Patrones De Diseño

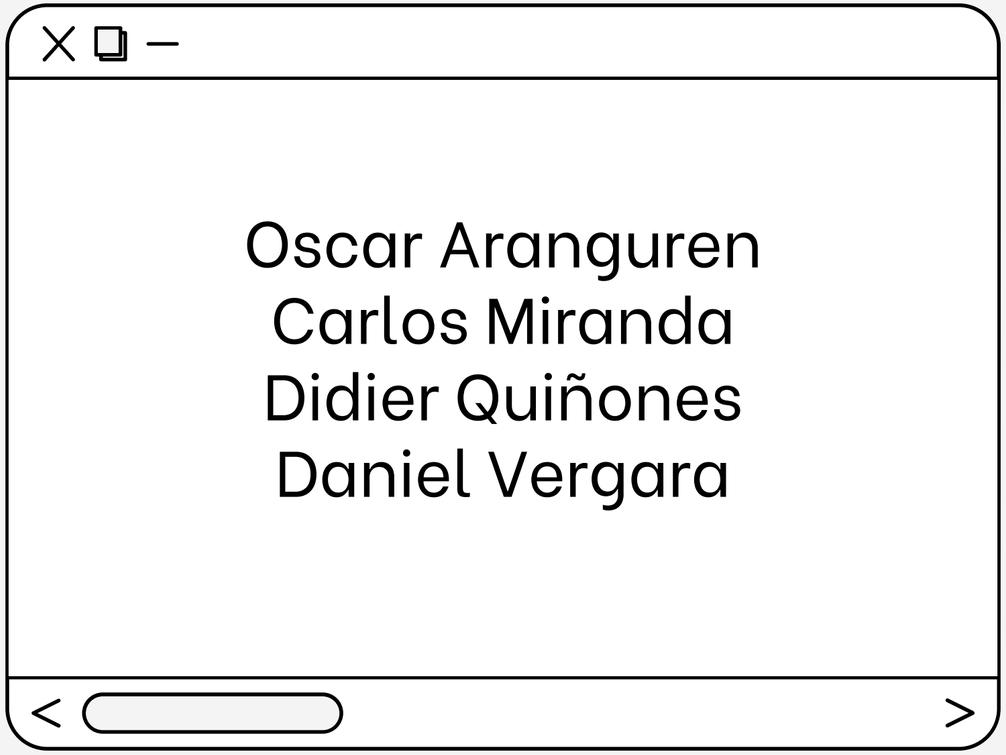
