UNIVERSIDAD MADERO

Gestor de tareas

Plataformas Propietarias

Emanuel Rendón Veloz

Ingeniería en Software

Nicolás Arrioja Landa Cosio

Introducción

Manejar eficientemente las múltiples tareas de una persona, los gestores de tareas se han convertido en herramientas fundamentales para administrar la productividad del trabajo, este proyecto organiza las tareas pendientes con mensajes diferentes según la prioridad, la prioridad es asignada según el tiempo restante para la realización de la tarea.

Antecedentes

Herramientas de Gestión de Tareas: Plataformas como “Trello”, “Asana” y “Todoist” permiten organizar tareas, pero su capacidad para notificar eventos críticos es limitada.

Notificaciones Inteligentes: Sistemas como “Slack” y “Microsoft Teams” utilizan canales de comunicación específicos (push, email) para alertar a los usuarios sobre eventos importantes.

Este proyecto busca integrar estas ideas en una solución única que mejore la experiencia del usuario al gestionar tareas y recibir notificaciones oportunas.

Descripción del Problema

El problema abordado en el código se centra en la gestión ineficiente de tareas diarias o laborales. Muchas personas enfrentan dificultades para organizar, priorizar y realizar un seguimiento de sus actividades pendientes, lo que puede resultar en retrasos, olvidos o una mala administración del tiempo. Además, no existe una forma clara de recibir notificaciones oportunas sobre las tareas próximas a vencer o ya vencidas, lo que puede afectar la productividad.

Este sistema busca resolver estos problemas al proporcionar una herramienta que permita al usuario gestionar sus tareas de manera efectiva mediante la creación, modificación, eliminación y respaldo de tareas, así como la implementación de notificaciones personalizadas según la cercanía de las fechas límite.

Objetivo general

Desarrollar un sistema de gestión de tareas que permita al usuario organizar, modificar y eliminar actividades, además de recibir notificaciones automáticas basadas en la proximidad de las fechas límite, con el fin de mejorar la productividad y la administración del tiempo.

Objetivos específicos

1. Gestión de Tareas :
    - Permitir al usuario agregar nuevas tareas con detalles como nombre, descripción y fecha límite.
    - Facilitar la eliminación de tareas que ya no sean necesarias.
    - Ofrecer la posibilidad de modificar los detalles de una tarea existente, incluyendo su nombre, descripción y fecha límite.
2. Notificaciones Personalizadas :
    - Implementar un sistema de notificaciones que alerte al usuario sobre tareas pendientes utilizando diferentes métodos (correo electrónico, SMS y PopUp) según la urgencia de la tarea.
3. Respaldo de Tareas :
    - Proporcionar una funcionalidad para crear un respaldo de las tareas actuales, asegurando que la información no se pierda accidentalmente.
4. Interacción con el Usuario :
    - Diseñar una interfaz de consola intuitiva que permita al usuario interactuar fácilmente con el sistema mediante preguntas claras y opciones simples (como "Si" o "No").
5. Validación de Datos :
    - Asegurar que las tareas sean aprobadas por el usuario antes de ser agregadas al sistema y que las fechas límite puedan ser ajustadas si es necesario.
6. Guardar datos XML:
    - Se guardan las tareas en XML para que se legible y fácil de compartir.

