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 de software

TÍTULO ACTIVIDAD:

Examen unidad 4 y 5

UNIDAD A EVALUAR:
4 y 5

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

Acuña Gomez Carlos Yael 21212320

NOMBRE DEL MAESTRO (A):

Maribel Guerrero Luis

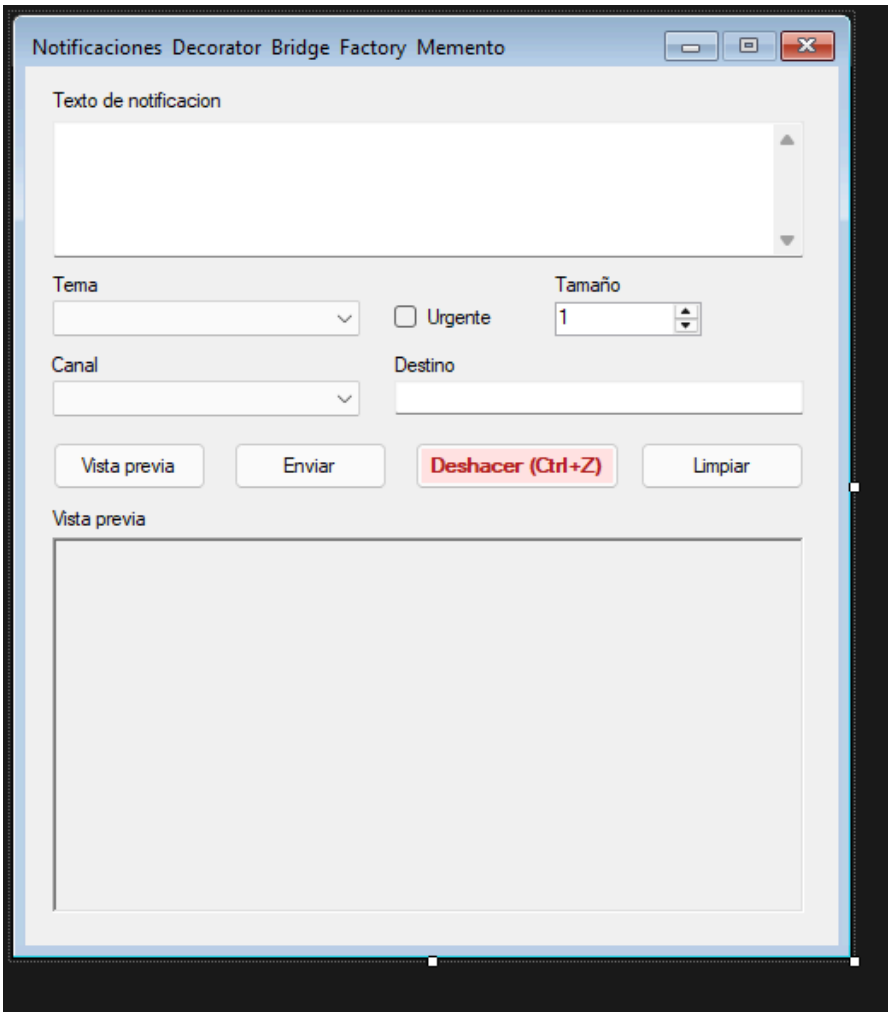
Código
Program.cs

```
using System;

namespace NotificacionesWinForms
{
    internal static class Program
    {
        [STAThread]
        static void Main()
        {
            System.Windows.Forms.Application.EnableVisualStyles();
            System.Windows.Forms.Application.SetCompatibleTextRenderingDefault(false);
            System.Windows.Forms.Application.Run(new Presentation.MainForm());
        }
    }
}
```

DisplayForm.cs

```
using System.Drawing;
using System.Windows.Forms;

namespace NotificacionesWinForms.Presentation
{
    3 referencias
    public class DisplayForm : Form
    {
        private readonly RichTextBox _rtb = new RichTextBox();

        2 referencias
        public DisplayForm(string titulo, string contenido)
        {
            Text = titulo;
            Width = 520;
            Height = 460;
            StartPosition = FormStartPosition.CenterParent;
            FormBorderStyle = FormBorderStyle.FixedDialog;
            MaximizeBox = false;
            MinimizeBox = false;

            _rtb.Dock = DockStyle.Fill;
            _rtb.ReadOnly = true;
            _rtb.Font = new System.Drawing.Font("Consolas", 10);
            _rtb.WordWrap = false;

            var panel = new Panel { Dock = DockStyle.Fill, Padding = new Padding(8) };
            panel.Controls.Add(_rtb);
            Controls.Add(panel);

            PreviewStyler.Apply(_rtb, contenido);
        }
    }
}
```

