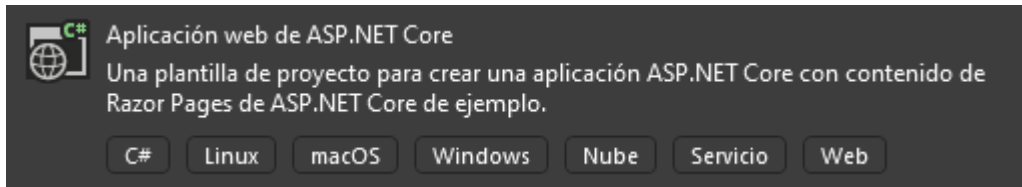


## Implementación:

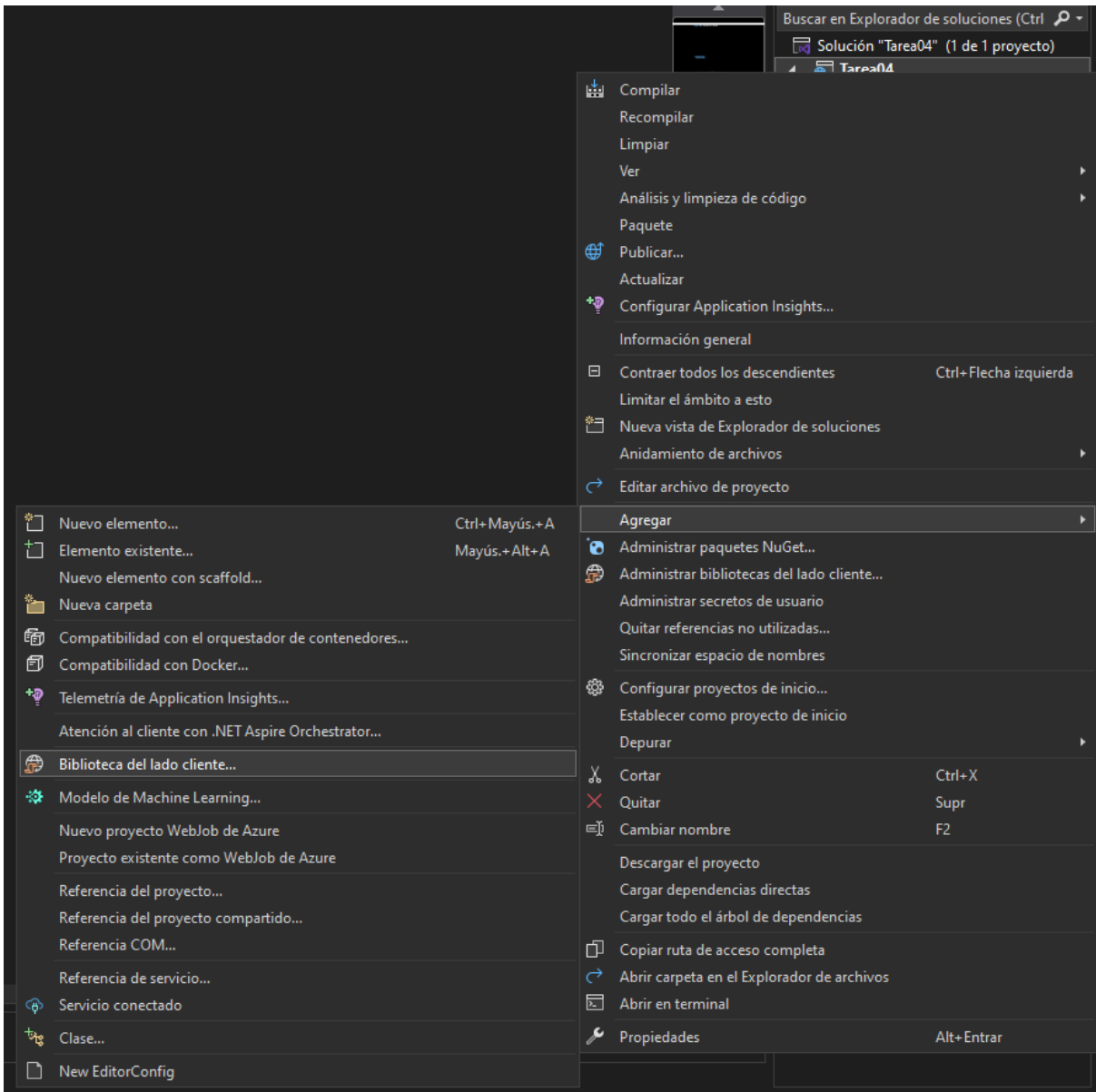
Paso 1: Crear un proyecto utilizando la plantilla correspondiente.

