

Práctica. Frontend en .NET con autenticación.

Instrucciones. Elabore una aplicación web de .NET utilizando el patrón de diseño de software MVC con controladores, vistas y modelos que utilice el método de autenticación Cookies y se comunique con un backend API utilizando JWT.

1. Sigue las instrucciones de la práctica en la siguiente página.
2. Crea un reporte de práctica en un **archivo de Word** con una portada con tu nombre. Guárdalo con el nombre **NombreAlumno_FrontendNetAuth.docx**.
3. Coloca **las siguientes capturas de pantalla**, cada una con una **descripción textual** arriba de la captura.
 - a. Coloca las capturas de pantalla de tu código fuente de tus archivos donde se agrega la funcionalidad de la aplicación como controladores, modelos, vistas, middleware y servicios.
 - b. Coloca la captura de pantalla donde se vea la página de inicio de sesión.
 - c. Coloca la captura de pantalla donde se vea la lista de categorías.
 - d. Coloca la captura de pantalla donde se vea la lista de películas.
 - e. Coloca la captura de pantalla donde se vea la lista de usuarios.
 - f. Coloca las capturas de pantalla donde se vea la edición de películas, categorías y usuarios.
 - g. Publica tu código fuente en un proyecto público de GitHub. **Coloca la URL** del código fuente publicado en GitHub.
4. Sube tu reporte de práctica archivo Word a la actividad en Eminus.

Prerrequisitos

1. **Esta práctica tiene como prerequisito haber realizado las prácticas anteriores de Backend con seguridad.**
2. Esta práctica tiene como prerequisito tener instalado el software **Visual Studio Code** con la extensión para C# y la extensión REST Client.
Puede seguir las instrucciones de cómo instalar Visual Studio Code desde la práctica [Instalación de Visual Studio Code](#).
3. Esta práctica tiene como prerequisito tener instalado tener instalado el SDK de .NET descárguelo desde <https://dotnet.microsoft.com/en-us/download/dotnet/8.0>.
4. Esta práctica tiene como prerequisito tener instalado MySQL en su equipo.
Puede seguir las instrucciones de cómo instalar MySQL desde la práctica [Instalación de MySQL](#).
5. Puede seguir las instrucciones de cómo publicar un proyecto en Visual Studio Code a GitHub desde la práctica [Como publicar un proyecto a GitHub desde Visual Studio Code](#).

Instrucciones. Defina su Frontend.

Lo primero que tenemos que realizar, es definir las páginas de su frontend. no comience a escribir código y crear estructuras sin saber lo que se quiere realizar. Esta práctica crea la siguiente aplicación:

URL	Descripción	Rol requerido
GET /	Inicio de la aplicación	Ninguno
GET /home	Inicio de la aplicación	Ninguno
GET /auth	Inicio de sesión de usuario	Ninguno
GET /peliculas	Obtener todas las películas	Usuario, Administrador
GET /peliculas?{s=titulo}	Obtener todas las películas que contengan en el título la cadena s	Usuario, Administrador
GET /peliculas/detalle/{id}	Detalles de una película por ID	Usuario, Administrador
GET /peliculas/crear	Agregar una nueva película	Administrador
GET /peliculas/editar/{id}	Actualizar una película existente	Administrador
GET /peliculas/eliminar/{id}	Eliminar una película	Administrador
GET /categorias	Obtener todas las categorías	Administrador
GET /categorias/detalle/{id}	Obtener un categoría por ID	Administrador
GET /categorias/crear	Agregar una nueva categoría	Administrador
GET /categorias/editar/{id}	Actualizar una categoría existente	Administrador
GET /categorias/eliminar/{id}	Eliminar una categoría	Administrador
GET /usuarios	Obtener todas los usuarios	Administrador
GET /usuarios/detalle/{id}	Obtener un usuario por ID	Administrador
GET /usuarios/crear	Agregar un nuevo usuario	Administrador
GET /usuarios/editar/{id}	Actualizar un usuario existente	Administrador
GET /usuarios/eliminar/{id}	Eliminar un usuario	Administrador

Es momento de comenzar la construcción de la aplicación.

¿Qué es la autenticación basada en cookies?

La autenticación basada en cookies ha sido el método predeterminado (y comprobado) para manejar la autenticación de usuarios durante mucho tiempo.

La autenticación basada en cookies presenta un estado (es stateful).

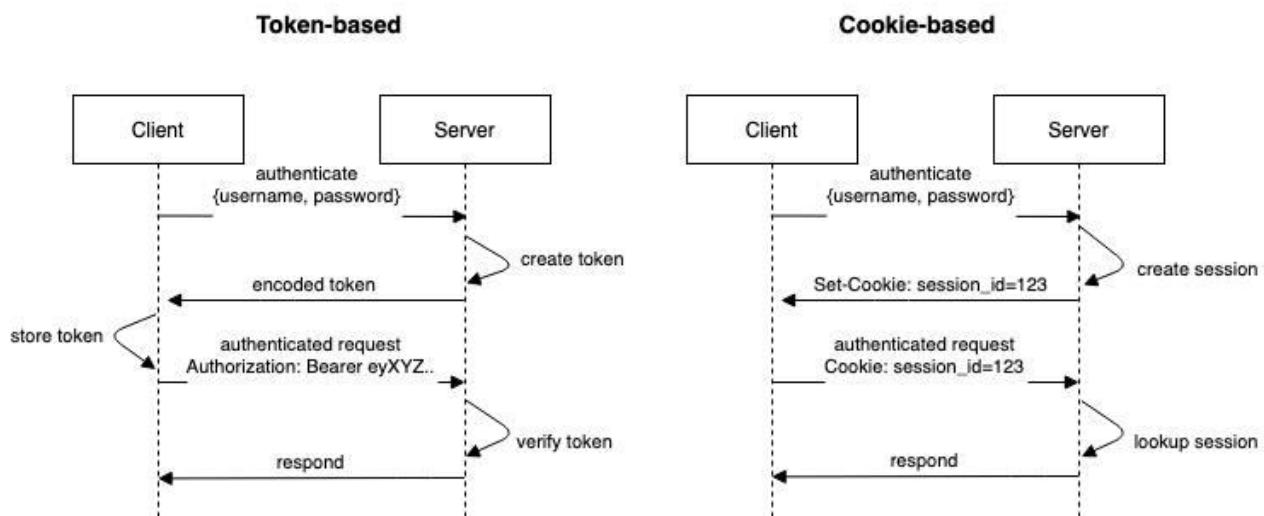
Al iniciar sesión, luego que un usuario envía sus credenciales (y estas se validan), el servidor registra datos (con el fin de recordar que el usuario se ha identificado correctamente). *Estos datos que se registran en el backend*, en correspondencia con el identificador de sesión, es lo que se conoce como *estado*.

En el lado del cliente una cookie es creada para almacenar el identificador de sesión, mientras que los datos se almacenan en el servidor (y son llamados variables de sesión).

¿Cómo funciona la autenticación basada en cookies?

El flujo que sigue este sistema de autenticación tradicional es el siguiente:

- Un usuario ingresa sus credenciales (datos que le permiten iniciar sesión).
- El servidor verifica que las credenciales sean correctas, y crea una sesión (esto puede corresponderse con la creación de un archivo, un registro nuevo en una base de datos, o alguna otra solución server-side).
- Una cookie con el `session ID` es puesta en el navegador web del usuario.
- En las peticiones siguientes, el `session ID` es comparado con las sesiones creadas por el servidor.
- Una vez que el usuario se desconecta, la sesión es destruida en ambos lados (tanto en el cliente como en el servidor).



Instrucciones: Creación de la aplicación.

1. Asegúrese de contar con una carpeta de trabajo en **C:\codigo**. Si aún no ha creado la carpeta **C:\codigo** hágalo antes de continuar.
2. Seleccione **Archivo > Abrir carpeta** en el menú principal.
3. En el cuadro de diálogo **Abrir carpeta**, cree una carpeta en la ruta **C:\codigo\tw\frontendnet** y selecciónela. Luego haga clic en **Seleccionar carpeta**.
4. El nombre de la carpeta se convierte en el nombre del proyecto y el nombre del espacio de nombres de forma predeterminada.
5. Abra una Terminal de consola presionando **Ctrl + ñ**.
6. Verifique que tenga instalado el SDK de .NET para crear aplicaciones.

```
dotnet --version  
8.0.200
```

7. Si recibe un mensaje de comando no reconocido, es necesario que descargue e instale el SDK de .NET desde <https://dotnet.microsoft.com/en-us/download/dotnet/8.0>
8. Ejecute el comando siguiente para crear su proyecto web: **dotnet new web**.

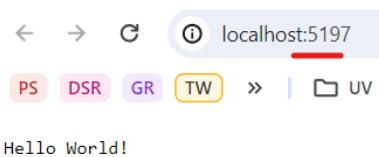
```
PS C:\codigo\tw\frontendnet> dotnet new web  
La plantilla "ASP.NET Core vacío" se creó correctamente.
```

```
Procesando acciones posteriores a la creación...  
Restaurando C:\codigo\tw\frontendnet\frontendnet.csproj:  
  Determinando los proyectos que se van a restaurar...  
  Se ha restaurado C:\codigo\tw\frontendnet\frontendnet.csproj (en 103 ms).  
Restauración realizada correctamente.
```

9. Esto creará un **proyecto web vacío**. Existe también una plantilla disponible que ya genera un proyecto [MVC completo](#), pero en este caso, vamos a hacerlo nosotros desde cero.
10. Su web app esta lista para ser ejecutada. En la consola escriba **dotnet build**. Y verifique que no marque errores.
11. Escriba **dotnet run** o **dotnet watch run** para ejecutar su aplicación. La diferencia con ambos, es que la segunda aplica los cambios al guardar el archivo de código fuente modificado sin necesidad de reiniciar todo el servidor web de desarrollo.

```
PS C:\codigo\tw\frontendnet> dotnet run  
Compilando...  
[info]: Microsoft.Hosting.Lifetime[14]  
  Now listening on: http://localhost:5197  
[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:\codigo\tw\frontendnet
```

12. Sino se abre una ventana de su navegador, presione **ctrl + clic** sobre la dirección **http://localhost:puerto** o copie y péguela para abrirla en Chrome y verifique que su sitio pueda verse.



