Decisiones de diseño

Casos de Uso:

- Consideramos que los actores son Miembro de comunidad, Administrador de comunidad y Proveedor de plataforma porque son los únicos actores que acceden a las funcionalidades del sistema. Entendemos que hay una entidad Usuario pero lo tomamos como algo interno al sistema y no un actor externo.

- Consideramos que todos los usuarios tienen que iniciar sesión teniendo que registrar su usuario una única vez.

- Consideramos que el proveedor de plataforma no inicia sesión debido a su cargo y puede designar un administrador. La validación de que no tienen conflictos de intereses es ajena al sistema.

Diagrama de Clases:

- Consideramos que los servicios públicos tienen un atributo tipo, que representa si es subte o tren, sólo como etiqueta ya que un subte no tiene diferentes funcionalidades que un tren, o al menos por ahora.

- Consideramos que las comunidades tienen una lista de usuarios que pueden ser miembros o administradores.

- Consideramos que el estado de un servicio representa si está activo o inactivo.

- Decidimos modelar una clase que represente los medios de elevación ya que el enunciado los diferencia de los demás servicios aunque por ahora es todo esquemático porque ningún servicio en particular cumple con funcionalidades.

- La responsabilidad de validar las credenciales es del validador, que no está en el diagrama de clases porque no pertenece al dominio y nadie lo usa, o al menos por ahora.

Algoritmo Validador de Contraseñas:

- La clase Validador es la clase principal que contiene el algoritmo validador de contraseñas.

- Decidimos que las validaciones respetan la interfaz validación, que tiene un método validar. Las validaciones son EsDebil, UsaCredencialPorDefecto y

RespetaPoliticasNIST. Dentro del constructor de EsDebil se lee el archivo que contiene el top 10.000 peores contraseñas para guardarlas en una lista.

- La clase CredencialDeAcceso funciona como un agrupador de todas las cuestiones que el validador debe verificar.

- Consideramos que la creación de una interfaz Condicion, dentro de la clase RespetaPoliticaNIST, es útil ya que nos aporta extensibilidad y mantenibilidad a la hora crear los métodos tieneCaracterEspecial, tieneMayuscula y tieneNumero.