Decisiones de diseño

Casos de Uso:

- Consideramos al actor usuario de plataforma como una persona que tendrá acceso a sus funcionalidades dependiendo del rol que ocupe.

- Un usuario puede unirse a una comunidad.

- 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.

- Modelamos el Rol como una clase abstracta usando el patrón Strategy para encapsular distintas formas de resolver el mismo problema (Acceder a las funcionalidades) en diferentes clases.

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

- 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 contemplan una funcionalidad distinta a los demás servicios aunque por ahora es todo esquemático.

- Las estaciones tienen una lista de prestaciones de servicios, estas prestaciones de servicios son agrupaciones de servicios que incluyen un conjunto de servicios, este conjunto puede ser de tamaño >= 1.

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

Algoritmo Validador de Contraseñas:

- La clase Validador es la clase principal que contiene el algoritmo validador de contraseñas. En el constructor de esta clase, se abre el archivo con el top 10.000 peores contraseñas y se agregan a una lista para luego verificar si la contraseña es débil.

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

- Consideramos que la clase Validador tiene la responsabilidad de validar si una contraseña de un usuario respeta las políticas de NIST, que la misma no sea débil y que no utilice credenciales por defecto.

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