Informe

Trabajo Final
Sistema de Alquileres

Integrantes

Aguirre, Tomás Ezquiel
tomi_aguirre01@hotmail.com

Ibarra, Ezequiel Agustín
ibarra.ezequiel01@gmail.com

Volosin, Josefina
volosinjosefina@gmail.com

2do cuatrimestre, 2024

Programación con Objetos II

Departamento de Ciencia y Tecnología

Universidad Nacional de Quilmes

Introducción

A continuación, se presentan las decisiones de diseño, detalles de implementación y patrones de diseño aplicados en el desarrollo de un sistema de alquileres temporales.

Decisiones de Diseño

Gestión de Notificaciones (Observer)

Dado que ciertos eventos (como cancelaciones de reservas y cambios de precio) deben ser comunicados a usuarios específicos, se aplicó el patrón Observer. NotificationManager se encarga de gestionar las suscripciones y emitir notificaciones a los listeners interesados, brindando flexibilidad y escalabilidad al sistema de alertas sin requerir modificaciones en el código base.

Búsqueda de Propiedades (Composite)

Para ofrecer búsquedas avanzadas, donde los filtros se puedan combinar de forma flexible, se implementó el patrón Composite. Este patrón permite anidar y combinar filtros usando operadores lógicos (como AndFilter y OrFilter), permitiendo que el sistema maneje criterios de búsqueda complejos.

Modularización del Proceso de Checkout

El proceso de checkout en BookingManager se organiza en una secuencia de pasos bien definida, delegando la lógica de actualización de rankings a RankingManager. Esta separación permite a BookingManager centrarse en la gestión de reservas, mientras que RankingManager gestiona los puntajes, asegurando una mayor cohesión en cada clase.

Cálculo de Rankings (Strategy)

Dado que el cálculo de rankings puede variar según diferentes criterios, se implementó el patrón Strategy en RankingManager. Esto permite que el sistema aplique diferentes estrategias de evaluación sin necesidad de modificar RankingManager, facilitando la adaptabilidad del sistema a futuras necesidades de cálculo de ranking.

Políticas de Cancelación (Strategy)

Cada propiedad puede definir su propia política de cancelación, con reglas específicas sobre reembolsos según el tiempo restante hasta la fecha de inicio de la reserva. Con el patrón Strategy, estas políticas se encapsulan en clases específicas (FreeCancellationPolicy, NoCancellationPolicy, etc.), permitiendo personalizar las condiciones de cancelación sin modificar BookingManager.

Detalles de Implementación

Notificaciones y Propiedades

Para diferenciar qué usuarios deben recibir ciertos mensajes en función del tipo de evento, NotificationManager utiliza un diccionario que organiza listeners por tipo de evento y propiedad específica, permitiendo una comunicación precisa y selectiva. PropertyEvent encapsula la información del evento y la envía a los listeners suscritos, que implementan interfaces específicas (HomePagePublisher o PopUpWindow).

Cálculo de Puntuación y Políticas de Cancelación con Strategy

RankingManager aplica SimpleRankingStrategy por defecto, pero el sistema permite agregar nuevas estrategias de evaluación sin modificar la clase principal. Del mismo modo, las políticas de cancelación se implementan en diferentes clases para encapsular los distintos criterios de reembolso. Esto permite a BookingManager aplicar la política seleccionada en cada caso sin cargarlo con la lógica de reembolso específica de cada política.

Patrones de Diseño

Observer (Notificaciones)

- Observable/Subject -> NotificationManager: Permite que múltiples observadores (listeners) se suscriban y desuscriban de eventos específicos.
- Observer -> EventListener: Define la interfaz para los objetos que desean recibir notificaciones de cambios en las propiedades.
- ConcreteObserver -> WebSitePublisher y MobileAppListener: Reaccionan a eventos específicos, como una cancelación de reserva o una baja de precio.
- ConcreteObservable/Subject -> PropertyEvent: Representa el evento específico que interesa a los observadores.

Composite (Búsqueda de Propiedades)

- Client -> PropertiesManager: Utiliza la interfaz Filter para realizar búsquedas de propiedades.
- Component -> Filter: Define la interfaz común para todos los filtros de búsqueda.
- Composite -> AndFilter y OrFilter: Permiten combinar varios filtros simples en una estructura jerárquica.
- Leaf -> CityFilter, PriceFilter, DateFilter, y GuestFilter: Representan filtros individuales que no tienen hijos.

Strategy (Ranking y Políticas de Cancelación)

Ranking:

- Strategy -> RankingStrategy: Define una interfaz común para las distintas estrategias de cálculo de puntaje.
- ConcreteStrategy -> SimpleRankingStrategy: Implementa una estrategia de cálculo de ranking simple.

- Context -> RankingManager: Colabora una estrategia concreta y utilizándola para calcular puntajes.

Políticas de Cancelación:

- Strategy -> CancellationPolicy: Declara una interfaz común para las políticas de reembolso que se aplican en caso de cancelación.
- ConcreteStrategy -> FreeCancellationPolicy, NoCancellationPolicy, y IntermediateCancellationPolicy: Cada clase concreta implementa una política específica de reembolso en caso de cancelación.
- Context -> BookingManager: Colabora una estrategia concreta y utilizándola para calcular reembolsos.

Template Method (User)

- Abstract Class -> User: Define la estructura del algoritmo de calificación a través del método rateAfterCheckout.
- Concrete Class -> Tenant y Owner: Personalizan pasos específicos del proceso de calificación.