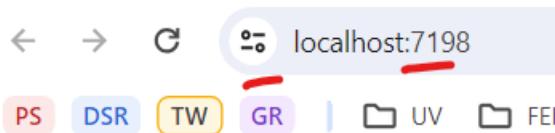
13. Observe como la aplicación se ejecuta en un puerto distinto al estándar 80 (HTTP) o al 443 (HTTPS).
14. En caso de que quisiéramos ejecutar la aplicación con HTTPS, escriba `dotnet dev-certs https --trust` para confiar en el certificado de desarrollo HTTPS local.



15. En caso de que se le solicite, acepte el certificado SSL auto firmado.
16. Para utilizar SSL, utilice `dotnet run --launch-profile https` o `dotnet watch run --launch-profile https` según corresponda.

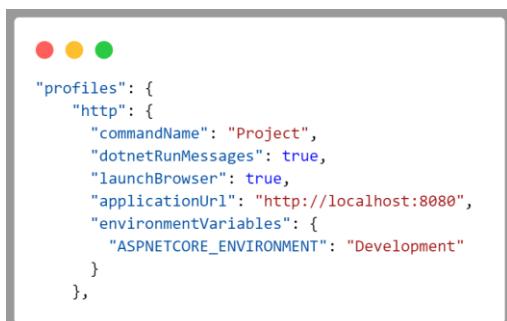
```
PS C:\codigo\tw\mvc> dotnet run --launch-profile https
Compilando...
[info]: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:7198
[info]: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5101
[info]: Microsoft.Hosting.Lifetime[0]
  Application started. Press Ctrl+C to shut down.
[info]: Microsoft.Hosting.Lifetime[0]
  Hosting environment: Development
[info]: Microsoft.Hosting.Lifetime[0]
  Content root path: C:\codigo\tw\mvc
```

17. Observe como se sirve el contenido con un certificado SSL.



Hello World!

18. Para terminar la ejecución escriba en la consola `Ctrl + Q` en Visual Studio Code.
19. Para cambiar el puerto en el que se ejecutará su aplicación, abra el archivo `/Properties/launchSettings.json`.
20. Los perfiles disponibles cuando usted ejecuta su aplicación en el servidor de desarrollo son `http`, `https` y si lo tuviera instalado en Windows, `IIS Express`.
21. Cambie el puerto del perfil default `http` por `8080`.



22. Ejecute su aplicación usando `dotnet run` o `dotnet watch run` y ahora puede acceder a la ventana de su navegador en el nuevo puerto.

```
PS C:\codigo\tw\frontendnet> dotnet run
Compilando...
[info]: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:8080
[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]
```

23. Felicidades, ha creado su aplicación de manera correcta.

Instrucciones: Configuración inicial.

1. Primeramente vamos a obtener una lista de librerías de cliente necesarias para el proyecto. Se obtendrán desde redes de distribución de contenidos y se muestran a continuación.

Librería de cliente	URL
jquery	https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js
jquery-validate	https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.20.0/jquery.validate.min.js
jquery-validation-unobtrusive	https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-unobtrusive/4.0.0/jquery.validate.unobtrusive.min.js
bootstrap	https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.3/js/bootstrap.bundle.min.js https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.3/css/bootstrap.min.css
bootstrap-icons	https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css

2. En la barra de herramientas superior del **Explorador de archivos**, seleccione **Nueva carpeta** y cree las siguientes carpetas:
 - **Controllers**. Aquí van los controladores.
 - **Middlewares**. Aquí van los middlewares.
 - **Models**. Aquí almacenaremos los modelos.
 - **Services**. Aquí van los servicios.
 - **Views**. Aquí van las vistas. Debemos crear subcarpetas con el nombre de cada controlador.
 - **Views/Archivos**. Aquí van las vistas para el controlador Archivos.
 - **Views/Auth**. Aquí van las vistas para el controlador Auth.
 - **Views/Bitacora**. Aquí van las vistas para el controlador Bitacora.
 - **Views/Categorías**. Aquí van las vistas para el controlador Categorías.
 - **Views/Home**. Aquí van las vistas para el controlador Home.
 - **Views/Películas**. Aquí van las vistas para el controlador Películas.
 - **Views/Perfil**. Aquí van las vistas para el controlador Perfil.
 - **Views/Shared**. Aquí van las vistas compartidas.
 - **Views/Usuarios**. Aquí van las vistas para el controlador Usuarios.
 - **wwwroot**. Aquí va el contenido estático.
 - **wwwroot/css/**. Aquí vas las hojas de estilo.
 - **wwwroot/images/**. Aquí van las imágenes.
 - **wwwroot/js/**. Aquí van los scripts js.
3. Descargue [el código](#) para cambiar el tema del sitio y colóquelo en la carpeta `wwwroot/js/cambio_tema.js`.
4. Descargue el logo de su aplicación que se verá en el menú del sitio desde [logo.png](#) y colóquelo en la carpeta `wwwroot/images/logo.png`.
5. Descargue el fondo de su aplicación desde [fondo.jpg](#) y colóquelo en la carpeta `wwwroot/images/fondo.jpg`.
6. Descargue el icono de su aplicación que se ve en la pestaña del navegador desde [favicon.ico](#) y colóquelo en la carpeta `wwwroot/favicon.ico`.
7. En la carpeta `wwwroot/css/` agregue un nuevo archivo llamado `site.css` con el siguiente código.



```
html[data-bs-theme="dark"] body {  
    background: linear-gradient(rgba(0, 0, 0, 0.85), rgba(0, 0, 0, 0.85)), url('/images/fondo.jpg');  
}
```

8. En la carpeta `wwwroot/css/` agregue un nuevo archivo llamado `guest.css` con el siguiente código.



```
html[data-bs-theme="dark"] body {  
    background: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.7)), url('/images/fondo.jpg');  
}  
  
html[data-bs-theme="dark"] .card {  
    background-color: rgba(0, 0, 0, 0.5);  
}
```

9. Abra su archivo `appsettings.json` y agregue la conexión hacia su backend API. En este ejemplo es la siguiente, pero puede variar en su equipo.



```
},  
    "AllowedHosts": "*",  
    "URLWebAPI": "http://localhost:3000"  
}
```

10. Enhorabuena. Ha configurado correctamente su aplicación.

Instrucciones. Agregar los modelos con validación de entrada de usuario.

En .NET para realizar la validación de entrada del usuario, se puede utilizar los atributos de validación de modelos por medio de la clase `System.ComponentModel.DataAnnotations`. Esto ofrece la funcionalidad de validar la entrada tanto desde código de dinámico como desde la `Vista` utilizando **JavaScript** por medio de librería `jquery-validate` que forma parte del paquete que incluye .NET en sus plantillas para frontend.

1. En la carpeta `Models` agregue un nuevo archivo llamado `AuthUser.cs` con el siguiente código.

```
● ● ●  
namespace frontendnet.Models;  
  
public class AuthUser  
{  
    public required string Email { get; set; }  
    public required string Nombre { get; set; }  
    public required string Rol { get; set; }  
    public required string Jwt { get; set; }  
}
```

2. En la carpeta `Models` agregue un nuevo archivo llamado `Categoría.cs` con el siguiente código.

```
● ● ●  
using System.ComponentModel.DataAnnotations;  
  
namespace frontendnet.Models;  
  
public class Categoría  
{  
    [Display(Name = "Id")]  
    public int? CategoríaId { get; set; }  
  
    [Required(ErrorMessage = "El campo {0} es obligatorio.")]  
    public required string Nombre { get; set; }  
  
    [Display(Name = "Eliminable")]  
    public bool Protegida { get; set; } = false;  
}
```

3. En la carpeta `Models` agregue un nuevo archivo llamado `Login.cs` con el siguiente código.



```
using System.ComponentModel.DataAnnotations;

namespace frontendnet.Models;

public class Login
{
    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [EmailAddress(ErrorMessage = "El campo {0} no es correo válido.")]
    [Display(Name = "Correo electrónico")]
    public required string Email { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [DataType(DataType.Password)]
    [Display(Name = "Contraseña")]
    public required string Password { get; set; }
}
```

4. Observe los atributos de validación como `Required` y `EmailAddress`. Observe como utilizaremos este modelo mostrar el inicio de sesión en la vista correspondiente.
5. En la carpeta `Models` agregue un nuevo archivo llamado `Pelicula.cs` con el siguiente código



```
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc;

namespace frontendnet.Models;

public class Pelicula
{
    [Display(Name = "Id")]
    public int? PeliculaId { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    public required string Titulo { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [DataType(DataType.MultilineText)]
    public string Sinopsis { get; set; } = "Sin sinopsis";

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [Range(1950, 2024, ErrorMessage = "El valor del campo {0} debe estar entre {1} y {2}.")]
    [Display(Name = "Año")]
    public int Anio { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [Remote(action: "ValidaPoster", controller: "Peliculas", ErrorMessage = "El campo {0} debe ser una dirección URL válida o N/A.")]
    public string Poster { get; set; } = "N/A";

    [Display(Name = "Eliminable")]
    public bool Protegida { get; set; } = false;
    public ICollection<Categoria>? Categorias { get; set; }
}
```

