

## Diagrama de Clases

### - Persona

Una **persona** es aquella que interactúa con el sistema. Una persona puede ser **Administrador** o **Usuario**. Toda persona puede **navegar los hechos**, **solicitar eliminar un hecho**, y **subir un hecho**.

Un usuario tiene un **rol**: visualizador o contribuyente. Este rol puede cambiar, y cada uno tiene comportamiento propio, por lo tanto se plantea utilizar el **patrón strategy** para resolver esto. Un visualizador puede **subir un hecho**. Cuando lo hace pasa a ser contribuyente. Un contribuyente puede **subir hechos**. Los hechos que suba se asocian a su registro (extract class). Un usuario puede registrarse o no. El registro consta de nombre (obligatorio), apellido y edad.

Un administrador puede **aceptar o rechazar una solicitud de eliminación de un hecho**, le envía un mensaje a todas las colecciones que cambien su estado de activo a eliminado. Además, puede **crear colecciones** que toman los hechos de una **fuentes (archivo .csv)**.

### - Colección

Una **colección** extrae todos los datos de una fuente (dataset), los filtra según un **criterio de pertenencia** y, a aquellos que cumplen el filtro, los instancia y los agrega a su lista interna de hechos.

Le permite a una persona filtrar, buscar y listar hechos en base a su lista interna, para no tener que leer de vuelta la fuente cada vez que se quiera realizar una operación.

Clases Gestoras: pensamos que podría ser útil una clase gestor que se encargue de almacenar las solicitudes y colecciones, como otras listas a futuro. Pero por ahora lo dividimos en dos (GestorSolicitudes y GestorColecciones)

- GestorColecciones: Utilizamos el patrón creacional Singleton para que exista una única instancia de gestor para todo el sistema, y todas las personas puedan acceder a todas las colecciones del sistema.

### Criterios:

Decidimos crear una interfaz Criterio, que utiliza el method strategy para implementar los distintos criterios que pueden elegir los usuarios, permitiendo que sea escalable/extensible (para un futuro poder agregar más sin problema) diferenciando las características de cada uno, aprovechando el polimorfismo.

### Solicitud

Los usuarios crean solicitudes de eliminación. Estas las gestiona el gestor de solicitudes. Su rol es llevar un seguimiento de las solicitudes que hay creadas en el sistema. Cuando un administrador acepta o rechaza una solicitud, esta se elimina

de la lista que tiene el gestor de solicitudes. Cuando se remueve de la lista, no hay una referencia a la solicitud, por lo que se elimina.

### **Diagrama de Casos de Uso**

- Un usuario se puede registrar al subir un hecho (sólo si lo desea)
- Añadimos al “usuario” quien representa al visualizador y contribuyente (ya que ambos realizan las mismas acciones)

### **DUDAS:**

- No sabemos si es necesario registrarse para subirHecho, o si simplemente cambia que al hecho se le adjudique el nombre del usuario registrado.
- Sí visualizador y contribuyente tienen exactamente los mismos permisos, que sentido tiene hacer un strategy para cambiar el rol en tiempo de ejecución. Hasta ahora lo único que pasa es que al usuario (visualizador por default) le cambia el rol cuando sube un hecho.
- No sabemos todavía cómo se diferencia un hecho de texto con otros tipos de hechos futuros.