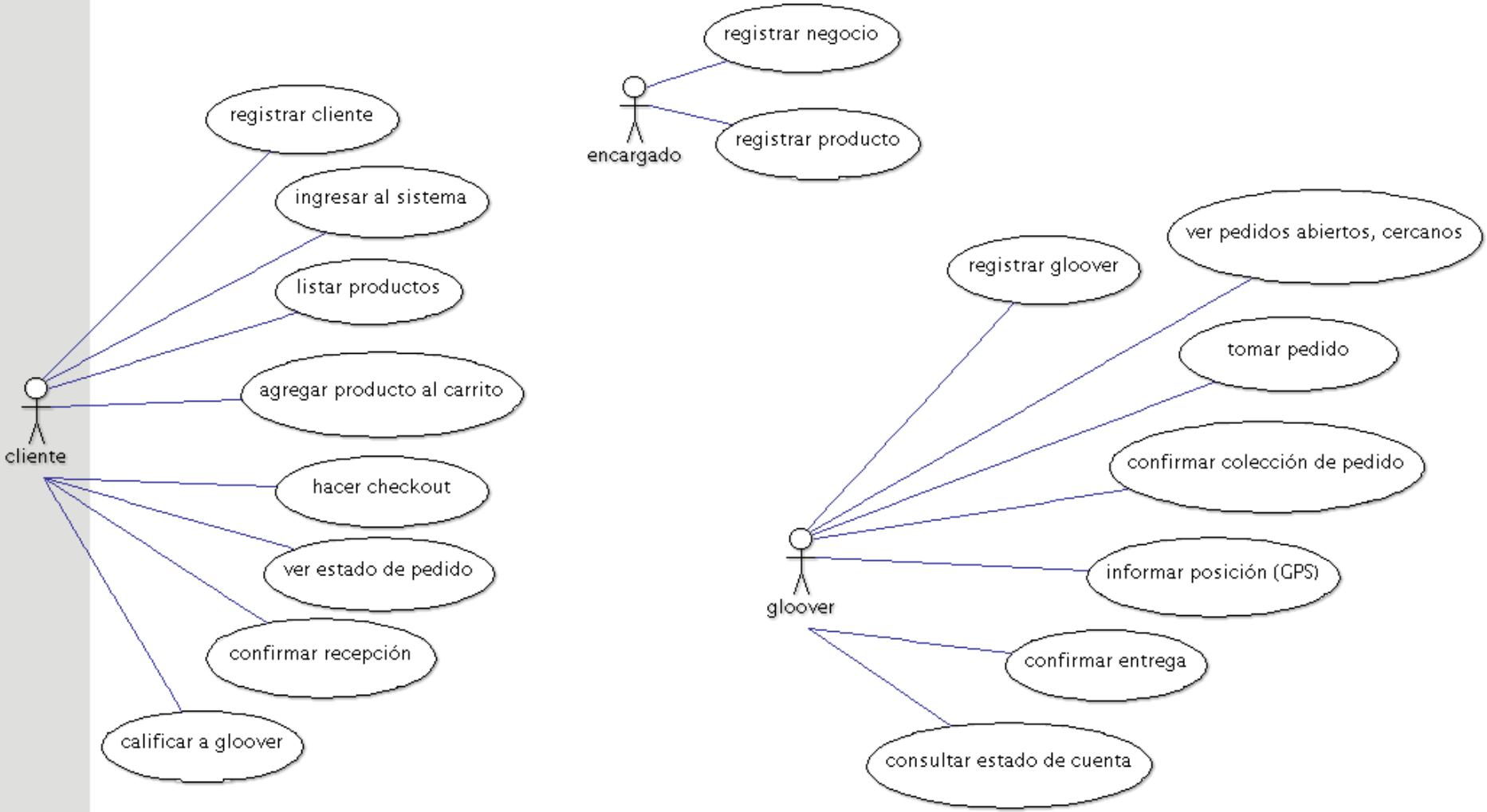
inicializarNotificadorTelegram (idTelegram)

notificador= new NotificadorTelegram(idTelegram)

notificador initialize

Como y cuando se disparan estos metodos?

Cobrarle al cliente



Gloover entrega un pedido

- Opciones:

  1-Cuando Gloover entrega se cobra al cliente

  2-Recien lo hacemos cuando el cliente confirma que lo recibio

- La opcion 1 parece mas cercana a los estandars de comercio electronico. De hecho se cobra cuando se envia el producto y no hay notificacion de recepcion

- La opcion 2 parece mas “end user-friendly” pero es poco realista

Gloover entrega pedido

- En Clase EmpresaDePedidos

pedidoEntregado (pedido)

this.actualizarPedidosPendientes (pedido)

(pedido.cliente).pagar (pedido)