6. En la carpeta `Models` agregue un nuevo archivo llamado `PeliculaCategoria.cs` con el siguiente código.



```
using System.ComponentModel.DataAnnotations;

namespace frontendnet.Models;

public class PeliculaCategoria
{
    [Display(Name = "Categoría")]
    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    public int? CategoriaId { get; set; }

    public string? Nombre { get; set; }

    public Pelicula? Pelicula { get; set; }
}
```

7. En la carpeta **Models** agregue un nuevo archivo llamado **Rol.cs** con el siguiente código.



```
using System.ComponentModel.DataAnnotations;

namespace frontendnet.Models;

public class Rol
{
    public required string Id { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [Display(Name = "Rol")]
    public required string Nombre { get; set; }
}
```

8. En la carpeta **Models** agregue un nuevo archivo llamado **Usuario.cs** con el siguiente código.



```
using System.ComponentModel.DataAnnotations;

namespace frontendnet.Models;

public class Usuario
{
    [Display(Name = "Id")]
    public string? Id { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [EmailAddress(ErrorMessage = "El campo {0} no es correo válido.")]
    public required string Email { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    public required string Nombre { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    public required string Rol { get; set; }
}
```

9. En la carpeta **Models** agregue un nuevo archivo llamado **UsuarioPwd.cs** con el siguiente código.



```
using System.ComponentModel.DataAnnotations;

namespace frontendnet.Models;

public class UsuarioPwd
{
    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [EmailAddress(ErrorMessage = "El campo {0} no es correo válido.")]
    [Display(Name = "Correo electrónico")]
    public required string Email { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    [MinLength(6, ErrorMessage = "El campo {0} debe tener un mínimo de {1} caracteres.")]
    [DataType(DataType.Password)]
    [Display(Name = "Contraseña")]
    public required string Password { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    public required string Nombre { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]
    public required string Rol { get; set; }
}
```

10. Enhorabuena, ha creado correctamente este apartado.

Instrucciones. Agregar middlewares.

Los middlewares son funciones de código que se ejecutan antes de que una petición HTTP llegue al manejador de rutas o antes de que un cliente reciba una respuesta, lo que da al framework la capacidad de ejecutar un script típico antes o después de la petición de un cliente.

1. En la carpeta **Middlewares** agregue un nuevo archivo llamado **EnviaBearerDelegatingHandler.cs** con el siguiente código.

```
● ● ●  
using System.Security.Claims;  
  
namespace frontendnet.Middlewares;  
  
public class EnviaBearerDelegatingHandler(IHttpContextAccessor httpContextAccessor) : DelegatingHandler  
{  
    protected override Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)  
    {  
        request.Headers.Add("Authorization", "Bearer " + httpContextAccessor.HttpContext?.User.FindFirstValue("jwt"));  
        return base.SendAsync(request, cancellationToken);  
    }  
}
```

2. En la carpeta **Middlewares** agregue un nuevo archivo llamado **RefrescaTokenDelegatingHandler.cs** con el siguiente código.

```
● ● ●  
using System.Security.Claims;  
using frontendnet.Services;  
  
namespace frontendnet.Middlewares;  
  
public class RefrescaTokenDelegatingHandler(AuthClientService auth, IHttpContextAccessor httpContextAccessor) : DelegatingHandler  
{  
    protected override async Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)  
    {  
        var response = await base.SendAsync(request, cancellationToken);  
        response.EnsureSuccessStatusCode();  
        // Revisa si el servidor nos envió un nuevo token  
        if (response.Headers.Contains("Set-Authorization"))  
        {  
            string jwt = response.Headers.GetValues("Set-Authorization").FirstOrDefault();  
            var claims = new List<Claim>()  
            {  
                // Todo esto se guarda en la Cookie  
                new(ClaimTypes.Name, httpContextAccessor.HttpContext?.User.FindFirstValue(ClaimTypes.Name)!),  
                new(ClaimTypes.GivenName, httpContextAccessor.HttpContext?.User.FindFirstValue(ClaimTypes.GivenName)!),  
                new("jwt", jwt),  
                new(ClaimTypes.Role, httpContextAccessor.HttpContext?.User.FindFirstValue(ClaimTypes.Role)!)  
            };  
            auth.IniciaSesionAsync(claims);  
        }  
        return response;  
    }  
}
```

3. Enhorabuena, ha creado correctamente este apartado.

Instrucciones. Agregar los Servicios.

1. Agregue un nuevo archivo llamado `AuthClientService.cs` en la carpeta `Services` con el siguiente código.

```
● ● ●
using frontendnet.Models;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using System.Security.Claims;

namespace frontendnet.Services;

public class AuthClientService(HttpClient client, IHttpContextAccessor httpContextAccessor)
{
    public async Task<AuthUser> ObtenTokenAsync(string email, string password)
    {
        Login usuario = new() { Email = email, Password = password };
        // Realizo la llamada al Web API
        var response = await client.PostAsJsonAsync("api/auth", usuario);
        var token = await response.Content.ReadFromJsonAsync<AuthUser>();

        return token!;
    }

    public async void IniciaSesionAsync(List<Claim> claims)
    {
        var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
        var authProperties = new AuthenticationProperties();
        await httpContextAccessor.HttpContext?.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity), authProperties)!;
    }
}
```

2. Agregue un nuevo archivo llamado `CategoriasClientService.cs` en la carpeta `Services` con el siguiente código.

```
● ● ●
using frontendnet.Models;

namespace frontendnet.Services;

public class CategoriasClientService(HttpClient client)
{
```

3. Dentro de la clase agregue los siguientes métodos.



```
public async Task<List<Categoria>?> GetAsync()
{
    return await client.GetFromJsonAsync<List<Categoria>>("api/categorias");
}
```



```
public async Task<Categoria?> GetAsync(int id)
{
    return await client.GetFromJsonAsync<Categoria>($"api/categorias/{id}");
}
```



```
public async Task<bool> PostAsync(Categoria categoria)
{
    var response = await client.PostAsJsonAsync($"api/categorias", categoria);
    return response.IsSuccessStatusCode;
}
```



```
public async Task<bool> PutAsync(Categoria categoria)
{
    var response = await client.PutAsJsonAsync($"api/categorias/{categoria.CategoriaId}", categoria);
    return response.IsSuccessStatusCode;
}
```



```
public async Task<bool> DeleteAsync(int id)
{
    var response = await client.DeleteAsync($"api/categorias/{id}");
    return response.IsSuccessStatusCode;
}
```

4. Agregue un nuevo archivo llamado `PeliculasClientService.cs` en la carpeta `Services` con el siguiente código.



```
using frontendnet.Models;

namespace frontendnet.Services;

public class PeliculasClientService(HttpClient client)
{
    public async Task<List<Pelicula>> GetAsync(string? search)
    {
        return await client.GetFromJsonAsync<List<Pelicula>>($"api/peliculas?s={search}");
    }

    public async Task<Pelicula?> GetAsync(int id)
    {
        return await client.GetFromJsonAsync<Pelicula>($"api/peliculas/{id}");
    }

    public async Task<bool> PostAsync(Pelicula pelicula)
    {
        var response = await client.PostAsJsonAsync($"api/peliculas", pelicula);
        return response.IsSuccessStatusCode;
    }

    public async Task<bool> PutAsync(Pelicula pelicula)
    {
        var response = await client.PutAsJsonAsync($"api/peliculas/{pelicula.PeliculaId}", pelicula);
        return response.IsSuccessStatusCode;
    }

    public async Task<bool> DeleteAsync(int id)
    {
        var response = await client.DeleteAsync($"api/peliculas/{id}");
        return response.IsSuccessStatusCode;
    }

    public async Task<bool> PostAsync(int id, int categoriaid)
    {
        var response = await client.PostAsJsonAsync($"api/peliculas/{id}/categoria", new { categoriaid });
        return response.IsSuccessStatusCode;
    }

    public async Task<bool> DeleteAsync(int id, int categoriaid)
    {
        var response = await client.DeleteAsync($"api/peliculas/{id}/categoria/{categoriaid}");
        return response.IsSuccessStatusCode;
    }
}
```

5. Agregue un nuevo archivo llamado `PerfilClientService.cs` en la carpeta `Services` con el siguiente código.



```
namespace frontendnet.Services;

public class PerfilClientService(HttpClient client)
{
    public async Task<string> ObtenTiempoAsync()
    {
        return await client.GetStringAsync($"api/auth/tiempo");
    }
}
```

6. Agregue un nuevo archivo llamado `RolesClientService.cs` en la carpeta `Services` con el siguiente código.



```
using frontendnet.Models;

namespace frontendnet.Services;

public class RolesClientService(HttpClient client)
{
    public async Task<List<Rol>?> GetAsync()
    {
        return await client.GetFromJsonAsync<List<Rol>>("api/roles");
    }
}
```

7. Agregue un nuevo archivo llamado `UsuariosClientService.cs` en la carpeta `Services` con el siguiente código.



```
using frontendnet.Models;

namespace frontendnet.Services;

public class UsuariosClientService(HttpClient client)
{}
```

8. Dentro de la clase agregue los siguientes métodos.



```
public async Task<List<Usuario>> GetAsync()
{
    return await client.GetFromJsonAsync<List<Usuario>>("api/usuarios");
}

public async Task<Usuario?> GetAsync(string email)
{
    return await client.GetFromJsonAsync<Usuario>($"api/usuarios/{email}");
}

public async Task<bool> PostAsync(UsuarioPwd usuario)
{
    var response = await client.PostAsJsonAsync($"api/usuarios", usuario);
    return response.IsSuccessStatusCode;
}

public async Task<bool> PutAsync(Usuario usuario)
{
    var response = await client.PutAsJsonAsync($"api/usuarios/{usuario.Email}", usuario);
    return response.IsSuccessStatusCode;
}

public async Task<bool> DeleteAsync(string email)
{
    var response = await client.DeleteAsync($"api/usuarios/{email}");
    return response.IsSuccessStatusCode;
}
```

9. Enhorabuena, ha creado correctamente este apartado.

Instrucciones. Agregar los Controladores.