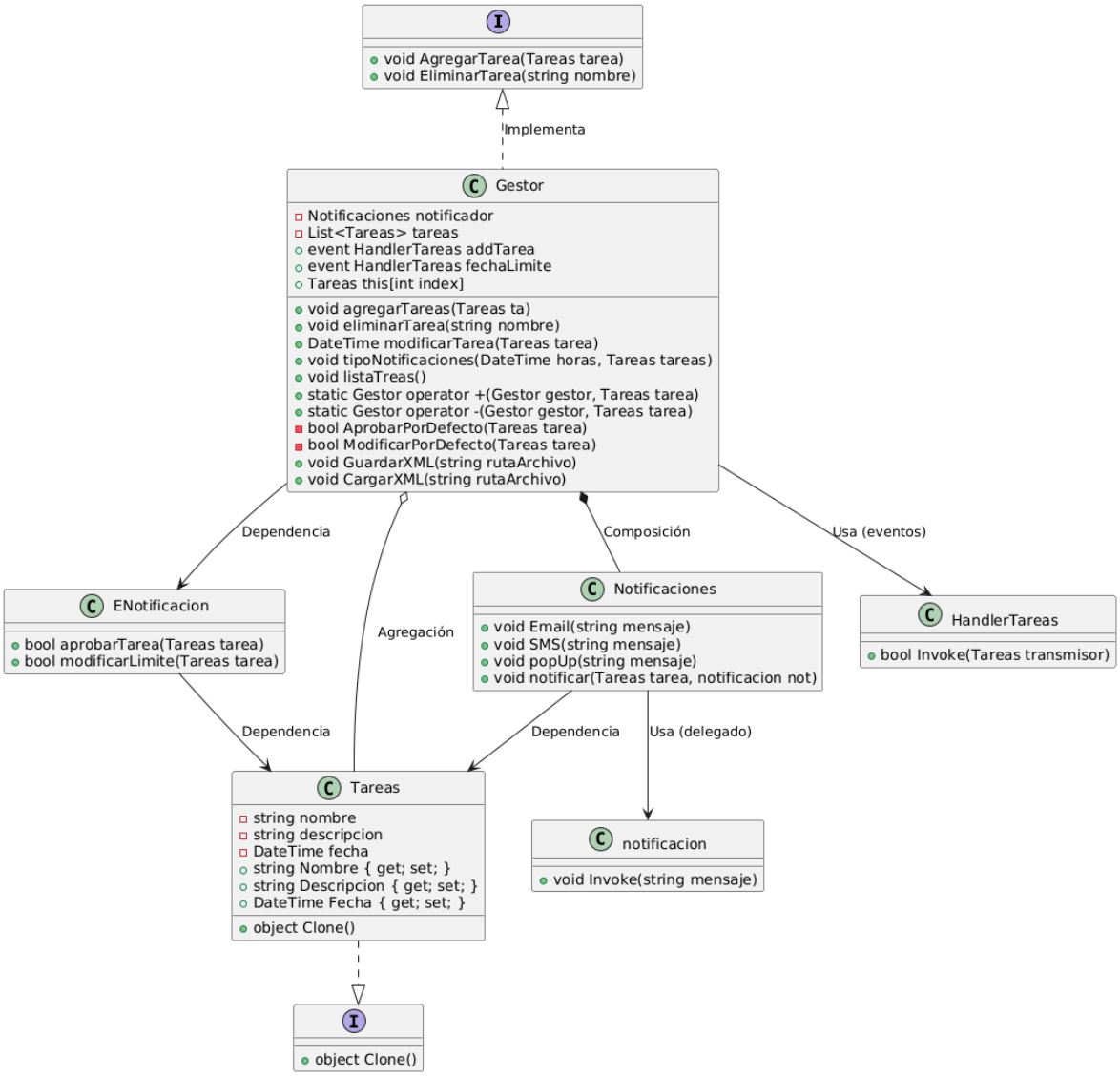
Alcances

El proyecto esta limitado solo a simular un gestor de tareas.

Limitaciones

Es un programa simple de gestión de tareas.

