INSTITUTO TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto-Diciembre 2025

CARRERA:
Ingeniería Informática

MATERIA:
Patrones de diseño

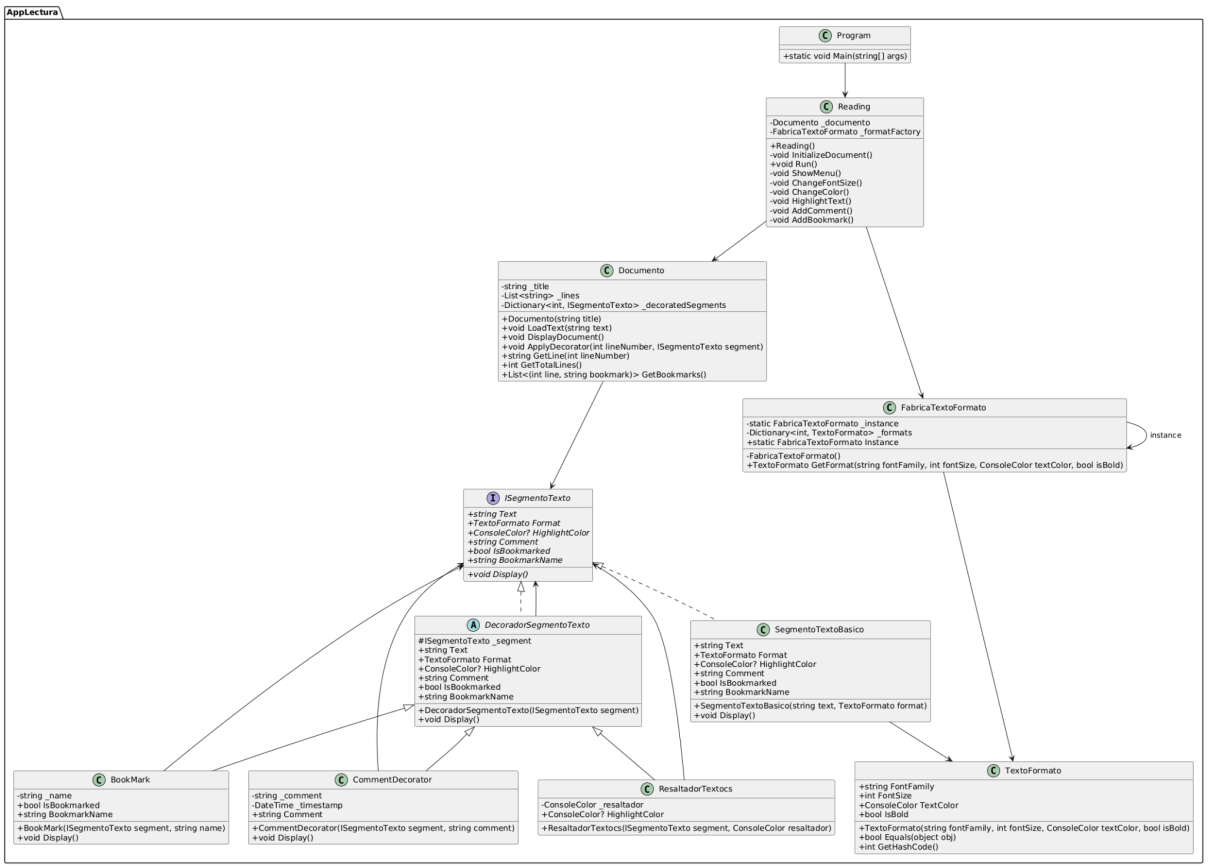
NOMBRE DEL TRABAJO:
Reporte de práctica (examen)