Controlador: Archivos

1. Agregue un nuevo archivo llamado `ArchivosController.cs` en la carpeta `Controllers` con el siguiente código.

```
● ● ●  
using Microsoft.AspNetCore.Mvc;  
  
namespace frontendnet;  
  
public class ArchivosController : Controller  
{  
    public IActionResult Index()  
    {  
        return View();  
    }  
}
```

Controlador: Auth

2. Agregue un nuevo archivo llamado `AuthController.cs` en la carpeta `Controllers` con el siguiente código.

```
● ● ●  
using System.Security.Claims;  
using frontendnet.Models;  
using frontendnet.Services;  
using Microsoft.AspNetCore.Authentication;  
using Microsoft.AspNetCore.Authentication.Cookies;  
using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
  
namespace frontendnet;  
  
public class AuthController(AuthClientService auth) : Controller  
{  
  
}
```

3. Dentro de la clase agregue los siguientes métodos.



```
[AllowAnonymous]
public IActionResult Index()
{
    return View();
}
```



```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> IndexAsync(Login model)
{
    if (ModelState.IsValid)
    {
        try
        {
            // Esta función verifica en backend que el correo y contraseña sean válidos
            var token = await auth.ObtenTokenAsync(model.Email, model.Password);
            var claims = new List<Claim>
            {
                // Todo esto se guarda en la Cookie
                new(ClaimTypes.Name, token.Email),
                new(ClaimTypes.GivenName, token.Nombre),
                new("jwt", token.Jwt),
                new(ClaimTypes.Role, token.Rol),
            };
            auth.IniciaSesionAsync(claims);
            // Usuario válido, lo envía a la lista de Películas
            return RedirectToAction("Index", "Peliculas");
        }
        catch (Exception)
        {
            ModelState.AddModelError("Email", "Credenciales no válidas. Inténtelo nuevamente.");
        }
    }
    return View(model);
}
```



```
[Authorize(Roles = "Administrador, Usuario")]
public async Task<IActionResult> SalirAsync()
{
    // Cierra la sesión
    await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
    // Sino, se redirige a la página inicial
    return RedirectToAction("Index", "Auth");
}
```

Controlador: Bitácora

4. Agregue un nuevo archivo llamado `BitacoraController.cs` en la carpeta `Controllers` con el siguiente código.

```
● ● ●  
using Microsoft.AspNetCore.Mvc;  
  
namespace frontendnet;  
  
public class BitacoraController : Controller  
{  
    public IActionResult Index()  
    {  
        return View();  
    }  
}
```

Controlador: Categorías

5. Agregue un nuevo archivo llamado `CategoriasController.cs` en la carpeta `Controllers` con el siguiente código.

```
● ● ●  
using frontendnet.Models;  
using frontendnet.Services;  
using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
  
namespace frontendnet;  
  
[Authorize(Roles = "Administrador")]  
public class CategoriasController(CategoriasClientService categorias) : Controller  
{  
}
```

6. Dentro de la clase agregue los siguientes métodos.

```
● ● ●  
public async Task<IActionResult> Index()  
{  
    List<Categoria>? lista = [];  
    try  
    {  
        lista = await categorias.GetAsync();  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(lista);  
}
```

```
● ● ●  
public async Task<IActionResult> Detalle(int id)  
{  
    Categoria? item = null;  
    try  
    {  
        item = await categorias.GetAsync(id);  
        if (item == null) return NotFound();  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(item);  
}
```

```
● ● ●  
public IActionResult Crear()  
{  
    return View();  
}
```

```
● ● ●  
[HttpPost]  
public async Task<IActionResult> CrearAsync(Categoría itemToCreate)  
{  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            await categorias.PostAsync(itemToCreate);  
            return RedirectToAction(nameof(Index));  
        }  
        catch (HttpRequestException ex)  
        {  
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
                return RedirectToAction("Salir", "Auth");  
        }  
    }  
  
    ModelState.AddModelError("Nombre", "No ha sido posible realizar la acción. Inténtelo nuevamente.");  
    return View(itemToCreate);  
}
```

```
● ● ●  
public async Task<IActionResult> EditarAsync(int id)  
{  
    Categoria? itemToDelete = null;  
    try  
    {  
        itemToDelete = await categorias.GetAsync(id);  
        if (itemToDelete == null) return NotFound();  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(itemToDelete);  
}
```

```
● ● ●  
[HttpPost]  
public async Task<IActionResult> EditarAsync(int id, Categoria itemToDelete)  
{  
    if (id != itemToDelete.CategoriaId) return NotFound();  
  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            await categorias.PutAsync(itemToDelete);  
            return RedirectToAction(nameof(Index));  
        }  
        catch (HttpRequestException ex)  
        {  
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
                return RedirectToAction("Salir", "Auth");  
        }  
    }  
  
    ModelState.AddModelError("Nombre", "No ha sido posible realizar la acción. Inténtelo nuevamente.");  
    return View(itemToDelete);  
}
```

```
● ● ●  
public async Task<IActionResult> Eliminar(int id, bool? showError = false)  
{  
    Categoria? itemToDelete = null;  
    try  
    {  
        itemToDelete = await categorias.GetAsync(id);  
        if (itemToDelete == null) return NotFound();  
  
        if (showError.GetValueOrDefault())  
            ViewData["ErrorMessage"] = "No ha sido posible realizar la acción. Inténtelo nuevamente.";  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(itemToDelete);  
}
```

```
[HttpPost]
public async Task<IActionResult> Eliminar(int id)
{
    if (ModelState.IsValid)
    {
        try
        {
            await categorias.DeleteAsync(id);
            return RedirectToAction(nameof(Index));
        }
        catch (HttpRequestException ex)
        {
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
                return RedirectToAction("Salir", "Auth");
        }
    }
    return RedirectToAction(nameof(Eliminar), new { id, showError = true });
}
```

Controlador: Home

7. Agregue un nuevo archivo llamado `HomeController.cs` en la carpeta `Controllers` con el siguiente código.

```
using Microsoft.AspNetCore.Mvc;

namespace frontendnet;

public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }

    public IActionResult Error([FromServices] IHostEnvironment hostEnvironment)
    {
        return View();
    }

    public IActionResult AccessDenied()
    {
        return View();
    }
}
```

Controlador: Películas

8. Agregue un nuevo archivo llamado `PeliculasController.cs` en la carpeta `controllers` con el siguiente código.

```
● ● ●  
using System.Security.Claims;  
using frontendnet.Models;  
using frontendnet.Services;  
using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.AspNetCore.Mvc.Rendering;  
  
namespace frontendnet;  
  
[Authorize(Roles = "Administrador, Usuario")]  
public class PeliculasController(PeliculasClientService peliculas, CategoriasClientService categorias) : Controller  
{  
}  
}
```