Diagrama de Clases



Código

```
/// Esta clase es una Interface
/// Autor: Emanuel Rendón Veloz
/// Fecha: 20-05-2025
/// Versión: 1.0.0

internal interface Itarea
 {
     void AgregarTarea(Tareas tarea);
     void EliminarTarea(string nombre);
 }


/// Esta clase se encarga de manejar las notificaciones relacionadas con la aprobación y modificación de tareas.
/// Autor: Emanuel Rendón Veloz
/// Fecha: 16-05-2025
/// Versión: 1.0.0
/// Modificación: 17-05-2025
internal class ENotificacion
{
    /// Método para aprobar una tarea antes de agregarla retornando un valor booleano.
    /// <param name="tarea">La tarea que se desea aprobar.</param>
    ///  <returns>True si el usuario aprueba la tarea, False en caso contrario.</returns>
    public bool aprobarTarea(Tareas tarea)
    {
        string respuesta = "";
        Console.ForegroundColor = ConsoleColor.Blue;

        // Solicita al usuario confirmación para agregar la tarea
        Console.WriteLine("¿Quieres agregar la siguiente tarea? {0}, con {1} horas de realización",
            tarea.Nombre, Math.Round((tarea.Fecha - DateTime.Now).TotalHours));
        Console.WriteLine("(Si/No)");
        respuesta = Console.ReadLine();

        // Verifica la respuesta del usuario
        if (respuesta.ToLower() == "si")
            return true;
        else
            return false;
    }

    /// Método para modificar la fecha límite de una tarea retornando un valor booleano.
    ///  <param name="tarea">La tarea cuya fecha límite se desea modificar.</param>
    /// <returns>True si el usuario desea modificar la fecha, False en caso contrario.</returns>
```

```
    public bool modificarLimite(Tareas tarea)
    {
        string respuesta = "";
        Console.ForegroundColor = ConsoleColor.Blue;

        // Solicita al usuario confirmación para modificar la fecha límite
        Console.WriteLine("¿Quieres modificar las horas de realización de la
siguiente tarea: {0}? ", tarea.Nombre);
        Console.WriteLine("(Si/No)");
        respuesta = Console.ReadLine();

        // Verifica la respuesta del usuario
        if (respuesta.ToLower() == "si")
            return true;
        else
            return false;
    }
}
```

```
/// Esta clase gestiona las operaciones relacionadas con las tareas, como agregar,
eliminar y modificar.
/// Autor: Emanuel Rendón Veloz
/// Fecha: 20-05-2025
/// Versión: 1.0.0
internal class Gestor : Itarea
{
    // Delegado para eventos relacionados con las tareas
    public delegate bool HandlerTareas(Tareas transmisor);

    // Instancia de la clase Notificaciones
    private Notificaciones notificador = new Notificaciones();

    // Eventos para aprobar y modificar la fecha de una tarea
    public event HandlerTareas addTarea;
    public event HandlerTareas fechaLimite;

    // Lista para almacenar tareas
    private List<Tareas> tareas = new List<Tareas>();

    /// Constructor de la clase Gestor
    public Gestor()
    {
        // Inicializa los eventos con métodos predeterminados
        addTarea += AprobarPorDefecto;
        fechaLimite += ModificarPorDefecto;
    }

    /// Método para agregar una tarea a la lista.
    /// <param name="ta">La tarea que se desea agregar.</param>
    public void agregarTareas(Tareas ta)
    {
        // Verifica si la tarea es aprobada mediante el evento
        if (addTarea.Invoke(ta))
        {
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("Aprobado");

            // Actualiza la fecha límite de la tarea
            ta.Fecha = modificarTarea(ta);

            // Agrega la tarea a la lista
            tareas.Add(ta);

            // Envía notificación según la fecha límite
            tipoNotificaciones(ta.Fecha, ta);
        }
    }

    /// Método para eliminar una tarea de la lista.
    /// <param name="nombre">El nombre de la tarea que se desea eliminar.</param>
    public void eliminarTarea(string nombre)
    {
        // Recorre la lista de tareas en orden inverso
```

```
        for (int i = tareas.Count - 1; i >= 0; i--)
        {
            if (tareas[i].Nombre == nombre)
            {
                tareas.RemoveAt(i);
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine("Tarea Eliminada");
            }
        }
    }

    /// Método para verificar y modificar la fecha límite de una tarea.
    ///   <param   name="tarea">La   tarea   cuya   fecha   límite   se   desea
modificar.</param>
    /// <returns>La nueva fecha límite de la tarea.</returns>
    public DateTime modificarTarea(Tareas tarea)
    {
        int nhoras;
        string leer;

        // Verifica si la fecha límite fue modificada mediante el evento
        if (fechaLimite.Invoke(tarea))
        {
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("Escribe las nuevas horas");
            nhoras = Convert.ToInt32(Console.ReadLine());

            // Calcula la nueva fecha límite
            return  tarea.Fecha.AddHours((nhoras  -  Math.Round((tarea.Fecha  -
