PATRONES DE DISEÑO

Patrones de Comportamiento –

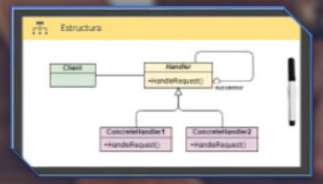
Chain of Responsability

PATRON DE COMPORTAMIENTO - MEDIATOR

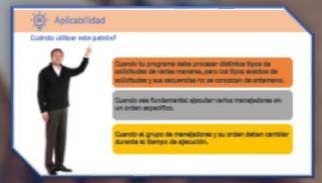
Propósito

Problema

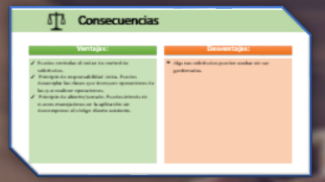
Solución



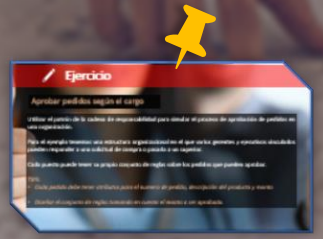
Estructura



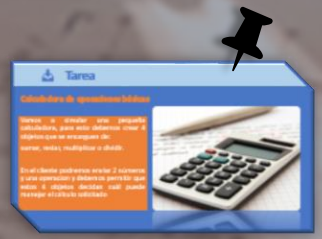
Aplicabilidad



Consecuencias

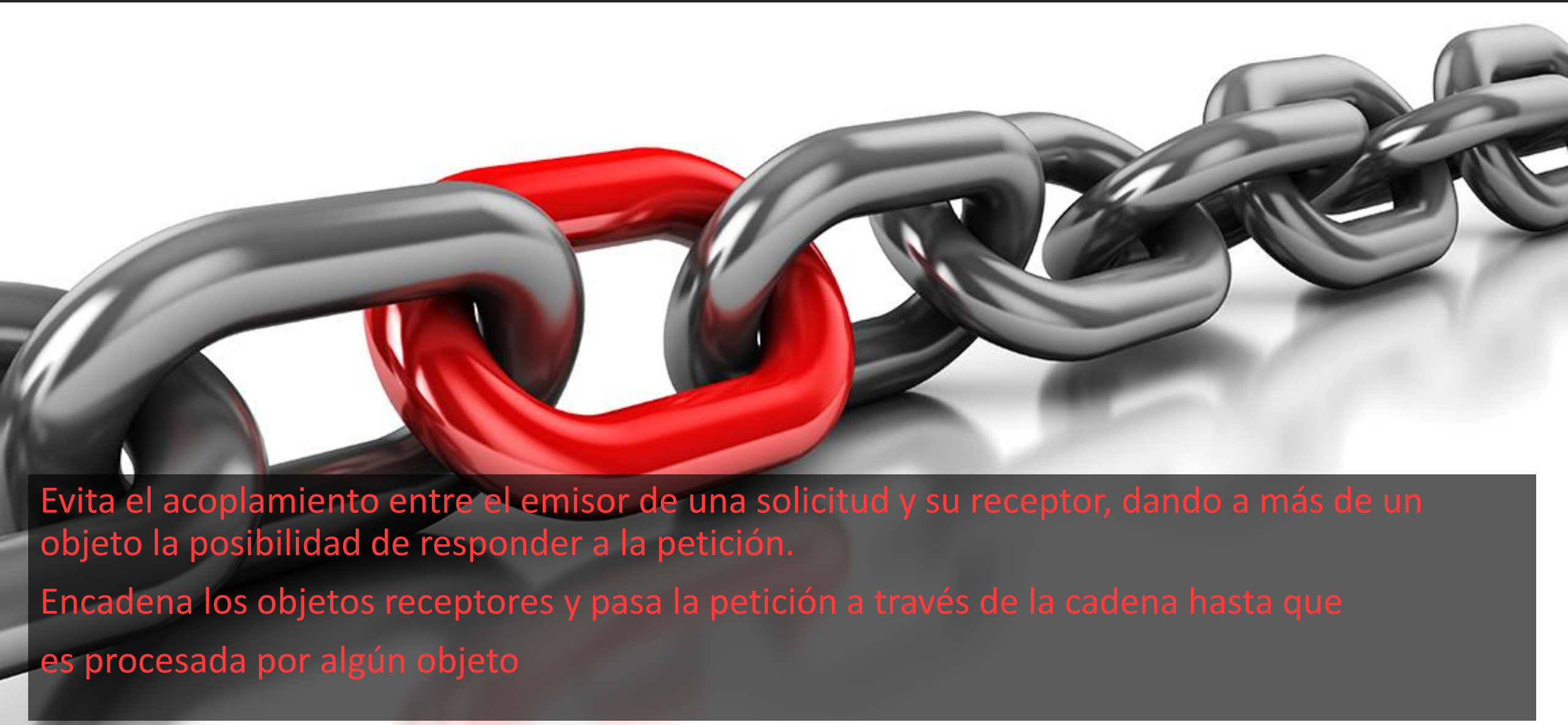


Ejercicio



Tarea

Propósito



Evita el acoplamiento entre el emisor de una solicitud y su receptor, dando a más de un objeto la posibilidad de responder a la petición.

Encadena los objetos receptores y pasa la petición a través de la cadena hasta que

