Universidad Nacional de Quilmes

PROGRAMACIÓN CON OBJETOS II

TRABAJO PRÁCTICO FINAL

SISTEMA DE ALQUILERES TEMPORALES

PROFESORES:

- DIEGO TORRES
- DIEGO CANO
- MATIAS BUTTI

ALUMNOS:

- JULIÁN VÁZQUEZ // *julivazquez09@gmail.com*
- ALAN PACHECO // *pachechoalannahuel@gmail.com*
- DAVID CARREVEDO // *davidcarrevedo@gmail.com*

https://github.com/Nahuel-P/unqui-po2-alquileres-temporales

INDICE

https://github.com/Nahuel-P/unqui-po2-alquileres-temporales

ALCANCE

DESCRIPCION DEL SISTEMA

Se implementa la lógica de un Sitio Web con la capacidad de comunicar Usuarios que poseen propiedades en alquiler, y Usuarios interesados en alquilarlas.

Se desarrolla en lenguaje Java, con cobertura de Tests JUnit 5 mayor al 95% y el uso de Mockito como asistencia en los Tests.

DECISIONES DE DISEÑO

WEB RESERVAS

Se pensó en este objeto como dispositivo de entrada al sistema, con la idea de manejar las funcionalidades más generales del programa e influyendo en varias de las otras clases, pero sin perder de vista el Single Responsability. Es por esto que las funcionalidades complejas fueron delegadas a otras clases.

BUSQUEDA MEDIANTE FILTRADO

Originalmente se planteó la búsqueda siguiendo un patrón Composite, donde se partiría de una búsqueda básica para luego ir sumando parámetros que hicieran la búsqueda más compleja.

Luego se pensó (opción que finalmente se implementó) en el objeto Buscador, que recibirá Publicaciones y tendra una relación de dependencia con la clase Filtro. Ésta última es una clase abstracta, y sus calses hijas serán diferentes tipos de Filtros que aplicarán cada una un criterio propio de búsqueda. De esta manera llegamos al patrón Template Method, dejando la posibilidad de agregar futuros filtros que únicamente deberan extender de la clase Filtro.

USUARIO

También susfrió modificaciones en su idea original. En un principio fue pensado como una clase abstracta, que tendría clases hijas que definirían el comportamiento del usuario, luego se llegó a la conclusión de que una misma instancia de usuario podría querer tener distintos tipos de funcionalidades y que no se lo podía limitar, por eso se mantuvo como clase concreta y única.

Entre sus funciones, se destacan las de crear y recibir reservas/publicaciones

https://github.com/Nahuel-P/unqui-po2-alquileres-temporales

PUBLICACION

Una publicación será la encargada de combinar información de distintas clases en un lugar. Guardará un único Inmueble y podrá acceder a todos sus datos; y a través suyo, un Usuario también podrá acceder a datos de Inmuebles.

Conoce a una clase Observer y posee un método de notificación, de esta manera podra avisar a distintos observadores dependiendo sus intereses *(ver OBSERVADORES DE RESERVAS Y PUBLICACIONES)*.

Una publicación, además, podrá tener diferentes precios dependiendo la fecha en la que sea reservada *(ver TEMPORADAS ESPECIALES)*

RESERVA

Es la concreción de la interacción entre 2 Usuarios.

Mediante el uso del patrón Template Method, se establecen distintos tipos de reserva, que dado que tienen métodos similares los heredarán de una clase padre abstracta, EstadoReserva, y los sobrescribirán. De esta manera, se deja abierta la posibilidad de seguir agregando futuros tipos, respetando así el principio SOLID de Open/Closed.

BIBLIOTECA DE RESERVAS Y PUBLICACIONES

Se basan en la lógica del patrón de diseño Facade.

El objetivo es concentrar en distintas bibliotecas (fachadas) las distintas reservas y publicaciones que se vayan concretando. De esta manera logramos quitar a la web la responsabilidad del guardado, dándole independencia y desacoplandola de los clientes que necesiten dicha información. Incluso se vuelve reutilizable con clases futuras que pudieran requerirla.

OBSERVADORES DE RESERVAS Y PUBLICACIONES

En ambos casos, los observadores tendrán la posibilidad de suscribirse a la clase de la cual deseen recibir notificaciones. También, ambas clases lo hacen a través de una interfaz, dejando abierta la posibilidad de recibir distintos tipos de observadores y respetando Open/Closed.

En el caso puntual de las reservas, el observador será con listeners, haciendo que nuestra interfaz sea un poco más compleja. Esto choca contra el principio de Segregación de Interfaces, pero aporta la correcta funcionalidad que se buscaba.

CALIFICAIONES

Tanto Usuarios como Inmuebles, son calificables. Esto hizo pensar en que debería existir una interfaz que nos permitiera tratarlos polimórficamente, incluso a clases que pudieran agregarse en un futuro y recibir calificaciones.

A su vez las calificaciones serán guardadas en una Biblioteca de Calificaciones, volviendo a uitilizar la lógica de Facade, dándole independencia al Sitio Web y haciendo reutilizables los datos.

TEMPORADAS ESPECIALES

Surgen de la necesidad de tener una forma de crear distintos precios para distintas fechas, ya que un Usuario puede determinar varias temporadas con precios de descuento.

Es así que la Publicación delegará la tarea de establecer estos precios a la clase Precio Temporal, y ella (la Publicación) se encargará únicamente de almacenarlos.

(continúa abajo)

PATRONES DE DISEÑO UTILIZADOS

OBSERVER

- Publicación / IPriceObserver
  - ConcreteSubject: Publicación
  - Subject: Observer
  - Observer: IPriceObserver
  - ConcreteObserver: clase que implemente la interfaz IPriceObserver

OBSERVER CON LISTENER

- Reserva / IBookingListener
  - ConcreteSubject: Reserva
  - Subject: Reserva
  - Observer: BookingListener
  - ConcreteObserver: clase que implemente la interfaz IBookingListener

FACADE

- WebReservas / BibliotecaDeReservas / Reserva
  - Client: WebReservas
  - Facade: BibliotecaDeReservas
  - SubClass: Reserva

- WebReservas / BibliotecaDePublicaciones / Publicacion
  - Client: WebReservas
  - Facade: BibliotecaDePublicaciones
  - SubClass: Publicacion

- WebReservas / BibliotecaDeCalificaciones / Calificacion
  - Client: WebReservas
  - Facade: BibliotecaDeCalificaciones
  - SubClass: Calificacion

TEMPLATE METHOD

- Reserva / EstadoReserva / Estados
  - Client: Reserva
  - Strategy: EstadoReserva
  - Implementation: Los 6 estados disponibles para una reserva

TEMPLATE METHOD

- Buscador / Filtro / Filtros
  - Client: Buscador
  - AbstractTemplate: Filtro
  - Implementation: Los 3 filtros existentes

https://github.com/Nahuel-P/unqui-po2-alquileres-temporales