



El futuro digital  
es de todos

MinTIC



# Autenticación con ASP .NET Core Identity Laboratorio 01



Universidad de Caldas

# Objetivo

Este workshop tiene el objetivo de aplicar los conceptos sobre autenticación utilizando el framework ASP .NET Core Identity en un escenario común: **un proyecto existente que no está configurado con ningún tipo de autenticación.**



## Descarga y prueba de funcionamiento

El proyecto en el que vamos a trabajar es el de hospitalización en casa (Parcial que trabajamos en el tema maestro-detalle).

Puede descargarlo de

<https://github.com/samurilloretic/UCParking23>

## Pasos

- Como de costumbre, vamos primero a ubicarnos en la carpeta raíz de la solución, en donde aparecen todos los proyectos que hemos creado.

```
C:\MisionTIC-2021-Ciclo3-BD>dir
  El volumen de la unidad C es Windows
  El n mero de serie del volumen es: 0200-6BDE

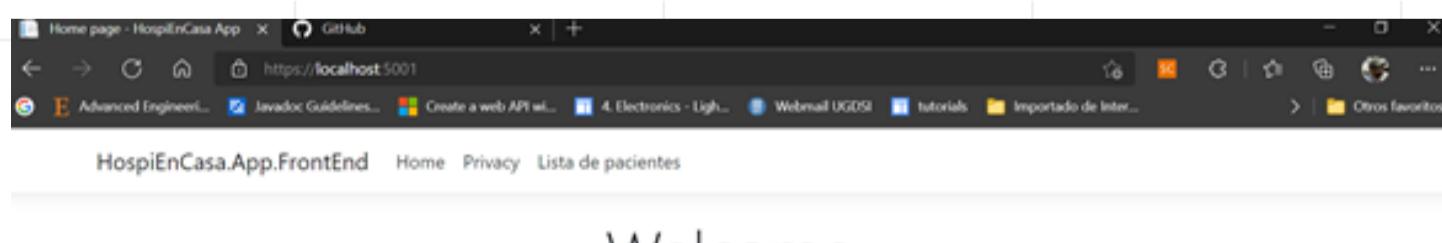
  Directorio de C:\MisionTIC-2021-Ciclo3-BD

  05/10/2021  03:34    <DIR>      .
  05/10/2021  03:34    <DIR>      ..
  30/09/2021  18:36    <DIR>      .vscode
  24/08/2021  17:15    <DIR>      ConsoleApp
  05/10/2021  11:40    <DIR>      HospiEnCasa.App.Consola
  24/08/2021  17:15    <DIR>      HospiEnCasa.App.Dominio
  05/10/2021  10:32    <DIR>      HospiEnCasa.App.FrontEnd
  05/10/2021  08:06    <DIR>      HospiEnCasa.App.Persistence
  24/08/2021  17:15    <DIR>      HospiEnCasa.App.Servicios
  24/08/2021  17:15            522 HospiEnCasa.App.sln
  06/10/2021  09:37            90 README.md
                           2 archivos       612 bytes
                           9 dirs   103.647.948.000 bytes libres

C:\MisionTIC-2021-Ciclo3-BD>
```

**2** Antes de continuar Podemos comprobar que esta funcionando correctamente

```
C:\MisionTIC-2021-Ciclo3-BD\HospiEnCasa.App.FrontEnd>dotnet run
Compilando...
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\MisionTIC-2021-Ciclo3-BD\HospiEnCasa.App.FrontEnd
```



# Scaffolding en ASP .NET Core

En computación el término Scaffolding se refiere a la generación de código relacionado con acceso a las bases de datos de un proyecto tipo web, también hace referencia a la generación de proyectos. En nuestro caso la usaremos para incorporar el Identity Package de asp.net core a nuestro proyecto (que no incluye autenticación). El scaffolder generará la mayoría de código necesario para incorporar la autenticación, sin embargo, es importante entender el proceso ya que como veremos es necesario configurar un poco el código generado automáticamente.

## Instalación del generador de código

- 3 Si no se ha instalado previamente el ASP .NET Core Scaffolder, debe instalarlo:

```
dotnet tool install -g dotnet-aspnet-codegenerator --version="3.1"
```

```
dotnet tool update --global dotnet-aspnet-codegenerator
```

```
C:\MisionTIC-2021-Ciclo3-BD\HospitEnCasa.App.FrontEnd>dotnet tool install -g dotnet-aspnet-codegenerator
Puede invocar la herramienta con el comando siguiente: dotnet-aspnet-codegenerator
La herramienta "dotnet-aspnet-codegenerator" (versión '5.0.2') se instaló correctamente.
```

# Instalación de los paquetes necesarios para trabajar con la Autenticación

- 4 Para trabajar con el ASP .NET Core Identity es necesario adicionar una serie de paquetes al proyecto en donde se va a implementar la autenticación, En nuestro caso UCP.App.FrontEnd. Antes de realizar la instalación verifique en la consola que se encuentra ubicado en el directorio del proyecto indicado:

 Símbolo del sistema

```
C:\MisionTIC-2021-Ciclo3-BD\HospiEnCasa.App.FrontEnd>
```

Adicione uno a uno los siguientes paquetes

Verificar que en el csproj de Frontend este con net5.0

```
dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design  
--version 5.0.2
```

```
dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.9
```

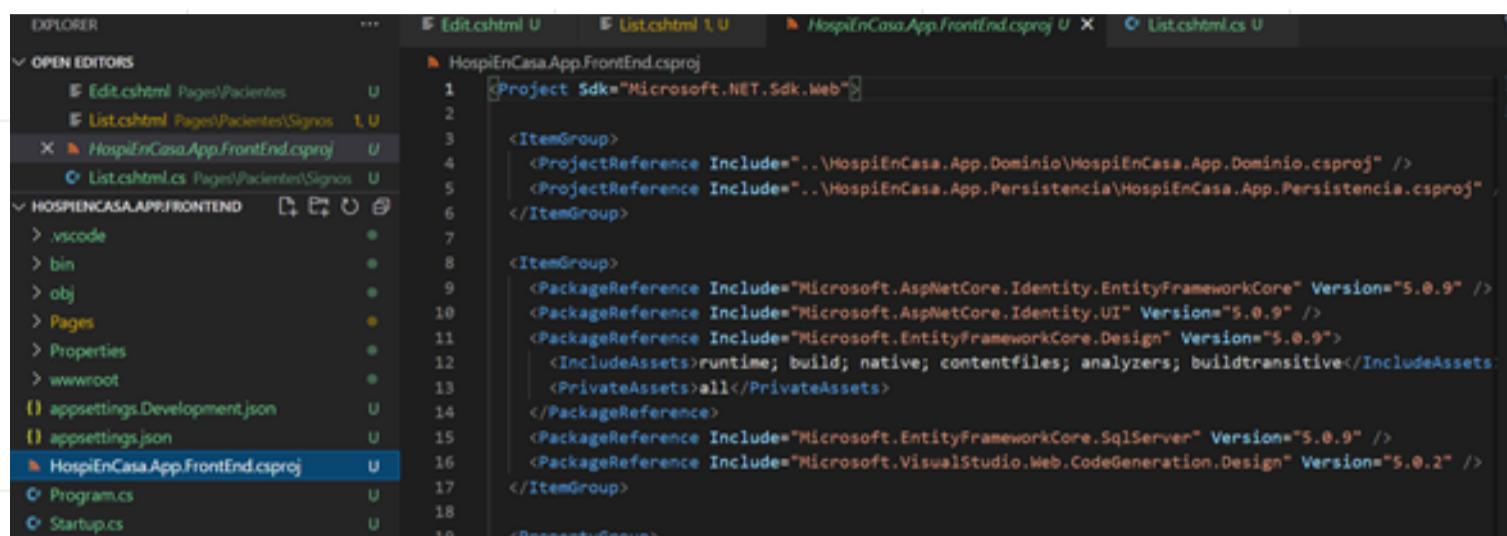
```
dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore  
--version 5.0.9
```

```
dotnet add package Microsoft.AspNetCore.Identity.UI --version 5.0.9
```

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 5.0.9
```

```
8 <PropertyGroup>  
9 | <TargetFramework>net5.0</TargetFramework>  
10 </PropertyGroup>
```

- 5** Puede verificar que efectivamente se han instalado estos paquetes en el proyecto ingresando al Visual Studio code y abriendo el archivo ConfArch.Web.csproj



```
Project Sdk="Microsoft.NET.Sdk.Web"

<ItemGroup>
  <ProjectReference Include="..\HospitEnCasa.App.Dominio\HospitEnCasa.App.Dominio.csproj" />
  <ProjectReference Include="..\HospitEnCasa.App.Persistence\HospitEnCasa.App.Persistence.csproj" />
</ItemGroup>

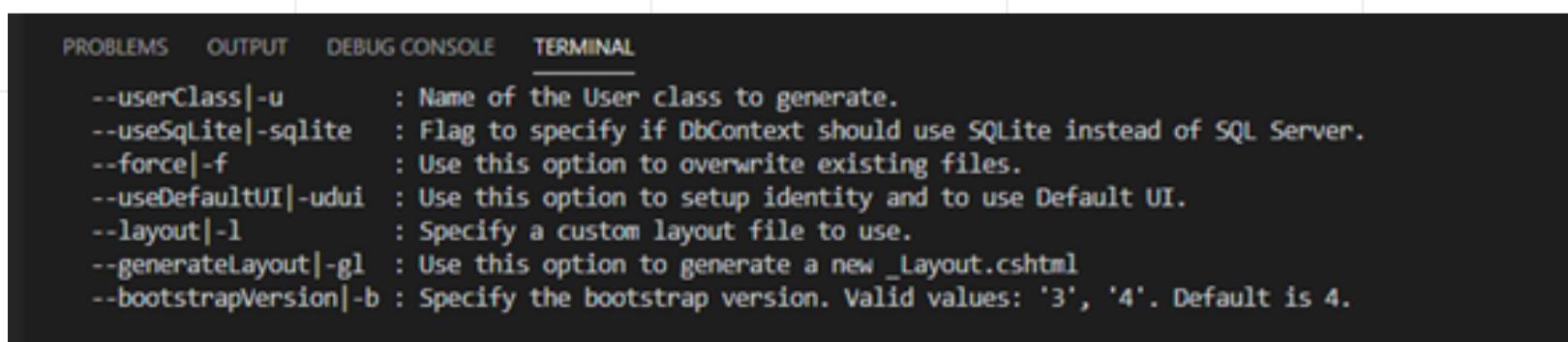
<ItemGroup>
  <PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="5.0.9" />
  <PackageReference Include="Microsoft.AspNetCore.Identity.UI" Version="5.0.9" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="5.0.9" />
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    <PrivateAssets>all</PrivateAssets>
</PackageReference>
  <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="5.0.9" />
  <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="5.0.2" />
</ItemGroup>
```

## Generación de código

- 6** Para confirmar que efectivamente se ha instalado la herramienta que genera el código ejecute el siguiente comando en la consola/terminal. **COMPILAR**

```
dotnet aspnet-codegenerator identity -h
```

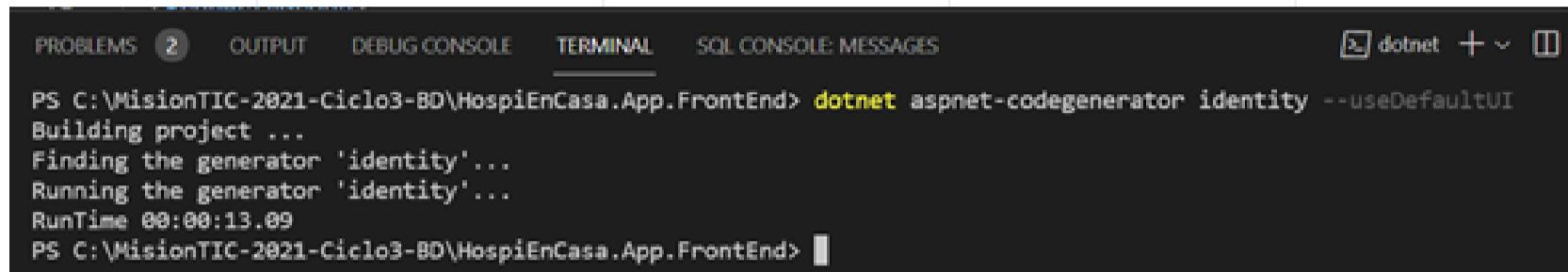
- 7** Si todo está bien deben aparecer las opciones del generador de código de ASP .NET Core para la autenticación por Identity. En particular nos interesa la opción –useDefaultUI ya que permite la generación de código para la autenticación, el cual se adicionará en el proyecto en el que estamos



```
--userClass|-u      : Name of the User class to generate.
--useSQLite|-sqlite : Flag to specify if DbContext should use SQLite instead of SQL Server.
--force|-f          : Use this option to overwrite existing files.
--useDefaultUI|-udui : Use this option to setup identity and to use Default UI.
--layout|-l          : Specify a custom layout file to use.
--generateLayout|-gl : Use this option to generate a new _Layout.cshtml
--bootstrapVersion|-b : Specify the bootstrap version. Valid values: '3', '4'. Default is 4.
```

**8** Ahora si vamos a generar el código y a ahorrarnos una cantidad de trabajo!! ejecute el siguiente comando en la consola CLI

```
dotnet aspnet-codegenerator identity --useDefaultUI
```

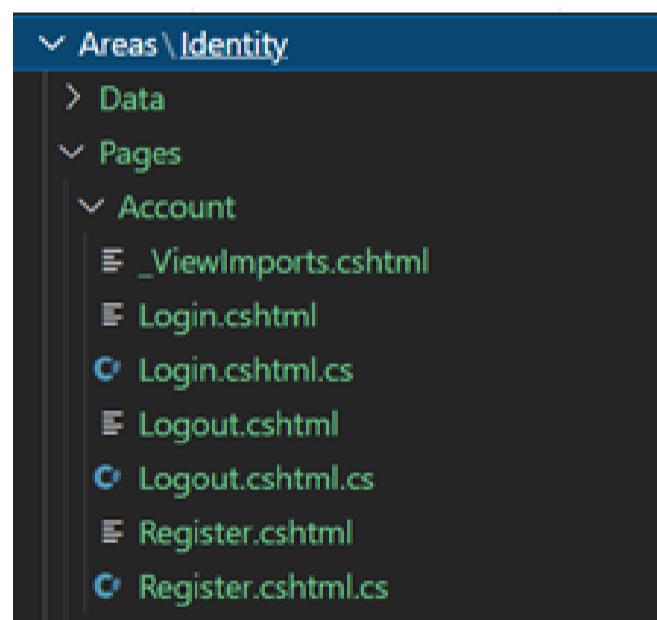


A screenshot of a terminal window titled "dotnet". The window shows the command "dotnet aspnet-codegenerator identity --useDefaultUI" being run. The output of the command is displayed, including "Building project ...", "Finding the generator 'identity'...", "Running the generator 'identity'...", and a runtime of "RunTime 00:00:13.09". The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and SQL CONSOLE: MESSAGES.

**9** Ahora ejecute el siguiente comando que nos adiciona los formularios web (Razor) para el Login, Logout y Register.

```
dotnet aspnet-codegenerator identity --files 'Account.Login;  
Account.Logout;Account.Register' --force
```

**10** Podemos verificar la generación de estas vistas en la carpeta Areas\Identity\Pages



## Verificar la Configuración

**11** Es necesario verificar algunos servicios y configuraciones, entre ellas: recursos estáticos (css, html, js) y el middleware de autenticación deben estarse invocando en el método Configure de la clase Startup.cs

```

public class Startup
{
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        app.UseStaticFiles();
        app.UseRouting();
        app.UseAuthentication();
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Conference}/{action=Index}/{id?}");
        });
    }
}

```

Asegurarse de que queden en este orden.

Asegurarse de que tenga  
endpoints.MapControllerRoute(  
name: "default",  
pattern:  
"{controller=Home}/  
{action=Index}/{id?}");

Agregar  
ReturnUrl = returnUrl??Url.Content("~/");

en el register y el login

**12** También debe verificarse que están configurados los servicios MVC (AddControllersWithViews) y Razor Pages en el método ConfigureServices.

```

public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
    services.AddDbContext<HospitalesEnCasa.App.Persistencia.AppContext>();
    services.AddControllersWithViews();
}

```

Al igual que habilitar los endpoints para las páginas Razor en el método Configure.

```

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Conference}/{action=Index}/{id?}");
    endpoints.MapRazorPages();
})

```

Agregar en el login

```
var result = await _signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: true);
```

```

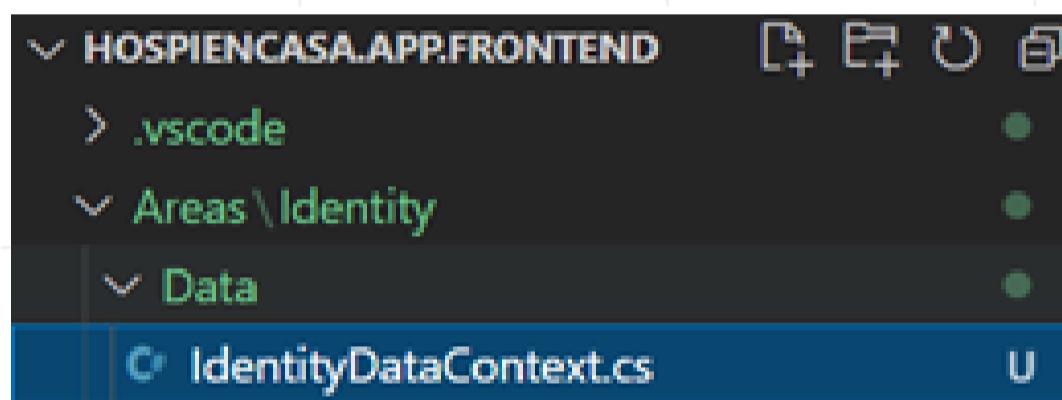
if (ModelState.IsValid)
{
    // This doesn't count login failures towards account lockout
    // To enable password failures to trigger account lockout, set
    //var result = await _signInManager.PasswordSignInAsync(Input.E
    var result = await _signInManager.PasswordSignInAsync(Input.Ema
    if (result.Succeeded)
    {
}

```

# La base de datos para los usuarios

Ahora se debe crear una base de datos para el manejo de usuarios, para ello utilizaremos una migración para el identity framework.

**13** Cuando generamos el código para la autenticación se creó un nuevo DbContext



**14** En el archivo appsettings.json se adicionó una nueva cadena de conexión. Cambie el nombre de la base de como se muestra a continuación (Solo por claridad, pídele dejarse con el nombre que genera):

```
"ConnectionStrings": {  
    "IdentityDataContextConnection": "Server=(localdb)\\mssqllocaldb;Database=HospitEnCasa.App.Data;Trusted_Connection=True;MultipleActiveResultSets=true"  
}
```

**15** Ahora se puede crear la migración, nótese que debido a que el proyecto tiene varios DbContexts es necesario especificar para cual se desea hacer la migración. Ejecute el siguiente comando en la consola.

```
dotnet ef migrations add IdentityInitial --context IdentityDataContext
```

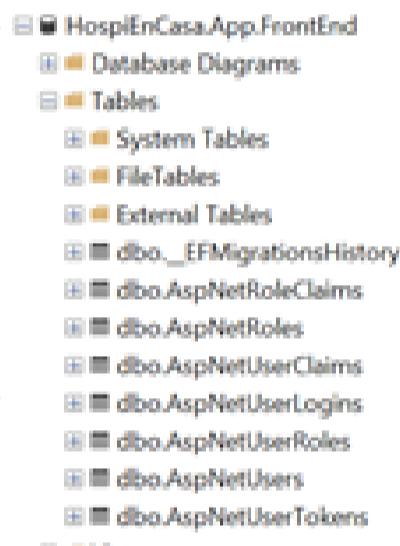
A screenshot of a terminal window titled 'TERMINAL'. The command 'dotnet ef migrations add IdentityInitial --context IdentityDataContext' is typed in and executed. The output shows 'Build started...' and 'Build succeeded.' indicating the migration was created successfully.

**16** Ahora se debe actualizar la base de datos, especificando el contexto

```
dotnet ef database update --context IdentityDataContext
```

```
PS C:\VisionTIC-2021-Ciclo3-BD\HospiEnCasa.App.FrontEnd> dotnet ef database update --context IdentityDataContext
Build started...
Build succeeded.
```

**17** Puede verificar la generación de la base de datos y sus tablas.



## Startup para el Identity

Cuando se hace la generación del código para la autenticación por Identity se genera una clase `IdentityHostingStartup` que tiene un funcionamiento similar al de la clase `Setup` que ya hemos visto y se ejecuta con la aplicación.

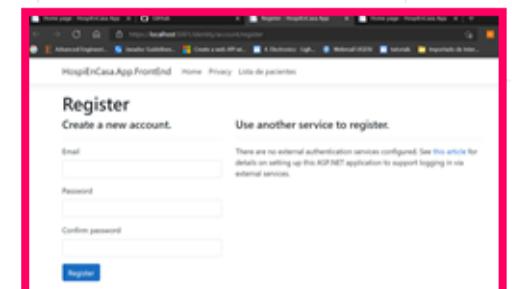
A screenshot of a code editor showing the `IdentityHostingStartup.cs` file. The file is located in the `HOSPIENCASAAPP\FRONTEND\Areas\Identity` folder. The code uses the `IdentityHostingStartup` template from the .NET Core Identity scaffolding. It includes imports for System, `HospiEnCasa.App.FrontEnd.Areas.Identity.Data`, `Microsoft.AspNetCore.Hosting`, `Microsoft.AspNetCore.Identity`, `Microsoft.AspNetCore.Identity.UI`, `Microsoft.EntityFrameworkCore`, `Microsoft.Extensions.Configuration`, and `Microsoft.Extensions.DependencyInjection`. The class itself is annotated with `[assembly: HostingStartup(typeof(HospiEnCasa.App.FrontEnd.Areas.Identity.IdentityHostingStartup))]` and has a `namespace HospiEnCasa.App.FrontEnd.Areas.Identity` declaration.

Aunque las configuraciones que se hacen en esta clase tambien pueden ir en el archivo `Setup.cs` es una buena práctica mantenerlas separadas.

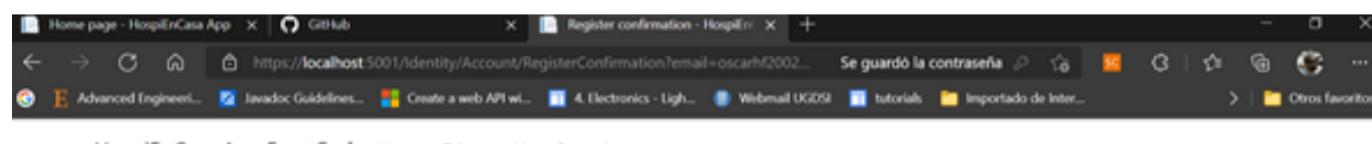
# Prueba inicial

Ejecute el proyecto y pruebe navegar a la opción de Register  
<https://localhost:5001/identity/account/register>

Y se despliega el formulario de registro de usuarios



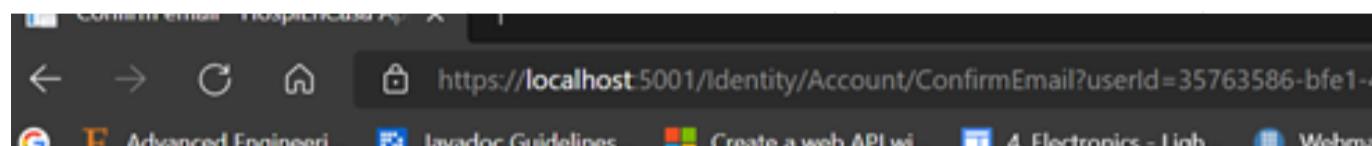
**19** Ahora por fin pruebe la aplicación (regístrese y luego autentíquese) verifique lo que sucede en las tablas de la base de datos. (¡¡Recuerde que estamos solo autenticando!!).



## Register confirmation

This app does not currently have a real email sender registered, see [these docs](#) for how to configure a real email sender. Normally this would be emailed:  
[Click here to confirm your account](#)

Como no tenemos servicio de email, vamos a simular la confirmación del registro dando click donde lo solicita



HospiEnCasa.App.FrontEnd Home Privacy Lista de pacientes

## Confirm email



**Mision  
TIC 2022**

The logo features the text "Mision TIC 2022" in a bold, sans-serif font. The word "Mision" is in blue, "TIC" is in red, and "2022" is in blue. A red curved line starts from the top of the letter "i" in "Mision" and ends at the bottom of the letter "c" in "2022". The background of the logo is a white circle with a gray halftone pattern.

Universidad de Caldas