UNIDAD A EVALUAR:
Unidad 3

ALUMNA:
Reyes Morales Andrea - 21212358

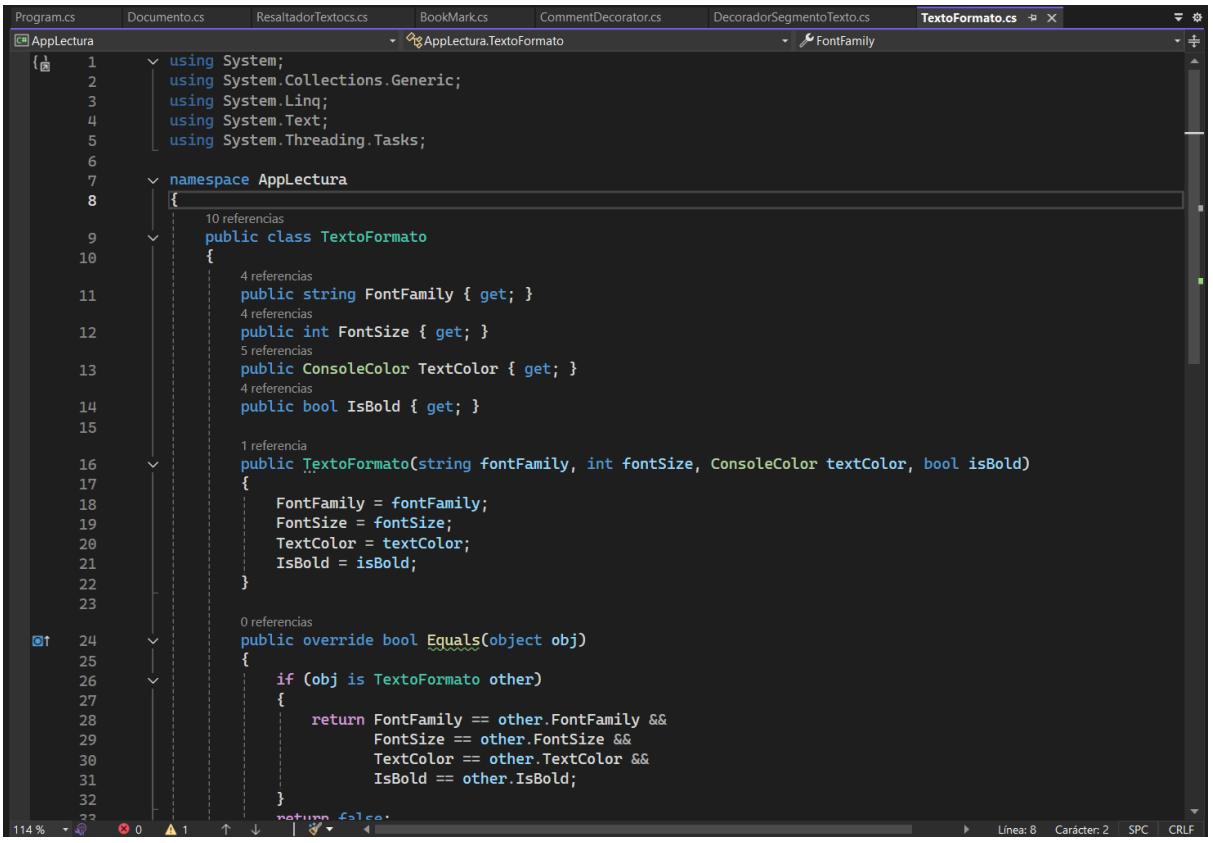
MAESTRA:
Maribel Guerrero Luis

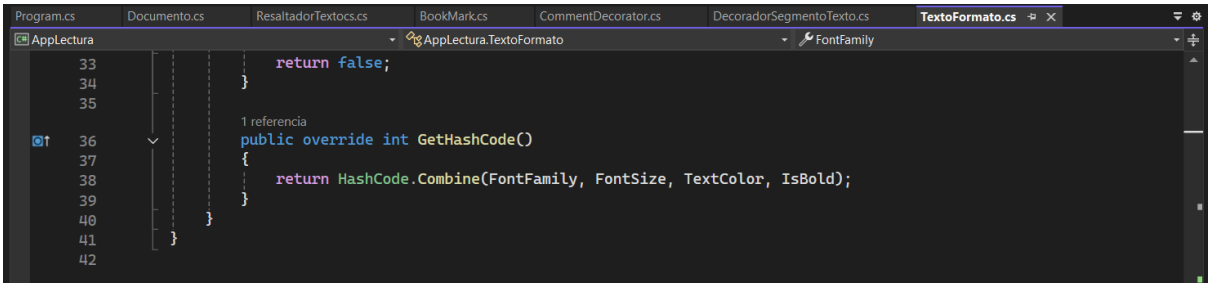
1. Diagrama UML



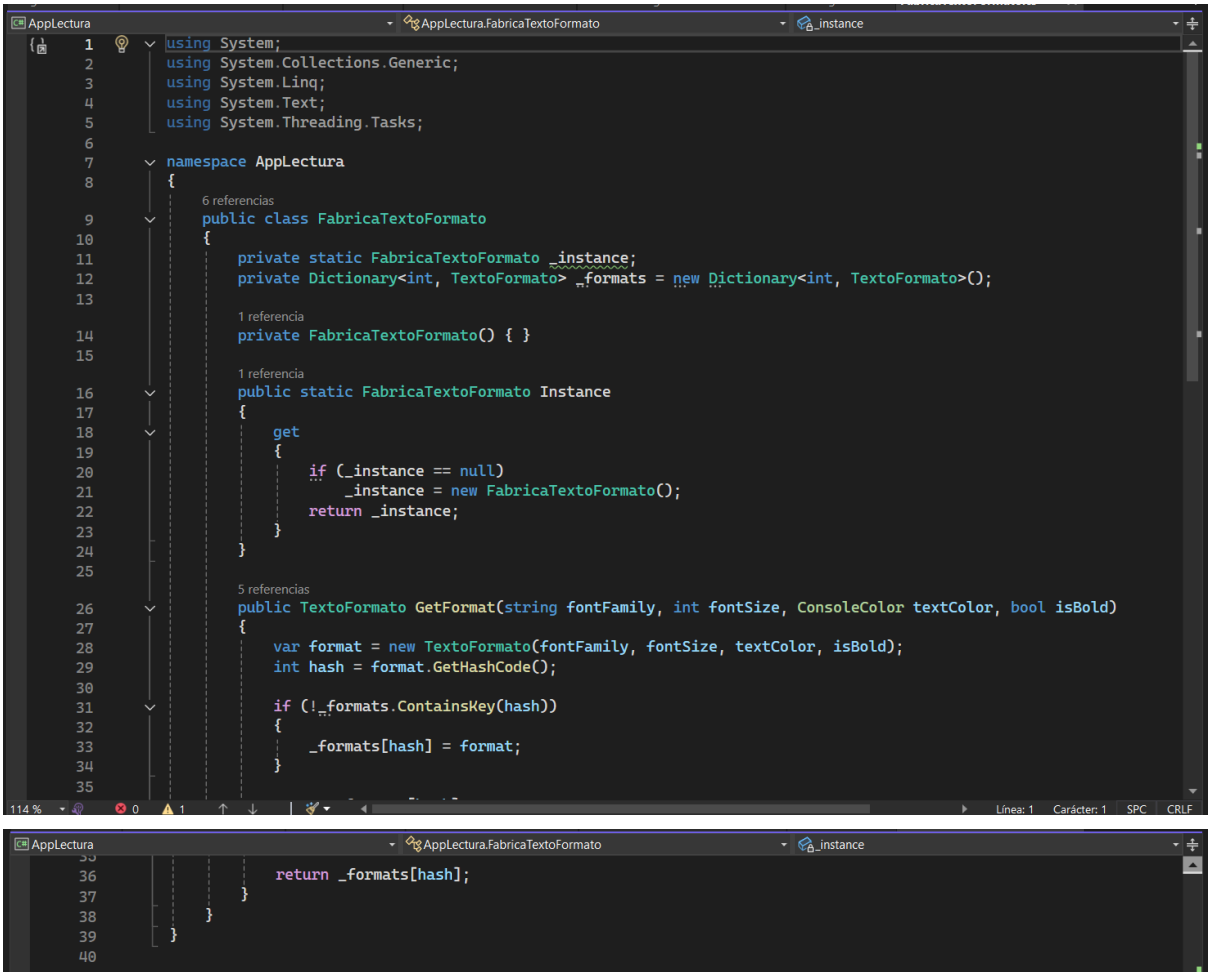
2. Captura del código

TextoFormato.cs





FabricaTextoFormato.cs

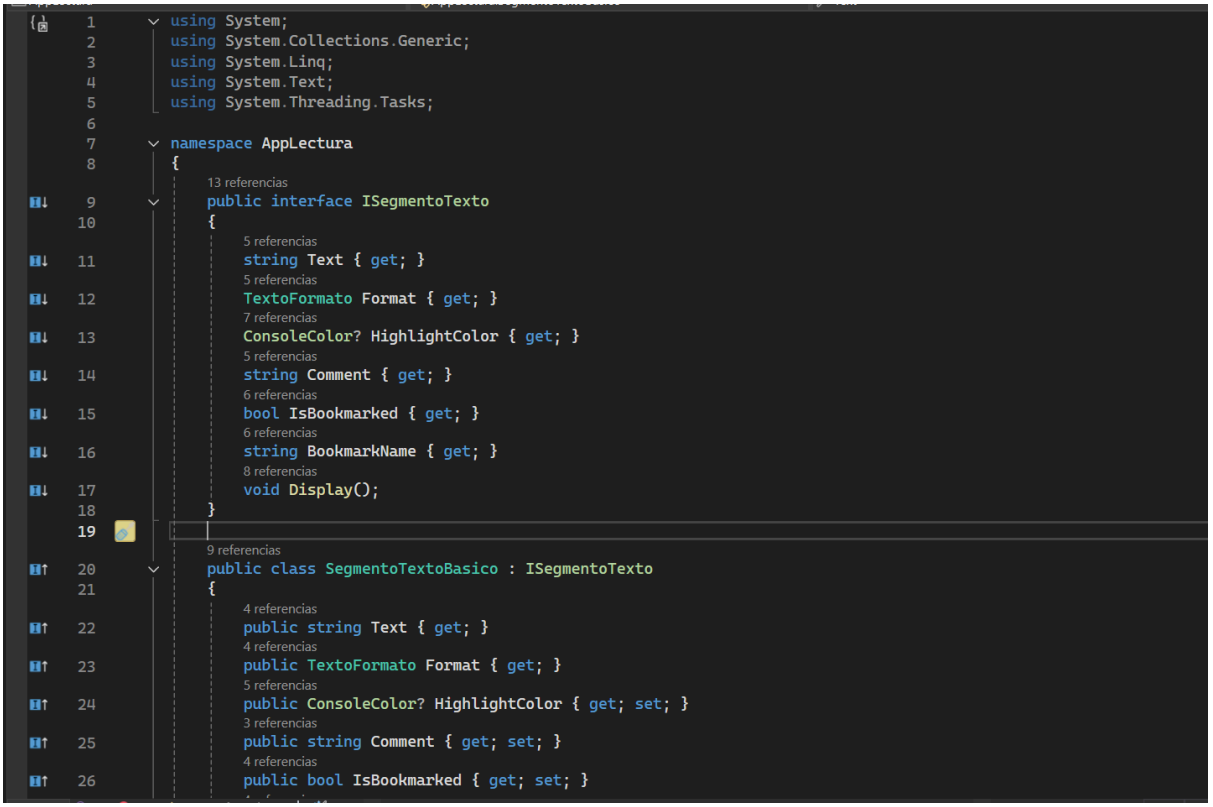


```
AppLectura | AppLectura.FabricaTextoFormato | _instance

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AppLectura
{
    6 referencias
    public class FabricaTextoFormato
    {
        private static FabricaTextoFormato _instance;
        private Dictionary<int, TextoFormato> _formats = new Dictionary<int, TextoFormato>();

        1 referencia
        private FabricaTextoFormato() { }

        1 referencia
        public static FabricaTextoFormato Instance
        {
            get
            {
                if (_instance == null)
                    _instance = new FabricaTextoFormato();
                return _instance;
            }
        }

        5 referencias
        public TextoFormato GetFormat(string fontFamily, int fontSize, ConsoleColor textColor, bool isBold)
        {
            var format = new TextoFormato(fontFamily, fontSize, textColor, isBold);
            int hash = format.GetHashCode();

            if (!_formats.ContainsKey(hash))
            {
                _formats[hash] = format;
            }
```

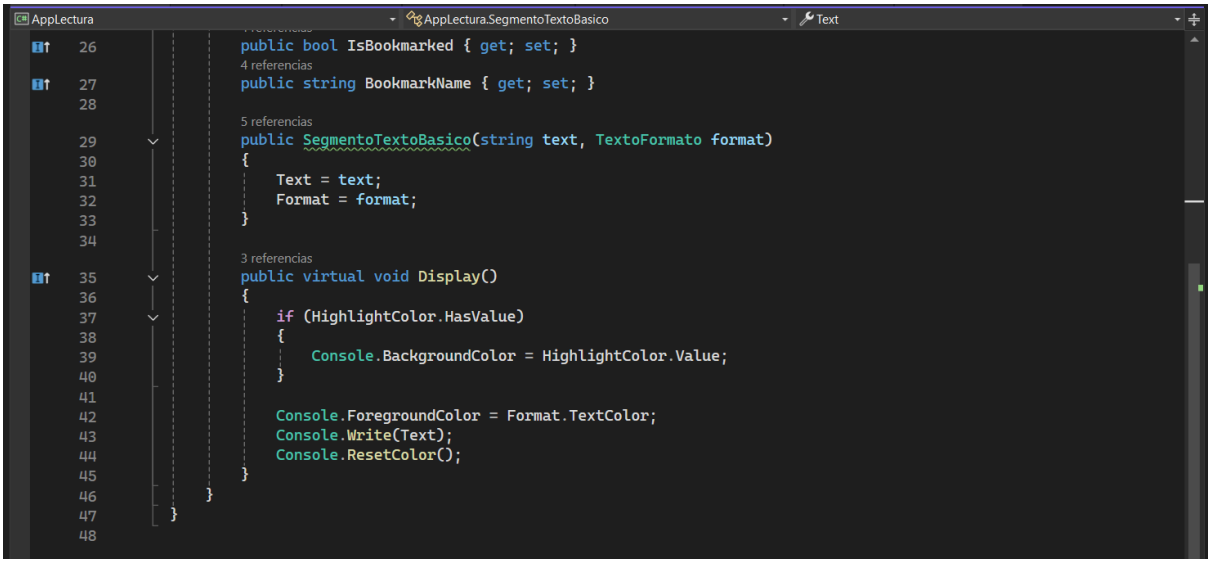
114 % | 0 | 1 | Línea: 1 | Carácter: 1 | SPC | CRLF


```
AppLectura | AppLectura.FabricaTextoFormato | _instance

            return _formats[hash];
        }
    }
}
```