MainForm.cs


Notificaciones Decorator Bridge Factory Memento
Texto de notificacion
Tema
Urgente
Tamaño
1
Canal
Destino
Vista previa
Enviar
Deshacer (Ctrl+Z)
Limpiar
Vista previa

PreviewStyler.cs

```
using System;
using System.Drawing;
using System.Globalization;
using System.Text;
using System.Windows.Forms;

namespace NotificacionesWinForms.Presentation
{
    3 referencias
    public static class PreviewStyler
    {
        3 referencias
        public static void Apply(RichTextBox rtb, string text)
        {

            if (string.IsNullOrEmpty(text))
            {
                MessageBox.Show("El texto de la notificación está vacío.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            rtb.Clear();
            var temaColor = Color.Gray;

            string normAll = Norm(text);
            if (normAll.Contains("[NAVIDAD]")) temaColor = Color.Green;
            else if (normAll.Contains("[HALLOWEEN]")) temaColor = Color.Orange;
            else if (normAll.Contains("[CUMPLEANOS]")) temaColor = Color.CadetBlue;

            bool urgente = normAll.Contains("[URGENTE]");

            foreach (var raw in text.Replace("\r\n", "\n").Split('\n'))
            {
                string line = raw;
                string trimmed = line.Trim();
                string normLine = Norm(trimmed);

                if (urgente && (normLine == "[URGENTE]" || IsBarOf(line, '!')))
                {
                    AppendLine(rtb, line, Color.Red);
                    continue;
                }
```

```
                if (IsStarLine(line) || normLine == "[NAVIDAD]" || normLine == "[HALLOWEEN]" || normLine == "[CUMPLEANOS]" ||
                    normLine == "FELIZNAVIDAD" || normLine == "NOCHEDEHALLOWEEN" || normLine == "FELIZCUMPLEANOS")
                {
                    AppendLine(rtb, line, temaColor);
                    continue;
                }

                if (normLine.StartsWith("[TAMANIOX"))
                {
                    AppendLine(rtb, line, Color.CornflowerBlue);
                    continue;
                }

                AppendLine(rtb, line, rtb.ForeColor);
            }
        }
```

```
private static void AppendLine(RichTextBox rtb, string text, Color color)
{
    int start = rtb.TextLength;
    rtb.AppendText(text + Environment.NewLine);
    rtb.Select(start, text.Length);
    rtb.SelectionColor = color;
    rtb.Select(rtb.TextLength, 0);
    rtb.SelectionColor = rtb.ForeColor;
}

1 referencia
private static bool IsStarLine(string s)
{
    string t = s.Replace(" ", "");
    if (string.IsNullOrWhiteSpace(t)) return false;
    if (t.Length < 5) return false;
    foreach (var ch in t) if (ch != '*') return false;
    return true;
}


private static bool IsBarOf(string s, char c)
{
    string t = s.Replace(" ", "");
    if (t.Length < 5) return false;
    int count = 0;
    foreach (var ch in t) if (ch == c) count++;
    return count >= (int)(t.Length * 0.9);
}
```

```
2 referencias
private static string Norm(string s)
{
    string noSpaces = s.Replace(" ", "").ToUpperInvariant();
    string decomposed = noSpaces.Normalize(NormalizationForm.FormD);
    var sb = new StringBuilder(decomposed.Length);
    foreach (var ch in decomposed)
    {
        var uc = CharUnicodeInfo.GetUnicodeCategory(ch);
        if (uc != UnicodeCategory.NonSpacingMark) sb.Append(ch);
    }
    return sb.ToString().Normalize(NormalizationForm.FormC);
}
```