es procesada por algún objeto

Problema

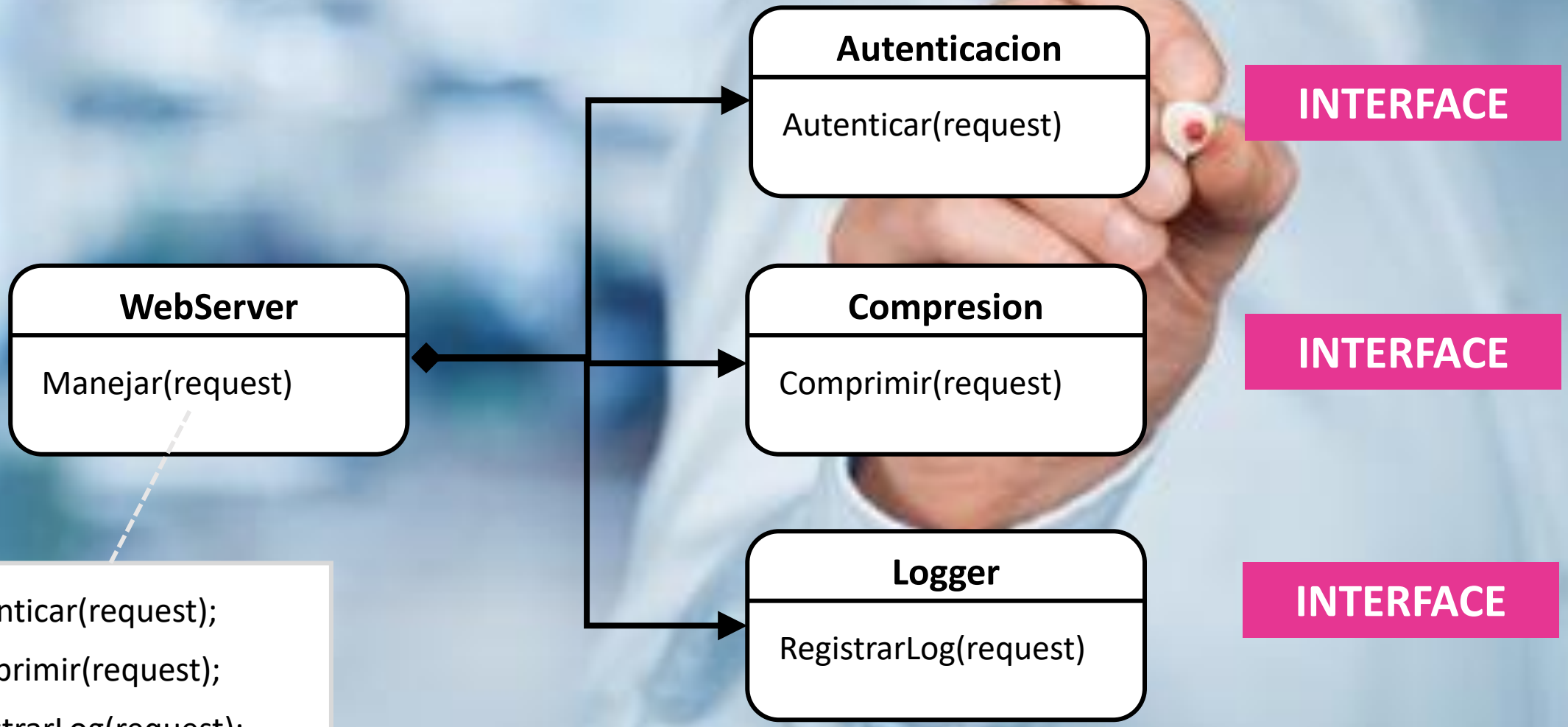

HTTP request
HTTP response
Server
Client


HTTP Request
HTTP Response