DateTime.Now).TotalHours)));
        }
        else
        {
            return tarea.Fecha;
        }
    }

    /// Método para definir la notificación según la fecha límite.
    /// <param name="horas">La fecha límite de la tarea.</param>
    /// <param name="tareas">La tarea asociada.</param>
    public void tipoNotificaciones(DateTime horas, Tareas tareas)
    {
        double horasRestantes = (horas - DateTime.Now).TotalHours;

        if (horasRestantes <= 12)
        {
            notificador.notificar(tareas, notificador.Email);
        }
        else if (horasRestantes > 12 && horasRestantes < 24)
        {
            notificador.notificar(tareas, notificador.SMS);
        }
        else if (horasRestantes > 24)
        {
```

```
        notificador.notificar(tareas, notificador.popUp);
    }
}

/// Método para imprimir la lista de tareas agregadas.
public void listaTreas()
{
    foreach (Tareas tarea in tareas)
    {
        Console.WriteLine("-----------------------------------------");
        tipoNotificaciones(tarea.Fecha, tarea);
        Console.WriteLine("-----------------------------------------");
    }
}

/// Indexador para acceder a las tareas por índice.
public Tareas this[int index]
{
    get { return tareas[index]; }
    set { tareas[index] = value; }
}

/// Sobrecarga del operador + para agregar una tarea.
public static Gestor operator +(Gestor gestor, Tareas tarea)
{
    gestor.agregarTareas(tarea);
    return gestor;
}

/// Sobrecarga del operador - para eliminar una tarea.
public static Gestor operator -(Gestor gestor, Tareas tarea)
{
    gestor.eliminarTarea(tarea.Nombre);
    return gestor;
}

/// Implementación explícita de la interfaz para agregar una tarea.
void Itarea.AgregarTarea(Tareas tarea)
{
    agregarTareas(tarea);
}

/// Implementación explícita de la interfaz para eliminar una tarea.
void Itarea.EliminarTarea(string nombre)
{
    eliminarTarea(nombre);
}

/// Método por defecto para aprobar una tarea.
private bool AprobarPorDefecto(Tareas tarea)
{
    return true;
}
```

```
    /// Método por defecto para modificar la fecha límite de una tarea.
    private bool ModificarPorDefecto(Tareas tarea)
    {
        return false;
    }


        /// <summary>
        /// Guarda la lista de tareas en un archivo XML.
        /// </summary>
        /// <param name="rutaArchivo">Ruta del archivo XML.</param>
        public void GuardarXML(string rutaArchivo)
        {
            //Crea un serializador XML para trabajar con listas de objetos
Tareas
            XmlSerializer serializer = new XmlSerializer(typeof(List<Tareas>));

            //Abre un StreamWriter para escribir en la ruta especificada
            //    (using asegura la liberación correcta de recursos)
            using (StreamWriter writer = new StreamWriter(rutaArchivo))
            {
                // Serializa la lista 'tareas' y la escribe en el archivo
                serializer.Serialize(writer, tareas);
            } //  Cierra automáticamente el StreamWriter al salir del bloque
using

            //  Imprime la representación en string del objeto serializer
            //    (Nota: Esto mostrará el nombre del tipo, no datos útiles)
            Console.WriteLine(serializer);

            //  Confirma la operación al usuario
            Console.WriteLine("Datos guardados en formato XML.");
        }

        /// <summary>
        /// Carga las tareas desde un archivo XML.
        /// </summary>
        /// <param name="rutaArchivo">Ruta del archivo XML.</param>
        public void CargarXML(string rutaArchivo)
        {
            // Verifica si el archivo existe
            if (File.Exists(rutaArchivo))
            {
                //  Crea un serializador XML para listas de Tareas
                XmlSerializer serializer = new
XmlSerializer(typeof(List<Tareas>));

                //  Abre un StreamReader para leer el archivo
                using (StreamReader reader = new StreamReader(rutaArchivo))
                {
                    //  Deserializa el contenido del archivo y hace casting a
List<Tareas>
                    tareas = (List<Tareas>)serializer.Deserialize(reader);

                    //  Imprime la representación en string de la lista
'tareas'
                    //    (Requiere override de ToString() en Tareas para ser
útil)
```

```
                    Console.WriteLine(tareas);
                } //  Cierra automáticamente el StreamReader

                //  Confirma la operación al usuario
                Console.WriteLine("Datos cargados desde XML.");
            }
            else
            {
                //  Manejo básico de caso de archivo inexistente
                Console.WriteLine("El archivo no existe.");
            }
        }
}
```

```
/// Esta clase se encarga de enviar notificaciones de diferentes tipos (Email,
SMS, PopUp).
```

```
/// Autor: Emanuel Rendón Veloz
/// Fecha: 20-05-2025
/// Versión: 1.0.0
internal class Notificaciones
{
    /// Delegado para definir métodos de notificación.
    public delegate void notificacion(string mensaje);

    /// Método para enviar notificaciones por correo electrónico.
    public void Email(string mensaje)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("Email: {0}, TAREA URGENTE D:", mensaje);
    }

    /// Método para enviar notificaciones por SMS.
    public void SMS(string mensaje)
    {
        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine("SMS: {0}, la fecha está cerca de vencer :)", mensaje);
    }

    /// Método para enviar notificaciones por PopUp.
    public void popUp(string mensaje)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("PopUp: {0}, tienes tiempo de sobra :D", mensaje);
    }

    /// Método para notificar según el tipo de notificación seleccionado.
    public void notificar(Tareas tarea, notificacion not)
    {
        string mensaje = $"Tarea: {tarea.Nombre}, Vence en:
{Math.Round((tarea.Fecha - DateTime.Now).TotalHours)} horas, Descripción:
{tarea.Descripcion}";
        not.Invoke(mensaje);
    }
}


/// Esta clase representa una tarea con sus atributos y métodos.
/// Autor: Emanuel Rendón Veloz
/// Fecha: 16-05-2025
/// Versión: 1.0.0
/// Modificación: 16-05-2025

[Serializable]
public class Tareas : ICloneable
{
    private string nombre; //!< El nombre de la tarea
    private string descripcion; //!< La descripción de la tarea
    private DateTime fecha; //!< La fecha límite de la tarea

    /// Propiedades públicas para acceder y modificar los atributos privados
```

```
    public string Nombre { get { return nombre; } set { nombre = value; } }
    public string Descripcion { get { return descripcion; } set { descripcion =
value; } }
    public DateTime Fecha { get { return fecha; } set { fecha = value; } }

    /// Constructor para inicializar una tarea
    public Tareas(string Nombre, string Descripcion, DateTime Fecha)
    {
        nombre = Nombre;
        descripcion = Descripcion;
        fecha = Fecha;
    }

    /// Método para clonar una tarea
    public object Clone()
    {
        return new Tareas(Nombre, Descripcion, Fecha);
    }
}
```

```
/// Esta clase contiene el punto de entrada del programa y maneja la interacción
con el usuario.
/// Autor: Emanuel Rendón Veloz
/// Fecha: 20-05-2025
```

```
/// Versión: 1.0.0

internal class Program
{
    static void Main(string[] args)
    {
        /// Declaración de variables
        string respuesta = "";
        string nombre;
        string descripcion;
        int horas;
        int indice;

        /// Instancia de la clase gestor para manejar las tareas
        Gestor gestor = new Gestor();

        /// Instancia de la clase ENotificacion para manejar eventos relacionados
con las tareas
        ENotificacion en = new ENotificacion();

        /// Asociación de los eventos del gestor con los métodos de ENotificacion
        /// Evento para aprobar una tarea y modificar una fecha
        gestor.addTarea += en.aprobarTarea;
        gestor.fechaLimite += en.modificarLimite;

        /// Creación de tareas de ejemplo
        Tareas tarea1 = new Tareas("Revisar informe", "Revisar el informe mensual
de ventas", DateTime.Now.AddHours(10));
        gestor.agregarTareas(tarea1);

        Tareas tarea2 = new Tareas("Enviar correo", "Enviar recordatorio al
equipo", DateTime.Now.AddHours(15));
        gestor.agregarTareas(tarea2);

        Tareas tarea3 = new Tareas("Preparar presentación", "Preparar slides para
la reunión", DateTime.Now.AddHours(30));
        gestor.agregarTareas(tarea3);

        do
        {
            /// Pregunta si el usuario quiere hacer un respaldo de sus tareas
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("¿Quieres hacer un respaldo de tus tareas?");
            Console.WriteLine("(Si/No)");
            respuesta = Console.ReadLine();

            /// If que verifica si la respuesta es "si"
            if (respuesta.ToLower() == "si")
            {
                /// Solicita el nombre de la tarea a ser respaldada
                Gestor Rgestor = new Gestor();
                Tareas t1 = (Tareas)tarea1.Clone();
                Rgestor.agregarTareas(t1);
                Tareas t2 = (Tareas)tarea2.Clone();
```

```
                Rgestor.agregarTareas(t1);
                Tareas t3 = (Tareas)tarea3.Clone();
                Rgestor.agregarTareas(t1);

                Console.WriteLine("Respaldo exitoso");
            }

            /// El ciclo do-while continuará mientras la respuesta sea "no"
        } while (respuesta.ToLower() != "no");

        /// Bucle do-while que permite al usuario agregar tareas
        do
        {
            /// Pregunta al usuario si quiere agregar una nueva tarea
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("¿Quieres agregar una nueva tarea?");
            Console.WriteLine("(Si/No)");
            respuesta = Console.ReadLine();

            /// If que verifica si la respuesta es "si"