AppSender.cs

```
using NotificacionesWinForms.Presentation;

namespace NotificacionesWinForms.Infrastructure
{

    public class AppSender : ISender
    {

        public string Nombre => "App";
        2 referencias
        public void Send(string texto, string destino)
        {
            using (var f = new DisplayForm($"[App a {destino}]", texto))
                f.ShowDialog();
        }
    }
```

EmailSender.cs

```
using NotificacionesWinForms.Presentation;

namespace NotificacionesWinForms.Infrastructure
{
    1 referencia
    public class EmailSender : ISender
    {
        1 referencia
        public string Nombre => "Email";
        2 referencias
        public void Send(string texto, string destino)
        {
            using (var f = new DisplayForm($"[Email a {destino}]", texto))
                f.ShowDialog();
        }
    }
}
```

ISender.cs

```
namespace NotificacionesWinForms.Infrastructure
{
    5 referencias
    public interface ISender
    {
        2 referencias
        string Nombre { get; }
        3 referencias
        void Send(string texto, string destino);
    }
}
```

SenderFactory.cs

```
namespace NotificacionesWinForms.Infrastructure
{
    1 referencia
    public static class SenderFactory
    {
        1 referencia
        public static ISender Create(int canal)
        {
            switch (canal)
            {
                case 1: return new AppSender();
                case 2: return new EmailSender();
                default: return new AppSender();
            }
        }
    }
}
```

DraftConfig.cs

```
namespace NotificacionesWinForms.Domain
{

    public class DraftConfig
    {

        public string Texto { get; set; } = "";
        6 referencias
        public int Tema { get; set; } = 0;
        6 referencias
        public bool Urgente { get; set; } = false;
        6 referencias
        public int Tamanio { get; set; } = 1;
        5 referencias
        public int Canal { get; set; } = 1;
        5 referencias
        public string Destino { get; set; } = "";

        1 referencia
        public DraftMemento CreateMemento()
            => new DraftMemento(Texto, Tema, Urgente, Tamanio, Canal, Destino);

        1 referencia
        public void Restore(DraftMemento m)
        {
            if (m == null) return;
            Texto = m.Texto;
            Tema = m.Tema;
            Urgente = m.Urgente;
            Tamanio = m.Tamanio;
            Canal = m.Canal;
            Destino = m.Destino;
        }
    }
}
```

DraftMemento.cs

```
namespace NotificacionesWinForms.Domain
{
    public class DraftMemento
    {
        2 referencias
        public string Texto { get; }
        2 referencias
        public int Tema { get; }
        2 referencias
        public bool Urgente { get; }
        2 referencias
        public int Tamanio { get; }
        2 referencias
        public int Canal { get; }
        2 referencias
        public string Destino { get; }

        1 referencia
        internal DraftMemento(string texto, int tema, bool urgente, int tamanio, int canal, string destino)
        {
            Texto = texto;
            Tema = tema;
            Urgente = urgente;
            Tamanio = tamanio;
            Canal = canal;
            Destino = destino;
        }
    }
}
```

History.cs

```
using System.Collections.Generic;

namespace NotificacionesWinForms.Domain
{
    2 referencias
    public class History
    {
        private readonly Stack<DraftMemento> _stack = new Stack<DraftMemento>();
        1 referencia
        public void Push(DraftMemento m) { if (m != null) _stack.Push(m); }
        1 referencia
        public DraftMemento PopOrNull() => _stack.Count > 0 ? _stack.Pop() : null;
        1 referencia
        public void Clear() => _stack.Clear();
    }
}
```

IMensaje.cs

```
namespace NotificacionesWinForms.Domain
{
    7 referencias
    public interface IMessage { string GetText(); }
}
```

Notification.cs

```
namespace NotificacionesWinForms.Domain
{
    using NotificacionesWinForms.Infrastructure;

    2 referencias
    public class Notification
    {
        private readonly ISender _sender;
        1 referencia
        public Notification(ISender sender) { _sender = sender; }
        1 referencia
        public void Publicar(IMessage contenido, string destino) => _sender.Send(contenido.GetText(), destino);
    }
}
```

PlainMessage.cs

```
namespace NotificacionesWinForms.Domain
{
    2 referencias
    public class PlainMessage : IMessage
    {
        private readonly string _text;
        1 referencia
        public PlainMessage(string text) { _text = text; }
        4 referencias
        public string GetText() => _text;
    }
}
```

