

Trabajo Práctico

Diseño de Sistemas

Grupo N°: 6

Docente:

Auxiliar Docente:

Curso:

Año: 2023

| Alumno | Legajo |
| --- | --- |
| Gianfranco Riccelli | 203.997-7 |
| Andrés Martín Trias | 203.367-7 |
| Federico Tomas Perez | 176.611-9 |
| Franco Pesce | 203.954-0 |
| Manuel Torrente | 204.116-9 |

# Decisiones de Diseño

**Entidades y Establecimientos**

Para esta entrega decidimos modificar completamente el diseño de la entrega 1 sobre líneas de transporte y estaciones de las mismas porque si bien tratamos de mantener las clases abiertas para la expansión pero cerradas para su modificación, en este caso el enunciado nos habla de que se amplían los servicios a monitorear y que las líneas de transporte son una instancia particular de una entidad y las estaciones instancias particulares de establecimientos. Por ende, para satisfacer este nuevo requerimiento decidimos modelar una clase genérica de entidad que tendrá además de un nombre para poder identificarlo correctamente en la interfaz de usuario, una colección de establecimientos, que son también una clase genérica con una lista de servicios prestados y además nos permite evitar la repetición de una misma instancia de un establecimiento dentro de una misma entidad.

Este diseño genérico nos permite la posibilidad que si queremos agregar una nueva entidad con distintos establecimientos simplemente se puede crear una nueva clase en momento de ejecución, como por ejemplo una clase banco como entidad y que tenga una colección de sucursales como establecimientos.   
  
Dado que con este diseño no podríamos conocer la estacion inicial y final de una línea de transporte, pensamos en que a lo mejor se podía crear una clase que herede de la clase entidad pero viendo las siguientes entregas, llegamos a la conclusión en una de las últimas iteraciones de que no era importante conocer la estacion inicial y final para el sistema y si en algún momento fuese necesario, se puede agregar una lista de atributos clave valor en el que estén especificados.

**Entidades Prestadoras y organismos de control**

Debido a que entendemos que el organismo de control debe regular a las prestadores, el mismo tiene una colección de estas, mientras que a su vez, la prestadora tiene una colección de entidades.

Ambas clases tienen una persona designada como atributo de su clase debido a que ambas tienen como requerimiento que designen una persona a la que le llegará información sobre las problemáticas de los servicios que se ofrecen. Hablando de este mismo requerimiento, como aun no sabemos qué información nos va a llegar ni cómo nos va a llegar decidimos hacer un adapter para poder seguir adelante con el diseño sin saber cuándo se resolverá esa parte pero sabiendo que en algún momento se tendrá que resolver.

Finalmente para modelar la carga de datos de estas clases a partir de un archivo CSV decidimos separar por un lado que datos contiene el CSV y por otro lado como se procesa este archivo.

Para los datos que necesitamos en el mismo decidimos que para el organismo de control simplemente necesitamos el nombre mientras que para la entidad se necesita además del nombre, la localización. Para lograr esto, pensamos en un inicio, un único archivo con los datos mencionados anteriormente y que además se agregue en el inicio un campo ID con 0 o 1 para diferenciar si esa línea del CSV está cargando una prestadora o un organismo de control. Esta solución fue descartada porque entendemos que no es la idea de un CSV tener líneas que tengan diferente cantidad de campos además de que su implementación se dificulta. Finalmente, llegamos a la solución de utilizar dos archivos CSV distintos, uno para cada clase

Las relaciones entre estas clases, es decir, la colección de prestadores que tiene el organismo de control no decidimos modelar en el archivo CSV ya que para estos se requieren de una parte perteneciente a la capa de datos y de persistencia que será implementada más adelante

Para el procesamiento del CSV, pensamos en dos soluciones. La primera era utilizar una clase por cada clase que se deba instancias. Es decir, un instanciador de prestadora y un instacionador del organismo de control. Todas estas clases utilizan el lector de CSV para tomar la ruta del CSV, abrir el archivo y devolver una lista de todas las líneas del csv en el que cada elemento es una lista con sus atributos. Sin embargo, esta solución la descartamos debido a que sus métodos eran muy parecidos, se llamaban en el mismo orden pero con una implementación distinta. Debido a esto, llegamos a la segunda solución.

Para esta solución, que fue la implementada, los instanciadores utilizan un patrón template method, ya que se requieren 2 pasos, leer el csv para transformarlo en arrays (paso 1), y parsear las listas a los objetos. El paso 1 es el mismo en todas las clases, pero el paso 2 cambia dependiendo del objeto que sea el instanciado. Esta forma de cargar el csv permite que si en un futuro se desea cargar por ejemplo, entidades, simplemente se requiere de una nueva clase que redefina el método levantarObjeto() a su forma con sus atributos.

**Localización**

La localización se obtiene desde la API GEOREF de Argentina utilizando un patrón adapter de por medio ya que es un servicio externo.

Decidimos crear un value object ubicación que tiene una provincia, un municipio o una localidad como atributo. Si una ubicación cambia porque se cambia uno de sus atributos, simplemente se crea una nueva clase y se desecha la anterior.