9. Dentro de la clase agregue los siguientes métodos.

```
● ● ●  
public async Task<IActionResult> Index(string? s)  
{  
    List<Pelicula>? lista = [];  
    try  
    {  
        lista = await peliculas.GetAsync(s);  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    if (User.FindFirstValue(ClaimTypes.Role) == "Administrador")  
        ViewBag.SoloAdmin = true;  
  
    ViewBag.search = s;  
    return View(lista);  
}
```

```
● ● ●  
public async Task<IActionResult> Detalle(int id)  
{  
    Pelicula? item = null;  
    try  
    {  
        item = await peliculas.GetAsync(id);  
        if (item == null) return NotFound();  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(item);  
}
```

```
● ● ●  
[Authorize(Roles = "Administrador")]  
public IActionResult Crear()  
{  
    return View();  
}
```

```
● ● ●  
[HttpPost]  
[Authorize(Roles = "Administrador")]  
public async Task<IActionResult> CrearAsync(Pelicula itemToCreate)  
{  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            await peliculas.PostAsync(itemToCreate);  
            return RedirectToAction(nameof(Index));  
        }  
        catch (HttpRequestException ex)  
        {  
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
                return RedirectToAction("Salir", "Auth");  
        }  
    }  
    // En caso de error  
    ModelState.AddModelError("Nombre", "No ha sido posible realizar la acción. Inténtelo nuevamente.");  
    return View(itemToCreate);  
}
```

```
● ● ●  
[Authorize(Roles = "Administrador")]  
public async Task<IActionResult> EditarAsync(int id)  
{  
    Pelicula? itemToDelete = null;  
    try  
    {  
        itemToDelete = await peliculas.GetAsync(id);  
        if (itemToDelete == null) return NotFound();  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(itemToDelete);  
}
```



```
[HttpPost]
[Authorize(Roles = "Administrador")]
public async Task<IActionResult> EditarAsync(int id, Pelicula itemToDelete)
{
    if (id != itemToDelete.PeliculaId) return NotFound();

    if (ModelState.IsValid)
    {
        try
        {
            await peliculas.PutAsync(itemToDelete);
            return RedirectToAction(nameof(Index));
        }
        catch (HttpRequestException ex)
        {
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
                return RedirectToAction("Salir", "Auth");
        }
    }
    // En caso de error
    ModelState.AddModelError("Nombre", "No ha sido posible realizar la acción. Inténtelo nuevamente.");
    return View(itemToDelete);
}
```



```
[Authorize(Roles = "Administrador")]
public async Task<IActionResult> Eliminar(int id, bool? showError = false)
{
    Pelicula? itemToDelete = null;
    try
    {
        itemToDelete = await peliculas.GetAsync(id);
        if (itemToDelete == null) return NotFound();

        if (showError.GetValueOrDefault())
            ViewData["ErrorMessage"] = "No ha sido posible realizar la acción. Inténtelo nuevamente.";
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    return View(itemToDelete);
}
```



```
[HttpPost]
[Authorize(Roles = "Administrador")]
public async Task<IActionResult> Eliminar(int id)
{
    if (ModelState.IsValid)
    {
        try
        {
            await peliculas.DeleteAsync(id);
            return RedirectToAction(nameof(Index));
        }
        catch (HttpRequestException ex)
        {
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
                return RedirectToAction("Salir", "Auth");
        }
    }
    // En caso de error
    return RedirectToAction(nameof(Eliminar), new { id, showError = true });
}
```



```
[AcceptVerbs("GET", "POST")]
[Authorize(Roles = "Administrador")]
public IActionResult ValidaPoster(string Poster)
{
    if (Uri.IsWellFormedUriString(Poster, UriKind.Absolute) || Poster.Equals("N/A"))
        return Json(true);
    return Json(false);
}
```



```
[Authorize(Roles = "Administrador")]
public async Task<IActionResult> Categorias(int id)
{
    Pelicula? itemToView = null;
    try
    {
        itemToView = await peliculas.GetAsync(id);
        if (itemToView == null) return NotFound();
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    ViewData["PeliculaId"] = itemToView?.PeliculaId;
    return View(itemToView);
}
```



```
[Authorize(Roles = "Administrador")]
public async Task<IActionResult> CategoriasAgregar(int id)
{
    PeliculaCategoria? itemToView = null;
    try
    {
        Pelicula? pelicula = await peliculas.GetAsync(id);
        if (pelicula == null) return NotFound();

        await CategoriasDropDownListAsync();
        itemToView = new PeliculaCategoria { Pelicula = pelicula };
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    return View(itemToView);
}
```

```
● ● ●  
[HttpPost]  
[Authorize(Roles = "Administrador")]  
public async Task<IActionResult> CategoriasAgregar(int id, int categoriaid)  
{  
    Pelicula? pelicula = null;  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            pelicula = await peliculas.GetAsync(id);  
            if (pelicula == null) return NotFound();  
  
            Categoria? categoria = await categorias.GetAsync(categoriaid);  
            if (categoria == null) return NotFound();  
  
            await peliculas.PostAsync(id, categoriaid);  
            return RedirectToAction(nameof(Categorias), new { id });  
        }  
        catch (HttpRequestException ex)  
        {  
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
                return RedirectToAction("Salir", "Auth");  
        }  
    }  
    // En caso de error  
    ModelState.AddModelError("id", "No ha sido posible realizar la acción. Inténtelo nuevamente.");  
    await CategoriasDropDownListAsync();  
    return View(new PeliculaCategoria { Pelicula = pelicula });  
}
```

```
● ● ●  
[Authorize(Roles = "Administrador")]  
public async Task<IActionResult> CategoriasRemover(int id, int categoriaid, bool? showError = false)  
{  
    PeliculaCategoria? itemToView = null;  
    try  
    {  
        Pelicula? pelicula = await peliculas.GetAsync(id);  
        if (pelicula == null) return NotFound();  
  
        Categoria? categoria = await categorias.GetAsync(categoriaid);  
        if (categoria == null) return NotFound();  
  
        itemToView = new PeliculaCategoria { Pelicula = pelicula, CategoriaId = categoriaid, Nombre = categoria.Nombre };  
  
        if (showError.GetValueOrDefault())  
            ViewData["ErrorMessage"] = "No ha sido posible realizar la acción. Inténtelo nuevamente";  
    }  
    catch (HttpRequestException ex)  
    {  
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
            return RedirectToAction("Salir", "Auth");  
    }  
    return View(itemToView);  
}
```

```
● ● ●  
[HttpPost]  
[Authorize(Roles = "Administrador")]  
public async Task<IActionResult> CategoriasRemover(int id, int categoriaid)  
{  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            await peliculas.DeleteAsync(id, categoriaid);  
            return RedirectToAction(nameof(Categorias), new { id });  
        }  
        catch (HttpRequestException ex)  
        {  
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  
                return RedirectToAction("Salir", "Auth");  
        }  
    }  
    // En caso de error  
    return RedirectToAction(nameof(CategoriasRemover), new { id, categoriaid, showError = true });  
}
```

```
● ● ●  
private async Task CategoriasDropDownListAsync(object? itemSeleccionado = null)  
{  
    var listado = await categorias.GetAsync();  
    ViewBag.Categoría = new SelectList(listado, "CategoriaId", "Nombre", itemSeleccionado);  
}
```

Controlador: Perfil

10. Agregue un nuevo archivo llamado `PerfilController.cs` en la carpeta `Controllers` con el siguiente código.

```
● ● ●  
using System.Security.Claims;  
using frontendnet.Models;  
using frontendnet.Services;  
using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
  
namespace frontendnet;  
  
[Authorize]  
public class PerfilController(PerfilClientService perfil) : Controller  
{  
}
```

11. Dentro de la clase agregue los siguientes métodos.



```
public async Task<IActionResult> IndexAsync()
{
    AuthUser? usuario = null;
    try
    {
        ViewBag.timeRemaining = await perfil.ObtenTiempoAsync();

        // Obtiene el perfil del usuario
        usuario = new AuthUser
        {
            Email = User.FindFirstValue(ClaimTypes.Name)!,  

            Nombre = User.FindFirstValue(ClaimTypes.GivenName)!,  

            Rol = User.FindFirstValue(ClaimTypes.Role)!,  

            Jwt = User.FindFirstValue("jwt")!
        };
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    return View(usuario);
}
```

Controlador: Usuarios

12. Agregue un nuevo archivo llamado `UsuariosController.cs` en la carpeta `Controllers` con el siguiente código.



```
using System.Security.Claims;
using frontendnet.Models;
using frontendnet.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace frontendnet;

[Authorize(Roles = "Administrador")]
public class UsuariosController(UsuariosClientService usuarios, RolesClientService roles) : Controller
{}
```

13. Dentro de la clase agregue los siguientes métodos.



```
public async Task<IActionResult> Index()
{
    List<Usuario>? lista = [];
    try
    {
        lista = await usuarios.GetAsync();
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    return View(lista);
}
```



```
public async Task<IActionResult> Detalle(string id)
{
    Usuario? item = null;
    try
    {
        item = await usuarios.GetAsync(id);
        if (item == null) return NotFound();
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    return View(item);
}
```



```
public async Task<IActionResult> Crear()
{
    await RolesDropDownListAsync();
    return View();
}
```



```
[HttpPost]
public async Task<IActionResult> CrearAsync(UsuarioPwd itemToCreate)
{
    if (ModelState.IsValid)
    {
        try
        {
            await usuarios.PostAsync(itemToCreate);
            return RedirectToAction(nameof(Index));
        }
        catch (HttpRequestException ex)
        {
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
                return RedirectToAction("Salir", "Auth");
        }
    }

    ModelState.AddModelError("Email", "No ha sido posible realizar la acción. Inténtelo nuevamente.");
    await RolesDropDownListAsync();
    return View(itemToCreate);
}
```



```
[HttpGet("[controller]/[action]/{email}")]
public async Task<IActionResult> EditarAsync(string email)
{
    Usuario? itemToDelete = null;
    try
    {
        itemToDelete = await usuarios.GetAsync(email);
        if (itemToDelete == null) return NotFound();
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    ViewBag.PuedeEditar = !(User.Identity?.Name == email);
    await RolesDropDownListAsync(itemToDelete?.Rol);
    return View(itemToDelete);
}
```



```
[HttpPost("[controller]/[action]/{email}")]
public async Task<IActionResult> EditarAsync(string email, Usuario itemToDelete)
{
    if (email != itemToDelete.Email) return NotFound();

    if (ModelState.IsValid)
    {
        try
        {
            await usuarios.PutAsync(itemToDelete);
            return RedirectToAction(nameof(Index));
        }
        catch (HttpRequestException ex)
        {
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
                return RedirectToAction("Salir", "Auth");
        }
    }

    ModelState.AddModelError("Nombre", "No ha sido posible realizar la acción. Inténtelo nuevamente.");
    ViewBag.PuedeEditar = !(User.Identity?.Name == email);
    await RolesDropDownListAsync(itemToDelete?.Rol);
    return View(itemToDelete);
}
```



```
public async Task<IActionResult> Eliminar(string id, bool? showError = false)
{
    Usuario? itemToDelete = null;
    try
    {
        itemToDelete = await usuarios.GetAsync(id);
        if (itemToDelete == null) return NotFound();

        if (showError.GetValueOrDefault())
            ViewData["ErrorMessage"] = "No ha sido posible realizar la acción. Inténtelo nuevamente.";
    }
    catch (HttpRequestException ex)
    {
        if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
            return RedirectToAction("Salir", "Auth");
    }
    ViewBag.PuedeEditar = !(User.Identity?.Name == id);
    return View(itemToDelete);
}
```



```
[HttpPost]
public async Task<IActionResult> Eliminar(string id)
{
    if (ModelState.IsValid)
    {
        try
        {
            await usuarios.DeleteAsync(id);
            return RedirectToAction(nameof(Index));
        }
        catch (HttpRequestException ex)
        {
            if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)
                return RedirectToAction("Salir", "Auth");
        }
    }
    return RedirectToAction(nameof(Eliminar), new { id, showError = true });
}
```



```
private async Task RolesDropDownListAsync(object? rolSeleccionado = null)
{
    var listado = await roles.GetAsync();
    ViewBag.Rol = new SelectList(listado, "Nombre", "Nombre", rolSeleccionado);
}
```

14. Enhorabuena, ha creado correctamente este apartado.

Instrucciones: Crear la plantilla de frontend en el motor de plantillas.

