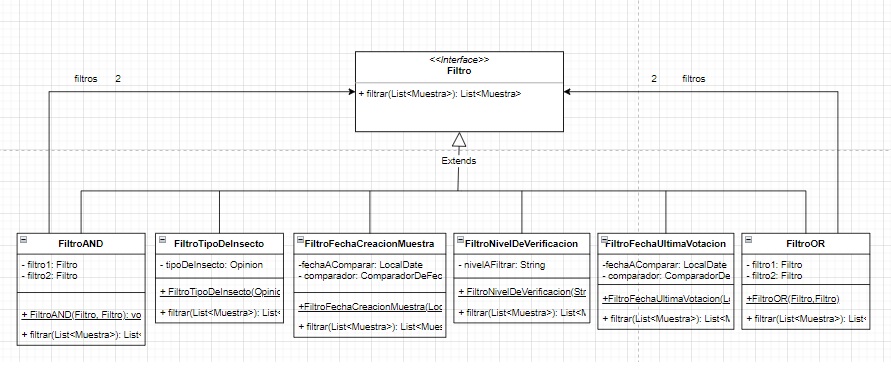
Patrones de diseño

Utilizamos el patrón Composite para implementar los filtros de búsqueda. Esto se decidió luego de una charla entre los integrantes del grupo y el profesor Matías Butti, que nos recomendó utilizar un diseño de patrón Composite que nos iba a facilitar el diseño de los filtros AND y OR.

En este caso, tenemos un **Component** que es la interfaz genérica Filtro, la cual tiene como **leafs** los filtros de tipo de insecto, nivel de verificación y por fecha de creación y ultima votación. Y los filtros AND y OR, son los **composite,** los cuales están compuestos por otros dos Filtro.

De esta manera, se puede tener un mensaje polimorfico “filtrar(List<Muestra>)” el cual van a saber responder ya sea una leaf o también un composite.

Decisiones de diseño

Se decidió tener una clase AplicacionWeb, la cual se encarga de manejar los usuarios, las muestras, las opiniones de las muestras, el filtrado de resultados según los filtros que se quieran utilizar, las organizaciones no gubernamentales y las zonas de cobertura. Esto para que el modelo se asimile a la realidad de como funcionaria una aplicación, donde todo debe pasar por ella.

Esto evita problemas como que un usuario que no este registrado en la aplicación, quiera hacer una revisión de una muestra, así como también quiera hacerla sobre una muestra que no existe en el sistema, entre otros casos.

Los usuarios si bien se registran en la aplicación, decidimos que se instancien fuera de esta para poder “mockearlos” a la hora de hacer los test de la aplicación.

Los filtros fueron implementados con un patrón Composite que nos permite una sencilla implementación de los filtros compuestos, ya que contienen dos objetos de tipo Filtro, los cuales pueden ser una leaf o un composite, y dentro de ese composite tener otros dos composite y así sucesivamente hasta llegar a dos leaf.

Se decidió que haya una clase “ComparadorDeFechas”. Esta se compone de una interfaz que luego es implementada por dos clases ComparadorMenor y ComparadorMayor, que tienen un método comparar(fecha1, fecha2) que depende el caso devuelve un booleno según sea mayor o menor fecha1 de fecha2.

Los filtros según fechas, tienen una variable de instancia que es ComparadorDeFechas, que se declara si es por menor o por mayor al momento de instanciar el filtro.

Problemas encontrados a la hora de diseñar

Uno de los problemas que apareció al momento del diseño, fue como implementar que los filtros de búsqueda por fecha, puedan ser antes o después de esa fecha. Luego de consultarlo con el profesor Diego Cano, nos recomendó utilizar una clase que nosotros denominamos “ComparadorDeFechas” explicada anteriormente.