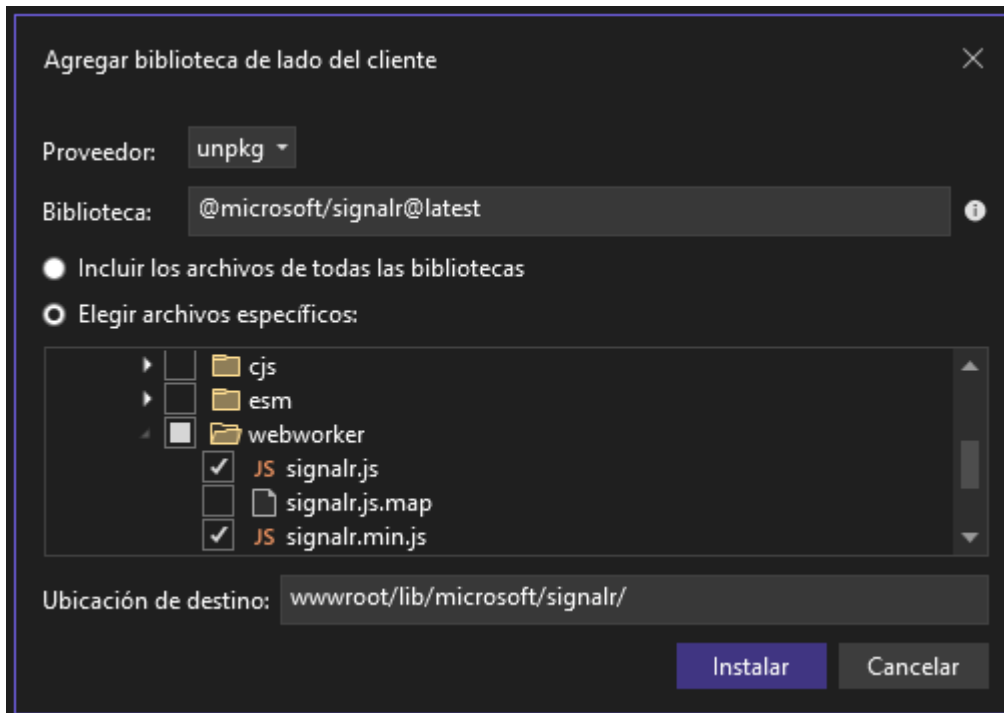


Paso 2: Crear una página Razor para el login

```
Login.cshtml PaginaBienvenida.cshtml
1 @page "/"
2 <!-- notar como la barra determina que esta página sea la pagina por defecto -->
3 @model Tarea04.Pages.LoginModel
4 @{
5     Layout = null;
6 }
7
8 <!-- El siguiente código fue generado por v0
9     https://v0.dev/chat/m0uKEDV40rS
10 -->
11 <!-- del lado del cliente necesito la librería de signalr -->
12 <script src="~/lib/microsoft/signalr/dist/webworker/signalr.js"></script>
13
14 <head>
15     <meta charset="UTF-8">
16     <meta name="viewport" content="width=device-width, initial-scale=1.0">
17     <title>Login Form</title>
18     <style>
19     <body {
20         font-family: Arial, sans-serif;
21         display: flex;
22         justify-content: center;
23         align-items: center;
24         height: 100vh;
25         margin: 0;
26         background-color: #f0f0f0;
27     }
28     .login-form {
29         background-color: white;
30         padding: 2rem;
31         border-radius: 8px;
32         box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
33         width: 300px;
34     }
35     h2 {
36         text-align: center;
37         margin-bottom: 1.5rem;
38     }
39     .form-group {
40         margin-bottom: 1rem;
41     }
42     label {
43         display: block;
44         margin-bottom: 0.5rem;
45     }
46     input {
47         width: 100%;
48         padding: 0.5rem;
49         border: 1px solid #ccc;
50     }
51     .btn {
52         width: 100%;
53         padding: 0.5rem;
54         background-color: #007bff;
55         color: white;
56         border: none;
57     }
58     .error {
59         color: red;
60         margin-top: 10px;
61     }
62     }
63 </style>
64 </head>
65 <div>
66     <h2>Login</h2>
67     <div>
68         <input type="text" value="" />
69     </div>
70     <div>
71         <input type="password" value="" />
72     </div>
73     <button type="submit">Login</button>
74 </div>
75 </body>
76 </html>
```

Paso 3: Agregar la biblioteca de signalr:

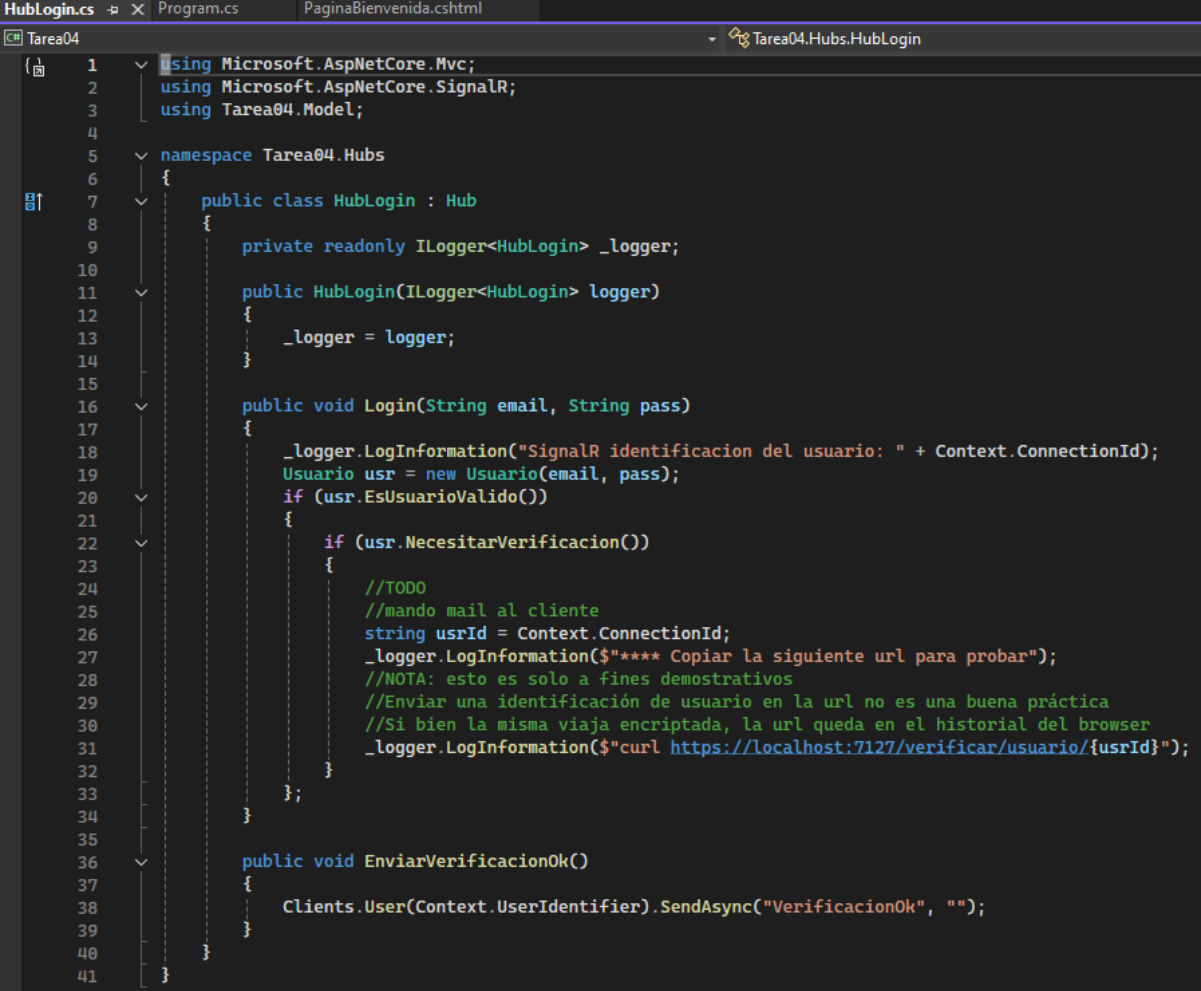




Modificamos el program.cs para que conozca la librería.

```
1  using Microsoft.AspNetCore.SignalR;  
2  using Tarea04.Hubs;  
3  
4  var builder = WebApplication.CreateBuilder(args);  
5  
6  // Add services to the container.  
7  builder.Services.AddRazorPages();  
8  builder.Services.AddSignalR();
```