ISegmentoTexto.cs



```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AppLectura
{
    13 referencias
    public interface ISegmentoTexto
    {
        5 referencias
        string Text { get; }
        5 referencias
        TextoFormato Format { get; }
        7 referencias
        ConsoleColor? HighlightColor { get; }
        5 referencias
        string Comment { get; }
        6 referencias
        bool IsBookmarked { get; }
        6 referencias
        string BookmarkName { get; }
        8 referencias
        void Display();
    }

    9 referencias
    public class SegmentoTextoBasico : ISegmentoTexto
    {
        4 referencias
        public string Text { get; }
        4 referencias
        public TextoFormato Format { get; }
        5 referencias
        public ConsoleColor? HighlightColor { get; set; }
        3 referencias
        public string Comment { get; set; }
        4 referencias
        public bool IsBookmarked { get; set; }
```



```cs
AppLectura | AppLectura.SegmentoTextoBasico | Text

        public bool IsBookmarked { get; set; }
        4 referencias
        public string BookmarkName { get; set; }

        5 referencias
        public SegmentoTextoBasico(string text, TextoFormato format)
        {
            Text = text;
            Format = format;
        }

        3 referencias
        public virtual void Display()
        {
            if (HighlightColor.HasValue)
            {
                Console.BackgroundColor = HighlightColor.Value;
            }

            Console.ForegroundColor = Format.TextColor;
            Console.Write(Text);
            Console.ResetColor();
        }
    }
}
```

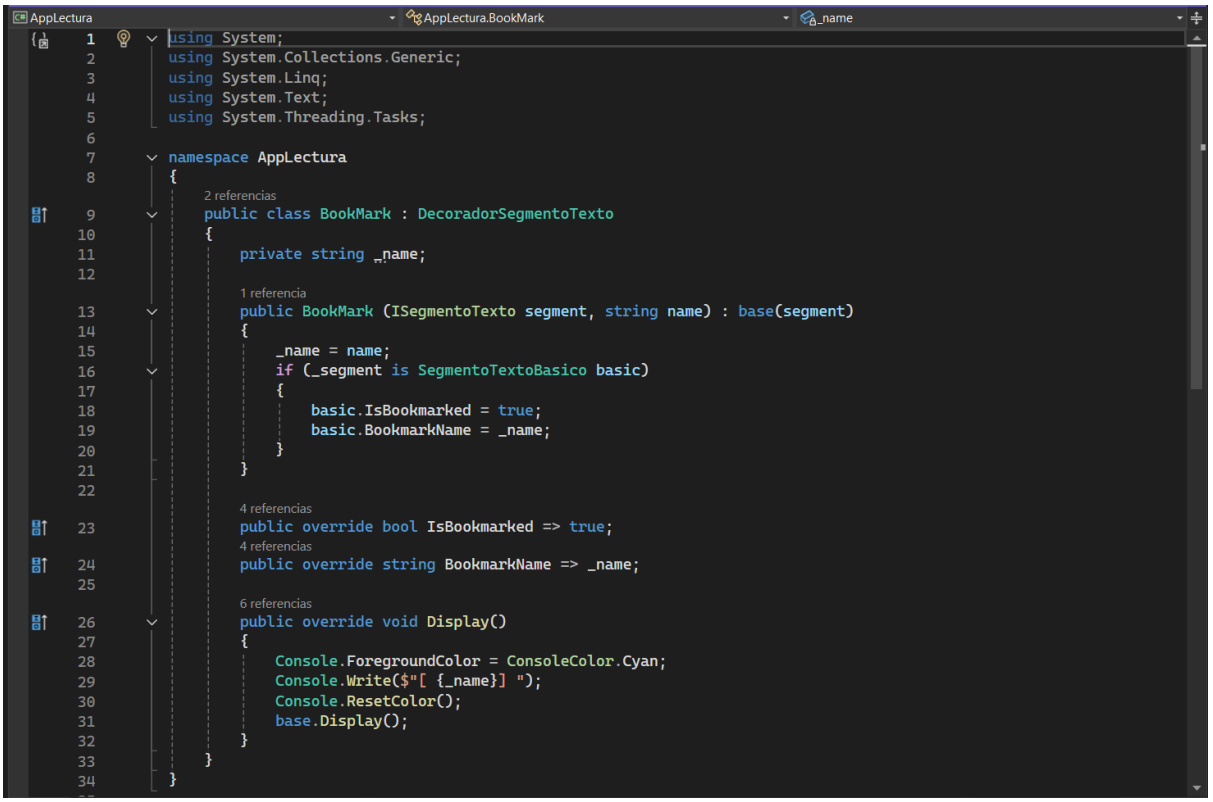
DecoradorSegmentoTexto.cs

```
AppLectura | AppLectura.DecoradorSegmentoTexto | BookmarkName

using System.Text;
using System.Threading.Tasks;

namespace AppLectura
{
    7 referencias
    public abstract class DecoradorSegmentoTexto : ISegmentoTexto
    {
        protected ISegmentoTexto _segment;

        3 referencias
        public DecoradorSegmentoTexto(ISegmentoTexto segment)
        {
            _segment = segment;
        }

        2 referencias
        public virtual string Text => _segment.Text;
        2 referencias
        public virtual TextoFormato Format => _segment.Format;
        3 referencias
        public virtual ConsoleColor? HighlightColor => _segment.HighlightColor;
        3 referencias
        public virtual string Comment => _segment.Comment;
        4 referencias
        public virtual bool IsBookmarked => _segment.IsBookmarked;
        4 referencias
        public virtual string BookmarkName => _segment.BookmarkName;

        7 referencias
        public virtual void Display()
        {
            _segment.Display();
        }

    }

}
```