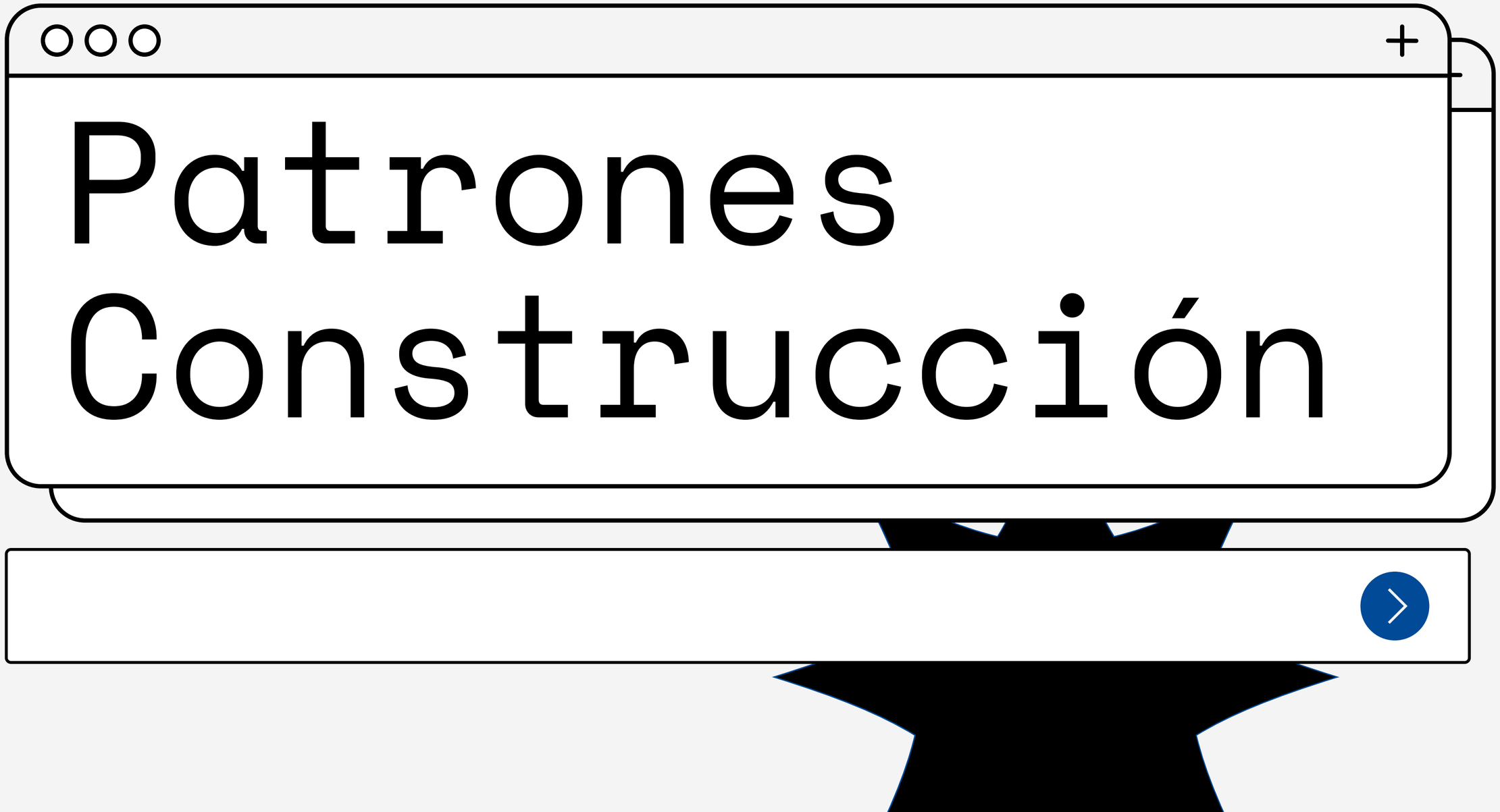
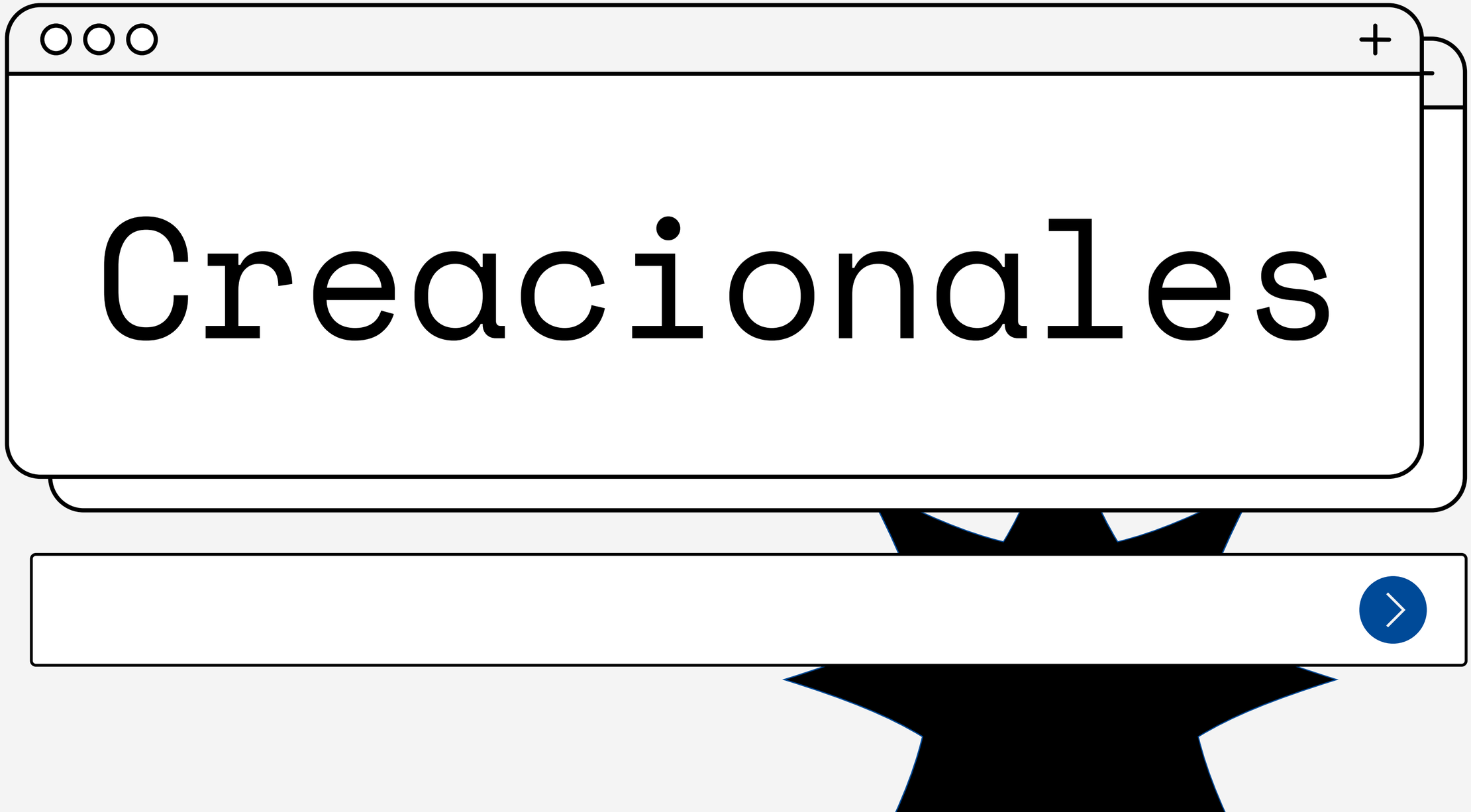