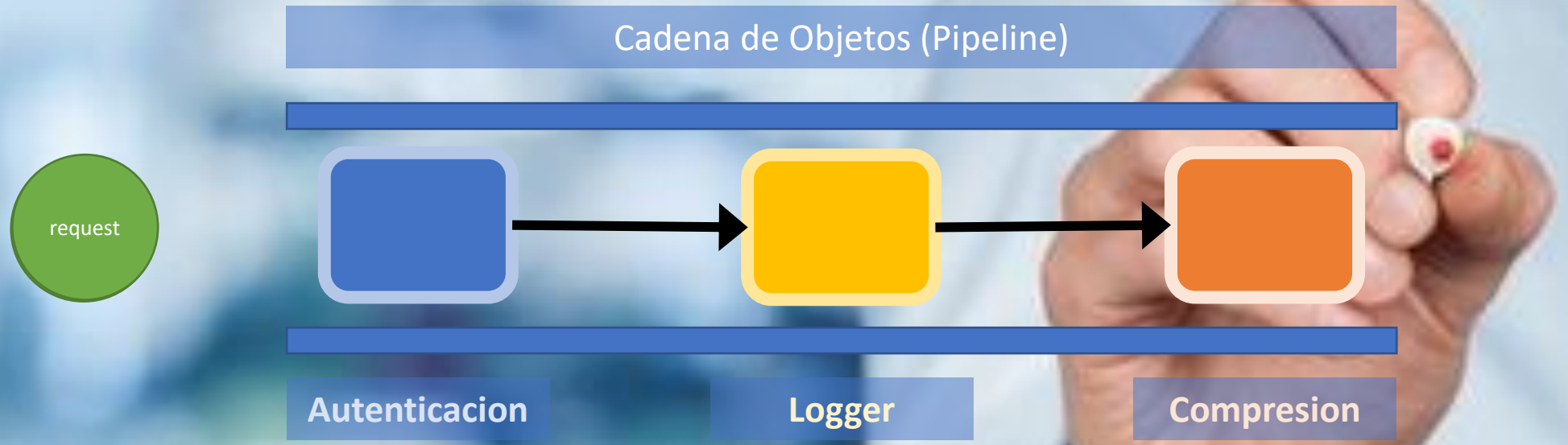
Demo

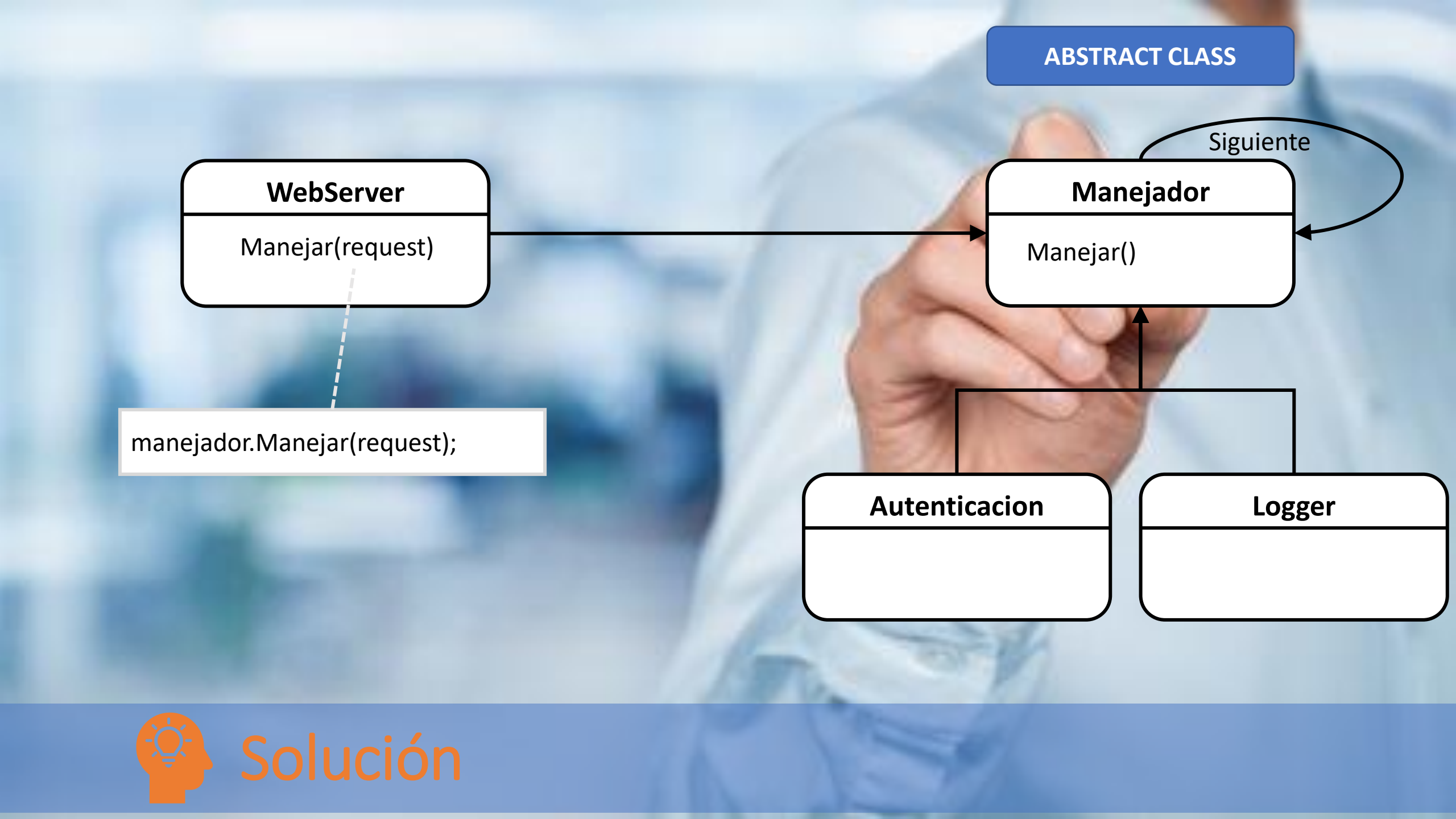
REVISEMOS EN CÓDIGO LA IMPLEMENTACIÓN DEL PROBLEMA

Solución


Autenticacion
Autenticar(request)
INTERFACE
WebServer
Manejar(request)
Compresion
Comprimir(request)
INTERFACE
Logger
RegistrarLog(request)
INTERFACE
Autenticar(request);
Comprimir(request);
RegistrarLog(request);