SimpleDecorator.cs

```
using System;

namespace NotificacionesWinForms.Domain
{
    2 referencias
    public class SimpleDecorator : IMessage
    {
        private readonly IMessage _inner;
        private readonly int _tema;
        private readonly bool _urgente;
        private readonly int _tamanio;

        1 referencia
        public SimpleDecorator(IMessage inner, int tema, bool urgente, int tamanio)
        {
            _inner = inner;
            _tema = tema;
            _urgente = urgente;
            _tamanio = tamanio;
        }

        4 referencias
        public string GetText()
        {
            string t = _inner.GetText();

            if (_tema != 0)
            {
                string marco = new string('*', Math.Max(10, t.Length));
                if (_tema == 1) t = $"[NAVIDAD]\n{marco}\nFELIZ NAVIDAD\n{t}\n{marco}";
                else if (_tema == 2) t = $"[HALLOWEEN]\n{marco}\nNOCHE DE HALLOWEEN\n{t}\n{marco}";
                else if (_tema == 3) t = $"[CUMPLEAÑOS]\n{marco}\nFELIZ CUMPLEAÑOS\n{t}\n{marco}";
            }

            if (_tamanio > 1)
            {
                var lineas = t.Split('\n');
                for (int i = 0; i < lineas.Length; i++)
                    lineas[i] = string.Join(" ", lineas[i].ToUpperInvariant().ToCharArray());

                t = $"[Tamanio x{_tamanio}]\n" + string.Join("\n", lineas);
            }
```

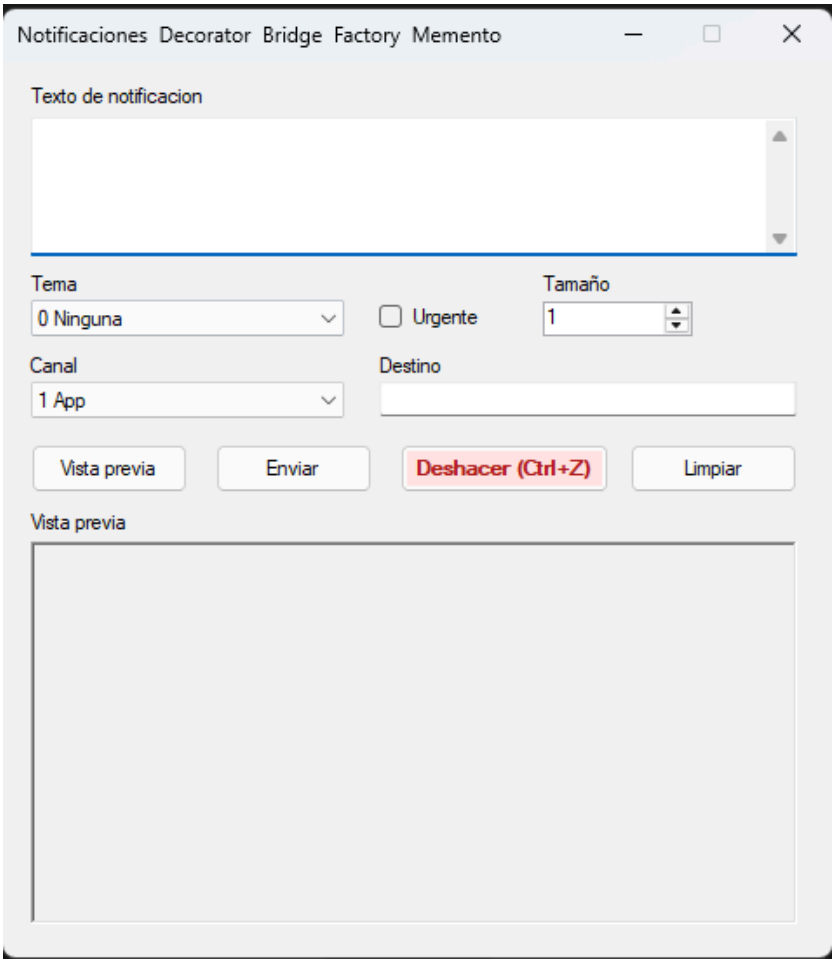
```
        if (_urgente)
        {
            string barra = new string('!', 24);
            t = "[URGENTE]\n" + t + "\n" + barra;
        }
        else
        {
            t = "[NORMAL]\n" + t;
        }

        return t;
    }
}
```

