Patrón Builder

Por Tomás Delgado

Problemática

1

Complejidad de Construcción

Aborda la creación de objetos complejos sin sobrecargar el constructor.

2

Configuración Flexible

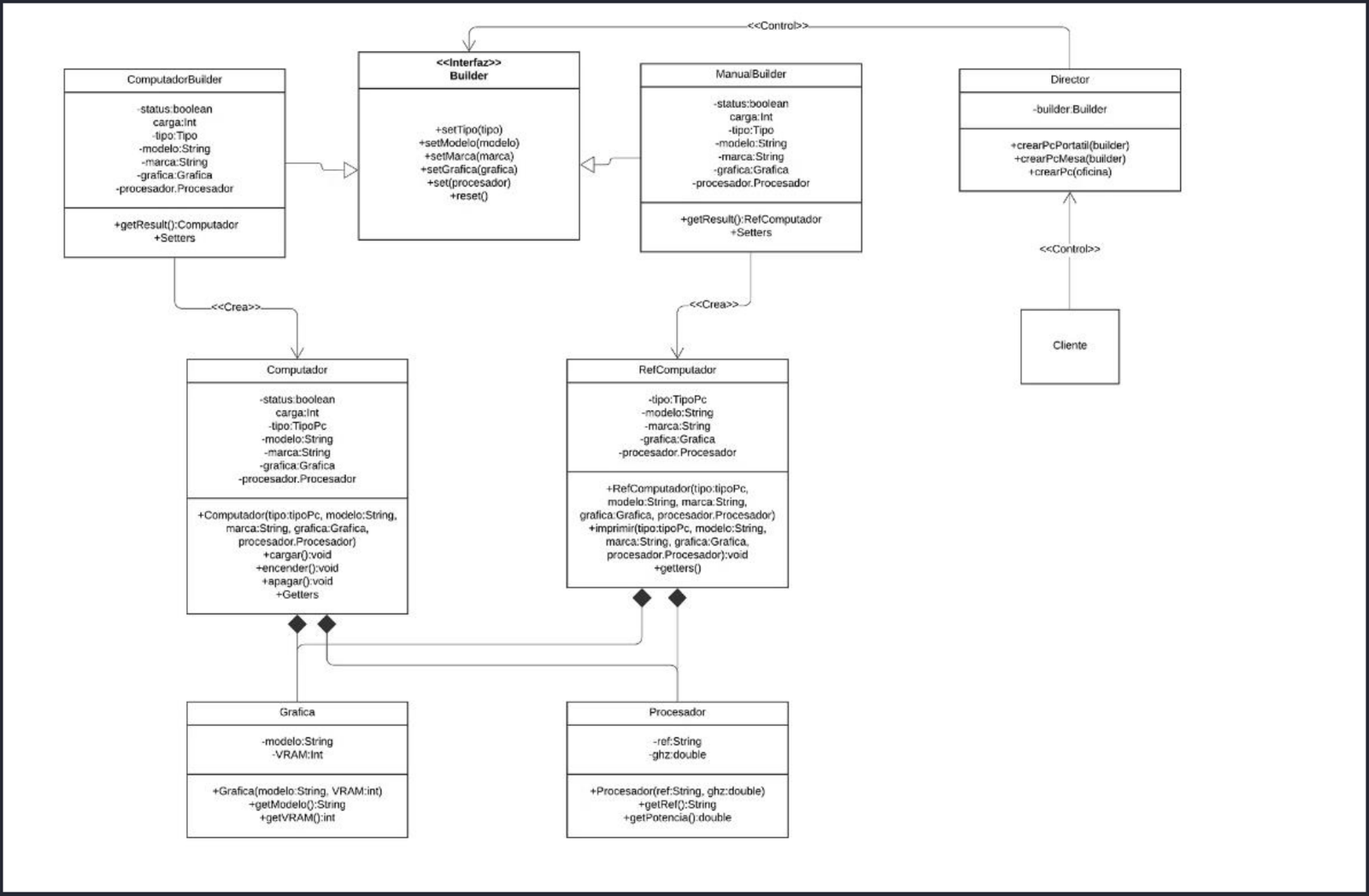
Permite crear diferentes representaciones del mismo conjunto de componentes.

3

Separación de la Construcción y Representación

Evita la construcción directa de objetos complejos en el cliente.

Estructura del patrón Builder



Ventajas y desventajas del patrón

Ventajas

Encapsulamiento de los procesos de creación y construcción de objetos, permitiendo un codigo más limpio y ordenado, así como una configuración flexible según requiera el caso.

Desventajas

Al requerir de más clases para la direccion de procesos de creación hace la sintaxis más larga y complicada aunque aporte mayor maleabilidad y escalabilidad.

Comparación con otros patrones de diseño

Builder vs. Abstract Factory

Builder se centra en la construcción paso a paso, mientras que Abstract Factory se centra en la creación de familias de objetos.

Builder vs. Prototype

Builder se utiliza para construir objetos complejos paso a paso, mientras que Prototype se utiliza para clonar objetos existentes.

Consideraciones

1

implementación en objetos deconstruibles

Recomendado para la creación de objetos complejos con múltiples partes.

2

Evaluar la escalabilidad

Analizar si la separación de la construcción y representación aporta suficientes beneficios.

3

Considerar Alternativas

En ocasiones concretas es mas factible trabajar con patrones de diseño más escalables según se requiera la fabricación de familias de objetos grandes u objetos que compartan atributos sin necesidad de ser modificados.