Trabajo Final
Programación Orientada a Objetos II
2024

ALUMNOS:

- Hernán Moreyra – hernan_hello@hotmail.com
- Matías Galarza – matias.galarza.unq@gmail.com
- Jorge Terradillos – jorgenterradillos@gmail.com

SEM

El Sistema de Estacionamiento Medido *(SEM)*, administra el ingreso de registros de estacionamiento y compra, producidos desde los Puntos de Venta *(PuntoDeVenta)*, o por medio la App *(AppSem)*, que permiten que cualquier dueño de un vehículo que se encuentre dentro de las Zonas de Estacionamiento del SEM *(ZonaDeEstacionamiento)* pueda registrar su estacionamiento dentro del sistema, además de poder finalizarlos.
Además, permite que entidades puedan suscribirse al sistema y así obtener notificaciones sobre cada registro y finalización de un estacionamiento, además de los registros de recarga que se realicen en los Puntos de Venta.
También, permite que inspectores puedan buscar si en el sistema existe un estacionamiento dada una patente determinada, y registrar infracciones.
El sistema tiene registrados todos los RegistrosDeEstacionamiento realizados *(tanto vigentes como no vigentes)*, los celulares con sus números de teléfono y respectivos saldos disponibles de cada recarga realizada en un Punto de Venta, las Zonas de Estacionamiento del sistema, los Registros de Compra *(tanto por estacionamiento puntual como recarga de saldo)*, cada Infracción realizada por el Inspector de cada zona, y las Entidades Observadoras.
A la hora de registrar un estacionamiento, el SEM depende del estado en el que se encuentre según el horario, ya que fuera del horario vigente es imposible generar estacionamientos nuevos. Esto se representa por medio de un **patrón State**, en el cuál el mismo SEM es el contexto del cuál depende la delegación de registrar estacionamiento en el estado de abierto o cerrado.

EntidadObservadora

Para las entidades observadoras, utilizamos un **patrón Observer**, el cual permite la suscripción y desuscripción a las entidades interesadas en observar las acciones del SEM. Está implementado de forma genérica para poder notificar a las entidades sobre cualquier cambio que se quiera notificar a futuro.
EntidadObservadora es la interfaz que cumple el rol del Observer. Cualquier entidad que quiera suscribirse debe implementar esta interfaz.
SEM es la clase que cumple el rol del Subject *(u Observable)*. La misma notifica a las entidades suscritas sobre las acciones que se realicen en el SEM.

Notificacion

Las notificaciones están diseñadas para almacenar el estado de las acciones de registrar un estacionamiento y finalizar un estacionamiento del SEM. También otorgan alertas para ayudar a los clientes a gestionar adecuadamente sus movimientos.
Estas notificaciones son luego enviadas a la AppSem.
La Notificacion es una clase abstracta que se asegura que aquellas clases que extiendan de ella, deban implementar el método “getMensaje()”, la cual es posteriormente usada por AppSem. Las clases que extienden de Notificacion son:

- **NotificacionDeInicioExitoso**: Almacena la información del registro de un estacionamiento que fue registrado con éxito.
- **NotificacionMensajePersonalizado**: Almacena un mensaje personalizado *(es decir, el mensaje puede ser modificado)*. Utilizado mayormente para armar mensajes de, por ejemplo, operaciones denegadas en el SEM.
- **NotificacionDeFin**: Almacena la información del registro de un estacionamiento que fue finalizado con éxito.
- **AlertaDeInicio**: Almacena el mensaje de alerta para que el cliente que la reciba sea alertado sobre que su estacionamiento aún no fue registrado.
- **AlertaDeFin**: Almacena el mensaje de alerta para que el cliente que la reciba sea alertado sobre que su estacionamiento aún es vigente.

AppSem

El manejo de los pedidos de estacionamiento desde la aplicación es iniciado desde la clase *AppSem*. Estos pedidos de estacionamiento dependen, a su vez, de las señales recibidas mediante los mensajes definidos por una interfaz (*MovementSensor*) que indica el cambio de estado del usuario entre su desplazamiento a pie o en vehículo. Por ello, adoptamos un **patrón State**, el cuál tomando a AppSem como su contexto, emplea el modo de caminando o manejando según la ocasión, con las respectivas clases de *EstadoCaminando* y *EstadoManejando*.
Por otro lado, AppSem también debe prestar al usuario la posibilidad de alternar entre dos modos para registrar los estacionamientos. Así que para describir esta situación, se toma un **patrón Strategy** para representar la variación en lo que, por ahora, son dos posibilidades: el modo manual donde el usuario debe accionar la aplicación para registrar el estacionamiento, y el modo automático, donde la detección del cambio entre manejando y caminando se detecta de forma inmediata a la hora de registrar nuevos estacionamientos. Estas serían las dos posibilidades que plantea para resolver ese mismo problema, plasmadas en las clases de *ModoManual* y *ModoAutomatico*.

RegistroDeEstacionamiento

Los registros de los estacionamientos son tomados como objetos que tienen una serie de datos esenciales y que son guardados en el objeto SEM. En general, todos los registros deben tener que retornar cosas elementales como sus datos de inicio, la zona en donde se realizan y el estado de vigencia. Por lo tanto, podría verse un cierto patrón Template, en el sentido que tantos los estacionamientos realizados por medio de AppSem, como aquellos hechos de forma puntual en PuntosDeVenta entienden mensajes de este tipo. Sin embargo, dependiendo del tipo de estacionamiento realizado, pueden tener ciertos datos adicionales, como el númer desde el que se realiza en el caso de *RegistroDeEstacionamientoApp*, o las horas de duración en caso de *RegistroDeEstacionamientoPuntual*. La clase genérica para ambas formas de estacionamiento es un *RegistroDeEstacionamiento* que puede responder a todo salvo a las diferencias particulares entre los dos tipos de estacionamiento.

RegistroDeCompra

Similar a los registros de estacionamiento, el enunciado también pedía información de las transacciones hechas por medio del SEM, por lo que hemos creado una distinción entre las cargas de saldo y los registros puntuales, ambos originados desde el objeto *PuntoDeVenta*. En la clase genérica de *RegistroDeCompra*, tendremos el patrón de Template, ya que allí quedarán mensajes relacionados con los datos más comunes: la fecha en que se efecutó la transacción, un número de registro y el punto de venta desde donde se realizan. En cambio, en los *RegistroDeCompraPuntual* también hay información sobre cierto vehículo concreto que realiza el estacionamiento puntual,y, en el *RegistroDeRecarga* dejaremos constancia del número de celular que se recarga y el monto por el cuál se realizó la transacción.