NotifierService.cs

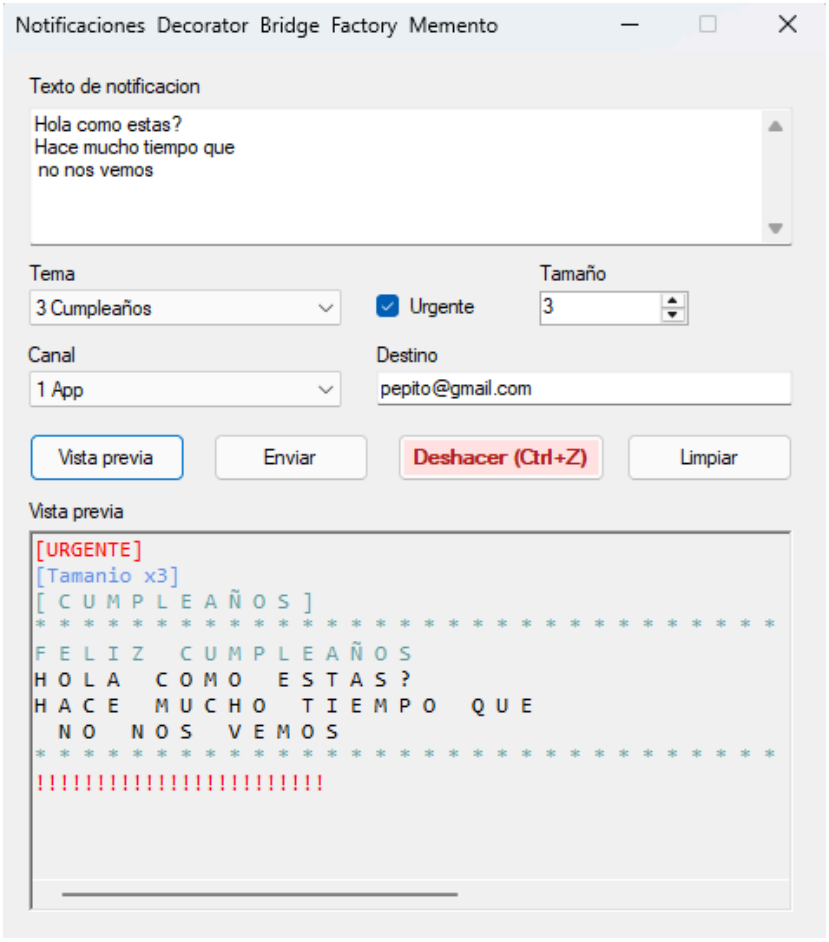
```
namespace NotificacionesWinForms.Application
{
    using NotificacionesWinForms.Domain;
    using NotificacionesWinForms.Infrastructure;

    2 referencias
    public class NotifierService
    {
        2 referencias
        private IMessage BuildMessage(DraftConfig cfg)
        {
            IMessage msg = new PlainMessage(cfg.Texto);
            if (cfg.Tema != 0 || cfg.Urgente || cfg.Tamanio > 1)
                msg = new SimpleDecorator(msg, cfg.Tema, cfg.Urgente, cfg.Tamanio);
            return msg;
        }

        2 referencias
        public string BuildPreview(DraftConfig cfg) => BuildMessage(cfg).GetText();

        1 referencia
        public void Send(DraftConfig cfg)
        {
            var sender = SenderFactory.Create(cfg.Canal);
            var noti = new Notification(sender);
            noti.Publicar(BuildMessage(cfg), cfg.Destino);
        }
    }
}
```

