

Practica 13

Continuando con nuestro proyecto, después de haber creado las vistas para el inicio de sesión y el formulario de registro de usuarios, es necesario agregar los modelos a nuestro proyecto. Posteriormente, modificaremos nuestra base de datos para reflejar estos cambios.

Crearemos una clase en la carpeta Model llamada Correo.

```
3 referencias
public class Correo
{
    2 referencias
    public string Para { get; set; }
    2 referencias
    public string Asunto { get; set; }
    2 referencias
    public string Contenido { get; set; }
}
```

Crearemos un modelo para los usuarios, le pondremos MUsuario:

```
0 referencias
public class MUsuario
{
    [Key]
    0 referencias
    public int idUsuario { get; set; }

    [Required]
    [StringLength(50)]
    [Display(Name = "Nombre del usuario")]
    0 referencias
    public string nombre { get; set; }

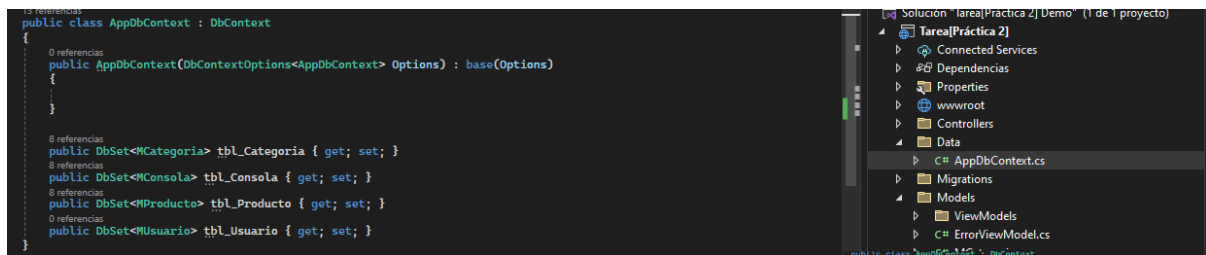
    [Required]
    [StringLength(50)]
    [Display(Name = "Correo del usuario")]
    0 referencias
    public string correo { get; set; }

    [Required]
    [StringLength(200)]
    [Display(Name = "Contraseña del usuario")]
    0 referencias
    public string clave { get; set; }

    0 referencias
    public bool restablecer { get; set; }
    0 referencias
    public bool confirmado { get; set; }
    0 referencias
    public string token { get; set; }
}
```

0 4 | Línea: 30 Carácter: 1 SPC CRLF

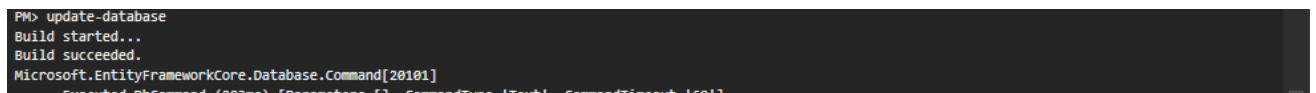
Procedemos agregar nuestro modelo en la clase **AppDbContext** que se encuentra en la carpeta Data



