

Programación Orientada a Objetos 2

*Trabajo Práctico Integrador*

*Alquileres - Hito 1*

Grupo: UndefinedObject

Integrantes:

Fernández Matías - fnandez.matias@gmail.com

Maguna Leandro - leandromaguna@gmail.com

Malsam Leandro - leandro.malsam@gmail.com

Introducción

El presente documento contiene las decisiones de diseño tomadas para la implementación de la solución, así como algunos detalles del proyecto Java correspondiente.

Generalidades

Se utilizó la librería externa Mockito para algunos tests, la cual se incluye en el mismo proyecto dentro de la carpeta lib.

Se decidió programar en inglés.

Solución propuesta

*Objetos auxiliares*

Englobamos dentro de este grupo una serie de objetos básicos, con poco o nulo comportamiento, que son colaboradores de otros objetos más complejos. En la mayoría de los casos los mismos podrían haber sido simplemente cadenas de texto, pero a fin de modelar correctamente la realidad y hacer más entendible la solución, se decidió por modelar estos objetos.

Los mismos son:

* PropertyKind: define el tipo de propiedad, ya sea Departamento, Casa, Cabaña, etc.
* Service: un servicio incluido en la propiedad, como ser WiFi, calefacción, y demás.
* Country: un país.
* City: una ciudad, que está ubicada en un país.
* Address: la dirección de una propiedad (país, ciudad, calle y número).
* ReviewCategory: una categoría a calificar, como podría ser “limpieza de la propiedad”, “cordialidad”, “solvencia ante problemas”, etc.

Todos esos objetos pueden ser registrados en el sistema en forma manual.

Inicialmente pensamos en modelar las monedas (pesos, dólares, etc.) y el precio como un objeto que tenga el valor y la moneda en que está expresado. Pero esto le agregaba complejidad a la solución, y al momento de comparar precios en distinta moneda había que agregar lógica adicional para convertir todo a una moneda única. Por lo que finalmente decidimos no utilizar estos objetos y expresar los precios con un simple número, asumiendo que todos estarían expresados en la misma moneda.

*Objetos de negocio principales*

Son los objetos principales de la solución, aquellos que tienen más colaboradores y comportamiento, y sobre los que se basa toda la gestión. En este grupo están:

* User: un usuario del sistema. El mismo puede ser “inquilino” (tenant) o “propietario” (owner), pero este rol está asociado a la acción, por lo que decidimos que no era necesario modelar el mismo. Es decir, si un usuario publica una propiedad, será su propietario, y si hace una reserva, será un inquilino.
* Property: una propiedad sobre la que se pueden crear avisos, incluye todos sus detalles de ubicación, servicios, etc.
* Listing: es una publicación, que contendrá una propiedad, un precio, los horarios de entrada y salida, etc. Pueden existir varias publicaciones para una misma propiedad.
* Booking: es la reserva que hace un usuario (inquilino), sobre una publicación, cuando decide alquilar una propiedad. La misma tendrá un estado, que permitirá seguir un flujo de transiciones e interacciones entre inquilino y propietario.
* Review: es una calificación, que incluye 3 categorías a evaluar. Estas revisiones pueden ser:
  + TenantToPropertyReview: El inquilino rankea al inmueble.
  + TenantToOwnerReview: El inquilino rankea al dueño del inmueble.
  + OwnerToTenantReview: El dueño del inmueble rankea al inquilino.
* BookingState: interfaz que define el comportamiento de los posibles estados (patrón State).
  + PendingBookingState: un inquilino hizo una reserva y queda a la espera de ser aprobada.
  + ApprovedBookingState: el propietario aceptó una reserva.
  + RejectedBookingState: el propietario rechazó una reserva.
  + CancelledBookingState: el inquilino canceló su reserva.

*El sistema*

Se modeló el sistema como objeto principal, desde el cual se llega a las distintas funcionalidades.

* BookingSystem: es la que contiene a todos los otros objetos del sistema y permite su administración, directa o indirectamente. Está modelado como un Singleton.
* RankingSubsystem: Otro objeto singleton, destinado a administrar todo lo referido a las revisiones, el ránking y promedios de calificaciones para inquilinos, propiedades y propietarios.
* PropertiesSubsystem: También un singleton, para administrar las propiedades y los objetos relacionados a las mismas.

Casos de Uso a programar

*2.a. Poder crear y publicar propiedades.*

Este punto está dividido en 2 casos distintos. Por un lado, el usuario puede crear una propiedad (Property), pero no necesariamente publicarla. Esto es administrado por el subsistema de propiedades.

Por otro lado, podrá armar una publicación (Listing) con alguna de sus propiedades y dejarla disponible para ser alquilada. Esto es manejado por el sistema principal, el de reservas.

Fuente del proyecto

GitHub: <https://github.com/madnotdead/PPO2_TPIntegrador>