**Observers vs Decorators**

**Desventajas del patrón Decorator.**

* Los objetos pierden identidad.
* Cierta complejidad para el armado de Builders.
* No es muy clara la implementación de, en este caso, las búsquedas, ya que hay objetos búsqueda que tienen otros objetos búsqueda adentro.

**Ventajas del patrón Observer.**

* Búsquedas más “limpias”, no tengo que instanciar clases de búsqueda nuevas o hacer que búsquedas tengan a otras búsquedas que realicen cierta lógica por ellas. Abstraigo en un objeto el comportamiento común de una búsqueda y este va a conservar su identidad.
* Los comportamientos adicionales que quiera agregar a una búsqueda están claramente definidos aparte de los objetos búsqueda, lo que hace más legible el código.
* Si quisiera agregar otro comportamiento, solo le “agregaría” a una búsqueda ese comportamiento, no tengo que crear una clase que tenga todos los comportamientos de una búsqueda común.

**Solución seleccionada.**

No pudimos encontrar ventajas considerables del patrón Decorator sobre el patrón Observer, pero si se cumple a la inversa. Por ende, decidimos utilizar al patrón Observer para realizar esta entrega.