# Integrador  Objetos 2

# Sistema de Alquileres temporales

## Integrantes:

Micaela Sans. Mica.sans@hotmail.com

Sebastian Correa. Sebastiancorreaemanuel@gmail.com

En este trabajo integrador se nos pide diseñar una solución completa de un sitio web sobre un sistema de alquileres temporales, donde se facilita el contacto entre personas que tienen algún inmueble para alquilar de forma temporal (por ejemplo, para vacaciones) y aquellos que desean alquilar, eliminando inmobiliarias y otros mediadores.

Se nos pide ciertos requisitos sobre:

*Publicación y Alquiler

*Búsqueda de inmuebles

*Ranking de inmuebles y propietarios

*Visualización y Reserva

*Concreción de una Reserva

*Administración de reservas para inquilinos

*Administración del sitio

*Políticas de cancelación
*Notificación

*Reserva Condicional

Al momento de diseñar el UML, habremos estado unas semanas, decidimos que la clase SistemaDeAlquiler, sea como una “base” de este, ya que el sitio web se basaba en este.

Cosas importantes al momento del diseño del UML pueden ser la decisión de darle un estado a publicación o a reserva, pero luego de entender bien la consigna decidimos dársela a reserva y que la publicación tenga una política de cancelación.

La implementación de la búsqueda y los filtros fueron muy difíciles de implementar, ya que el filtroBase debía ser obligatorio y se podría agregar otros.

Luego de tener una versión del primer UML, a la hora de programar se hacía todo más ameno. Lo que si a lo largo del trabajo fuimos agregando métodos y a veces clases al UML que lograban tener un sentido mientras programábamos.

Otro tema que fue de discusión fue si implementar una separación entre Inquilinos y Propietarios, pero por consigna supimos que un usuario podía/es ambos.

**Patrones de diseño utilizados:**

State: En la Interface “Estado”, utilizamos este patrón. Dependiendo el estado de la reserva cada mensaje tiene su propia implementación, y estas llevan a cambiar el estado de las reservas.

La interface “Estado” cumple el rol de Estado/State.

“Pendiente”, “Reservada”, “Obsoleta”, “Alquilada” cumplen el rol de EstadoConcreto/ConcreteState.

“Reserva” cumple el rol de Contexto o Context

Strategy: Este patrón se puede ver claramente en la interfaz “PoliticaDeCancelacion”, donde se elige la política a utilizar en la publicación, y dependiendo el método recibido implementa diferentes cosas. La diferencia con el patrón anterior es que estas políticas no cambian la política de publicación. a otra diferente.

La interfaz “PoliticaDeCancelacion”cumple el rol de Estrategia/Strategy.

“SinCancelacion”, “CancelacionGratuita”, “Intermedia” cumplen el rol de EstrategiaConcreta/ ConcreteStrategy

“Publicacion” cumple el rol de Context.

Observer: La clase “ObservadorDelSistema” utiliza el patrón de observer con suscriptores, y dependiendo el método recibido reaccionaran distintos suscriptores, o listeners

Luego de la primera corrección, tuvimos que hacer un gran cambio en el tema de las reservas condicionales, ya que teníamos un grave error de que se podían aceptar distintas reservas en la misma fecha, lo que llevándolo a la vida real podría causar un gran problema.

También tuvimos que hacer cambios en el tema de los filtros y códigos repetidos en los métodos relacionados con los rankings.

Implementamos una nueva clase para poder realizar fechas especiales con distintos precios, y no que sumen un porcentaje ( o resten) del precioBase de la publicación.