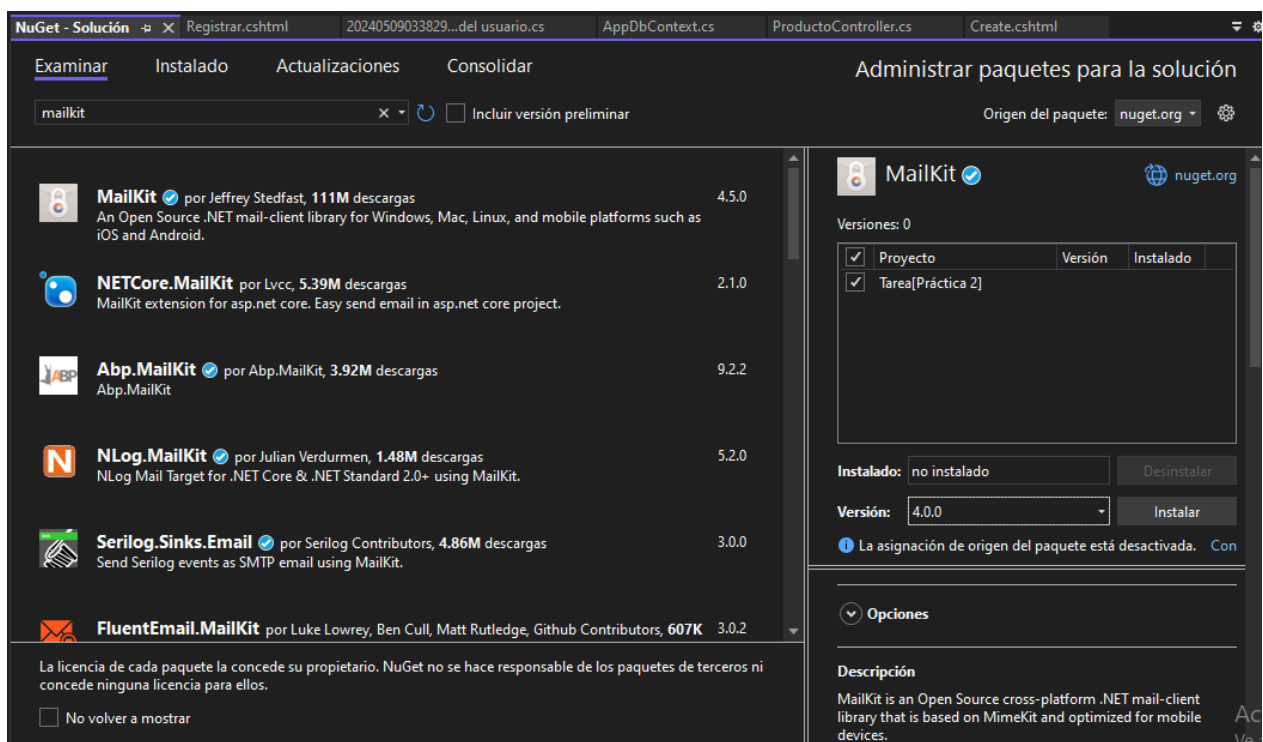
Luego tendremos que crear una migración:



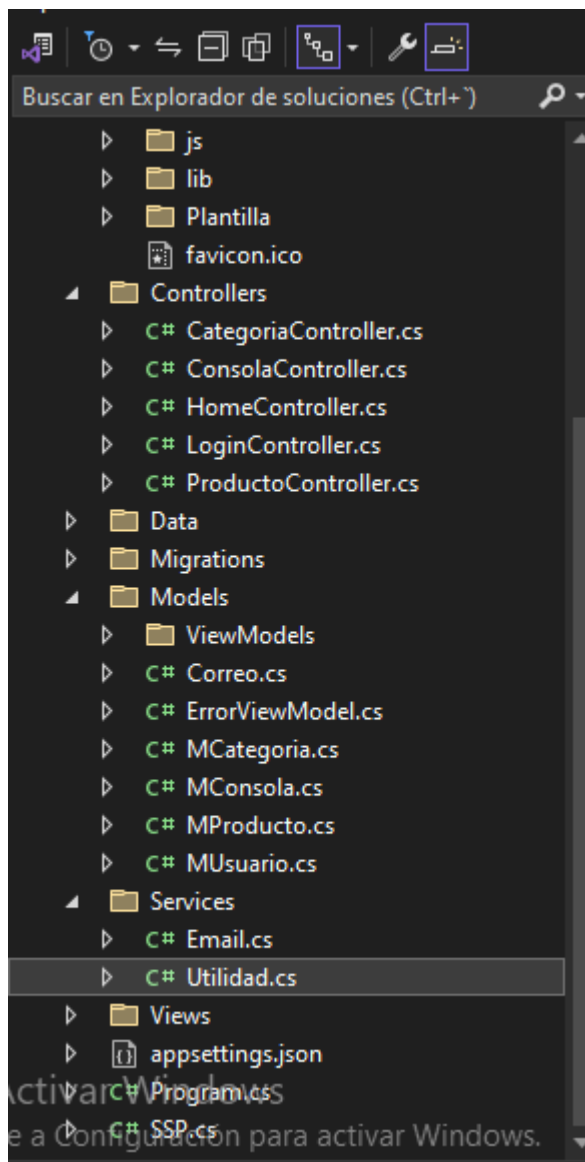
Procedemos actualizar nuestra bd



Luego en nuestro administrador de paquetes NuGet, vamos a instalar Mailkit la version 4.0



Crearemos una carpeta llamada Services la cual contendrá dos clases, que nos servira para mandar los emails y la utilidad que nos ayudará a encriptar la contraseña y generar un token, que lo usaremos para no mandar el id del usuario cuando desee activar la cuenta.



La clase Utilidad la configuraremos de esta manera:

```

2 referencias
public static class Utilidad
{
    //Metodo para encriptar
    1 referencia
    public static string ConvertirSHA256(string texto)
    {
        string hash=string.Empty;
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] hashValue = sha256.ComputeHash(Encoding.UTF8.GetBytes(texto));

            foreach (byte bt in hashValue)
            {
                hash += $"{bt:X2}";
            }
        }

        return hash;
    }

    1 referencia
    public static string GenerarToken()
    {
        string token = Guid.NewGuid().ToString("N");
        return token;
    }
}

```

La clase Email está configurada como se las enseñó el ingeniero en las clases con MailTrap, link del video por si se les olvido: <https://www.youtube.com/watch?v=KjVy5rduvQ0>

```

1 referencia
public static class Email
{
    private static string _Host = "sandbox.smtp.mailtrap.io";
    private static int _Puerto = 587;
    private static string _Nombre = "Julio Amaya";
    private static string _username = "329aad191d5842";
    private static string _Correo = "julioamaya988@gmail.com";
    private static string _Clave = "7dbcdac8237968";

    1 referencia
    public static bool Enviar(Correo correo)
    {
        try
        {
            var email = new MimeMessage();
            email.From.Add(new MailboxAddress(_Nombre, _Correo));
            email.To.Add(MailboxAddress.Parse(correo.Para));
            email.Subject = correo.Asunto;
            email.Body = new TextPart(TextFormat.Html)
            {
                Text = correo.Contenido
            };

            using (var smtp = new SmtplibClient())
            {
                smtp.Connect(_Host, _Puerto, false);
                smtp.Authenticate(_username, _Clave);
                smtp.Send(email);
                smtp.Disconnect(true);
            }

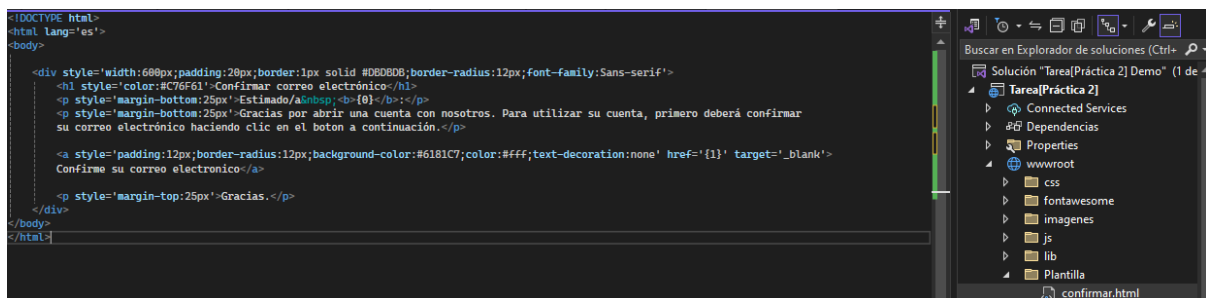
            return true;
        }
        catch (Exception ex)
        {
            throw ex;
            return false;
        }
    }
}