Oscar Aranguren
Carlos Miranda
Didier Quiñones
Daniel Vergara

Tabla De Contenido

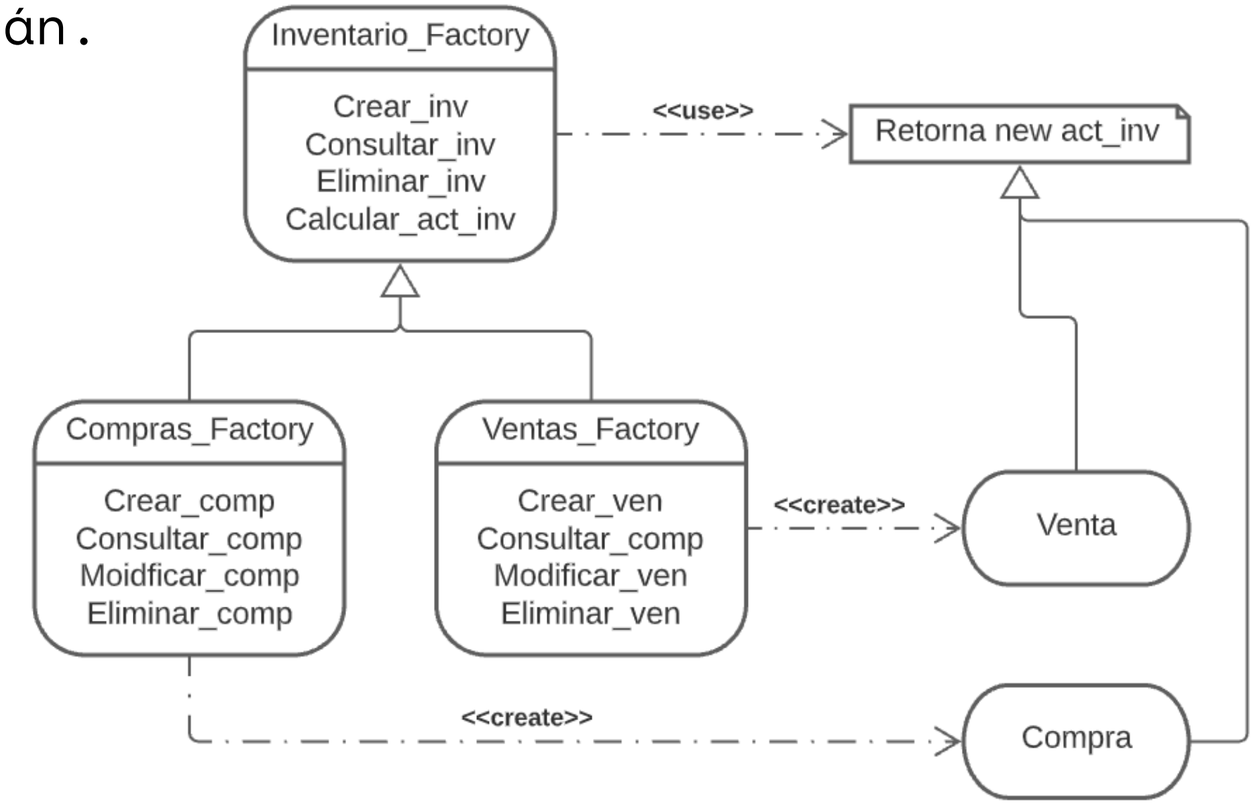
Patrones Construcción




Creacionales

Factory Method

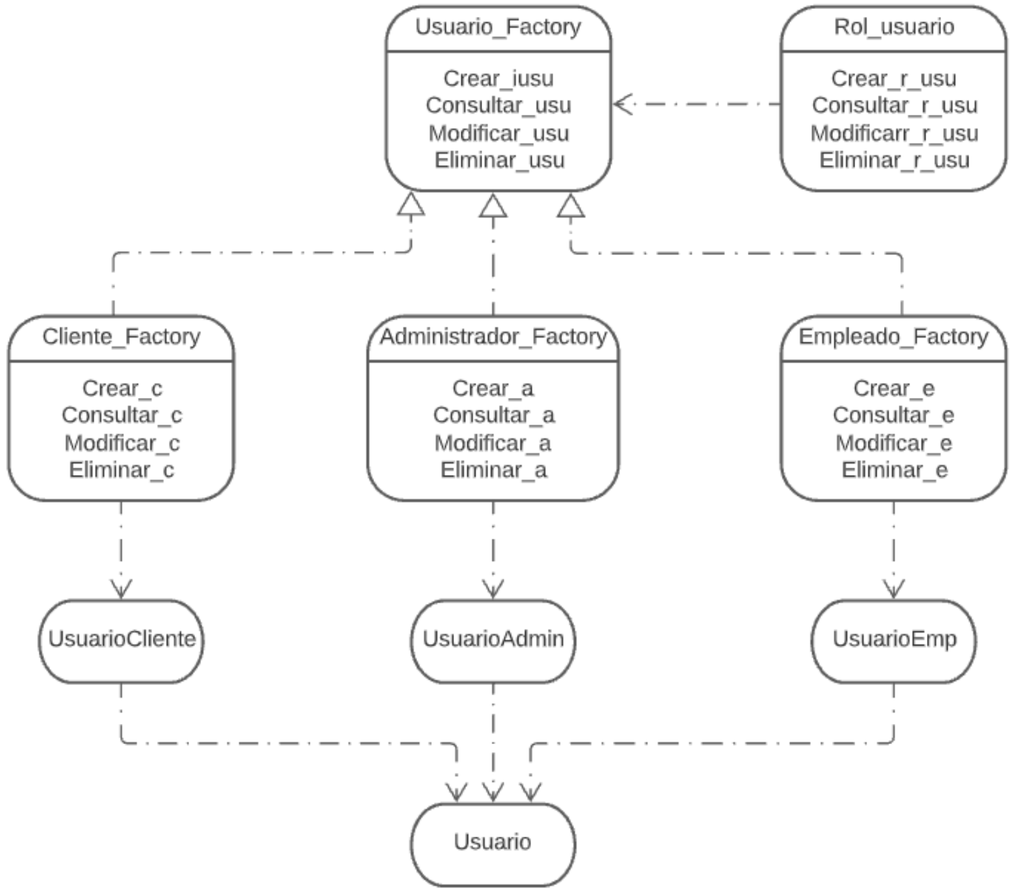
Es un patrón de diseño creacional que proporciona una interfaz para crear objetos en una superclase, mientras permite a las subclases alterar el tipo de objetos que se crearán.



