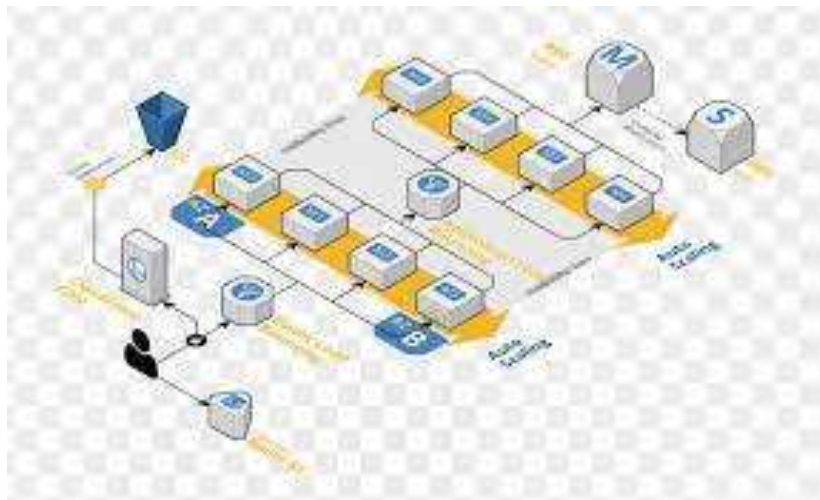


PATRÓN ADAPTER



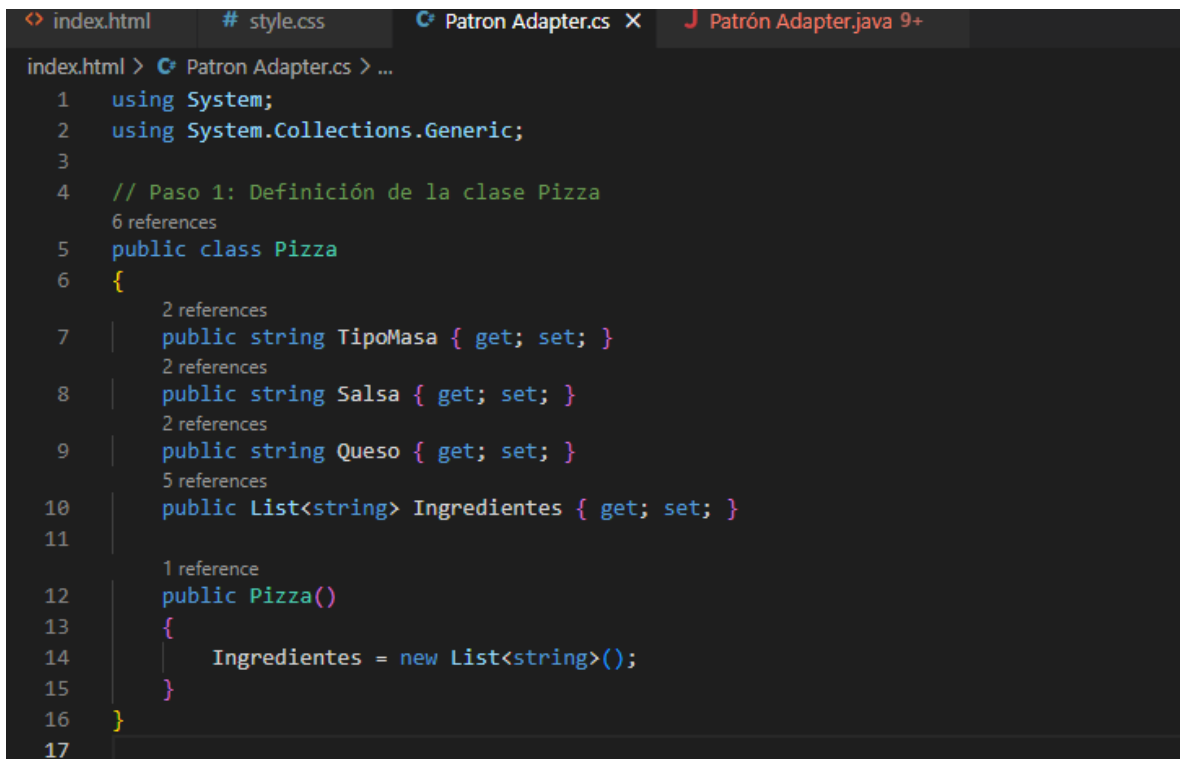
MODELADO Y DISEÑO

VIZCARRA MORENO JESUS DANIEL

GUSTAVO IBARRA

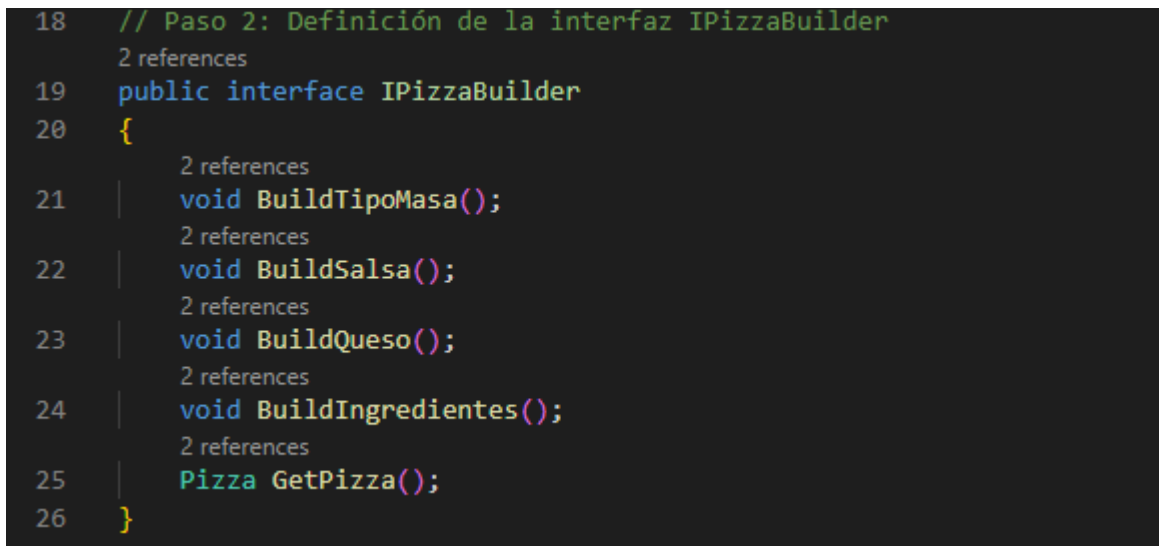
CESUN UNIVERSIDAD

PASO 1



```
index.html > Patron Adapter.cs > ...
1  using System;
2  using System.Collections.Generic;
3
4  // Paso 1: Definición de la clase Pizza
5  public class Pizza
6  {
7      public string TipoMasa { get; set; }
8      public string Salsa { get; set; }
9      public string Queso { get; set; }
10     public List<string> Ingredientes { get; set; }
11
12     public Pizza()
13     {
14         Ingredientes = new List<string>();
15     }
16 }
17
```

PASO 2



```
18 // Paso 2: Definición de la interfaz IPizzaBuilder
19 public interface IPizzaBuilder
20 {
21     void BuildTipoMasa();
22     void BuildSalsa();
23     void BuildQueso();
24     void BuildIngredientes();
25     Pizza GetPizza();
26 }
27
```

PASO 3

```
28 // Paso 3: Implementación del PizzaBuilder
29 2 references
30 public class PizzaBuilder : IPizzaBuilder
31 {
32     8 references
33     private Pizza _pizza;
34
35     1 reference
36     public PizzaBuilder()
37     {
38         _pizza = new Pizza();
39     }
40
41     2 references
42     public void BuildTipoMasa()
43     {
44         _pizza.TipoMasa = "masa delgada";
45     }
46
47     2 references
48     public void BuildSalsa()
49     {
50         _pizza.Salsa = "salsa de tomate";
51     }
52
53     2 references
54     public void BuildQueso()
55     {
56     }
```

```
57     public void BuildSalsa()
58     {
59         _pizza.Salsa = "salsa de tomate";
60     }
61
62     2 references
63     public void BuildQueso()
64     {
65         _pizza.Queso = "queso mozzarella";
66     }
67
68     2 references
69     public void BuildIngredientes()
70     {
71         _pizza.Ingredientes.Add("jamón");
72         _pizza.Ingredientes.Add("champiñones");
