Justificaciones:

Primera Entrega:

Para la distinción entre Ferrocarriles y Subte **(tipo de transporte)** consideramos usar un *identificador* en la clase o una *herencia*. Debido a la **falta de funcionalidad e información distintiva entre ellas** optamos por la primera opción. Como identificador pensamos en usar un *enum*, un *string* o una *clase identificador*. El enum **resulta limitante** en caso de que se agregue algún nuevo tipo de transporte, el string **da lugar a inconsistencias**, y la implementación de una *clase* Transporte da lugar a que se puedan agregar otro tipos de transporte (No subte/ferrocarriles) en tiempo real / futuro.

Decidimos hacer una distinción entre **Servicios** (escaleras, baños, ascensores, etc) de los **Servicios de Transporte** (líneas) porque creemos que cumplen funciones diferentes y no tienen suficiente lógica en común para ameritar el uso de *herencia*.

Para el **estado de un servicio**, utilizamos un *enum* para poder identificar si está en servicio o fuera de servicio debido a que no pensamos que vayan a agregarse nuevos posibles estados en un futuro.

En las **Estaciones**, con respecto a los **Servicios,** pensamos en poner una *lista* de servicios y una de servicios no incluidos pero decidimos tener una sola *lista.*

En cada uno de los **Servicios**, al no identificar ningún tipo de dato relevante para realizar código, decidimos hacer una identificación con un “Nombre” y una descripción de la tarea que realiza.

En la clase **Línea,** la *lista* de **Estaciones** están ordenadas según su recorrido. Es decir, la primera estación de la *lista* es la estación origen y la última de la lista es la estación destino. De esta forma, también se puede distinguir la línea que va de ida y la de vuelta.

Hicimos una distinción entre las *clases* **Usuario** y **Persona**. Una **Persona** puede tener acceso al sistema. Cada **Usuario** corresponderá a una **Persona** real.

Las **Comunidades** se relacionan con los servicios a través de una *lista* de servicios de interés para la misma.

Extra**:**

La Guía NIST 800-63-3, que fue publicada en 2017 y establece pautas para la autenticación digital, recomienda lo siguiente…

- *Para contraseñas de **nivel de seguridad medio**, una longitud mínima de 12 caracteres y se recomienda la inclusión de caracteres especiales, números y letras mayúsculas y minúsculas.*

Con respecto a *alinear la política de longitud, complejidad y rotación de contraseñas con las recomendaciones de la Sección 5.1.1.2 para Secretos Memorizados de la Guía NIST*

800-63, decidimos considerar las contraseñas del usuario como *contraseñas de* ***nivel de seguridad medio.***

Segunda Entrega:

Utilizamos una relación doble entre Establecimiento y Prestación De Servicio debido a la alta dependencia de Prestación De Servicio a un Establecimiento y al orden concreto de Instanciación de primer Establecimiento y luego Prestación De Servicio .

Los **Usuarios** tienen asignados roles los cuales les dan permisos para acceder a las diferentes funcionalidades del sistema. La compatibilidad de los roles se ve en la capa de controladores.

Los **Organismos de control** y **Entidades prestadoras** son dos clases que heredan de la clase **Entidad**. Al utilizar la herencia, podemos definir una clase base, OrganizacionExterna, que contiene los atributos y métodos comunes a todas las organizaciones externas. Las clases OrganismoDeControl y EntidadPrestadora pueden extender esta clase base y heredar estos atributos y métodos. Esto evita la duplicación de código y facilita la reutilización, ya que no es necesario volver a implementar los atributos y métodos compartidos en cada clase específica. Además, la herencia proporciona flexibilidad y extensibilidad al sistema. Si en el futuro se necesitarán agregar nuevos tipos de organizaciones externas, simplemente se podrían crear nuevas subclases de OrganizacionExterna sin modificar la clase base o el código existente. Esto permite adaptarse fácilmente a los cambios y ampliar la funcionalidad del sistema de manera modular.

Con respecto a la **localización** de las personas y entidades, decidimos darle uso al patrón Adapter en la integración de la API Geo ref para manejar la localización en el sistema para ganar interoperabilidad, flexibilidad, extensibilidad y separación de responsabilidades haciendo un sistema más robusto, mantenible y escalable del sistema en relación con el manejo de la información de localización.

Decidimos no asignar una **localización** específica a las **entidades** (como atributo) porque creemos que la localización de las mismas depende de todos sus establecimientos. Esta decisión nos da mejor flexibilidad y escalabilidad.