Al igual que con el abstract factory el factory method tambien fue aplicado a los dos tipos de movimiento en el inventario(entrada y salida).

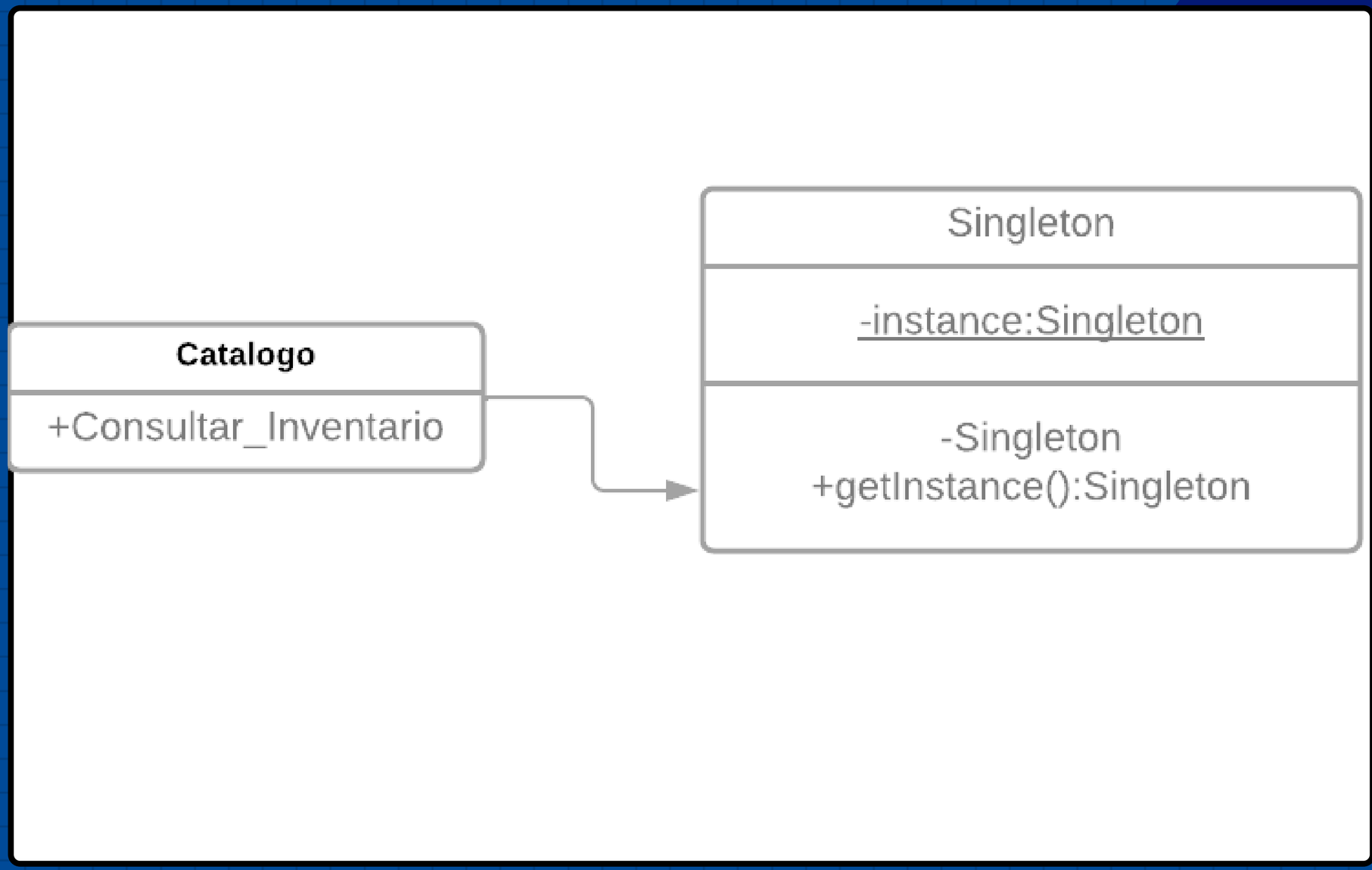
Abstract Factory

es un patrón de diseño creacional que nos permite producir familias de objetos relacionados sin especificar sus clases concretas.



