**Patrones De Diseño**

**Patrones Básicos**

***Constant Data Manager:*** Este Patrón básico no se aplica debido a que el nivel de constantes en nuestro proyecto es mínimo y no se hace necesario implementarlo.

***Objeto Inmutable:*** No lo implementaremos debido a que los objetos dentro de nuestro proyecto pueden llegar a necesitar cambios o modificaciones.

**Monitor:**

**Patrones Creacionales**

***Abstract Factory:*** No será implementado en nuestro proyecto, debido a que se extendería mucho el diseño de nuestro proyecto y además, ya estamos utilizando el Factory Method que nos ayuda a resolver la problemática al momento de crear los diferentes tipos de zonas comunes. No se genera sino solo una fábrica que es la encargada de escoger cual zona común se va a utilizar.

***Builder:*** No será implementado en el proyecto, debido a que ya tenemos definidos los patrones creacionales que vamos a usar y no vemos la necesidad de implementarlo.

No se utiliza por que no vamos a crear un objeto complejo ya que los que creamos no necesitan complementos.

**Prototype:** no necesitamos la duplicidad de la información ya que con la fábrica sabemos que zona se va a crear.

**Patrones Estructurales**

**Adapter:** no se utiliza ya que la interfaz creada cumple con los requisitos para ser utilizada por cada zona común.

**Flyweihgt:** en nuestra aplicación no tiene información idéntica ya que cada reserva y cada espacio solo puede ser utilizado en un estado a la vez y por un usuario a la vez. En el estado intrínseco se tomaría el tipo de zona común y el extrínseco las características de cada reserva.

**Facade:** No se implementó debido a que no tenemos subsistemas complejos.

**Decorator:** no se utiliza porque no queremos añadir nuevas alternativas al momento de la reserva. La información que tenemos es estática y la creación la realiza solo la fábrica.

**Composite:** este no se utiliza ya que no queremos grupos de zonas comunes, queremos la información uno a uno.

**Patrones Comportamentales**

**Chain of responsibility:** no se utiliza ya que el único que tiene responsabilidad en los eventos es reserva no existe ninguno más que pueda ejecutar la reserva.

**Iterator:** no se ve la necesidad de recorrer la estructura. Ya que nuestros datos van a estar almacenados en una BD que nos va a permitir mantener la persistencia de la información.

**Visitor:** aquí no queremos modificar la estructura de los objetos. Ya que cada uno de los objetos lo creamos con una estructura definida.