Correcciones 1º Entrega:

- Agregamos vínculo de uso de comunidad para Usuario, con quien instancia un miembro a quien agrega a una lista miembros
- Eliminamos interfaz de Permisos y se transformó en un tipo enum
- Eliminamos interfaz Servicios y se transformó en un tipo enum
- Servicio Público tiene una lista de estaciones de donde tiene la 1º y última estación de la línea.

Decisiones de diseño 2º Entrega:

Administración de Entidades y establecimientos:

Creamos una clase abstracta **Entidad**, la cual tiene las subclases **LineaTransporte** y **Organización**.
*Justificación: En esta segunda entrega se amplían los servicios, entre ellos el monitoreo de estas **Entidades**, las cuales para el sistema representan: servicio de transporte y establecimientos. En nuestro sistema en particular, **LineaTransporte** es quien representa nuestro "servicio de transporte". Así mismo, aún continúa teniendo atributos y métodos ya definidos de la entrega anterior. Por lo que para distinguir entre esta y la entidad **Organización**, decidimos diseñarlas como subclases de **Entidad**, compartiendo únicamente por el momento, una Lista de **Establecimientos** (Sucursales).*

Creamos un clase **Establecimiento**, quien para el dominio del sistema representará las sucursales y las estaciones. Además posee un atributo de tipo: **tipoEstablecimiento** que es un enum.
*Justificación: Tal como lo mencionado anteriormente, en el dominio de esta segunda entrega, se indica que un **Establecimiento** representa, para el uso interno del sistema, las sucursales y estaciones. Sin embargo, al ambas formar parte de esta entidad, pero no tener comportamiento definido alguno, decidimos identificar ambas dentro de una clase tipo ENUM, llamada **tipoEstablecimiento**. No solo por lo ya mencionado, sino también porque son elementos bien definidos y reducidos.*

Asignación de personas a servicios de interés

Agregamos a la clase **Usuario** el atributo **entidadesDeInteres** que representa una lista de entidades, que como lo dice su nombre, son de interés para el usuario.
*Justificación: Entendemos que un **Usuario** es de por sí, una persona. Dicho esto, como lo indica el dominio, cada persona tendrá asociadas entidades y servicios. Como se observa en el diagrama, consideramos que no es necesario agregar un "servicio asociado" como atributo en el **Usuario**, ya que este puede conocerlo a través de la propia entidad que a su vez conoce al establecimiento quien finalmente brindará el servicio que el usuario considere de interés.*

Administración de Localizaciones en personas y entidades:

Agregamos una clase llamada **Localización** que representará un espacio geográfico en el que estará ubicada una persona, o una entidad. Esta clase se asocia a la API de **ServicioGeoRef** (representada como una clase dentro de nuestro diagrama). Por otro lado, añadimos a la clase **Usuario**, el atributo **localizaciónInterés**.
*Justificación: Cada persona debe tener asociada a sí misma, una ubicación desde la cual podrá ver las entidades a las cuales está interesada, próximas a ella. Respecto a la **Localización**, decidimos modelar como una clase aparte ya que la tendremos que utilizar no solo para el usuario, sino que también para las entidades y quizá a futuro para otras organizaciones. Además para obtener una localización para el sistema, debemos hacer una adaptación en base a la API brindada por el enunciado, y utilizarla para nuestros requerimientos.*

Para que una **Entidad** tenga una **Localización**, la podrá obtener a través de la **ubicaciónGeografica** que ya posee un **Establecimiento**.
*Justificación: Esto es así ya que una **Entidad** está compuesta por establecimientos y servicios de transporte. Por lo que de esta manera tanto el establecimiento como los transportes tendrán una **Localización**.*

Para que una persona pueda acceder a las entidades que estén ubicados dentro o cercanos a su propia localización, se filtrarán las coincidencias obtenidas desde la clase **Localización** que ambas clases (**Usuario** y **Establecimiento**) conocen.

Entidades prestadoras de servicios de control

Creamos la clase **EntidadPropietaria** que representará las empresas propietarias de los servicios públicos u organismos de control. Esta entidad, tiene como atributo un usuarioEncargado y una lista de entidades de las cuales es propietari). Como comportamiento, actualmente solo se encargará de **recopilarProblematicas**() a través del **usuarioEncargado**.
*Justificación: Si bien una **EntidadPropietaria**, puede parecer tener la estructura de una **Entidad**, esto no es así y es por ello que decidimos modelar como una clase bien diferenciada, que conoce a la clase **Usuario**, por el **usuarioEncargado**, y conoce a la clase **Entidad** para obtener su lista de entidades.*
*En cuanto al método **recopilarProblematicas**(), forma parte de esta clase ya que en el dominio se menciona que esta **EntidadPropietaria** podrá designar una persona encargada de juntar información de problemáticas de los servicios que ofrecen, entonces, es un comportamiento que tendrá este **usuarioEncargado**.*