El abstract factory entonces consiste en dividir en subclases una clase principal con tal de evitar confusiones, esto fue usado en el usuario en nuestro aplicativo(ya que los productos con los que se trabaja no lo requerían).

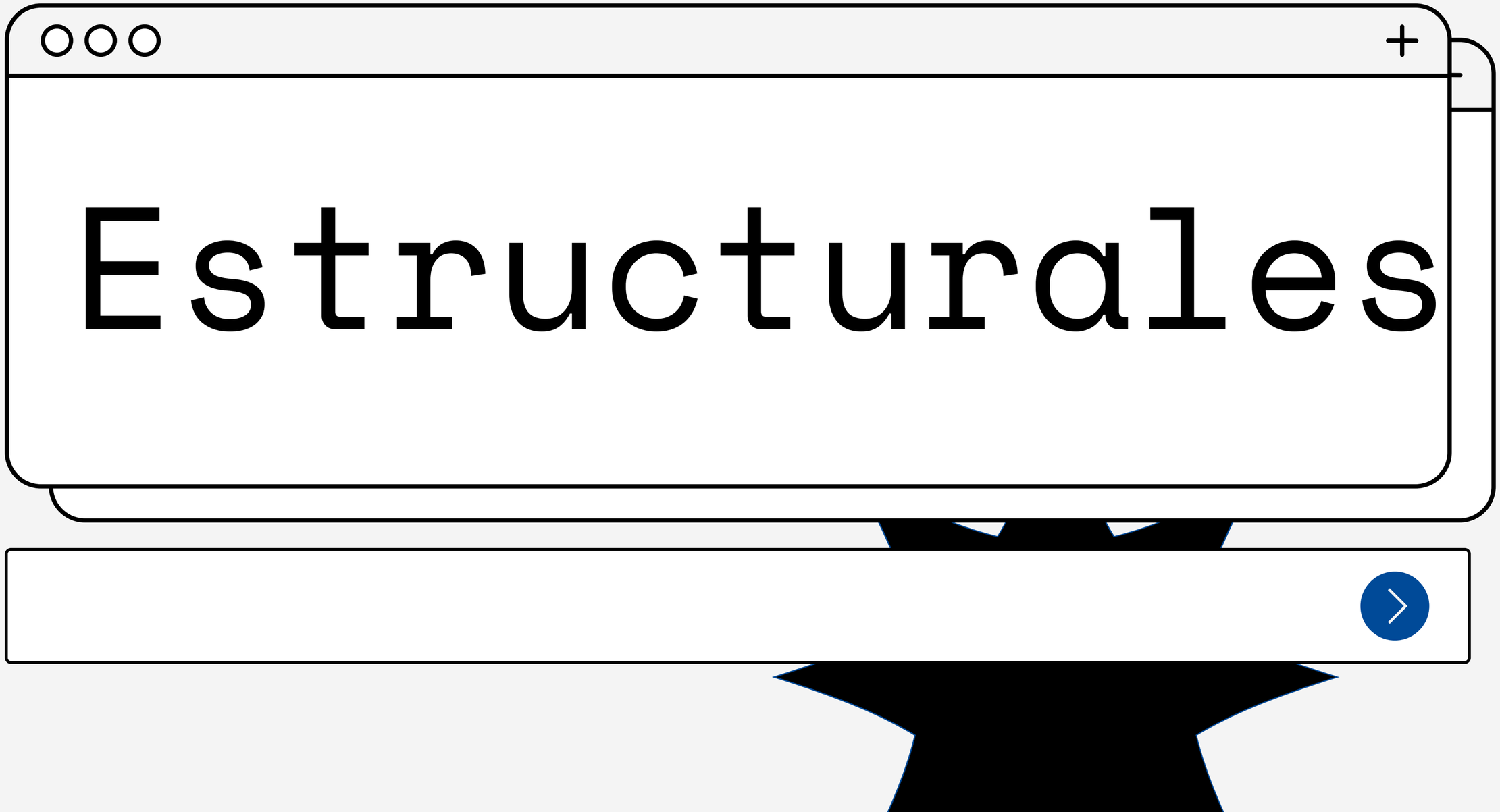
Singleton



Bueno procedere a explicar para que sirve el singleton, el singleton sirve para que una instacia o un campo quede con restricciones pero a la vez se pueda accder a esa instancia de todos los lados