73         _pizza.Ingredientes.Add("aceitunas");
74     }
75
76     2 references
77     public Pizza GetPizza()
78     {
79         return _pizza;
80     }
81 }
```

PASO 4 Y 5

```
66 // Paso 4: Uso del PizzaBuilder para construir una pizza
67 0 references
68 class Program
69 {
70     0 references
71     static void Main(string[] args)
72     {
73         // Creamos una instancia de PizzaBuilder
74         IPizzaBuilder pizzaBuilder = new PizzaBuilder();
75
76         // Utilizamos los métodos del builder para construir la pizza
77         pizzaBuilder.BuildTipoMasa();
78         pizzaBuilder.BuildSalsa();
79         pizzaBuilder.BuildQueso();
80         pizzaBuilder.BuildIngredientes();
81
82         // Obtenemos la pizza completa
83         Pizza pizza = pizzaBuilder.GetPizza();
84
85         // Paso 5: Resultado
86         Console.WriteLine("Se ha construido una deliciosa pizza con los siguientes ingredientes y características!");
87         Console.WriteLine($"Tipo de masa: {pizza.TipoMasa}");
88         Console.WriteLine($"Salsa: {pizza.Salsa}");
89         Console.WriteLine($"Queso: {pizza.Queso}");
90         Console.WriteLine("Ingredientes:");
91         foreach (var ingrediente in pizza.Ingredientes)
92         {
93             Console.WriteLine($"- {ingrediente}");
94         }
95     }
96 }
```

Conclusión:

El patrón Builder resulta útil cuando necesitamos crear objetos complejos con múltiples configuraciones posibles. Al separar el proceso de construcción de la representación interna del objeto, nos permite construir y combinar diferentes variaciones del objeto utilizando un proceso estructurado y modular. Además, facilita la adición de nuevos pasos de construcción o variaciones en el futuro, lo que lo hace altamente adaptable y escalable. En resumen, el patrón Builder es una herramienta valiosa para la creación de objetos complejos de manera flexible y mantenible.