CommentDecorator.cs

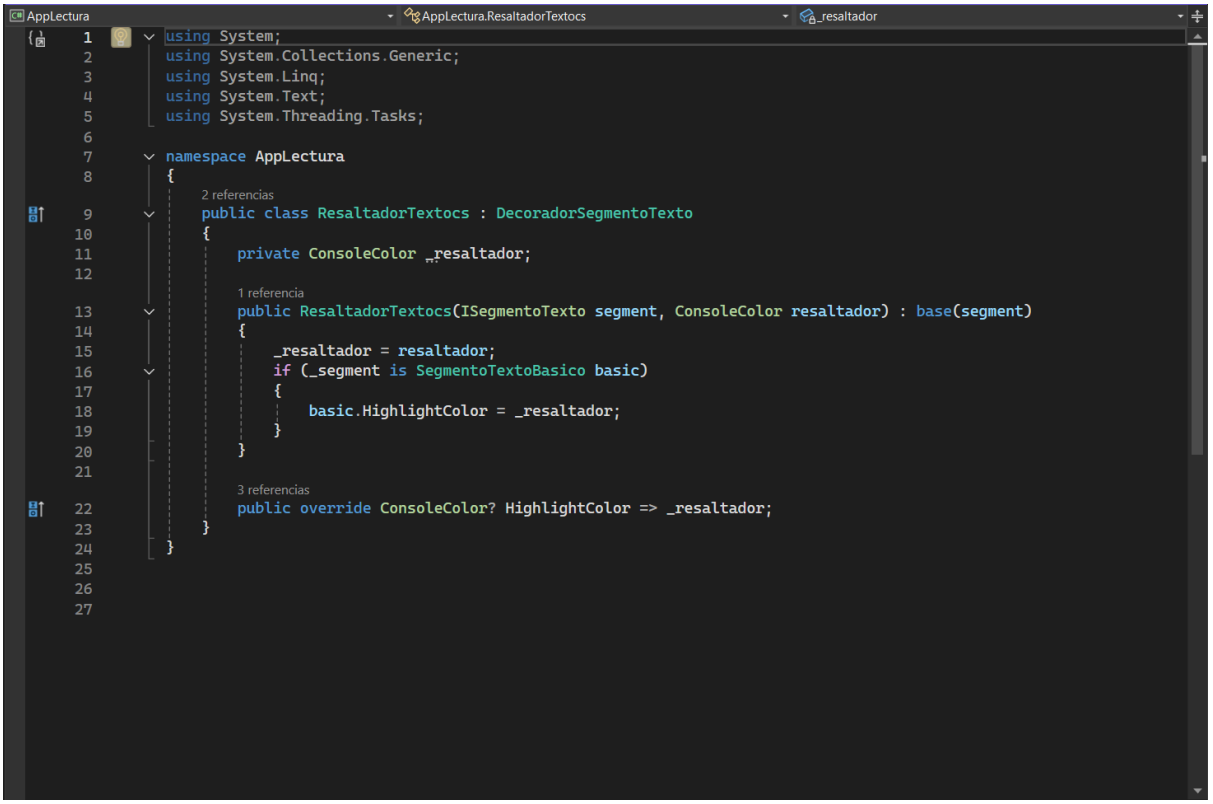
```
AppLectura | AppLectura.CommentDecorator | CommentDecorator(ISegmentoTexto segment, string comment)

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AppLectura
{
    2 referencias
    public class CommentDecorator : DecoradorSegmentoTexto
    {
        private string _comment;
        private DateTime _timestamp;

        1 referencia
        public CommentDecorator(ISegmentoTexto segment, string comment) : base(segment)
        {
            _comment = comment;
            _timestamp = DateTime.Now;
            if (_segment is SegmentoTextoBasico basic)
            {
                basic.Comment = _comment;
            }
        }

        3 referencias
        public override string Comment => _comment;

        6 referencias
        public override void Display()
        {
            base.Display();
            Console.ForegroundColor = ConsoleColor.DarkYellow;
            Console.Write($" [{_comment}]");
            Console.ResetColor();
        }

    }
}
```

BookMark.cs



```
AppLectura                AppLectura.BookMark                _name
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AppLectura
8  {
       2 referencias
9      public class BookMark : DecoradorSegmentoTexto
10     {
11         private string _name;
12
           1 referencia
13         public BookMark (ISegmentoTexto segment, string name) : base(segment)
14         {
15             _name = name;
16             if (_segment is SegmentoTextoBasico basic)
17             {
18                 basic.IsBookmarked = true;
19                 basic.BookmarkName = _name;
20             }
21         }
22
           4 referencias
23         public override bool IsBookmarked => true;
           4 referencias
24         public override string BookmarkName => _name;
25
           6 referencias
26         public override void Display()
27         {
28             Console.ForegroundColor = ConsoleColor.Cyan;
29             Console.Write($"[ {_name}] ");
30             Console.ResetColor();
31             base.Display();
32         }
33     }
34 }
```

ResaltadorTexto.cs



```
AppLectura                AppLectura.ResaltadorTextocs                _resaltador
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AppLectura
8  {
       2 referencias
9      public class ResaltadorTextocs : DecoradorSegmentoTexto
10     {
11         private ConsoleColor _resaltador;
12
           1 referencia
13         public ResaltadorTextocs(ISegmentoTexto segment, ConsoleColor resaltador) : base(segment)
14         {
15             _resaltador = resaltador;
16             if (_segment is SegmentoTextoBasico basic)
17             {
18                 basic.HighlightColor = _resaltador;
19             }
20         }
21
           3 referencias
22         public override ConsoleColor? HighlightColor => _resaltador;
23     }
24 }
25
26
27
```