Paso 4: Creamos e implementamos la clase HubLogin.cs en el directorio Hub que también debemos crear previamente.



```
1 using Microsoft.AspNetCore.Mvc;
2 using Microsoft.AspNetCore.SignalR;
3 using Tarea04.Model;
4
5 namespace Tarea04.Hubs
6 {
7     public class HubLogin : Hub
8     {
9         private readonly ILogger<HubLogin> _logger;
10
11         public HubLogin(ILogger<HubLogin> logger)
12         {
13             _logger = logger;
14         }
15
16         public void Login(String email, String pass)
17         {
18             _logger.LogInformation("SignalR identificacion del usuario: " + Context.ConnectionId);
19             Usuario usr = new Usuario(email, pass);
20             if (usr.EsUsuarioValido())
21             {
22                 if (usr.NecesitarVerificacion())
23                 {
24                     //TODO
25                     //mando mail al cliente
26                     string usrId = Context.ConnectionId;
27                     _logger.LogInformation($"**** Copiar la siguiente url para probar");
28                     //NOTA: esto es solo a fines demostrativos
29                     //Enviar una identificación de usuario en la url no es una buena práctica
30                     //Si bien la misma viaja encriptada, la url queda en el historial del browser
31                     _logger.LogInformation($"curl https://localhost:7127/verificar/usuario/{usrId}");
32                 }
33             }
34         }
35
36         public void EnviarVerificacionOk()
37         {
38             Clients.User(Context.UserId).SendAsync("VerificacionOk", "");
39         }
40     }
41 }
```

Notar que con la sentencia “\_logger.LogInformation(“SignalR identificacion del usuario: ” + Context.ConnectionId)” se genera un identificador para cada instancia existente del login por lo que no habrá 2 clientes con el mismo hash.

En la sentencia :

“\_logger.LogInformation(\$"curlhttps://localhost:7127/verificar/usuario/{usrId}")”;

El puerto debe ser el mismo con el que se levanta el servidor al ejecutar la aplicación.

De igual forma se debe observar lo mismo en el archivo de Login.

Paso 5: Creamos e implementamos la clase Usuario.cs en el directorio Models que debemos crear previamente.

```
1  namespace Tarea04.Model
2  {
3      public class Usuario
4      {
5          public String Email { get; set; }
6          public String Password { get; set; }
7
8
9          public Usuario(String email, String pass)
10         {
11             this.Email = email;
12             this.Password = pass;
13         }
14
15         public Boolean EsUsuarioValido()
16         {
17             //TODO implementar lógica de autenticacion
18             return true;
19         }
20
21         public Boolean NecesitarVerificacion()
22         {
23             //TODO implmentar lógica de verificación
24             return true;
25         }
26     }
27 }
28
```