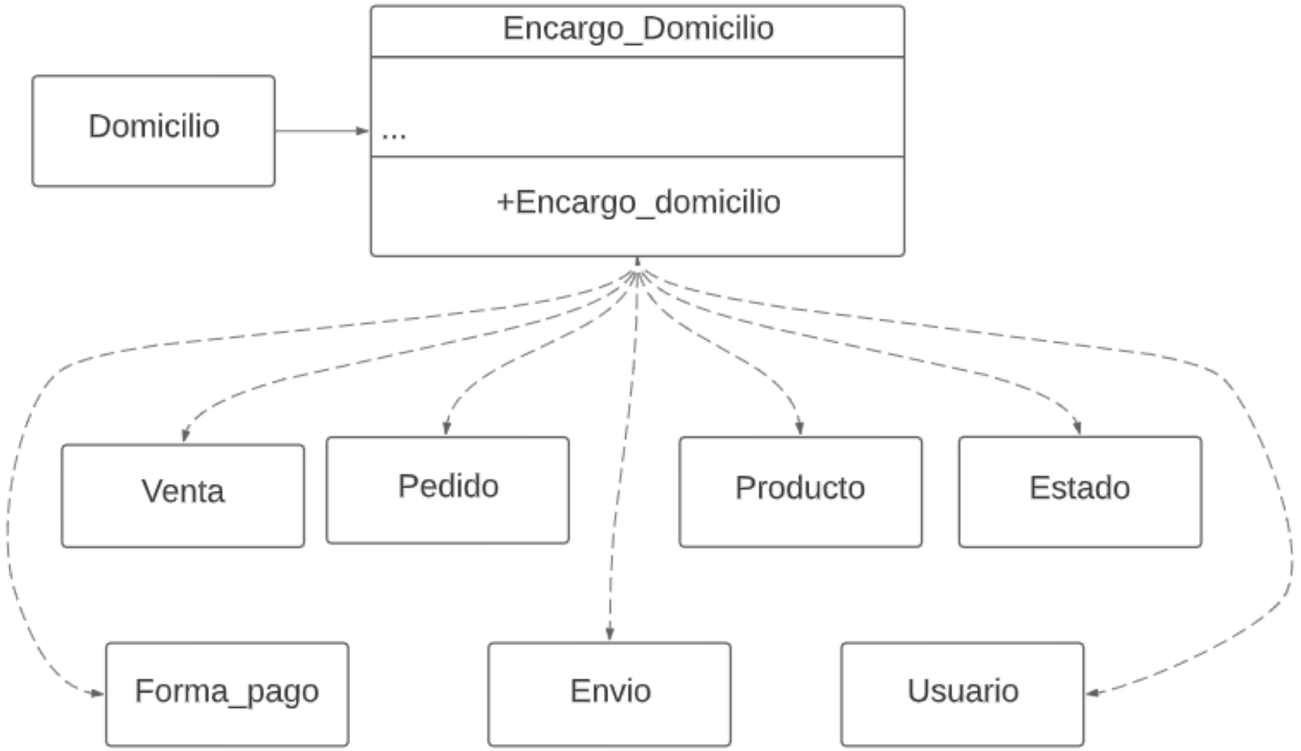
En el SI de DatStore el patron de diseño singleton nos serviria para tener una solo instancia del inventario, como nuestro proyecto esta orientado a tiendas de barrio estas no suelen tener mas de un inventario ponerle una de mas seria pues complicarnos ya que pondriamos algo que no existe

Estructurales



Facade

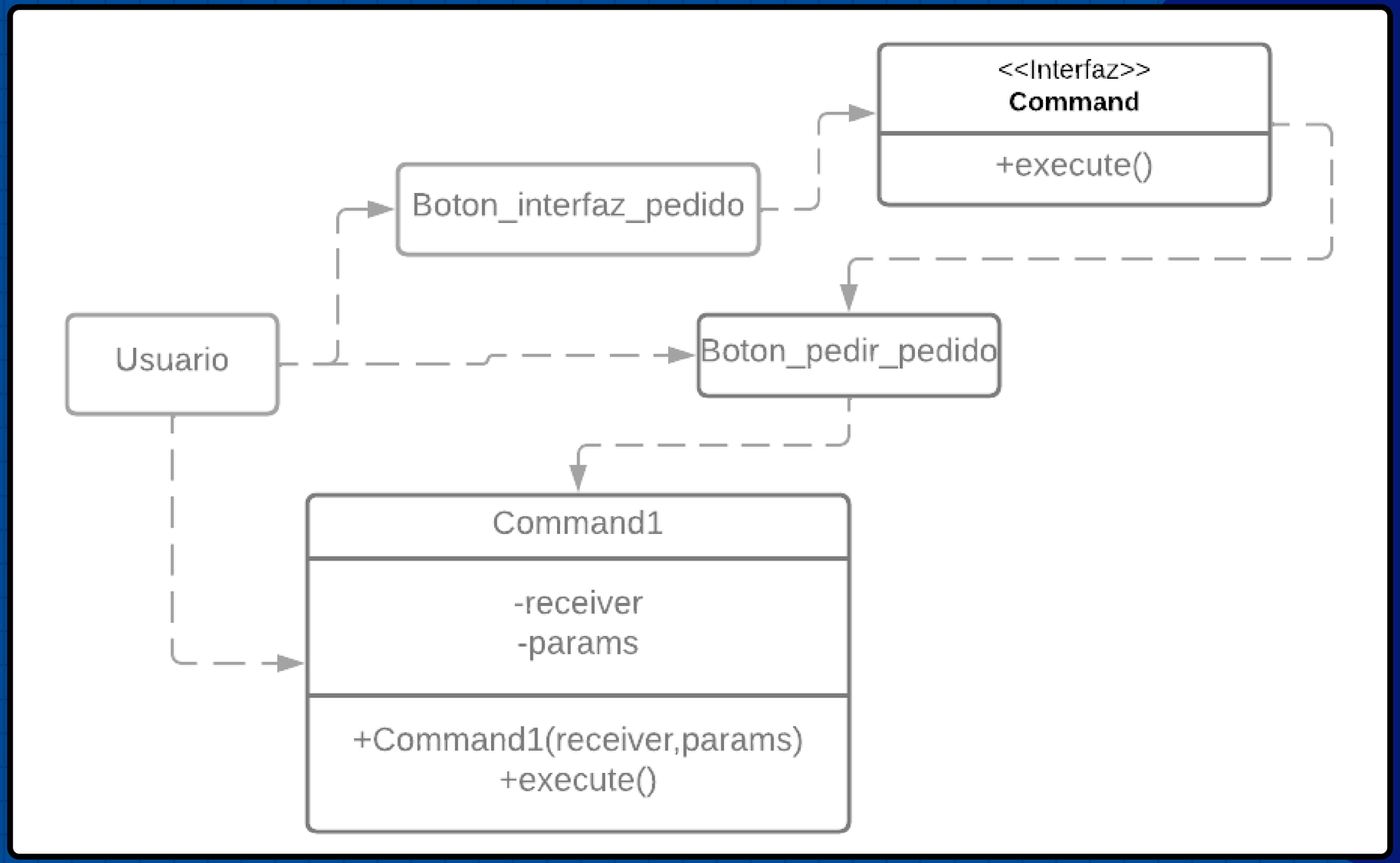
Facade es un patrón de diseño estructural que proporciona una interfaz simplificada a una biblioteca, un framework o cualquier otro grupo complejo de clases.