Documento.cs

```
AppLectura                    AppLectura.Documento                    DisplayDocument()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AppLectura
{
    3 referencias
    public class Documento
    {
        private string _title;
        private List<string> _lines;
        private Dictionary<int, ISegmentoTexto> _decoratedSegments = new Dictionary<int, ISegmentoTexto>();

        1 referencia
        public Documento(string title)
        {
            _title = title;
            _lines = new List<string>();
        }

        1 referencia
        public void LoadText(string text)
        {
            _lines = text.Split('\n').ToList();
        }

        1 referencia
        public void DisplayDocument()
        {
            Console.Clear();
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine($"\n{'=',70}");
            Console.WriteLine($" {_title}");
            Console.WriteLine($"{'=',70}\n");
            Console.ResetColor();

            for (int i = 0; i < _lines.Count; i++)
```

```
AppLectura                    AppLectura.Documento                    DisplayDocument()
            {
                Console.Write($"{i + 1,3}. ");

                if (_decoratedSegments.ContainsKey(i))
                {
                    _decoratedSegments[i].Display();
                }
                else
                {
                    Console.Write(_lines[i]);
                }
                Console.WriteLine();
            }

            Console.WriteLine($"\n{'=',70}");
        }

        5 referencias
        public void ApplyDecorator(int lineNumber, ISegmentoTexto segment)
        {
            if (lineNumber >= 0 && lineNumber < _lines.Count)
            {
                _decoratedSegments[lineNumber] = segment;
            }
        }

        5 referencias
        public string GetLine(int lineNumber)
        {
            if (lineNumber >= 0 && lineNumber < _lines.Count)
                return _lines[lineNumber];
            return null;
        }

        5 referencias
        public int GetTotalLines() => _lines.Count;

        0 referencias
        public List<(int line, string bookmark)> GetBookmarks()
```

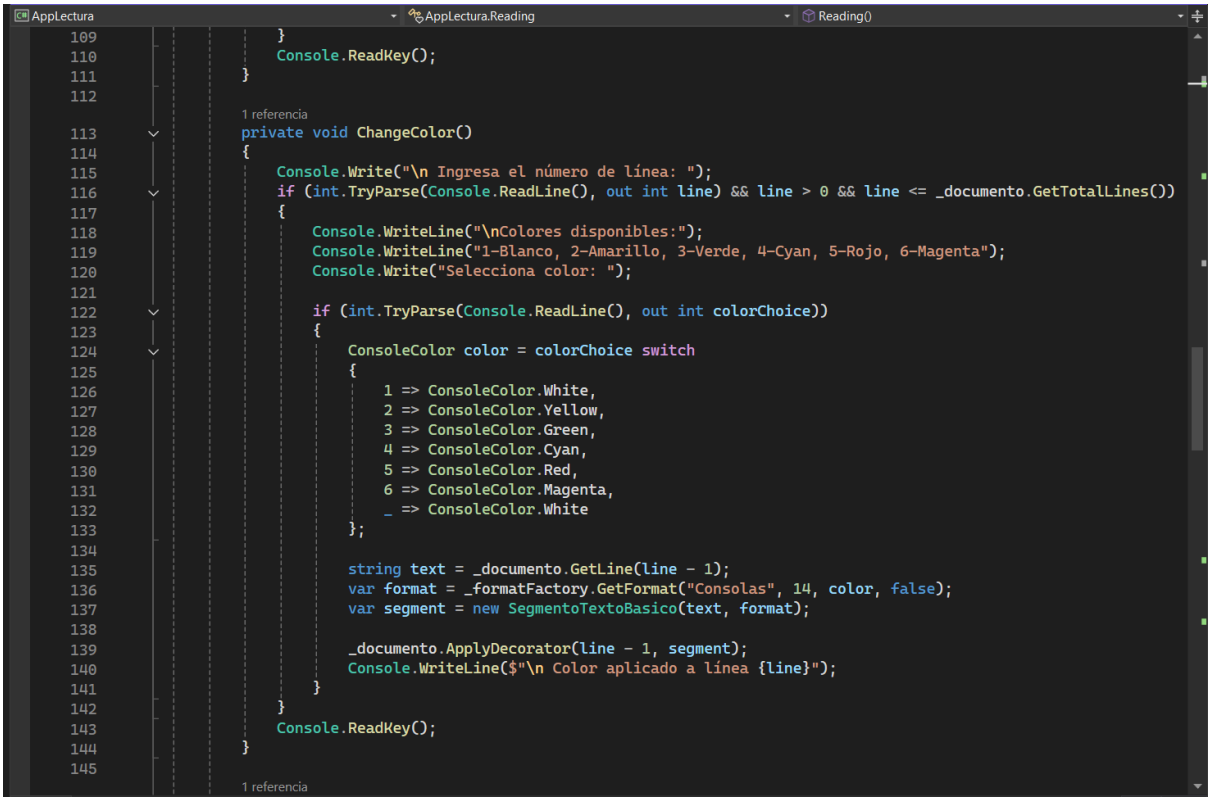
```
71          {
72              var bookmarks = new List<(int, string)>();
73              foreach (var kvp in _decoratedSegments)
74              {
75                  if (kvp.Value.IsBookmarked)
76                  {
77                      bookmarks.Add((kvp.Key, kvp.Value.BookmarkName));
78                  }
79              }
80              return bookmarks;
81          }
82      }
83  }
84
85
```

