**Justificaciones de las decisiones de diseño tomadas**

* Entrega Modelado en Objetos Parte I: Puesto a punto del entorno de desarrollo y primera iteración del diseño

En primer lugar, fuimos identificando y guardando información relevante relacionada al Estado Interno de cada clase, la cual consideramos que vamos a necesitar en un futuro para desarrollar el sistema.

En Servicios Públicos, creamos una sola clase que represente ferrocarriles y subterráneos porque no identificamos nada que los diferencie a la hora de utilizarlos en el sistema en base a la funcionalidad.

En Servicios, separamos los Medios de Elevación de los Baños ya que, tenemos que guardar información propia de cada una de esas clases. También diseñamos una superclase abstracta llamada Servicio para poder tratar a ambas clases de manera polimórfica y guardar el atributo que tienen en común (Estado). Por esto mismo, decidimos crear una clase abstracta y no una interfaz.

En Comunidades y Usuarios, decidimos crear una lista de Administradores como atributo perteneciente a la clase Comunidad porque si nosotros le asignamos un rol (como estado interno) a la clase Usuario íbamos a tener muchos problemas ya que hay usuarios que pueden tener un rol distinto según la comunidad a la que pertenezcan. Además esto mismo genera tener una relación de asociación doble, lo cual conlleva a tener más acoplamiento entre esas dos clases.

Utilizamos una relación de agregación entre las clases Comunidad y Miembro, ya que las comunidades necesitan a un miembro para existir, creemos que no debe haber comunidades sin miembros, pero los miembros pueden existir sin necesidad de pertenecer a una comunidad.

Estas dos clases, por el momento, solo contienen atributos debido a que no identificamos ninguna funcionalidad (método) necesaria para los requerimientos dados de esta entrega.

Para verificar si una contraseña es válida, creamos una clase Verificador la cual conoce y utiliza distintas clases de condición (Longitud, Credencial y Top 10000). Estas condiciones se basan en el patrón Strategy, donde consideramos cada condición como una estrategia, ya que sino cada vez que queramos agregar una nueva condición deberíamos modificar nuestra clase Verificador la cual tendría toda la responsabilidad, generando poca cohesión ya que se encargaría de hacer muchas cosas.

* Segunda entrega: Modelado en Objetos Parte II e incremento de funcionalidades:

Dado que para la presente entrega se agregaron las organizaciones y sus sucursales a los servicios a monitorear, estos mismos tuvieron que ser adaptados y generalizados cambiando el modelo para poder cumplir con la funcionalidad del sistema.

Por eso, nuestras clases “Servicio Público“ y “Estación“ se vieron afectadas. Se dividió por un lado, las líneas de servicio de transporte y las organizaciones en una clase llamada “Entidades”, y las estaciones y sucursales en otra llamada “Establecimientos”, cada una con sus respectivos atributos.

Por otra parte, la clase “Establecimientos” se relaciona con una interfaz “Localización”, la cual es implementada por las clases Municipio y Provincia, con el objetivo de establecer una ubicación para cada uno de los establecimientos y así saber dónde presta sus servicios la Entidad. Esto lo logramos mediante la implementación de una API REST que proporciona a través de la plataforma de datos abiertos del Gobierno Nacional Argentino. Además, la clase Usuario también se relaciona con dicha interfaz dado que los usuarios del sistema tienen interés en acceder a los servicios que se prestan en la localización en que se encuentran.

En la clase importador, la cual se encargará de la carga de datos de entidades prestadoras y organismos de control a través de un archivo csv, decidimos implementar el Patrón Strategy, así de esta manera evitar que la clase Importador sea altamente cohesiva y sea mantenible a lo largo del tiempo. Para esto creamos dos clases que corresponden a las estrategias de importación, tanto para las entidades prestadoras y los organismos de control, que implementan una interfaz, que se relaciona con la clase importadora.

Para la clase “Entidad” le agregamos una lista de Atributo Especiales como atributo para poder tener atributos específicos de cada tipo de entidad (línea de transporte o una organización). Tuvimos la necesidad de hacer esto, ya que la “Entidad” representa tanto servicios de transporte como diferentes organizaciones, pero aunque se comportan de la misma manera, cada uno tiene sus propios atributos (distintos a los de los demás).

Si nosotros queremos modelar una línea de servicio públicos, que cuenta por ejemplo con una lista de estaciones no tendría sentido que un banco tenga estos atributos, así que como solución se propone modelar una clase abstracta “AtributoEspecial” de la cual van a heredar diferentes clases tales como “AtributoInt”, “AtributoEstablecimiento”, “AtributoString”, etc. dependiendo del tipo de atributo que se quiera representar se va a relacionar con la entidad definida.

Por ejemplo, para una línea de trenes teniendo una lista de estaciones, en la primera entrega, si solo cambiamos el nombre de Línea por Entidad, iba a seguir teniendo esas estaciones que si la entidad llegaba a ser un banco, no tendría sentido que estén esos atributos.