El facade es una interfaz donde puedes hacer procesos muy difíciles mas sencillos con una interfaz y en nuestro proyecto se utilizaría en el que cliente pueda pedir un domicilio donde hay muchos procesos como formas de pago al hacer un pedido y hacer una entrega solo oprimiendo esa opción de una interfaz de domicilio

Comportamiento

Command

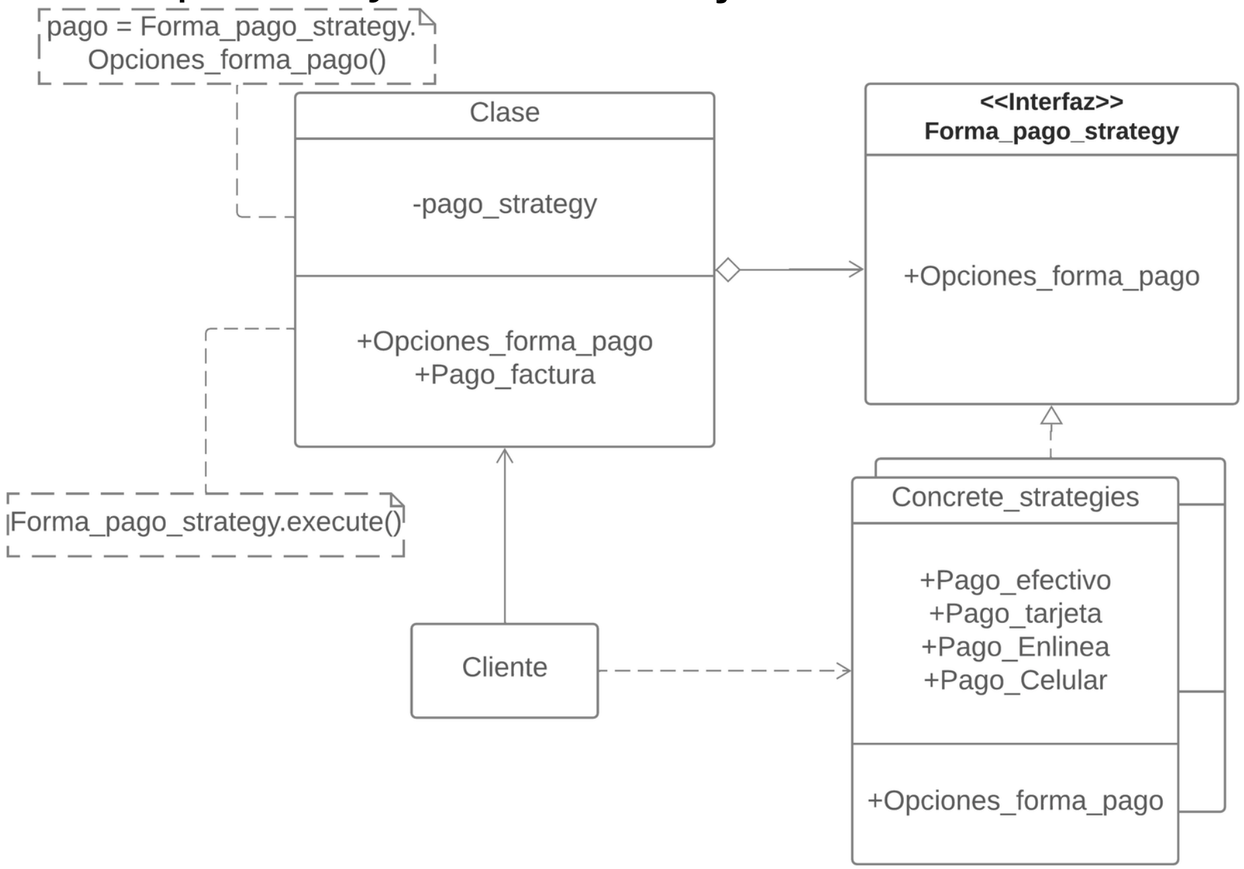


EL command es un patron de diseño que lo que hace es volver una solicitud en un objeto ademas de que la puede atrasar o poner en cola.