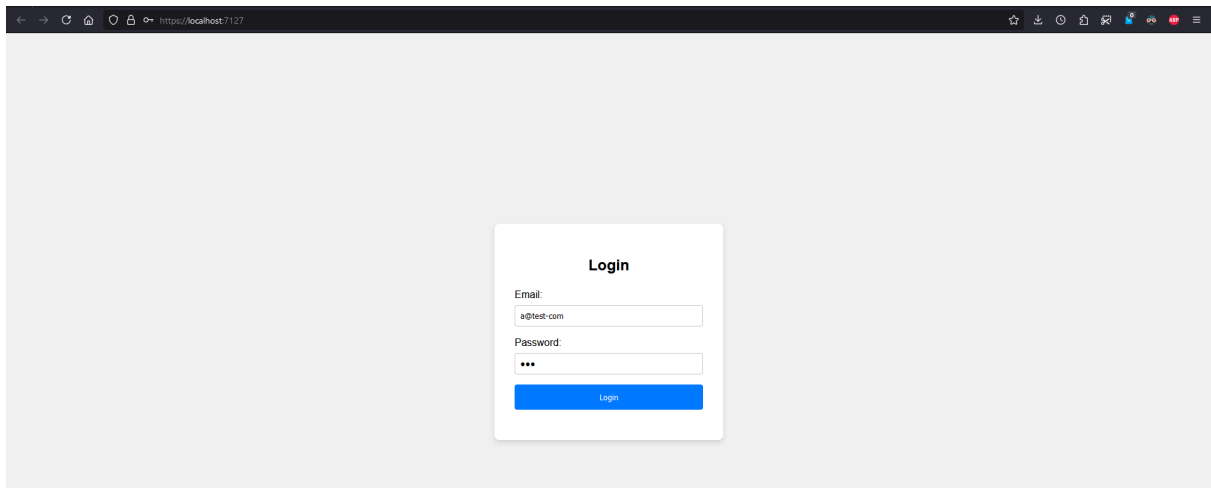
## Paso 6: Modificamos el program.cs para que pueda utilizar el Hub

```
1  using Microsoft.AspNetCore.SignalR;
2  using Tarea04.Hubs;
3
4  var builder = WebApplication.CreateBuilder(args);
5
6  // Add services to the container.
7  builder.Services.AddRazorPages();
8  builder.Services.AddSignalR();
9
10 var app = builder.Build();
11
12 // Configure the HTTP request pipeline.
13 if (!app.Environment.IsDevelopment())
14 {
15     app.UseExceptionHandler("/Error");
16     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
17     app.UseHsts();
18 }
19
20 app.UseHttpsRedirection();
21 app.UseStaticFiles();
22
23 app.UseRouting();
24
25 app.UseAuthorization();
26
27 app.MapRazorPages();
28 app.MapHub<HubLogin>("/login");
29
30 app.MapGet("/verificar/usuario/{userId}", (string userId,
31     ILogger<HubLogin> logger,
32     IHttpContext<HubLogin> hubContext) => {
33
34     logger.LogInformation($"Se notificara al cliente con id {userId}");
35     // Esta url la expongo para recibir los request que se disparan cuando el usuario hace click en el
36     //link que se envia por mail
37     hubContext.Clients.Client(userId).SendAsync("VerificacionOk", userId);
38 });
39
40 app.Run();
41
```

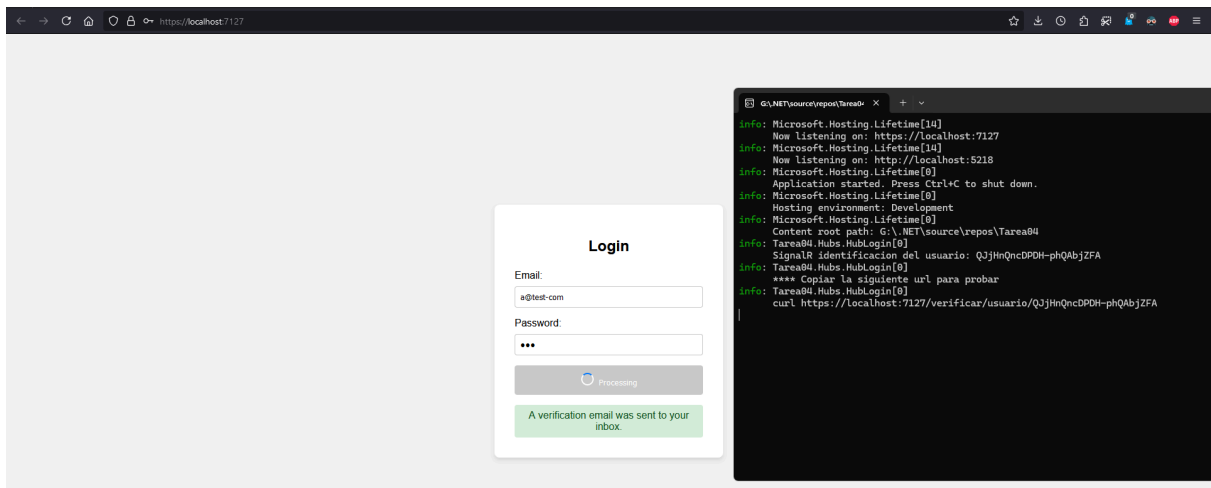
Paso 7: Creamos e implementamos la página de bienvenida para redireccionar luego del login.

```
PaginaBienvenida.cshtml*  X
1  @page
2  @model Tarea04.Pages.PaginaBienvenidaModel
3  @{
4      Layout = null;
5  }
6  <head>
7      <meta charset="UTF-8">
8      <meta name="viewport" content="width=device-width, initial-scale=1.0">
9      <title>Welcome Page</title>
10     <style>
11         body {
12             font-family: Arial, sans-serif;
13             background-color: #f0f0f0;
14             margin: 0;
15             padding: 0;
16             display: flex;
17             justify-content: center;
18             align-items: center;
19             height: 100vh;
20         }
21
22         .welcome-container {
23             background-color: white;
24             border-radius: 8px;
25             box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
26             padding: 40px;
27             text-align: center;
28         }
29
30         h1 {
31             color: #333;
32             margin-bottom: 10px;
33         }
34
35         .user-name {
36             color: #4a90e2;
37             font-size: 24px;
38             font-weight: bold;
39         }
40
41         p {
42             color: #666;
43             margin-top: 20px;
44         }
45     </style>
46 </head>
47 <body>
48     <div class="welcome-container">
49         <h1>Bienvenido</h1>
50         <div class="user-name">John Doe</div>
```

