Trabajo Patrones:

Jorge Escobar Pacheco

Oscar Apaza Surculento

Patrones aplicados:

1. Chain of responsability
2. Command
3. State

## Chain of responsability:

### Descripción

Este patrón permite la implementación de métodos que se ejecutarán de manera encadenada ( o secuencial) hasta que uno de los métodos sea identificado como el apropiado para las condiciones

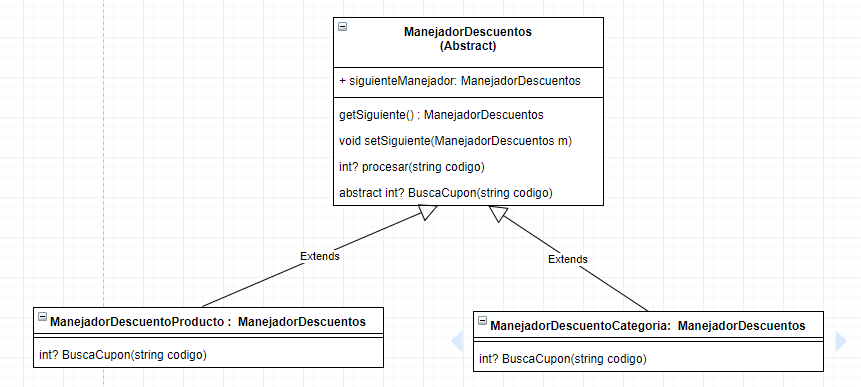
### Motivación

Este patrón se aplica al problema en cuestión dado que tenemos dos tipos de descuentos posibles:

* Productos
* Categorias

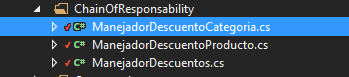
El patrón se puede aplicar enviando un código de cupón que no sabemos si aplica a categoría o producto, la cadena inicialmente revisará si es de producto y posteriormente si es de categoría.

### Implementación



La implementación de este patrón simplemente requiere un manejador general (clase abstracta) y tantas implementaciones que extiendan la misma como sea necesario.

En el proyecto se pueden apreciar en las clases



### Problema que soluciona

Es una manera muy ordenada de implementar varias sentencias de control que permiten extender la aplicación de código fácilmente.

## Command:

### Descripción

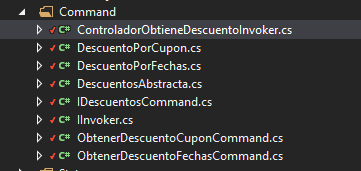
El patrón Command permite tener una forma muy sencilla de invocación de código

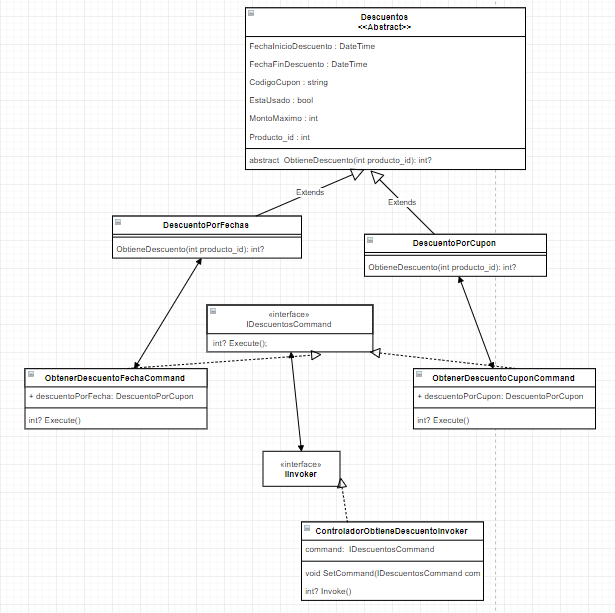
### Motivación

En el problema de los cupones se puede aplicar dado que para un producto podríamos requerir un descuento basado en cupón o por rango de fechas