```

Luego en nuestra clase SSP agregaremos la ruta de un archivo html que nos servirá como plantilla del correo:

```
public static class SSP
{
    public static string ProductoPath = @"\\imagenes\\producto";
    public static string PlantillaPath = @"\\Plantilla\\confirmar.html";
}
```

Creamos la carpeta Plantilla y dentro el archivo confirmar.html, **OJO** siempre en la ruta wwwroot



```
<!DOCTYPE html>
<html lang='es'>
<body>

<div style='width:600px;padding:20px;border:1px solid #D8DBDB;border-radius:12px;font-family:Sans-serif'>
<h1 style='color:#C76F61'>Confirmar correo electrónico</h1>
<p style='margin-bottom:25px'>Estimado/a<b></b></p>
<p style='margin-bottom:25px'>Gracias por abrir una cuenta con nosotros. Para utilizar su cuenta, primero deberá confirmar su correo electrónico haciendo clic en el botón a continuación.</p>
<a style='padding:12px;border-radius:12px;background-color:#6181C7;color:#fff;text-decoration:none' href='{1}' target='_blank'>
Confirme su correo electrónico</a>

<p style='margin-top:25px'>Gracias.</p>
</div>
</body>
</html>
```

Luego en nuestro controlador Login, en el constructor vamos a inicializar la clase y la interfaz a utilizar:

```
public class LoginController : Controller
{
    private readonly ApplicationDbContext _appDbContext;
    private readonly IWebHostEnvironment _webHostEnvironment;
    0 referencias
    public LoginController(ApplicationDbContext appDbContext, IWebHostEnvironment webHostEnvironment)
    {
        _appDbContext = appDbContext;
        _webHostEnvironment = webHostEnvironment;
    }
    0 referencias
    public IActionResult Index()
    {
        return View();
    }

    [HttpGet]
    0 referencias
    public IActionResult Registrar()
    {
        return View();
    }
}
```