## Probamos la interacción de SignalR:



Se verifica que los datos se están enviando





## La petición retorna “OK”

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

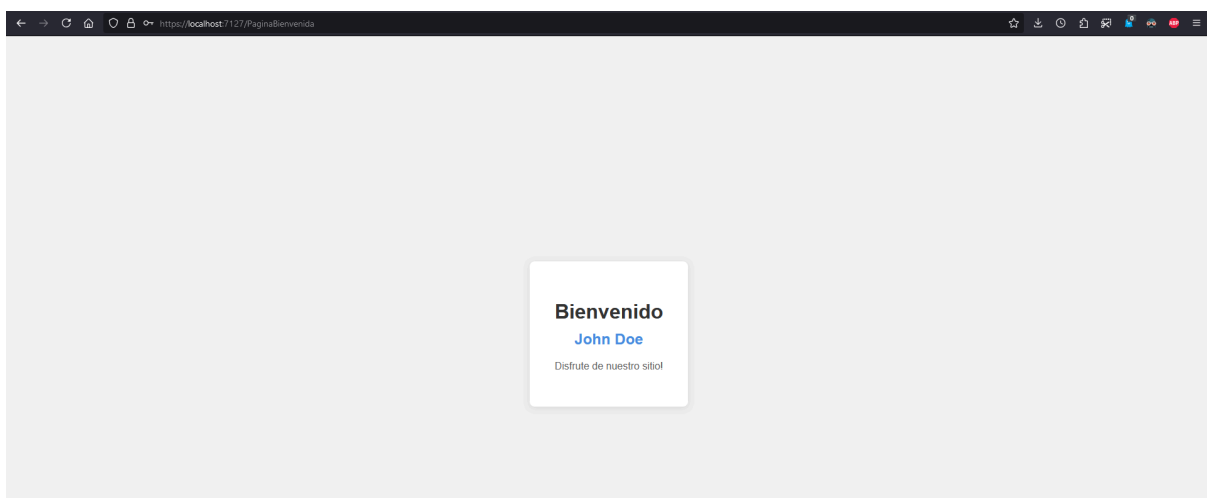
PS C:\Users\sky> curl https://localhost:7127/verificar/usuario/QJjHnQncDPDH-phQAbjZFA

StatusCode      : 200
StatusDescription : OK
Content         : {}
RawContent      : HTTP/1.1 200 OK
                  Content-Length: 0
                  Date: Mon, 09 Sep 2024 16:05:10 GMT
                  Server: Kestrel

Headers         : {[Content-Length, 0], [Date, Mon, 09 Sep 2024 16:05:10 GMT], [Server, Kestrel]}
RawContentLength : 0

PS C:\Users\sky> |
```

Luego de ejecutar la petición anterior se redirige a la página de bienvenida



## Repositorio de GitHub:

[https://github.com/AndresRomano/DotNET\\_Pr-cticos/tree/main/Tarea04](https://github.com/AndresRomano/DotNET_Pr-cticos/tree/main/Tarea04)

Nota: la mayor parte de los códigos fueron proporcionados por el docente en el repositorio:

[https://github.com/gabrielaramburu/TallerNET/tree/main/01\\_04\\_1\\_SignalREjemploLogin](https://github.com/gabrielaramburu/TallerNET/tree/main/01_04_1_SignalREjemploLogin)

posteriormente fueron adaptados para funcionar en el entorno de pruebas.