### Implementación

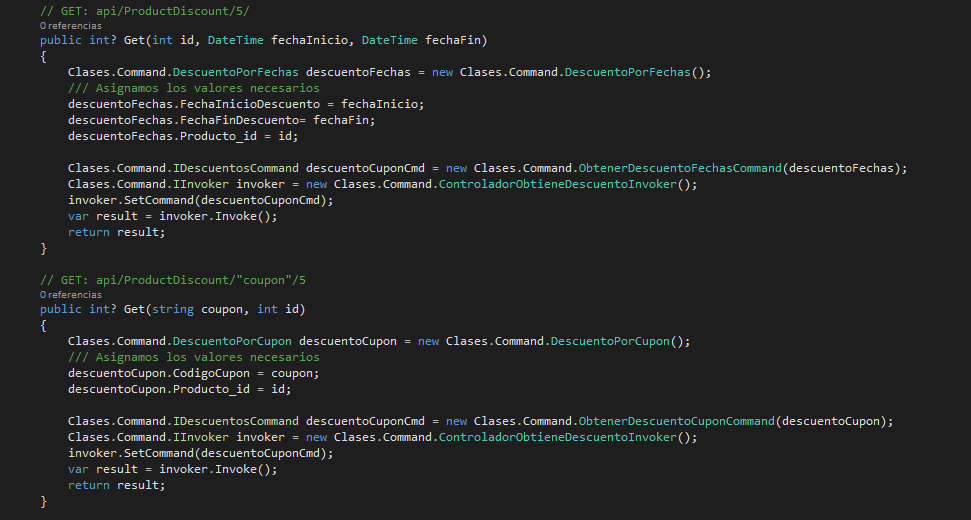
La implementación de este patrón es algo compleja y tiene los siguientes elementos:





### Problema que soluciona

Una vez establecidas las clases Interface command y las invoker, simplemente deben implementarse una clase command y otra implementación de la clase abstracta, esto permite tener implementaciones especificas y un solo punto de llamada:



## State:

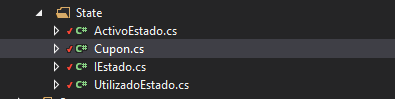
### Descripción

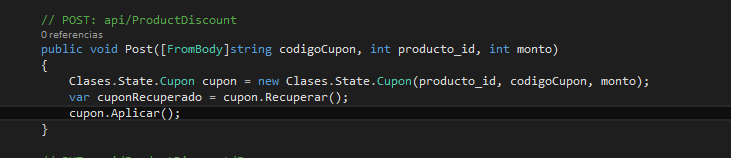
Asigna estados (o una combinación de propiedades) que determinan un comportamiento del objeto

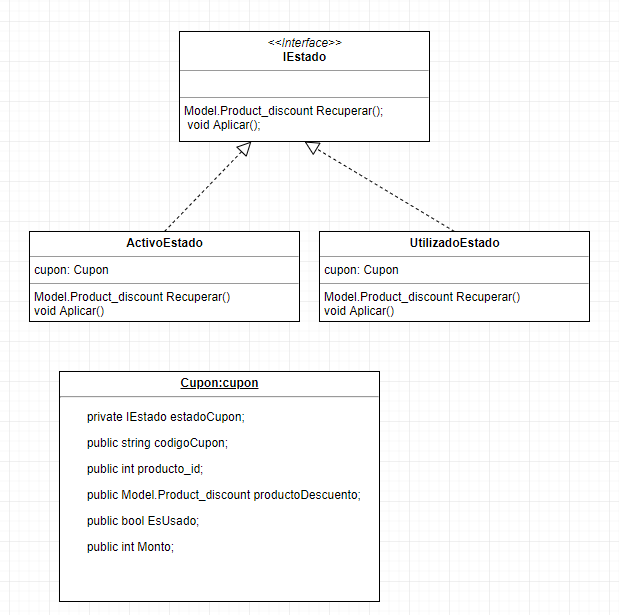
### Motivación

Este patrón nos permite abstraer a un nivel el concepto de cupon y podemos utilizar el patrón para realizar una acción si el mismo ya ha sido utilizado, verificar si ha sido usado antes de obtenerlo

### Implementación







### Problema que soluciona

Este patrón ayuda a separar los estados de una regla de negocio de una manera más mantenible y nos evita tener que condicionales dentro de métodos que ocultan la implementación.