Solución


Cadena de Objetos (Pipeline)
request
Autenticacion
Logger
Compresion

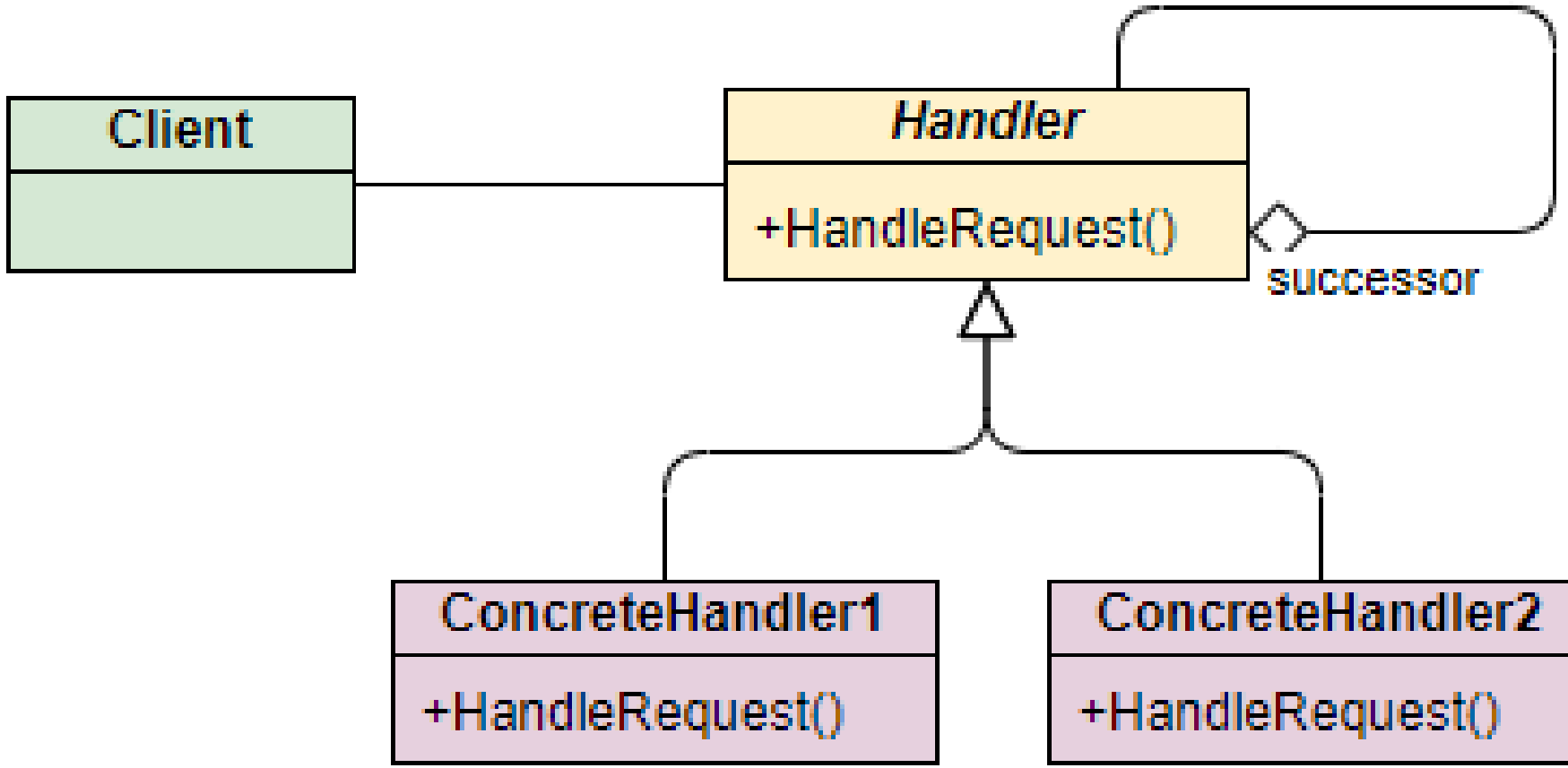

ABSTRACT CLASS
Siguiente
WebServer
Manejar(request)
Manejador
Manejar()
manejador.Manejar(request);
Autenticacion
Logger