Reading.cs

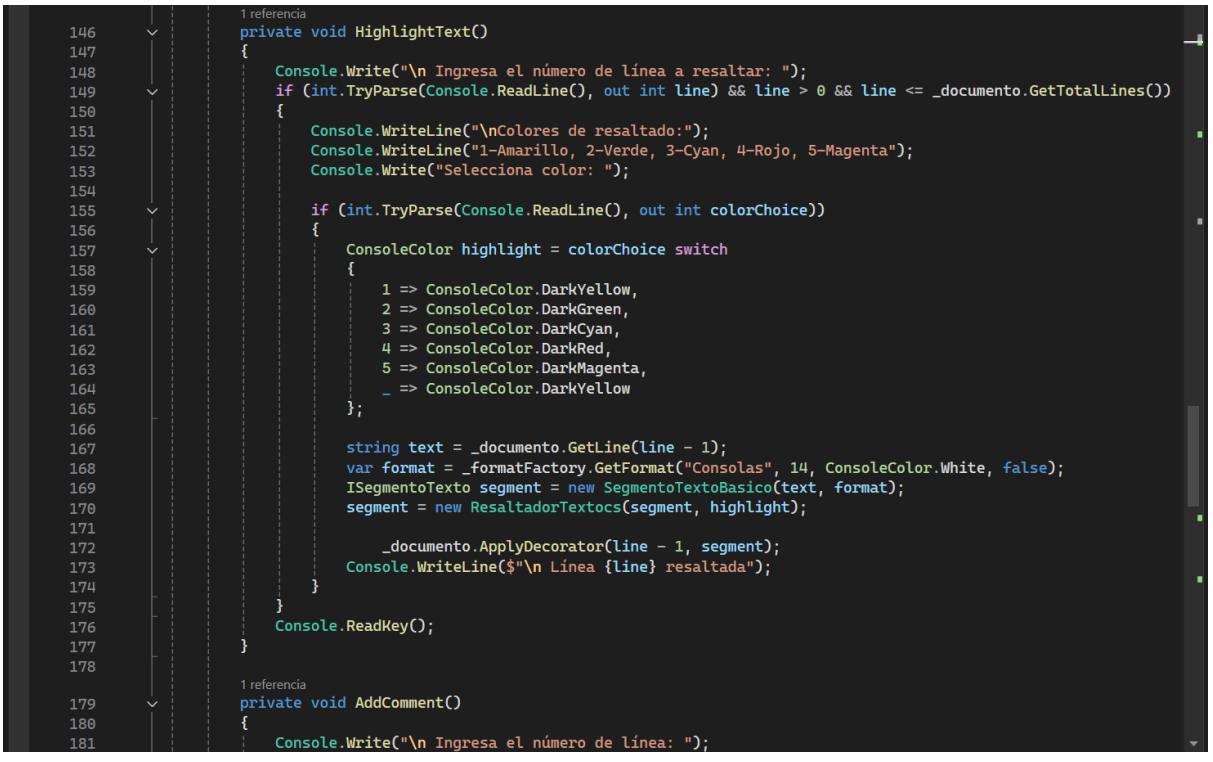
```
AppLectura                AppLectura.Reading                Reading()
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Reflection.Metadata;
5   using System.Text;
6   using System.Threading.Tasks;
7
8   namespace AppLectura
9   {
        3 referencias
10      class Reading
11      {
12          private Documento _documento;
13          private FabricaTextoFormato _formatFactory;
14
            1 referencia
15          public Reading()
16          {
17              _formatFactory = FabricaTextoFormato.Instance;
18              InitializeDocument();
19          }
20
            1 referencia
21          private void InitializeDocument()
22          {
23              _documento = new Documento("El Principito - Antoine de Saint-Exupéry");
24
25              string sampleText = @"Cuando yo tenía seis años vi una vez una lámina magnífica
26                                    en un libro sobre el Bosque Virgen que se llamaba 'Historias Vividas'.
27                                    Representaba una serpiente boa que se tragaba a una fiera.
28                                    He aquí la copia del dibujo.
29                                    En el libro decía: 'Las serpientes boas tragan su presa entera'.
30                                    Reflexioné mucho entonces sobre las aventuras de la jungla.
31                                    Lo esencial es invisible a los ojos.
32                                    Esta es una de las frases más importantes del libro.
33                                    El zorro le enseña al principito sobre la amistad y el amor.
34                                    Viví solo, sin nadie con quien hablar verdaderamente.";
35
36              _documento.LoadText(sampleText);
```

```
37          }
38
            1 referencia
39          public void Run()
40          {
41              bool running = true;
42
43              while (running)
44              {
45                  _documento.DisplayDocument();
46                  ShowMenu();
47
48                  string option = Console.ReadLine();
49
50                  switch (option)
51                  {
52                      case "1":
53                          ChangeFontSize();
54                          break;
55                      case "2":
56                          ChangeColor();
57                          break;
58                      case "3":
59                          HighlightText();
60                          break;
61                      case "4":
62                          AddComment();
63                          break;
64                      case "5":
65                          AddBookmark();
66                          break;
67                      case "6":
68                          running = false;
69                          Console.WriteLine("\n¡Gracias por usar la aplicación! \n");
70                          break;
71                      default:
```

```
AppLectura  AppLectura.Reading  Reading()
72                          Console.WriteLine("\n Opción inválida");
73                          Console.ReadKey();
74                          break;
75                  }
76              }
77          }
78
            1 referencia
79          private void ShowMenu()
80          {
81              Console.ForegroundColor = ConsoleColor.Cyan;
82              Console.WriteLine("\n MENÚ DE OPCIONES:");
83              Console.ResetColor();
84              Console.WriteLine("1. Cambiar tamaño de fuente");
85              Console.WriteLine("2. Cambiar color de texto");
86              Console.WriteLine("3. Resaltar línea");
87              Console.WriteLine("4. Añadir comentario");
88              Console.WriteLine("5. Crear marcador");
89              Console.WriteLine("6. Ver marcadores");
90              Console.WriteLine("7. Salir");
91              Console.Write("\n> Selecciona una opción: ");
92          }
93
            1 referencia
94          private void ChangeFontSize()
95          {
96              Console.Write("\n Ingresa el número de línea: ");
97              if (int.TryParse(Console.ReadLine(), out int line) && line > 0 && line <= _documento.GetTotalLines())
98              {
99                  Console.Write("Tamaño de fuente (10/12/14/16/18/20): ");
100                 if (int.TryParse(Console.ReadLine(), out int size))
101                 {
102                     string text = _documento.GetLine(line - 1);
103                     var format = _formatFactory.GetFormat("Consolas", size, ConsoleColor.White, false);
104                     var segment = new SegmentoTextoBasico(text, format);
105
106                     _documento.ApplyDecorator(line - 1, segment);
107                     Console.WriteLine($"\n Tamaño {size} aplicado a línea {line}");
108                 }
```




```
        }
        Console.ReadKey();
    }

    1 referencia
    private void ChangeColor()
    {
        Console.Write("\n Ingresa el número de línea: ");
        if (int.TryParse(Console.ReadLine(), out int line) && line > 0 && line <= _documento.GetTotalLines())
        {
            Console.WriteLine("\nColores disponibles:");
            Console.WriteLine("1-Blanco, 2-Amarillo, 3-Verde, 4-Cyan, 5-Rojo, 6-Magenta");
            Console.Write("Selecciona color: ");

            if (int.TryParse(Console.ReadLine(), out int colorChoice))
            {
                ConsoleColor color = colorChoice switch
                {
                    1 => ConsoleColor.White,
                    2 => ConsoleColor.Yellow,
                    3 => ConsoleColor.Green,
                    4 => ConsoleColor.Cyan,
                    5 => ConsoleColor.Red,
                    6 => ConsoleColor.Magenta,
                    _ => ConsoleColor.White
                };

                string text = _documento.GetLine(line - 1);
                var format = _formatFactory.GetFormat("Consolas", 14, color, false);
                var segment = new SegmentoTextoBasico(text, format);

                _documento.ApplyDecorator(line - 1, segment);
                Console.WriteLine($"\n Color aplicado a línea {line}");
            }
        }
        Console.ReadKey();
    }

    1 referencia
```



```
    1 referencia
    private void HighlightText()
    {
        Console.Write("\n Ingresa el número de línea a resaltar: ");
        if (int.TryParse(Console.ReadLine(), out int line) && line > 0 && line <= _documento.GetTotalLines())
        {
            Console.WriteLine("\nColores de resaltado:");
            Console.WriteLine("1-Amarillo, 2-Verde, 3-Cyan, 4-Rojo, 5-Magenta");
            Console.Write("Selecciona color: ");

            if (int.TryParse(Console.ReadLine(), out int colorChoice))
            {
                ConsoleColor highlight = colorChoice switch
                {
                    1 => ConsoleColor.DarkYellow,
                    2 => ConsoleColor.DarkGreen,
                    3 => ConsoleColor.DarkCyan,
                    4 => ConsoleColor.DarkRed,
                    5 => ConsoleColor.DarkMagenta,
                    _ => ConsoleColor.DarkYellow
                };

                string text = _documento.GetLine(line - 1);
                var format = _formatFactory.GetFormat("Consolas", 14, ConsoleColor.White, false);
                ISegmentoTexto segment = new SegmentoTextoBasico(text, format);
                segment = new ResaltadorTextocs(segment, highlight);

                    _documento.ApplyDecorator(line - 1, segment);
                Console.WriteLine($"\n Línea {line} resaltada");
            }
        }
        Console.ReadKey();
    }

    1 referencia
    private void AddComment()
    {
        Console.Write("\n Ingresa el número de línea: ");
```

```
AppLectura  AppLectura.Reading  Reading()
182             if (int.TryParse(Console.ReadLine(), out int line) && line > 0 && line <=  _documento.GetTotalLines())
183             {
184                 Console.Write("Escribe tu comentario: ");
185                 string comment = Console.ReadLine();
186
187                 if (!string.IsNullOrWhiteSpace(comment))
188                 {
189                     string text = _documento.GetLine(line - 1);
190                     var format = _formatFactory.GetFormat("Consolas", 14, ConsoleColor.White, false);
191                     ISegmentoTexto segment = new SegmentoTextoBasico(text, format);
192                     segment = new CommentDecorator(segment, comment);
193
194                     _documento.ApplyDecorator(line - 1, segment);
195                     Console.WriteLine($"\n Comentario añadido a línea {line}");
196                 }
197             }
198             Console.ReadKey();
199         }
200
                1 referencia
201         private void AddBookmark()
202         {
203             Console.Write("\n Ingresa el número de línea: ");
204             if (int.TryParse(Console.ReadLine(), out int line) && line > 0 && line <= _documento.GetTotalLines())
205             {
206                 Console.Write("Nombre del marcador: ");
207                 string name = Console.ReadLine();
208
209                 if (!string.IsNullOrWhiteSpace(name))
210                 {
211                     string text = _documento.GetLine(line - 1);
212                     var format = _formatFactory.GetFormat("Consolas", 14, ConsoleColor.White, false);
213                     ISegmentoTexto segment = new SegmentoTextoBasico(text, format);
214                     segment = new BookMark(segment, name);
215
216                     _documento.ApplyDecorator(line - 1, segment);
217                     Console.WriteLine($"\n Marcador '{name}' creado en línea {line}");
218                 }
219             }
```

```
AppLectura  AppLectura.Reading  Reading()
220             Console.ReadKey();
221         }
222     }
223 }
224
```

