TP Integrador Programación Orientada a Objetos 2

Integrantes:

- Duarte, David Ignacio (davidignacio20@outlook.com)
- Barberena, Jorge Daniel (daniel.barberena10@gmail.com)
- Díaz, Juan Pablo (fdiaz.juanpablo@gmail.com)

Decisiones de diseño

Comenzamos diagramando el UML más importantes (SitioInmuebles, Usuario, Inmueble) y luego surgieron aquellas a las que se les delega su comportamiento, tales como la clases Reserva, Rankeo, PrecioPeriodo y PoliticaCancelacion. Esta última la pensamos como una clase abstracta de modo que sirva para definir el comportamiento en común de cada una que la extienda para cumplir con lo esperado, aumentando la flexibilidad con respecto al agregado de nuevas políticas de cancelación.
Se creó la interfaz IRankeable debido a la necesidad de poder rankear tanto a inquilinos como a inmuebles, permitiendo al sistema agregar nuevas entidades que puedan ser rankeadas al implementar esta interfaz. A su vez, creamos la clase GestorDeRankeos que permite modularizar la lógica de IRankeable y gestionar al mismo tiempo las categorías con su respectivo puntaje y comentarios.
También se creó la interfaz IVisualizable, que permite mostrar en una Pantalla la información relevante de cada entidad (Inmueble, Usuario), permitiendo agregar más clases que implementen su interfaz.
Reserva funciona como una clase que brinda información referente a la operación del alquiler de un Inmueble, a la que se puede acceder en distintas instancias del proceso. Para un mejor manejo de estar es gestionada por la clase GestorDeReservas, incluida en SitioInmuebles, para encapsular la lógica y, en el caso de necesitar modificar su implementación, no es necesario cambiar la clase SItioInmuebles
Un enfoque similar se utilizó con la clase GestorDeNotificaciones, la cual contiene los sensores necesarios para notificar a diferentes entidades, mientras implementen la interfaz adecuada. La implementación de estos sensores se realizó utilizando el patrón de diseño Observer, donde el rol de Observer lo cumplen las interfaces IListenerBajaDePrecio, IListenerCancelacion, IListenerReserva. Los observadores concretos son aquellos que implementan estas interfaces, como por ejemplo Usuario. El rol de ConcreteSubject lo cumplen las clases SensorBajaDePrecio, SensorCancelacion y SensorReserva.
A modo de conclusión, con lo expuesto anteriormente creemos que respetamos los principios SOLID, ya que se buscó delegar en todo momento comportamientos de clases colaboradoras; se pensó usar súper clases para evitar tener que modificar clases ya existentes en caso de tener que agregar comportamiento nuevo o nuevas clases, de forma que extiendan el dominio. En cuanto a las interfaces, se buscó separarlas en protocolos específicos, como en el caso de los sensores.