            if (respuesta.ToLower() == "si")
            {
                /// Solicitud de los datos de la tarea para ser agregada
                Console.WriteLine("Escribe el nombre de la tarea");
                nombre = Console.ReadLine();

                Console.WriteLine("Escribe la descripción de la tarea");
                descripcion = Console.ReadLine();

                Console.WriteLine("Escribe el número de horas para su
realización");
                horas = Convert.ToInt32(Console.ReadLine());

                /// Creación y agregación de la nueva tarea
                Tareas tarea = new Tareas(nombre, descripcion,
DateTime.Now.AddHours(horas));
                gestor += tarea;

                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("¿Quieres agregar una nueva tarea?");
                Console.WriteLine("(Si/No)");
                respuesta = Console.ReadLine();
            }

            /// El ciclo do-while continuará mientras la respuesta sea "no"
        } while (respuesta.ToLower() != "no");

        Console.ForegroundColor = ConsoleColor.Yellow;

        /// Impresión de la lista de tareas que fueron agregadas
        Console.WriteLine("------------Lista de tareas-----------------");
        gestor.listaTreas();

        /// Bucle do-while que permite al usuario eliminar tareas
```

```
do
{
    /// Pregunta si el usuario quiere eliminar una tarea
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("¿Quieres eliminar una tarea?");
    Console.WriteLine("(Si/No)");
    respuesta = Console.ReadLine();

    /// If que verifica si la respuesta es "si"
    if (respuesta.ToLower() == "si")
    {
        /// Solicita el índice de la tarea a ser eliminada
        Console.WriteLine("Escribe el índice de la tarea");
        indice = Convert.ToInt32(Console.ReadLine());

        Tareas ETarea = gestor[indice];

        /// Elimina la tarea de la lista
        gestor -= ETarea;

        /// Pregunta si quiere eliminar otra tarea
        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine("¿Quieres eliminar una tarea?");
        Console.WriteLine("(Si/No)");
        respuesta = Console.ReadLine();
    }

    /// El ciclo do-while continuará mientras la respuesta sea "no"
} while (respuesta.ToLower() != "no");

/// Impresión de la lista de tareas que fueron agregadas
Console.WriteLine("------------Lista de tareas-----------------");
gestor.listaTreas();

/// Bucle do-while que permite al usuario modificar tareas
do
{
    /// Pregunta si el usuario quiere modificar una tarea
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("¿Quieres modificar una tarea?");
    Console.WriteLine("(Si/No)");
    respuesta = Console.ReadLine();

    /// If que verifica si la respuesta es "si"
    if (respuesta.ToLower() == "si")
    {
        /// Solicita el índice de la tarea a ser modificada
        Console.WriteLine("Escribe el índice de la tarea");
        indice = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Escribe el nuevo nombre de la tarea");
        string nuevoNombre = Console.ReadLine();

        Console.WriteLine("Escribe la nueva descripción de la tarea");
```

```
                string nuevaDescripcion = Console.ReadLine();

                Console.WriteLine("Escribe el número de horas para su
realización");
                int nuevasHoras = Convert.ToInt32(Console.ReadLine());

                gestor[indice] = new Tareas(nuevoNombre, nuevaDescripcion,
DateTime.Now.AddHours(nuevasHoras));

                Console.WriteLine("------------Lista de tareas-----------------
");
                gestor.listaTreas();

                /// Pregunta si quiere modificar otra tarea
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("¿Quieres modificar otra tarea?");
                Console.WriteLine("(Si/No)");
                respuesta = Console.ReadLine();
            }

            /// El ciclo do-while continuará mientras la respuesta sea "no"
        } while (respuesta.ToLower() != "no");
    do
    {
        /// Pregunta si el usuario quiere guardar las tareas en XML
        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine("¿Quieres guardar las tareas en XML?");
        Console.WriteLine("(Si/No)");
        respuesta = Console.ReadLine();
        if (respuesta.ToLower() == "si")
        {
            gestor.GuardarXML("tareas.xml");
        }
        /// Pregunta si el usuario quiere cargar las tareas en XML
        Console.WriteLine("¿Quieres cargar tareas desde XML?");
        Console.WriteLine("(Si/No)");
        respuesta = Console.ReadLine();
        if (respuesta.ToLower() == "si")
        {
            gestor.CargarXML("tareas.xml");
            Console.WriteLine("------------Lista de tareas-----------------");

        }

    /// El ciclo do-while continuará mientras la respuesta sea "no"
} while (respuesta.ToLower() != "no");

        Console.ReadKey();
    }
}
```

Bibliografía

nicosiored. (n.d.). YouTube. https://www.youtube.com/@nicosiored