---

program.cs

```
AppLectura  AppLectura.Program  Main(string[] args)
1   using System;
2
3   namespace AppLectura
4   {
        0 referencias
5       class Program
6       {
            0 referencias
7           static void Main(string[] args)
8           {
9               Console.OutputEncoding = System.Text.Encoding.UTF8;
10              Console.Title = " Aplicación de Lectura con Marcadores Inteligentes";
11
12              Reading app = new Reading();
13              app.Run();
14          }
15      }
16  }
17
```

3. Conclusión

El desarrollo de esta aplicación de lectura me permitió implementar varios patrones de diseño como el Decorador, Peso ligero y Singleton, aunque enfrenté varios problemas durante el proceso. Uno de los mayores obstáculos fue comprender cómo aplicar correctamente los

decoradores a las líneas de texto .Pero problema más crítico que permanece sin resolver es la imposibilidad de aplicar múltiples decoradores en una misma línea, como combinar un resaltado con un comentario o marcador, debido a que la estructura actual sobrescribe el decorador anterior en lugar de componerlos.

Esta limitación afecta la flexibilidad del sistema y sería un área clave para mejorar en futuras versiones, posiblemente implementando una colección de decoradores por línea en lugar de un único decorador. A pesar de estas dificultades, la aplicación logra su objetivo principal de ofrecer funcionalidades básicas de formato y anotación, demostrando el valor de los patrones de diseño en la creación de software mantenible y extensible.