1. Generalmente se debe crear una o más plantillas de diseño para ser utilizadas por varias páginas web, en lugar de definir el diseño en cada una de las páginas web.
2. El motor de vistas de frontend incluido con .NET se llama **Razor View Engine**. Debe configurarlo para poder darle toda la funcionalidad a sus vistas.

Carpeta: Views

3. Agregue un nuevo archivo llamado **_ViewImports.cshtml** en la carpeta **Views** con el siguiente contenido.

```
● ● ●  
@using frontendnet  
@using frontendnet.Models  
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

4. Este es un archivo especial de Razor Engine y siempre se incluirá en las vistas.
5. Agregue un nuevo archivo llamado **_ViewStart.cshtml** en la carpeta **Views** con el siguiente contenido.

```
● ● ●  
@{  
    Layout = "_Layout";  
}
```

6. Este es un archivo especial de Razor Engine y siempre se incluirá en las vistas.

Carpeta: Views/Shared

7. Agregue un nuevo archivo llamado **_ValidationScriptsPartial.cshtml** en la carpeta **Views/Shared** con el siguiente contenido.

```
● ● ●  
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.20.0/jquery.validate.min.js"></script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-unobtrusive/4.0.0/jquery.validate.unobtrusive.min.js"></script>  
<script>  
    var settings = {  
        validClass: "is-valid",  
        errorClass: "is-invalid"  
    };  
    $.validator.setDefaults(settings);  
    $.validator.unobtrusive.options = settings;  
</script>
```

8. Observe que son las URL que habíamos obtenido previamente. Este solo se debe incluir en las vistas que requieran validación de entradas.
9. Agregue un nuevo archivo llamado **Error.cshtml** en la carpeta **Views/Shared** con el siguiente contenido.



```
@{  
    ViewData["Title"] = "Aplicación no disponible";  
}  
  
<div class="container">  
    <h1 class="text-danger">@ViewData["Title"]</h1>  
    <p class="text-muted">La aplicación se encuentra no disponible. Intentelo nuevamente más tarde.</p>  
</div>
```

10. Agregue un nuevo archivo llamado `AccessDenied.cshtml` en la carpeta `Views/Shared` con el siguiente contenido.



```
@{  
    ViewData["Title"] = "Acceso denegado";  
}  
  
<div class="container">  
    <h1 class="text-danger">@ViewData["Title"]</h1>  
    <p class="text-muted">El usuario actual no tiene acceso a este apartado.</p>  
</div>
```

11. Agregue un nuevo archivo llamado `_MenuDerPartial.cshtml.cshtml` en la carpeta `Views/Shared` con el siguiente contenido.



```
<div class="navbar-nav small">  
    <div class="navbar-text form-check form-switch form-check-reverse me-2">  
        <input class="form-check-input" type="checkbox" id="darkModeSwitch">  
        <label class="form-check-label" for="darkModeSwitch"><i class="bi bi-moon-stars modo-luz"></i></label>  
    </div>  
    @if (User.Identity?.IsAuthenticated == true)  
    {  
        <a class="nav-link" asp-controller="Perfil">Hola @(User.Identity.Name)!</a>  
        <a class="nav-link" asp-controller="Auth" asp-action="Salir">Salir</a>  
    }  
    else  
    {  
        <a class="nav-link" asp-controller="Auth" asp-action="Index">Iniciar sesión</a>  
    }  
</div>
```

12. Este archivo es una vista parcial del menú principal del sitio.

13. Agregue un nuevo archivo llamado `_MenuIzqPartial.cshtml.cshtml` en la carpeta `Views/Shared` con el siguiente contenido.

```
● ● ●  
@using System.Security.Claims  
  
<div class="me-auto navbar-nav">  
    @if (User.Identity?.IsAuthenticated == true)  
    {  
        <a class="nav-link" asp-controller="Home" asp-action="Index">Inicio</a>  
        <a class="nav-link" asp-controller="Peliculas" asp-action="Index">Películas</a>  
        @if (User.FindFirstValue(ClaimTypes.Role) == "Administrador")  
        {  
            <a class="nav-link" asp-controller="Categorias" asp-action="Index">Categorías</a>  
            <a class="nav-link" asp-controller="Archivos" asp-action="Index">Archivos</a>  
            <a class="nav-link" asp-controller="Usuarios" asp-action="Index">Usuarios</a>  
            <a class="nav-link" asp-controller="Bitacora" asp-action="Index">Bitácora</a>  
        }  
    }  
</div>
```

14. Este archivo también es una vista parcial del menú principal del sitio.
15. Agregue un nuevo archivo llamado `_Layout.cshtml` en la carpeta `Views/Shared` con el siguiente contenido.

```
● ● ●  
<!DOCTYPE html>  
<html lang="es" data-bs-theme="dark">  
  
<head>  
    <meta charset="utf-8" />  
    <link rel="icon" href="~/favicon.ico">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <meta name="description" content="Películas Fei">  
    <title>Netflix: @ ViewData["Title"]</title>  
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.3/css/bootstrap.min.css" />  
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css">  
    @if (User.Identity?.IsAuthenticated == true)  
    {  
        <link rel="stylesheet" href="~/css/site.css" />  
    }  
    else  
    {  
        <link rel="stylesheet" href="~/css/guest.css" />  
    }  
</head>  
  
<body>  
    <header>  
        <nav class="navbar navbar-expand-md bg-transparent">  
            <div class="container-fluid">  
                <a class="navbar-brand" asp-controller="Home" asp-action="Index">  
                      
                </a>  
                <button class="navbar-toggler shadow-none border-0 btn btn-light" type="button" data-bs-toggle="collapse"  
                    data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent" aria-expanded="false"  
                    aria-label="Toggle navigation">  
                    <i class="bi bi-three-dots"></i>  
                </button>  
                <div class="navbar-collapse collapse">  
                    <partial name="_MenuIzqPartial" />  
                    <partial name="_MenuDerPartial" />  
                </div>  
            </div>  
        </nav>  
    </header>  
    <main role="main" class="mt-3 mb-5 container">  
        @RenderBody()  
    </main>  
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>  
    <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.3/js/bootstrap.bundle.min.js"></script>  
    <script src="~/js/cambio_tema.js"></script>  
    @await RenderSectionAsync("Scripts", required: false)  
</body>  
</html>
```

16. Enhorabuena, su plantilla está lista para ser utilizada.

Instrucciones. Agregar las vistas

Vista: Archivos

- Dentro de la carpeta llamada `Views/Archivos` agregue un nuevo archivo llamado `Index.cshtml` con el siguiente código.

```
● ● ●  
 @model AuthUser  
 {@  
     ViewData["Title"] = "Archivos";  
     ViewData["SubTitle"] = "Listado";  
 }  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-4">@ViewData["Title"]</h2>  
  
<div class="mt-5">  
    <div class="alert alert-warning" role="alert">  
        Este apartado queda de tarea al estudiante.  
    </div>  
</div>
```

Vista: Auth

2. Dentro de la carpeta llamada `Views/Auth` agregue un nuevo archivo llamado `Index.cshtml` con el siguiente código.

```
● ● ●

@model Login
 @{
     ViewData["Title"] = "Inicio de sesión";
 }

<form asp-action="Index">
    <div class="card mx-auto" style="max-width: 500px;">
        <div class="card-body">
            <h1 class="text-center text-danger">Bienvenido a Netflix</h1>
            <h2 class="text-center text-secondary-emphasis">Facultad de Estadística e Informática</h2>
            <h5 class="card-card-title text-center mt-4">@ViewData["Title"]</h5>
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-floating mb-3">
                <input asp-for="Email" class="form-control" placeholder="Email">
                <label asp-for="Email" class="form-label"></label>
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-floating mb-3">
                <input asp-for="Password" class="form-control" placeholder="Password">
                <label asp-for="Password" class="form-label"></label>
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <p class="card-text small">Ingrese sus credenciales para poder disfrutar del contenido de la plataforma.</p>
        </div>
        <div class="card-footer text-center">
            <button type="submit" class="btn btn-danger">Iniciar sesión</button>
        </div>
    </div>
</form>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Vista: Bitácora

3. Dentro de la carpeta llamada `Views/Bitacora` agregue un nuevo archivo llamado `Index.cshtml` con el siguiente código.



```
@model AuthUser
 @{
    ViewData["Title"] = "Bitácora";
    ViewData["SubTitle"] = "Listado";
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-4">@ViewData["Title"]</h2>

<div class="mt-5">
    <div class="alert alert-warning" role="alert">
        Este apartado queda de tarea al estudiante.
    </div>
</div>
```

Vista: Categorías

4. Dentro de la carpeta llamada `Views/Categorías` agregue un nuevo archivo llamado `_DetallePartial.cshtml` con el siguiente código.



```
@model Categoria
```

```
<div class="card mt-3">
    <div class="card-body">
        <div class="mb-3 ps-3">
            <label asp-for="CategoriaId" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.CategoriaId</div>
        </div>
        <div class="mb-3 ps-3">
            <label asp-for="Nombre" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Nombre</div>
        </div>
    </div>
</div>
```