- Observese que el cliente puede tener muchos pedidos abiertos

Cliente paga

- En Clase Cliente

pagar (pedido)

precio = pedido calcularPrecio

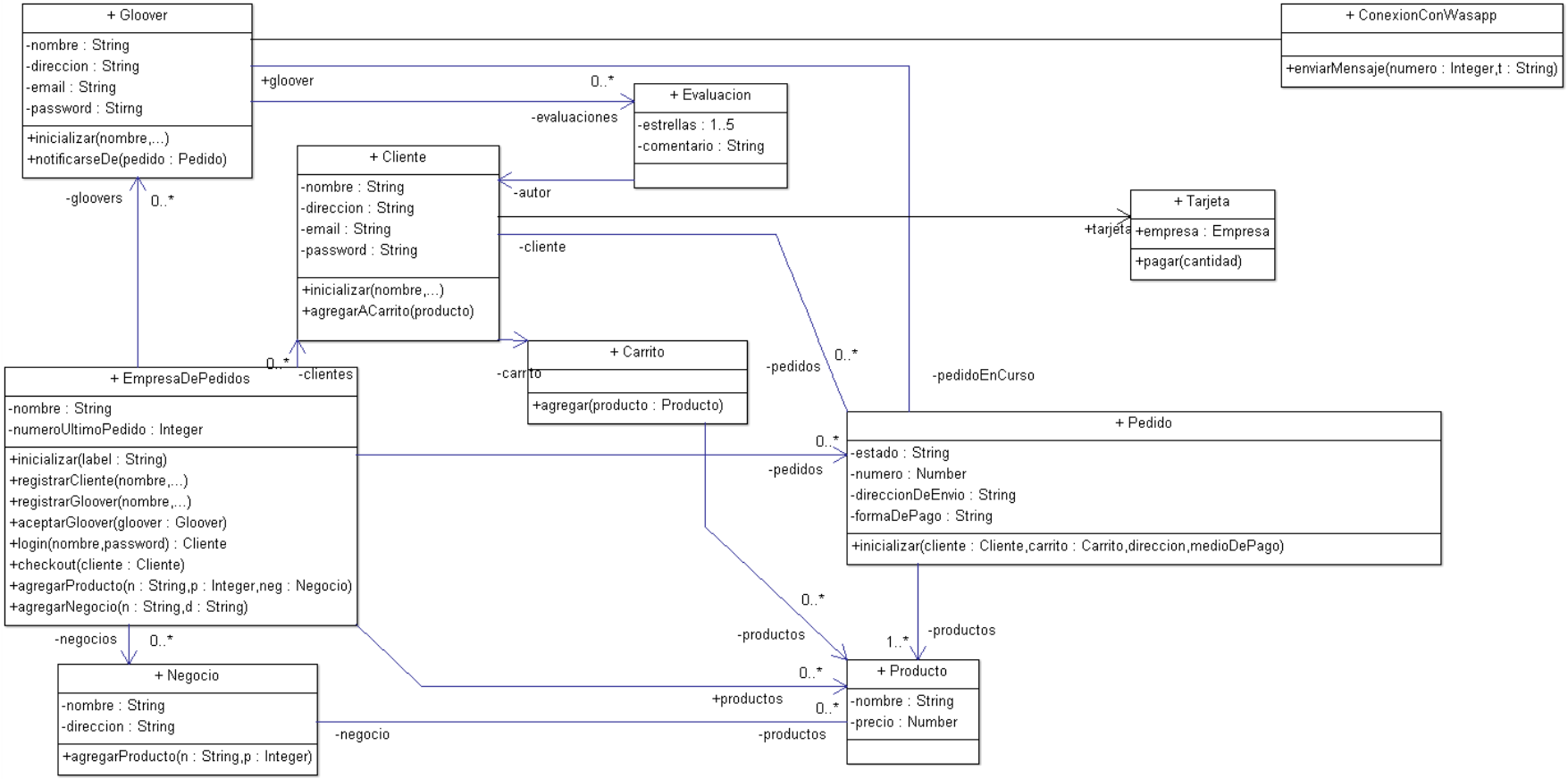
tarjeta.pagar (precio)

- En Clase Pedido

calcularPrecio

return (suma de precios de productos + precio envio)

Incluimos la tarjeta en el modelo



Mirando el pago con detalle

- La expresion:

  tarjeta.pagar (precio)

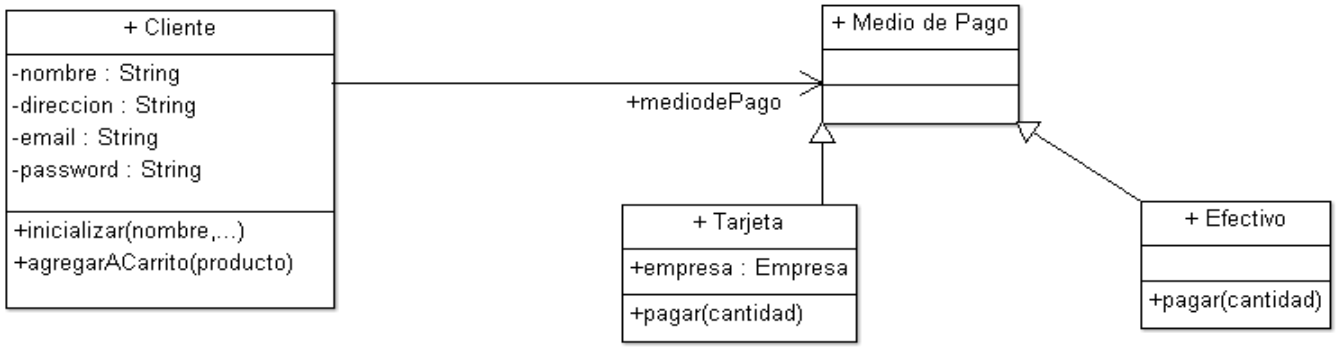
- Asume:

  El cliente conoce a un objeto “tarjeta” (de credito) que sabe “pagar”

- Como manejariamos el caso de otras formas de pago (efectivo, debito, etc)?

- La solucion es “generalizar” el concepto de tarjeta en “medio de pago”

Medio de pago



- Tenemos que redefinir la variable de instancia “tarjeta”
- Tenemos que implementar el metodo pagar en la clase Efectivo