UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Documentación del Trabajo Práctico Anual

Diseño de Sistemas 2023 - Grupo 3

Integrantes

Nicolás Aparicio - Mariano Iturriza - Juan Nardi

Agustín Ezequiel Quiroga - Adrián Tolaba

Primer entrega - 25/04/2023

Esta primera entrega se encuentra muy orientada al modelado de las clases y los casos de uso en base a los requerimientos sentados en la documentación del TPA. También se hace el desarrollo de las validaciones de requerimientos de contraseñas.

Establecer el dominio

En gran síntesis, el objetivo del sistema solución es el de proveer a sus usuarios información de la accesibilidad provista por diferentes servicios públicos de manera personalizada según sus necesidades/preferencias.
Otro aspecto fundamental es que los usuarios podrán organizarse en comunidades con tal de estar al tanto de una problemática en común o mejorar el seguimiento de sus servicios de interés.

Modelado de clases

Dentro del modelado de clases, las más importantes son:

- Estación: Es la encargada de proveer los servicios de accesibilidad para sus usuarios, lo cual es en gran parte la esencia del sistema. Partiendo de que todas las estaciones van a proveer la mayoría de los servicios en un principio, se decidió darle más prioridad al conocimiento de los servicios faltantes. De estas también es importante saber la localización de sus entradas, que son abstraídas en una clase Entrada que tiene la calle, altura y barrio de cada una. Así obtenemos la ubicación de cada una de las estaciones. En el atributo “servicioFaltante”, dimos por sentado que todas las estaciones tienen todo tipo de servicio, pero si había alguno que faltaba, que quede bien especificado en cada instancia de clase, según lo indicado en el enunciado.
- Línea: Estas son las que conectan las diferentes estaciones que formarán parte de los trayectos del usuario. De estas se conocerán la estación de origen, la de destino, y las estaciones intermedias de su trayecto. Sabiendo que estas líneas podrán pertenecer a diferentes medios de transporte, ya sea subte o colectivo, se guardará información del tipo de medio al que pertenece.
  Si bien todavía no se profundizó en las implementaciones, encontramos que mantener el listado de líneas ordenado nos permitirá saber su recorrido.
- Servicio: Esta podría ser la clase más importante, ya que es la que define el servicio que debe ser prestado. Además del nombre de cada servicio, se desea saber su tipo y los tramos de distancia que poseen desde la entrada y desde el andén.
  Consideramos muy importante que esta clase sea lo más genérica posible, puesto que todavía no tenemos conocimiento de todos los servicios que podrán ser abarcados por el sistema.

- Comunidad: Otro aspecto importante del sistema es que los usuarios podrán formar parte de comunidades, las cuales tendrán como ejes exponer interés en alguno de los servicios provistos y/o exponer una problemática que no haya sido tomada en consideración desde un principio. Entre los usuarios de las comunidades habrá moderadores, que luego de ser validados por los administradores del servicio, tendrán la posibilidad de administrar la comunidad.
  Una cuestión adicional que se encontró es la creación de comunidades, puesto que si bien se aclara que el proveedor puede designar administradores, no se aclara cómo se las creará en esta entrega.

Requerimientos de seguridad

Otro aspecto importante del sistema será la seguridad de las cuentas del usuario, lo cual se ve reflejado en los estrictos requerimientos para la creación de contraseñas y el inicio de sesión. La responsabilidad de establecer qué contraseñas son aceptables recae en la clase Validador.
Esta clase estará compuesta por un listado de métodos cuya responsabilidad reside en establecer si la contraseña que reciben como parámetro cumple cierto requerimiento, y un método que al recibir una contraseña como String discrimina si esta cumple los requerimientos del listado.
Para la creación de métodos se decidió que implementen la interfaz Validable, pensando en la necesidad de que a futuro se puedan agregar/modificar los requerimientos de las contraseñas sin tener que modificar la clase mencionada anteriormente.

Archivos de configuración

Si bien nuestro diseño se realiza tomando en cuenta la posibilidad de que se pueda utilizar un archivo de configuración externo, todavía no se comenzó la implementación de esto puesto que no tenemos suficiente información sobre los requerimientos de las futuras implementaciones.

Sugerencias Primer entrega

- Patrón Adapter: Consideramos que en la validación del top 10 mil peores contraseñas se podría usar el Adapter para desacoplar la conexión con el archivo lo más posible de nuestro dominio. Una posibilidad a futuro es la de utilizar múltiples archivos de configuración según las circunstancias, e implementar este patrón nos permitirá guardar múltiples rutas a archivos sin tener que modificar el código.
- Comunidad tiene miembros tipo usuario: Pensamos que en las comunidades, su lista de miembros sea tipo usuario ya que cada miembro está directamente ligado a un usuario en particular.
- Listado de Comunidades en Usuario: La clase Usuario debería tener un atributo “listadoComunidades” que contenga información de las comunidades a las que pertenece, ya que sino tendríamos que consultar a todas las comunidades si el usuario pertenece a ellas.

Segunda entrega - 09/06/2023

En la segunda entrega, se generaliza el dominio de las líneas y estaciones de transporte público, convirtiéndolas en clases más genéricas llamadas entidades y establecimientos respectivamente. Se profundiza en la localización, agregando herramientas externas para obtener información de esta. Y para finalizar, se agregan usuarios asociados a las entidades previamente mencionadas y a organismos de control de entidades, que pueden ser importados mediante un archivo CSV.

Expandiendo los proveedores de servicios

Las entidades y establecimientos pasan a ser las prestadores de servicios, ya que estas son capaces de abarcar un dominio mayor al de las líneas y estaciones sin la necesidad de sacrificar los atributos previamente definidos; esto último es posible incluyendo la información no genérica mediante composición.

Los usuarios y sus intereses

Para tener en cuenta los intereses de cada usuario, lo que definimos fue abstraer Interés en una clase aparte, y como se aclaraba que cada interés era específico de un usuario, decidimos ponerle un atributo usuario asociado al interés. También se agregan las entidades y servicios del interés, y de ahí planteamos un comportamiento hipotético para sacar todos los servicios de interés, que son los deshabilitados en dichas entidades que le interesan.

Localización

A fin de poder obtener información exacta de las localizaciones, se recurrió a una API provista por GeoRef. Esto nos da la posibilidad de conocer:

- Provincias
- Localidades
- Municipios
- Departamentos
- Centroides

La obtención de estos se hace mediante una interfaz *Localizable*, que nos da la posibilidad de modificar la API a utilizar sin necesidad de modificar el código pertinente a la clase *Localización*.

Peticiones a la API

Si bien hay muchas peticiones desde diferentes enfoques que se pueden hacer a la API Georef,decidimos inicialmente definir 4 tipos de peticiones.
Para empezar, a la hora de crear un nuevo usuario o establecimiento (a alto nivel, desde una pantalla), en algún lado del formulario se debe pedir la localización de lo que se vaya a crear. Al principio se va a pedir seleccionar una provincia a elegir entre todas (representaría una petición al listado de provincias). Luego, en base a la provincia elegida se van a filtrar

los municipios, departamentos y localidades a ser mostradas para elegir (peticiones a listados por id de provincia). Entonces, las peticiones definidas a la API nos quedaron como:

- Listado de provincias
- Listado de municipios por id de provincia
- Listado de departamentos por id de provincia
- Listado de localidades por id de provincia

Todas estas peticiones se van a tratar polimórficamente en nuestra interfaz Localizable, usando el patrón Adapter con el servicio externo correspondiente (Georef actualmente) y estas 4 peticiones se van a realizar independientemente del servicio externo que nosotros usemos, devolviendo los listados ya instanciados como clases en nuestro dominio.
Como planteo hipotético, teniendo en cuenta que en el enunciado mencionaba por encima que cada entidad podia conocer el espacio geográfico donde tenga actividad, habíamos pensado que cada una calcule dichos espacios por medio de la localización de sus establecimientos, llamando a sus centroides donde tenemos la latitud y longitud exacta.
También se mencionaba por arriba que cada usuario podría llegar a necesitar saber los servicios que tenga cerca, y esto se planteó buscando todos los establecimientos que tengan una localización coincidente con la del usuario en ese momento, y de ahí se podían sacar los servicios disponibles de c/u de ellos.
Cabe aclarar que estas fueron ideas que se nos ocurrieron, pero al final las descartamos ya que no se especificaba a fondo el funcionamiento de estas dos funcionalidades (obtener espacios en actividad de una entidad, y ver servicios cercanos por usuario).
Faltó mencionar un planteamiento para simplificar las peticiones a la API. Como no es muy posible en el corto plazo que se actualice el listado de provincias, se pensó hacer un sincronizador para hacer una petición de provincias a la API y guardarlas en memoria, asi en próximos llamados, se ahorra usar el servicio externo (ya que podría ser un costo evitable) y consultar directamente en memoria (Clases API Data, Main y APIThread).

Entidades y organismos de control como usuarios

Un nuevo requerimiento es que las entidades prestadores y sus organismos de control sean capaces de recibir reportes de los servicios que prestan y su estado. Si bien todavía no se conoce en detalle qué información será la requerida para generarlos, si se sabe que existirán usuarios asociados que los recibirán, surgiendo de esto otro requerimiento, la posibilidad de importar a estos usuarios dentro de nuestro sistema.
Esto último se hará mediante archivos en formato CSV, que definirán:

- Si el usuario pertenece a una entidad o a un organismo de control.
- El o los nombres de las entidades asociadas a este.
- Las credenciales de inicio de sesión del usuario (Usuario y contraseña).

La responsabilidad de importar a los usuarios corresponde a la clase *ImportadorPrestadores*, que al ser inicializada se le asociará una instancia de un *Importable*.
El objetivo de la interfaz *Importable* es independizar el archivo que contiene los datos de los usuarios asociados y su extensión del proceso de importación de estos, permitiendo a futuro ofrecer importaciones desde otras fuentes

Sugerencias Segunda entrega

- Desacoplar la lectura de líneas de los archivos del proceso de importación. Para esto se plantean 2 clases separadas:
    - Una clase cuya responsabilidad sea convertir la información de los archivos a un listado de Strings, en el cual cada elemento sea una línea del archivo.
    - Otra clase encargada de mapear los elementos generados por la clase previa y almacenarlos en una lista a retornar.