*UTN- Facultad Regional Buenos Aires*

*Diseño de Sistemas - 2023*

TRABAJO PRÁCTICO INTEGRADOR ANUAL

*TEMA: Monitoreo de Estado de Servicios de Transporte Público y de Establecimientos*

**Grupo Nº12**

| **Apellido, Nombre** | **Legajo** |
| --- | --- |
| Condori, Joselin Magali | 203.520-0 |
| de los Ríos, Carolina | 213.242-4 |
| Kassem, Agustin | 213.248-5 |
| Piccinnino, Franco Ezequiel | 203.958-8 |
| Vazquez, Juan Martin | 204.144-3 |

**Fecha Presentación**: 27/04/2023

**Profesores:** Ezequiel Escobar

Gaston Prieto

**Ayudante:** Uriel Soifer

Servicio

* El estado de la clase servicio va a ser de tipo boolean ya que tenemos solo “Disponible” (true) y “No Disponible” (false).
* La relacionamos con la clase TipoServicio para poder representar que esos tipos se van a poder ir cargando a medida que necesitemos nuevos.

TipoServicio

* Es la categoría de un servicio, la modelamos con una clase con un atributo nombre (un String), sirve principalmente para poder agruparlos por tipos cuando sea necesario.
* Tendrá tipos de servicios ya parametrizados y otros se podrán ir agregando en tiempo de ejecución, es decir es extensible.
* No tiene comportamiento

Usuario:

* Tendrá un usuario, una contraseña y una persona asociada.
* Cuando un usuario desee unirse a una comunidad, se creará una instancia de Miembro que tendrá asociado al usuario en cuestión.
* Puede tener el rol de AdministradorPlataforma, lo que le permite acceder a todas las funciones de ese rol.
* Para enfocar mejor el propósito de la clase decidimos relacionarlo con la clase Persona.

Miembro:

* Contiene un atributo comunidad que apunta a Comunidad de la que es miembro y los métodos que puede ejecutar un miembro de una comunidad, como por ejemplo, poder informar sobre los cambios en los servicios de la plataforma.
* Puede tener el rol de AdministradorComunidad, lo que le permite acceder a todas las funciones de ese rol.

AdministradorComunidad

* Se instancia cada vez que una comunidad es creada y se agrega al diccionario de administradores del administrador de plataforma.
* Contiene un atributo comunidad que apunta a la comunidad de la que es administrador, acceso al ServicioController y los métodos que puede ejecutar un administrador de una comunidad, como por ejemplo, eliminar un miembro de dicha comunidad.

AdministradorPlataforma

* Esta clase es un rol que pueden, o no, tener los usuarios. Esta les da acceso a los controladores de servicio y servicio público, obteniendo métodos de los mismos, como, por ejemplo, eliminar un servicio.

ServicioController

* Es la clase que posee todos los métodos para administrar los servicios, los administradores de comunidades y los administradores de plataforma tienen acceso a una instancia de la misma.
* Con esta clase queremos lograr una reducción de la complejidad en los módulos, o sea que son más simples, y en este caso tener un menor número de operaciones en la clase.
* Genera mayor cohesión al estar las tareas agrupadas para que se enfoque en un único tipo de responsabilidad.

ServicioPublicoController

* Es la clase que posee todos los métodos para administrar los servicios públicos, los administradores de plataforma tienen acceso a una instancia de la misma.
* Con esta clase queremos lograr una reducción de la complejidad en los módulos, o sea que son más simples, y en este caso tener un menor número de operaciones en la clase.

Comunidad

* Una comunidad está conformada por una lista de Miembros y un nombre que la identifica.

Línea

* Esta clase contiene de atributos una *estacionOrigen*, *estacionDestino y* una lista *estacionesIntermedias* de tipo *Estacion*, la pusimos para representar que entre ese origen y destino tenemos varias paradas que se pueden hacer.
* El *estado* de la línea también está representado como un tipo boolean (“Disponible” (true) y “No Disponible” (false)).
* La relacionamos con *TipoTransporte* ya que para esté caso tenemos sólo ferrocarril y subterráneo, pero al hacer una clase aparte para el tipo de transporte pensamos que a futuro se puede llegar a pedir algún otro tipo.

TipoTransporte

* Modelamos el tipo de transporte como una clase con un atributo nombre de tipo String y no como un enumerado por si luego se quieren agregar otro tipo de transportes compatibles con el sistema.

Estación

* Esta clase contendrá una lista de los servicios que se encuentran en dicha estación. Además, tendrá un nombre y una ubicación.

Validador de Contraseña

* Con esta clase vamos a validar todas las contraseñas de las cuentas de los usuarios en el momento de registrarse.

Validador

* Es una clase abstracta que se encarga de implementar cada una de las validaciones de las clases ValidadorLongitudContrasenia, Validador10kContrasenia, ValidadorCaracter, ValidadorContraseniaNoContieneUsuario con un método esValida(). Utilizamos un patron strategy porque todas las validaciones que agregamos son formas de hacer algo.
* Es una clase extensible porque podemos crear más subclases que hereden los atributos y métodos con usos diferentes.
* Utilizando este patrón logramos que se puedan llegar a reutilizar los módulos ante un posible cambio. También pensamos que esto lo hace un sistema mantenible y extensible porque si llegamos a modificar alguno de estos módulos no vamos a tener que hacer tantos cambios en los demás.

ValidadorLongitudContrasenia

* Hereda de la clase Validador y contiene la implementación del método esValida(), verifica que la contraseña tenga más de 8 caracteres.

Validador10kContrasenia

* Hereda de la clase Validador y contiene la implementación del método esValida(), verifica que no se encuentre dentro de la lista de 10 mil contraseñas comunes.

ValidadorCaracter

* Hereda de la clase Validador y contiene la implementación del método esValida().

ValidadorContraseniaNoContieneUsuario

* Hereda de la clase Validador y contiene la implementación del método esValida().

IniciadorDeSesion

* Esta clase es la que se encargará de administrar los inicios de sesión de todos los usuarios.
* Tiene una lista de “IntentoInicioSesion” donde guardará los intentos de todos los usuarios.
* Para que un intento de inicio de sesión no sea válido, se chequeara que los últimos tres intentos sean incorrectos, que el tiempo transcurrido entre el ante penúltimo intento y el último no sea mayor a 10 minutos, y además que el tiempo transcurrido entre el último intento y el momento de comprobación no sea mayor a 10 minutos.

IntentoInicioSesion

* Las instanciaciones de esta clase las tendra la clase “IniciadorDeSesion”.
* Esta clase tendrá los atributos: esCorrecto (Boolean), fechaHora (LocalTime) y usuario (String).
* Esta clase solo servirá para llevar un registro de los intentos de sesión de un usuario.