Ejecución

Interfaz sin información



Interfaz con datos y vista Previa



Mensaje Enviado

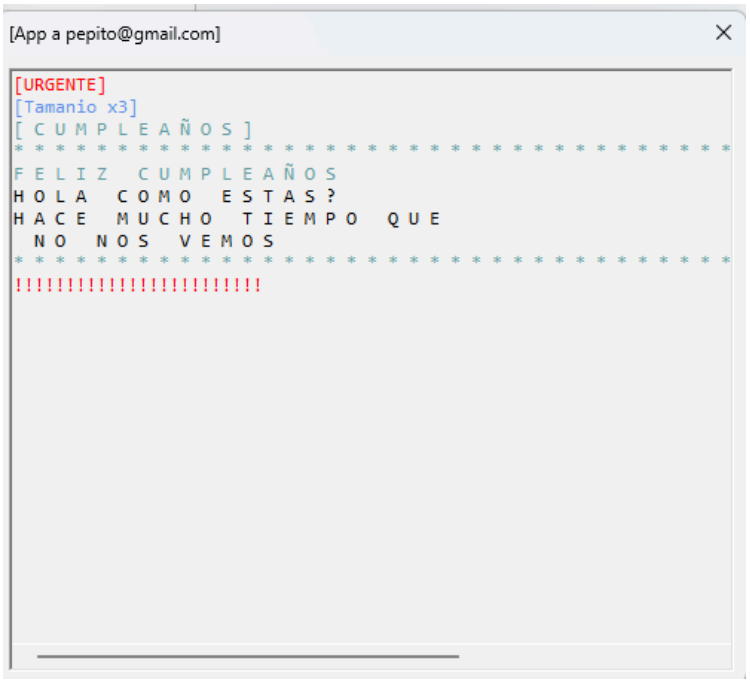
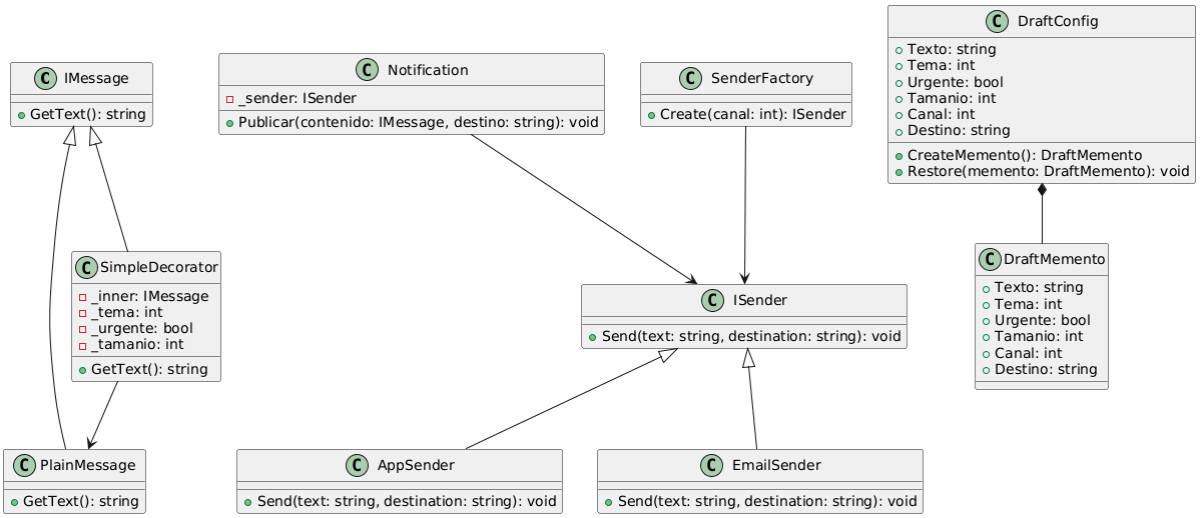


Diagrama UML



Conclusión

En conclusión el patrón decorator para este proyecto facilita la personalización de los mensajes, para poder mostrar algo con un mejor diseño que dejarlo simple, además con bridge y factory method me ayudaron en la comunicación del mensaje indistintamente de la plataforma que quiera enviar el mensaje, y por la parte de memento me ayuda a deshacer un cambio que quiera si no me gusto como quedo el diseño de mi mensaje y ya el patrón de arquitectura MVC me ayudó a la gestion y organizacion de mi codigo para separarlos y encontrar de mejor manera ciertos archivos o también acomodar archivos o funciones que tengan en una misma carpeta. Y para terminar me sentí muy agusto con este proyecto me gusto la personalización de mensajes y pequeñas cosas que le puedo poner a estos mismos.