Procedemos a configurar nuestro método Registrar que será el encargado de guardar en la bd el registro y mandar el correo para la confirmación, mientras “confirmado” no cambie en la bd no dejará ingresar a ningún usuario que no se haya confirmado el correo previamente en el sistema

```

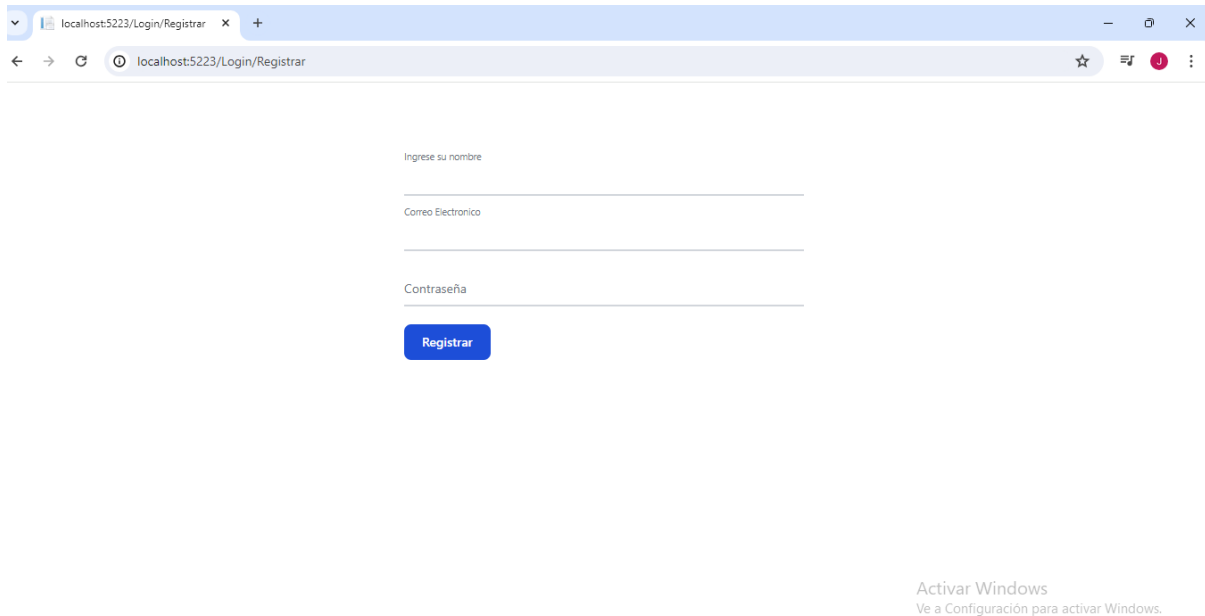
[HttpPost]
0 referencias
public IActionResult Registrar(MUsuario mUsuario)
{
    mUsuario.clave=Utilidad.ConvertirSHA256(mUsuario.clave);
    mUsuario.token = Utilidad.GenerarToken();
    mUsuario.restablecer = false;
    mUsuario.confirmado = false;
    _appDbContext.tbl_Usuario.Add(mUsuario);
    _appDbContext.SaveChanges();

    string path = _webHostEnvironment.WebRootPath;

    string upload = path + SSP.PlantillaPath;
    string content=System.IO.File.ReadAllText(upload);
    string url = string.Format("{0}://{1}{2}", HttpContext.Request.Scheme, Request.Headers["host"], "/Login/Confirmar?token="+mUsuario.token);
    string htmlBody = string.Format(content,mUsuario.nombre,url);
    Correo correo = new Correo()
    {
        Para=mUsuario.correo,
        Asunto= "Correo confirmacion",
        Contenido=htmlBody
    };
    bool enviado = Email.Enviar(correo);
    return RedirectToAction("Index");
}

```

Luego nos vamos a nuestro proyecto y en la url si digitamos /login/Registrar debería cargar el formulario de registro que les quedo de tarea



localhost:5223/Login/Registrar

Ingrese su nombre

Correo Electronico

Contraseña

Registrar

Activar Windows
Ve a Configuración para activar Windows.

Recuerden configurar bien las vistas, usando el model MUsuario y poniendo sus respectivos asp-for:

```
@model MUsuario;
@{
    Layout = null;
}
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>

<div class="mt-[85px]">
    <form class="max-w-md mx-auto" method="post">

        <div class="relative z-0 w-full mb-5 group">
            <input type="text" asp-for="nombre" class="block py-2.5 px-0 w-full text-sm text-gray-900 bg-transparent border-0 border-b-2 border-gra
            <label for="floating_repeat_password" class="peer-focus:font-medium absolute text-sm text-gray-500 dark:text-gray-400 duration-300 tran
        </div>

        <div class="relative z-0 w-full mb-5 group">
            <input type="email" asp-for="correo" class="block py-2.5 px-0 w-full text-sm text-gray-900 bg-transparent border-0 border-b-2 border-gr
            <label for="floating_email" class="peer-focus:font-medium absolute text-sm text-gray-500 dark:text-gray-400 duration-300 transform -tra
        </div>

        <div class="relative z-0 w-full mb-5 group">
            <input type="password" asp-for="clave" class="block py-2.5 px-0 w-full text-sm text-gray-900 bg-transparent border-0 border-b-2 border-
            <label for="floating_password" class="peer-focus:font-medium absolute text-sm text-gray-500 dark:text-gray-400 duration-300 transform -
        </div>

    </form>
</div>
</body>
</html>
```

Una vez todo está correctamente configurado debería llegarles un correo de esta manera:

