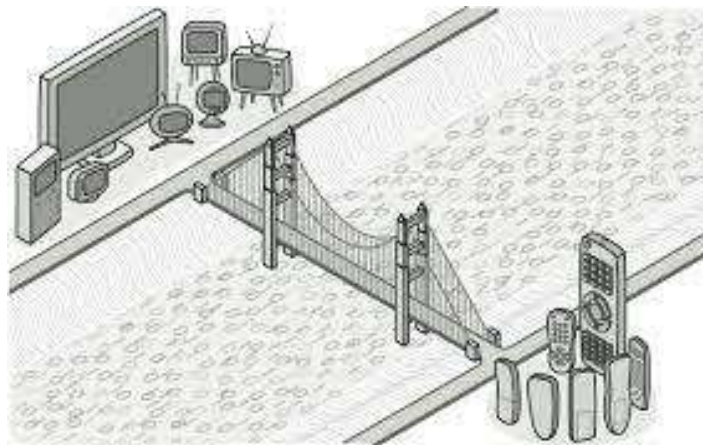


PATRÓN BRIGDE



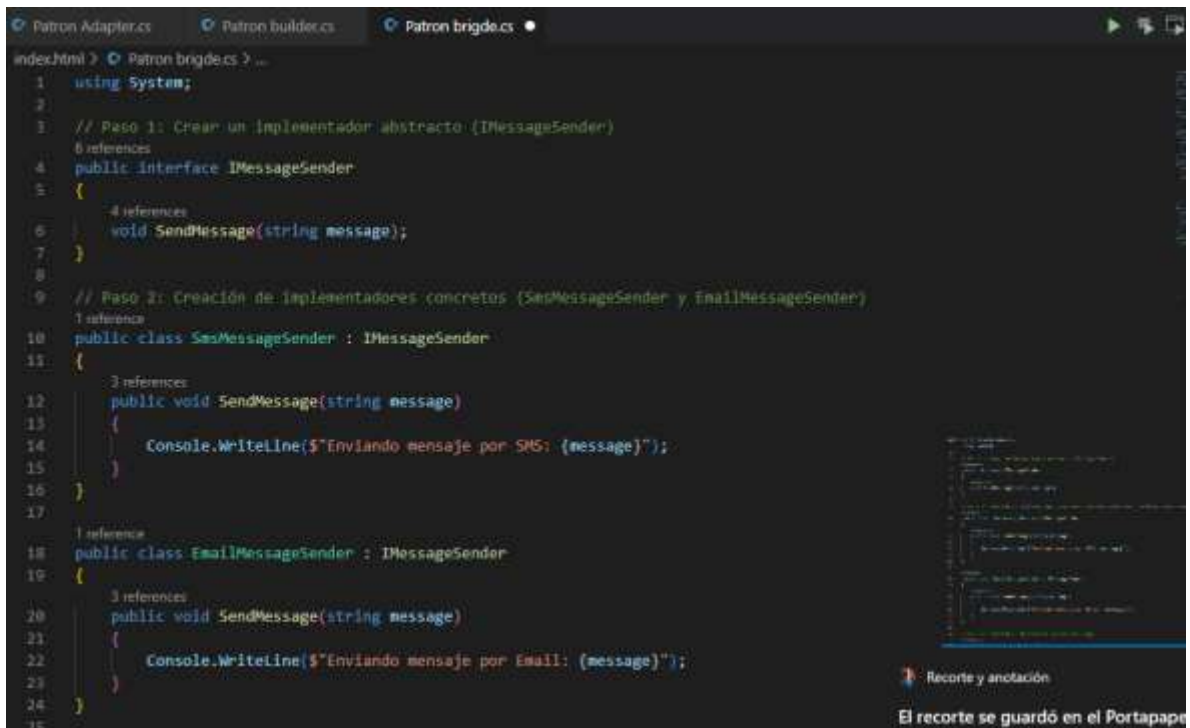
MODELADO Y DISEÑO

VIZCARRA MORENO JESUS DANIEL

GUSTAVO IBARRA

CESUN UNIVERSIDAD


PASO 1 y 2.



```
index.html > Patron brigde.cs > ...
1  using System;
2
3  // Paso 1: Crear un implementador abstracto (IMessageSender)
4  public interface IMessageSender
5  {
6      void SendMessage(string message);
7  }
8
9  // Paso 2: Creación de implementadores concretos (SmsMessageSender y EmailMessageSender)
10 public class SmsMessageSender : IMessageSender
11 {
12     public void SendMessage(string message)
13     {
14         Console.WriteLine($"Enviando mensaje por SMS: {message}");
15     }
16 }
17
18 public class EmailMessageSender : IMessageSender
19 {
20     public void SendMessage(string message)
21     {
22         Console.WriteLine($"Enviando mensaje por Email: {message}");
23     }
24 }
```

Recorte y anotación
El recorte se guardó en el Portapape

PASO 3



```
25
26 // Paso 3: Creación de abstracción (AbstractMessage)
27 public abstract class AbstractMessage
28 {
29     protected IMessageSender messageSender;
30
31     public AbstractMessage(IMessageSender sender)
32     {
33         messageSender = sender;
34     }
35
36     public abstract void SendMessage(string message);
37 }
```

PASO 4

```
// Paso 4: Creación de abstracción concreta (MensajeCorto y MensajeLargo)
2 references
public class MensajeCorto : AbstractMessage
{
    1 reference
    public MensajeCorto(IMessageSender sender) : base(sender)
    {
    }

    3 references
    public override void SendMessage(string message)
    {
        if (message.Length >= 10 && message.Length <= 20)
        {
            messageSender.SendMessage(message);
        }
        else
        {
            Console.WriteLine("El mensaje no puede ser enviado. Debe tener entre 10 y 20 caracteres.");
        }
    }
}

2 references
```

Paso 5

```
// Paso 5: Cliente
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        // Cliente elige el tipo de mensaje
        Console.WriteLine("¿Qué tipo de mensaje desea enviar? (corto/largo)");
        string tipoMensaje = Console.ReadLine().ToLower();

        AbstractMessage mensaje;

        // Crea objeto AbstractMessage según el tipo elegido
        switch (tipoMensaje)
        {
            case "corto":
                mensaje = new MensajeCorto(new SmsMessageSender());
                mensaje.SendMessage("Mensaje corto");
                break;
            case "largo":
                mensaje = new MensajeLargo(new EmailMessageSender());
                mensaje.SendMessage("Mensaje largo");
                break;
            default:
                Console.WriteLine("Tipo de mensaje no válido");
                return;
        }
    }
}
```

Conclusión:

Este código implementa el patrón Bridge en C#, que separa la abstracción de su implementación. Esto permite que los mensajes cortos y largos se envíen a través de diferentes medios (SMS o correo electrónico) sin que los clientes tengan que preocuparse por los detalles de implementación específicos.