Solución

Demo

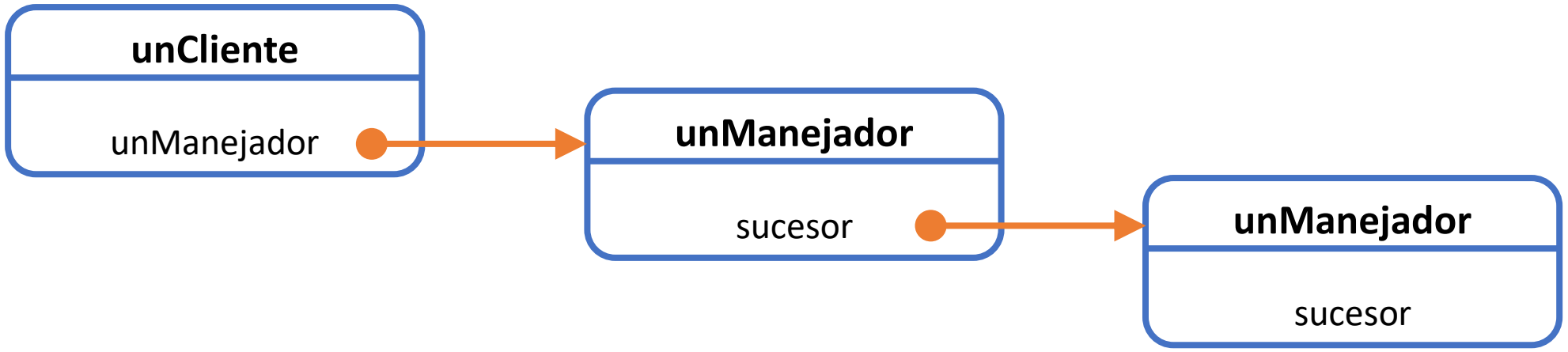
REVISEMOS EN CÓDIGO LA IMPLEMENTACIÓN DE LA SOLUCIÓN

Estructura


Client
Handler
+HandleRequest()
successor
ConcreteHandler1
+HandleRequest()
ConcreteHandler2
+HandleRequest()

Estructura

Una estructura de objetos típica podría parecerse a esta:



Aplicabilidad

Cuándo utilizar este patrón?

Cuando tu programa deba procesar distintos tipos de solicitudes de varias maneras, pero los tipos exactos de solicitudes y sus secuencias no se conozcan de antemano.

Cuando sea fundamental ejecutar varios manejadores en un orden específico.

Cuando el grupo de manejadores y su orden deban cambiar durante el tiempo de ejecución.

Consecuencias

Ventajas:

- ✓ Puedes controlar el orden de control de solicitudes.
- ✓ Principio de responsabilidad única. Puedes desacoplar las clases que invoquen operaciones de las que realicen operaciones.
- ✓ Principio de abierto/cerrado. Puedes introducir nuevos manejadores en la aplicación sin descomponer el código cliente existente.

Desventajas:

- ✗ Algunas solicitudes pueden acabar sin ser gestionadas.

Ejercicio

Aprobar pedidos según el cargo

Utilizar el patrón de la cadena de responsabilidad para simular el proceso de aprobación de pedidos en una organización.

Para el ejemplo tenemos una estructura organizacional en el que varios gerentes y ejecutivos vinculados pueden responder a una solicitud de compra o pasarla a un superior.

Cada puesto puede tener su propio conjunto de reglas sobre los pedidos que pueden aprobar.

TIPS:

- *Cada pedido debe tener atributos para el numero de pedido, descripción del producto y monto*
- *Diseñar el conjunto de reglas tomando en cuenta el monto a ser aprobado.*

Demo

REVISEMOS EN CÓDIGO LA IMPLEMENTACIÓN LA PRÁCTICA

Tarea

Calculadora de operaciones básicas

Vamos a simular una pequeña calculadora, para esto debemos crear 4 objetos que se encarguen de:

sumar, restar, multiplicar o dividir.

En el cliente podremos enviar 2 números y una operacion y debemos permitir que estos 4 objetos decidan cuál puede manejar el cálculo solicitado


Muchas Gracias…
los espero la siguiente clase