La ventaja que nos permite el tener este value object es que le podemos asignar una ubicación de manera distinta a cada entidad. Por ejemplo, en las entidades como un banco probablemente solo nos interese tener instanciada la provincia para que sea lo más genérico posible. En cambio, para un establecimiento o usuario seguramente nos interese tener una ubicación mucho más específica y por lo tanto nos interesa darle un valor a la localidad y al municipio.Además, cada ubicación se puede tratar polimórficamente en cualquier entidad/persona que lo use sin importar los atributos que tenga o no cargados

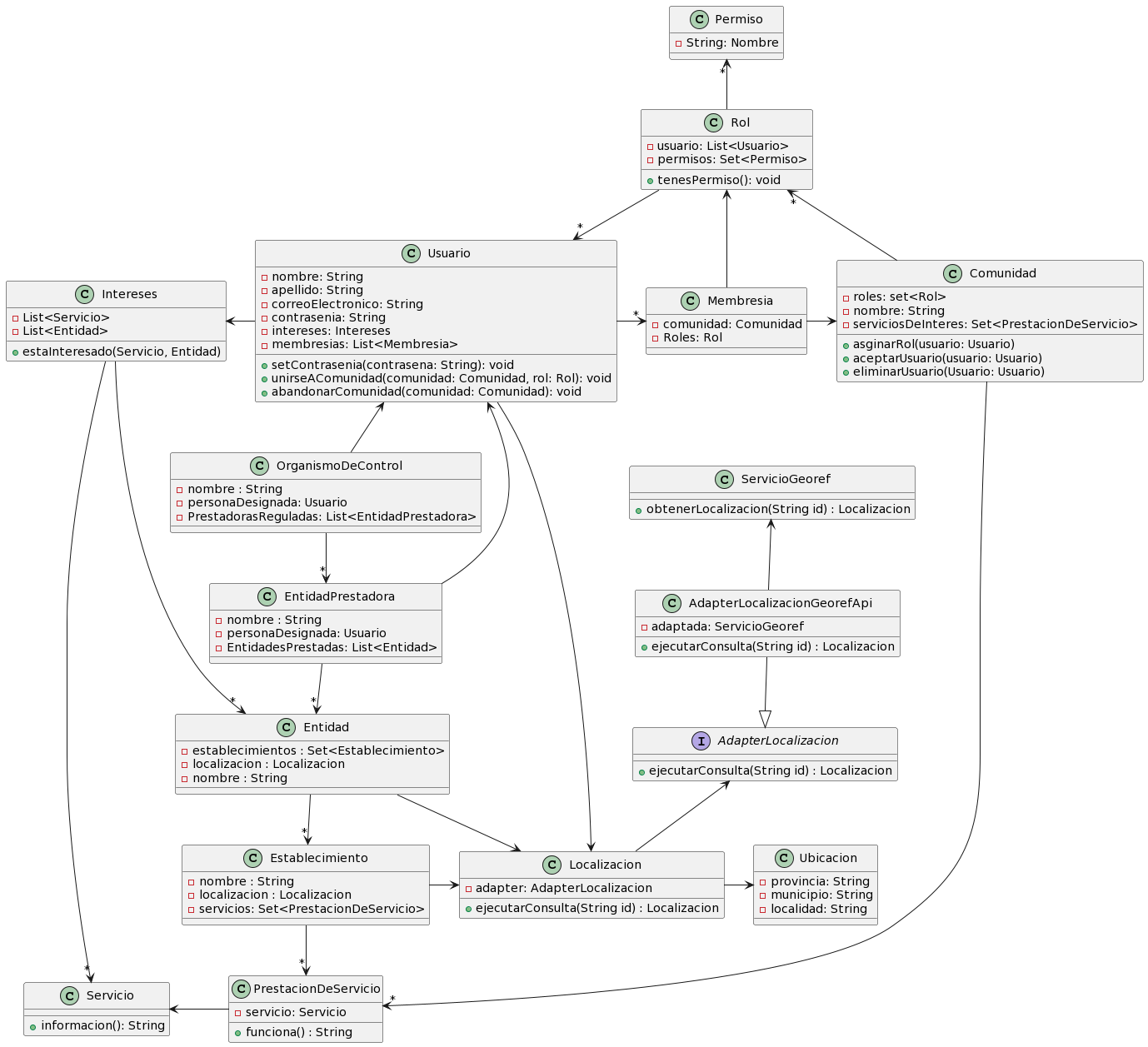
**Intereses**

Aunque esta parte podría haber ido directamente dentro de la clase Usuario,que fue lo pensado en un inicio, decidimos separarla para no darle demasiadas funcionalidades a esta ya que estábamos cayendo en que el usuario era una clase God.

Con este planteo, cada usuario tendrá su clase intereses, que a su vez tendrá una lista de servicios y entidades en los cuales el usuario está interesado. La clase interés tiene que poder contestar si un usuario está interesado en determinado servicio y en una entidad. Probablemente este método sea útil para la notificación de los incidentes en cada servicio prestado en una entrega futura.

**Usuario y comunidades**

La forma de modelar esto decidimos que sea de la siguiente forma:   
Un usuario va a tener una lista de membresias, una membresía va a ser un intermediario entre el usuario y la comunidad, esta tendrá como atributos Comunidad y Rol. Una comunidad va a tener una lista de roles y de prestación de servicios, y un rol va a tener una lista de usuarios y de permisos. Los roles serán distintos por cada comunidad, y estos se encargarán de conocer a los usuarios.



**CSV:**

