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.

Entrega 2:

Casos de Uso:

- Se agregaron dos actores nuevos, “Entidad Prestadora” y “Organismo de Control”, ambos pueden designar una persona destinada a recibir las problemáticas de los servicios que ofrecen las entidades asociadas.

Diagrama de Clases:

- Implementamos la clase Interés que conoce el Usuario, con el fin de contener la lista de entidades y servicios que le interesan al mismo.
- Por cuestiones de generalización del enunciado, lo que antes se modelaba como Servicio Público ahora pasó a llamarse Entidad, y lo que antes era Estación ahora es Establecimiento.
- Decidimos que la clase Entidad tenga una lista de establecimientos.

Implementación de la carga masiva de datos de entidades prestadoras y organismos de control:

- Decidimos implementar una clase LectorArchivoCSV que incluye un método leerArchivoCSV que actúa de lector genérico recibiendo como parámetro el nombre del archivo, un token y la cantidad de campos.
- Decidimos que la carga de Entidades prestadoras era independiente de la carga de Organismos de control y, por lo tanto, podrían tener un diseño de archivo csv distinto, aunque por ahora no es el caso.
- Cada clase CargaEntidadesPrestadoras y CargaOrganismosControl se encarga de llamar al LectorArchivoCSV definiendo las cuestiones asociadas a su diseño de archivo csv.

Implementación de la integración con el servicio GeoRef API:

- Consideramos que el encargado de realizar las llamadas a la API sería un controller, pero, como todavía no está modelado, no hay nadie que haga llamadas a georef. La integración igualmente está hecha para cuando el controller la llame.

Archivos CSV:

- Decidimos que ambos archivos CSV tengan por el momento un único campo que es el nombre.