
# PROGRAMACIÓN ORIENTADA A OBJETOS II

2do Cuatrimestre 2024

TRABAJO FINAL: SISTEMA DE ALQUILERES TEMPORALES

Integrantes:

- Coniglio, Lucila Victoria | lucilaconiglio@hotmail.com
- Lefloth, Fernando | mellolefloth@gmail.com
- Ventola, Gian Franco | gianfranco.ventola@gmail.com

## Publicación:

Se creo la clase **Publicacion** que representa la informacion de un Inmueble que un propietario desea alquilar, esta clase contiene toda la informacion del mismo. Además, **Publicacion** incluye una lista de periodos donde se registran los intervalos de fechas con precios diferentes. Esta lista se utiliza para calcular el precio final del alquiler.

## Busqueda de Inmueble:

Se implementó una lógica de búsqueda para filtrar inmuebles utilizando una interfaz de búsqueda (Search), que es implementada tanto por la clase FiltroBase como por la clase abstracta Filtro. La clase FiltroBase contiene los filtros obligatorios de la búsqueda, como la ciudad de destino y las fechas de entrada y salida, garantizando que solo se muestren los inmuebles disponibles en la ciudad y en el período solicitado.

Además, a FiltroBase se le pueden añadir otros filtros opcionales, como FiltroCantidadDeHuespedes para la cantidad de huéspedes y FiltroPrecioMin o FiltroPrecioMax para establecer un rango de precios. Estos filtros adicionales permiten al sistema ajustar los resultados de la búsqueda de acuerdo con los parámetros específicos ingresados por el usuario, ofreciendo flexibilidad y precisión en la selección de inmuebles.

## Ranking

Para implementar el sistema de ranking, se creó una interfaz Rankeable que permite a distintas entidades ser evaluadas en el sitio. La clase Publicacion implementa esta interfaz para que cada inmueble pueda ser rankeado. Además, las entidades Usuario, que se dividen en Inquilino y Propietario, también implementan Rankeable para ser evaluadas en su comportamiento y confiabilidad.

La lógica de cálculo y almacenamiento de los rankings se centralizó en la clase Ranking para evitar duplicación de código. Esta clase gestiona tanto los puntajes individuales como los promedios de las calificaciones de las diferentes entidades. Tanto el usuario como una publicacion conocen a su clase ranking.

## Concrecion de una reserva

Para manejar las reservas, se decidió implementar el **Patron State**, ya que permite gestionar los distintos estados de una reserva según las acciones de los usuarios.

Los estados que componen el patron state son los siguientes:

**EstadoPendienteDeAprobacion**:

- Este es el estado inicial de la reserva cuando el inquilino ha realizado una solicitud de reserva, pero el propietario aún no ha tomado una decisión.
- **Acciones posibles**: El propietario puede **aceptar** o **rechazar** la reserva.

**EstadoConsolidada**:

- Este estado indica que el propietario ha **aceptado** la reserva. En este momento, la reserva está confirmada.
- **Acciones posibles**: El inquilino puede **cancelar** la reserva (lo que cambiaría el estado a EstadoCancelada). Una vez realizada la estancia, el estado cambiará a EstadoFinalizada al realizar el check-out.

**EstadoCancelada**:

- Este estado refleja que la reserva ha sido **rechazada** por el propietario o **cancelada** por el inquilino después de haber sido aceptada. Si la reserva es rechazada, la acción es tomada por el propietario; si es cancelada, es una acción tomada por el inquilino después de la confirmación.
- **Acciones posibles**: Una vez que una reserva está en estado cancelado, no puede pasar a ningún otro estado útil, ya que ha sido anulada.

**EstadoFinalizada**:

- Este estado indica que la reserva ha llegado a su **finalización**. El inquilino ha hecho el check-out y el alquiler ha terminado correctamente.
- **Acciones posibles**: Después de que una reserva llega a EstadoFinalizada, no puede cambiar más, ya que el proceso de alquiler ha concluido.

Este enfoque permite una transición clara y controlada entre estados, asegurando que cada acción sea coherente con el estado actual de la reserva.

Participantes del Patron State:

- CONTEXT
  - Reserva
- STATE:
  - EstadoReserva
- CONCRETE STATE
  - EstadoPendienteDeAprobacion
  - EstadoConsolidada
  - EstadoCancelada
  - EstadoFinalizada

## Administracion del sitio:

El sitio maneja una lista de tipos de inmuebles y servicios, con sus categorías gestionadas por un Manager de Categorías. Este manager es una instancia única que centraliza la gestión de categorías en todo el sitio. Utiliza un mapa que recibe un objeto rankeable (como inquilino, propietario o publicación) y devuelve la lista de categorías correspondientes. El manager ofrece métodos para obtener las categorías de cada tipo de rankeable según sea necesario.

Se decidió usar Singleton para el manager de categorías porque:

- Centraliza el acceso a las categorías desde una única instancia.
- Optimiza recurso al evitar crear múltiples instancias.
- Garantiza consistencia en toda la aplicación al manejar las categorías desde un solo punto. Facilita el mantenimiento, ya que cualquier cambio se realiza en una única instancia.

Un Singleton es un patrón de diseño creacional que asegura que una clase tenga solo una instancia en toda la aplicación, y proporciona un punto de acceso global a esa instancia.

## Politicas de cancelación

Para manejar las políticas de cancelación, se decidió utilizar el **Patrón Strategy**, lo cual permite que una publicación pueda definir su propia estrategia de cancelación sin necesidad de cambiar la lógica interna de la clase Publicacion.

Se define una interfaz PoliticaDeCancelacion que contiene el método que determina cuánto debe pagar el inquilino dependiendo de la fecha de cancelación en relación con la fecha de reserva

Participantes del Patron Strategy:

- CONTEXT
  - Publicacion
- STRATEGY:
  - PoliticaDeCancelacion
- CONCRETE STRATEGY
  - CancelacionGratuita
  - Intermedia
  - SinCancelacion
  -

**Notificaciones Observer**

Para implementar este sistema de notificación flexible, en el que diferentes entidades (como el sitio externo y los usuarios móviles) se suscriben a eventos específicos, podemos seguir el **Patron Observer**. Este patrón permite que el sistema notifique a los "observadores" (suscriptores) cuando ocurre un evento, sin necesidad de modificar el código de los observadores o el sujeto que emite el evento.

Participantes del Patron Observer:

- SUBJECT
  - Publicacion
- OBSERVER:
  - EventListener
- CONCRETE OBSERVER
  - SitoExterno
  - UsuarioMovil

## Reserva condicional

La publicación tiene dos listas: una de reservas y una cola de reservas condicionales. Cuando se recibe una reserva, si hay conflicto de fechas, se agrega a la cola de reservas condicionales; de lo contrario, se guarda en la lista de reservas. Al cancelar una reserva, se revisa si hay alguna reserva condicional en espera, y si es el caso, se mueve a la lista de reservas.