En el SI de Datstore nos serviria bastante para poner por ejmplo pedidos en una cola de espera para sabar cual va primero y que no haya desorden ademas de que esto al estar ordenado aunmenta la eficiencia de los trabajadores ya que sabran cual es el primero y cual es el ultimo.

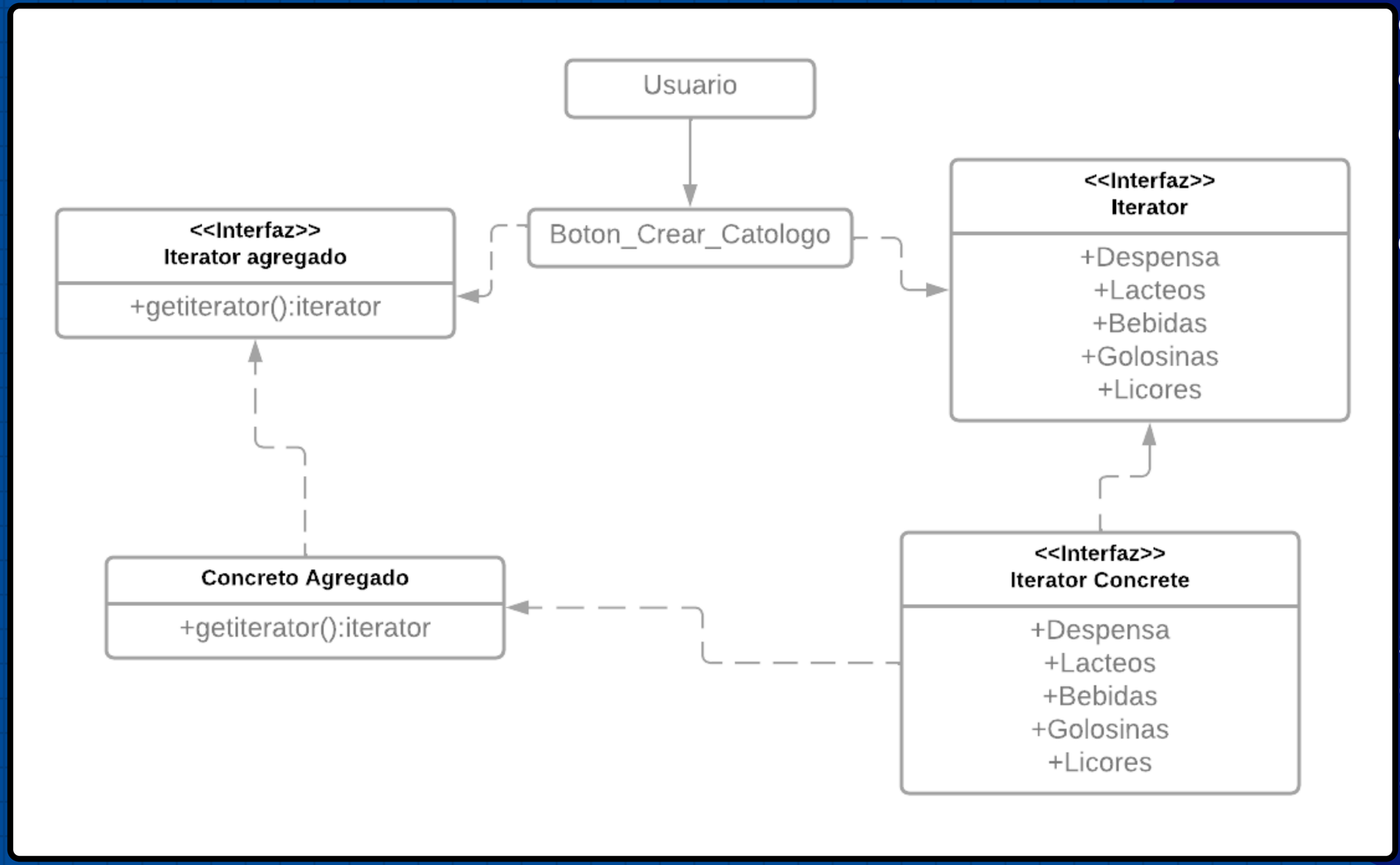
Strategy

Strategy es un patrón de diseño de comportamiento que te permite definir una familia de algoritmos, colocar cada uno de ellos en una clase separada y hacer sus objetos intercambiables.



El strategy son la diferentes opciones que tienes al hacer un proceso como lo seria la forma de pago que tiene un cliente al momento de hacer un pago sus strategy serian como pagarian ya sea pago en efectivo, con tarjeta crédito o debito, etc.

iterator



Define la interfaz para recorrer el agregado de elementos y acceder a ellos, de manera que el cliente no tenga que conocer los detalles y sea capaz de manejarlos de todos modos.

Un iterador le permitira al cliente hacer la ruta de catalogo a pedido a domicilio, aun asi si despues desea confirmar la información de alguno, no tendra que hacer el mismo recorrido secuencial, sino que podra ir directamente a lo que le interesa (y así con todas las otras funciones).

¡Gracias!