5. Dentro de la carpeta llamada **Views/Categorías** agregue un nuevo archivo llamado **Crear.cshtml** con el siguiente código.

```
● ● ●

@model Categoria
 @{
    ViewData["Title"] = "Categorías";
    ViewData["SubTitle"] = "Crear";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<form asp-action="Crear">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="card mt-3">
        <div class="card-body small">
            <div class="form-floating mb-3">
                <input asp-for="Nombre" placeholder="Nombre" class="form-control" />
                <label asp-for="Nombre" class="form-label"></label>
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
        </div>
    </div>
    <div class="text-center mt-3">
        <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Guardar</button>
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
    </div>
</form>
```

6. Dentro de la carpeta llamada **Views/Categorías** agregue un nuevo archivo llamado **Detalle.cshtml** con el siguiente código.

```
● ● ●  
@model Categoria  
{  
    ViewData["Title"] = "Categorías";  
    ViewData["SubTitle"] = "Detalle";  
}  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado"  
            asp-action="Index">Listado</a></li>  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
  
<partial name="_DetallePartial" />  
<div class="text-center mt-3">  
    <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>  
</div>
```

7. Dentro de la carpeta llamada **Views/Categorías** agregue un nuevo archivo llamado **Editar.cshtml** con el siguiente código.

```
● ● ●

@model Categoria
 @{
    ViewData["Title"] = "Categorías";
    ViewData["SubTitle"] = "Editar";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<form asp-action="Editar">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="card mt-3">
        <div class="card-body small">
            <div class="form-floating mb-3">
                <input asp-for="CategoriaId" placeholder="CategoriaId" readonly class="form-control-plaintext" />
                <label asp-for="CategoriaId" class="form-label"></label>
            </div>
            <div class="form-floating mb-3">
                <input asp-for="Nombre" placeholder="Nombre" class="form-control" />
                <label asp-for="Nombre" class="form-label"></label>
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
        </div>
    </div>
    <div class="text-center mt-3">
        <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Guardar</button>
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
    </div>
</form>
```

8. Dentro de la carpeta llamada **Views/Categorías** agregue un nuevo archivo llamado **Eliminar.cshtml** con el siguiente código.

```
● ● ●

@model Categoria
 @{
    ViewData["Title"] = "Categorías";
    ViewData["SubTitle"] = "Eliminar";
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

@if (!Model.Protegida)
{
    <p class="text-danger">¿Está seguro que desea eliminar este elemento?</p>

    <partial name="_DetallePartial" />
    <form asp-action="Eliminar">
        <div class="text-center mt-3">
            <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Eliminar</button>
            <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
        </div>
    </form>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning" role="alert">
            La categoría "<strong>@Model.Nombre</strong>" esta protegida y no puede eliminarse.
        </div>
    </div>

    <div class="text-center mt-3">
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
    </div>
}
```

9. Dentro de la carpeta llamada **Views/Categorías** agregue un nuevo archivo llamado **Index.cshtml** con el siguiente código.

```

● ● ●

@model List<Categoria>
{
    ViewData["Title"] = "Categorías";
    ViewData["SubTitle"] = "Listado";
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<div class="row small mb-3">
    <div class="col">
        <a class="text-decoration-none" asp-action="Crear" title="Crear nuevo">Crear nueva</a>
    </div>
    <div class="col text-end">
        Mostrando @Model.Count() elementos
    </div>
</div>

@if (Model.Count() > 0)
{
    <div class="table-responsive">
        <table class="table table-striped table-bordered small">
            <thead class="text-center">
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.First().CategoriaId)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Nombre)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Protegida)
                    </th>
                    <th></th>
                    <th></th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.CategoriaId)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Nombre)
                        </td>
                        <td>
                            @(item.Protegida ? "No" : "Sí")
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Detalle" asp-route-id="@item.CategoriaId">Detalle</a>
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Editar" asp-route-id="@item.CategoriaId">Editar</a>
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Eliminar" asp-route-id="@item.CategoriaId">Eliminar</a>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning" role="alert">
            No se han encontrado elementos. Inténtelo de nuevo más tarde.
        </div>
    </div>
}

```

Home

10. Dentro de la carpeta llamada `Views/Home` agregue un nuevo archivo llamado `Index.cshtml` con el siguiente código.

```
● ● ●  
{@  
    ViewData["Title"] = "Página principal";  
}  
  
<div class="fondo h-100">  
    <div class="card text-center bg-transparent border-0">  
        <h1 class="card-header bg-transparent border-0 text-danger">Netflix</h1>  
        <div class="card-body">  
            <h2 class="display-5 card-title">Películas y series ilimitadas y mucho más</h2>  
            <p class="card-text lead fw-semibold">Disfruta donde quieras. Cancela cuando quieras.</p>  
            <p class="card-text lead">¿Quieres ver Netflix ya? Presiona el siguiente botón para iniciar la experiencia en  
                Netflix.</p>  
            @if (User.Identity?.IsAuthenticated == true)  
            {  
                <a asp-controller="Películas" class="btn btn-danger">Ver el catálogo de Películas</a>  
            }  
            else  
            {  
                <a asp-controller="Auth" class="btn btn-danger">Comenzar</a>  
            }  
        </div>  
    </div>  
</div>
```

Películas

11. Dentro de la carpeta llamada `Views/Peliculas` agregue un nuevo archivo llamado `_CategoriasListadoPartial.cshtml` con el siguiente código.

```
● ● ●

@model List<Categoria>



<div class="col">
        <a class="text-decoration-none" asp-action="CategoriasAgregar" asp-route-id='@ViewData["PeliculaId"]' title="Crear nuevo">Agregar</a>
    </div>
    <div class="col text-end">
        Mostrando @Model.Count() elementos
    </div>
</div>

@if (Model.Count() > 0)
{
    <div class="table-responsive">
        <table class="table table-striped table-bordered small">
            <thead class="text-center">
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.First().CategoriaId)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Nombre)
                    </th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.CategoriaId)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Nombre)
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="CategoriasRemover" asp-route-id='@ViewData["PeliculaId"]' asp-route-categoriaid="@item.CategoriaId">Eliminar</a>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning small" role="alert">
            No se han encontrado elementos.
        </div>
    </div>
}


```

12. Dentro de la carpeta llamada `Views/Peliculas` agregue un nuevo archivo llamado `_CardPartial.cshtml` con el siguiente código.



```
@model Pelicula

<div class="card mt-3">
    <div class="row g-0">
        <div class="col-md-4">
            <img src='@((Model.Poster == "N/A") ? "https://via.placeholder.com/300x450" : Model.Poster)' alt="@Model.Titulo" class="img-fluid img-thumbnail mt-2 ms-2" style="max-height: 300px;">
        </div>
        <div class="col-md-8">
            <div class="card-body">
                <div class="mb-3 ps-3">
                    <label asp-for="PeliculaId" class="form-label form-text"></label>
                    <div class="form-text text-body">@Model.PeliculaId</div>
                </div>
                <div class="mb-3 ps-3">
                    <label asp-for="Titulo" class="form-label form-text"></label>
                    <div class="form-text text-body">@Model.Titulo</div>
                </div>
                <div class="mb-3 ps-3">
                    <label asp-for="Anio" class="form-label form-text"></label>
                    <div class="form-text text-body">@Model.Anio</div>
                </div>
                <div class="mb-3 ps-3">
                    <label asp-for="Poster" class="form-label form-text"></label>
                    <div class="form-text text-body">@Model.Poster</div>
                </div>
                <div class="mb-3 ps-3">
                    <label asp-for="Sinopsis" class="form-label form-text"></label>
                    <div class="form-text text-body">@Model.Sinopsis</div>
                </div>
            </div>
        </div>
    </div>
</div>
```

13. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **Categorias.cshtml** con el siguiente código.

```
● ● ●  
@model Pelicula  
{  
    ViewData["Title"] = "Peliculas";  
    ViewData["SubTitle"] = "Categorias";  
}  
  
@section Scripts {  
    <partial name="_ValidationScriptsPartial" />  
}  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado"  
            asp-action="Index">Listado</a></li>  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
  
<h4 class="my-3">Categorias</h4>  
@if (Model.Categorias != null)  
{  
    <partial name="_CategoriasListadoPartial" model="Model.Categorias" />  
}  
  
<h4 class="my-3">Película</h4>  
<partial name="_CardPartial" model="Model" />
```

14. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **CategoriasAgregar.cshtml** con el siguiente código.

```
● ● ●

@model PeliculaCategoria
 @{
    ViewData["Title"] = "Peliculas";
    ViewData["SubTitle"] = "Aregar categoría";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar a las categorías" asp-action="Categorias" asp-route-id="@Model.Pelicula?.PeliculaId">Categorías</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<h4 class="my-3">Aregar categoría</h4>
<form asp-action="CategoriasAgregar" asp-route-id="@Model.Pelicula?.PeliculaId">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="card mt-3">
        <div class="card-body small">
            <div class="form-floating mb-3">
                <select class="form-select" asp-for="CategoriaId" class="form-control" asp-items="ViewBag.Categoria">
                    <option value="">Seleccione una categoría</option>
                </select>
                <label asp-for="CategoriaId" class="control-label"></label>
                <span asp-validation-for="CategoriaId" class="text-danger" />
            </div>
        </div>
    </div>
    <div class="text-center mt-3">
        <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Aregar</button>
        <a class="btn btn-sm btn-outline-secondary" asp-action="Categorias" asp-route-id="@Model.Pelicula?.PeliculaId" title="Cancelar">Cancelar</a>
    </div>
</form>

<h4 class="my-3">Película</h4>
<partial name="_CardPartial" model="Model.Pelicula" />
```

15. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **CategoriasRemover.cshtml** con el siguiente código.

```
● ● ●

@model PeliculaCategoria
 @{
    ViewData["Title"] = "Peliculas";
    ViewData["SubTitle"] = "Remover categoría";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar a las categorías" asp-action="Categorias" asp-route-id="@Model.Pelicula?.PeliculaId">Categorías</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<h4 class="my-3">Eliminar categoría</h4>
<p class="text-danger">¿Está seguro que desea remover este elemento?</p>
<p class="text-danger">@ViewData["ErrorMessage"]</p>

<form asp-action="CategoriasRemover" asp-route-id="@Model.Pelicula?.PeliculaId">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="card mt-3">
        <div class="card-body small">
            <div class="form-floating mb-3">
                <div class="form-floating mb-3">
                    <input asp-for="CategoriaId" placeholder="CategoriaId" readonly class="form-control-plaintext" />
                    <label asp-for="CategoriaId" class="form-label"></label>
                </div>
                <div class="mb-3 ps-3">
                    <label asp-for="Nombre" class="form-label form-text"></label>
                    <div class="form-text text-body">@Model.Nombre</div>
                </div>
            </div>
        </div>
    </div>
    <div class="text-center mt-3">
        <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Remover</button>
        <a class="btn btn-sm btn-outline-secondary" asp-action="Categorias" asp-route-id="@Model.Pelicula?.PeliculaId" title="Cancelar">Cancelar</a>
    </div>
</form>

<h4 class="my-3">Película</h4>
<partial name="_CardPartial" model="Model.Pelicula" />
```

16. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **Crear.cshtml** con el siguiente código.

```
● ● ●  
@model Pelicula  
{  
    ViewData["Title"] = "Peliculas";  
    ViewData["SubTitle"] = "Crear";  
}  
  
@section Scripts {  
    <partial name="_ValidationScriptsPartial" />  
}  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
  
<form asp-action="Crear">  
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>  
    <div class="card mt-3">  
        <div class="row g-0">  
            <div class="col-md-4">  
                  
            </div>  
            <div class="col-md-8">  
                <div class="card-body small">  
                    <div class="form-floating mb-3">  
                        <input asp-for="Titulo" placeholder="Nombre" class="form-control" />  
                        <label asp-for="Titulo" class="form-label"></label>  
                        <span asp-validation-for="Titulo" class="text-danger"></span>  
                    </div>  
                    <div class="form-floating mb-3">  
                        <input asp-for="Anio" placeholder="Nombre" class="form-control" />  
                        <label asp-for="Anio" class="form-label"></label>  
                        <span asp-validation-for="Anio" class="text-danger"></span>  
                    </div>  
                    <div class="form-floating mb-3">  
                        <input asp-for="Poster" placeholder="Poster" class="form-control form-control-sm" />  
                        <label asp-for="Poster" class="form-label"></label>  
                        <span asp-validation-for="Poster" class="text-danger"></span>  
                        <div class="form-text small">Escriba N/A si la película no tiene un poster.</div>  
                    </div>  
                    <div class="form-floating mb-3">  
                        <textarea asp-for="Sinopsis" placeholder="Poster" class="form-control form-control-sm" style="height: 100px;"></textarea>  
                        <label asp-for="Sinopsis" class="form-label"></label>  
                        <span asp-validation-for="Sinopsis" class="text-danger"></span>  
                    </div>  
                </div>  
            </div>  
        </div>  
    </div>  
    <div class="text-center mt-3">  
        <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Guardar</button>  
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>  
    </div>  
</form>
```

17. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **Detalle.cshtml** con el siguiente código.

```
● ● ●  
@model Pelicula  
{  
    ViewData["Title"] = "Peliculas";  
    ViewData["SubTitle"] = "Detalle";  
}  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado"  
            asp-action="Index">Listado</a></li>  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
  
<partial name="_CardPartial" model="Model" />  
<div class="text-center mt-3">  
    <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>  
</div>
```

18. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **Editar.cshtml** con el siguiente código.

```
● ● ●
@model Pelicula
@{
    ViewData["Title"] = "Peliculas";
    ViewData["SubTitle"] = "Editar";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado"
            asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<form asp-action="Editar">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="card mt-3">
        <div class="row g-0">
            <div class="col-md-4">
                <img src='@(Model.Poster == "N/A") ? "https://via.placeholder.com/300x450" : Model.Poster'
                    alt="@Model.Titulo" class="img-fluid img-thumbnail mt-2 ms-2" style="max-height: 300px;">
            </div>
            <div class="col-md-8">
                <div class="card-body small">
                    <div class="form-floating mb-3">
                        <input asp-for="PeliculaId" placeholder="Id" readonly class="form-control-plaintext" />
                        <label asp-for="PeliculaId" class="form-label"></label>
                    </div>
                    <div class="form-floating mb-3">
                        <input asp-for="Titulo" placeholder="Nombre" class="form-control" />
                        <label asp-for="Titulo" class="form-label"></label>
                        <span asp-validation-for="Titulo" class="text-danger"></span>
                    </div>
                    <div class="form-floating mb-3">
                        <input asp-for="Anio" placeholder="Nombre" class="form-control" />
                        <label asp-for="Anio" class="form-label"></label>
                        <span asp-validation-for="Anio" class="text-danger"></span>
                    </div>
                    <div class="form-floating mb-3">
                        <input asp-for="Poster" placeholder="Poster" class="form-control form-control-sm" />
                        <label asp-for="Poster" class="form-label"></label>
                        <span asp-validation-for="Poster" class="text-danger"></span>
                        <div class="form-text small">Escriba N/A si la película no tiene un poster.</div>
                    </div>
                    <div class="form-floating mb-3">
                        <textarea asp-for="Sinopsis" placeholder="Poster" class="form-control form-control-sm" style="height: 100px"></textarea>
                        <label asp-for="Sinopsis" class="form-label"></label>
                        <span asp-validation-for="Sinopsis" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<div class="text-center mt-3">
    <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Guardar</button>
    <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
</div>
</form>
```

19. Dentro de la carpeta llamada `Views/Peliculas` agregue un nuevo archivo llamado `Eliminar.cshtml` con el siguiente código.

```
● ● ●  
@model Pelicula  
{  
    ViewData["Title"] = "Peliculas";  
    ViewData["SubTitle"] = "Eliminar";  
}  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado"  
            asp-action="Index">Listado</a></li>  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
  
<p class="text-danger">¿Está seguro que desea eliminar este elemento?</p>  
<p class="text-danger">@ViewData["ErrorMessage"]</p>  
  
<partial name="_CardPartial" model="Model" />  
<form asp-action="Eliminar">  
    <div class="text-center mt-3">  
        <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Eliminar</button>  
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>  
    </div>  
</form>
```

20. Dentro de la carpeta llamada **Views/Peliculas** agregue un nuevo archivo llamado **Index.cshtml** con el siguiente código.

```
● ● ●  
@model List<Pelicula>  
{  
    ViewData["Title"] = "Peliculas";  
    ViewData["SubTitle"] = "Listado";  
}  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
<div class="row small">  
    <div class="col-12 col-sm-6 col-md-4 col-lg-3">  
        <form asp-action="Index" method="get">  
            <div class="input-group input-group-sm mb-3">  
                <input name="s" id="s" value="@ViewBag.search" type="search" class="form-control" placeholder="Buscar por título">  
                <button class="btn btn-danger" type="submit" title="Buscar"><i class="bi bi-search"></i></button>  
            </div>  
        </form>  
    </div>  
</div>  
<div class="row small mb-3">  
    <div class="col">  
        @if (ViewBag.SoloAdmin == true)  
        {  
            <a class="text-decoration-none" asp-action="Crear" title="Crear nuevo">Crear nueva</a>  
        }  
    </div>  
    <div class="col text-end">  
        Mostrando @Model.Count() elementos  
    </div>  
</div>
```

21. Debajo, agregue el despliegue de la tabla con elementos.

```
● ● ●

@if (Model.Count() > 0)
{
    <div class="table-responsive">
        <table class="table table-striped table-bordered small">
            <thead class="text-center">
                <tr>
                    <th width="1">
                        @Html.DisplayNameFor(model => model.First().PeliculaId)
                    </th>
                    <th width="1">
                        @Html.DisplayNameFor(model => model.First().Poster)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Titulo)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Anio)
                    </th>
                    <th></th>
                    @if (ViewBag.SoloAdmin == true)
                    {
                        <th></th>
                        <th></th>
                        <th></th>
                    }
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.PeliculaId)
                        </td>
                        <td class="text-center">
                            <img src='@(item.Poster == "N/A") ? "https://via.placeholder.com/27x40" : item.Poster' alt="@item.Titulo" class="img-fluid img-thumbnail" style="max-height: 40px;">
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Titulo)<br />
                            <span class="text-secondary-emphasis small d-none d-md-block">@Html.DisplayFor(modelItem => item.Sinopsis)</span><br />
                            @foreach (var cat in item.Categorias!)
                            {
                                <span class="badge rounded-pill text-bg-secondary">@cat.Nombre</span>
                            }
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Anio)
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Detalle" asp-route-id="@item.PeliculaId">Detalle</a>
                        </td>
                        @if (ViewBag.SoloAdmin == true)
                        {
                            <td width="1">
                                <a class="text-decoration-none small text-uppercase" asp-action="Categorias" asp-route-id="@item.PeliculaId">Categorias</a>
                            </td>
                            <td width="1">
                                <a class="text-decoration-none small text-uppercase" asp-action="Editar" asp-route-id="@item.PeliculaId">Editar</a>
                            </td>
                            <td width="1">
                                <a class="text-decoration-none small text-uppercase" asp-action="Eliminar" asp-route-id="@item.PeliculaId">Eliminar</a>
                            </td>
                        }
                    </tr>
                }
            </tbody>
        </table>
    </div>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning" role="alert">
            No se han encontrado elementos. Inténtelo de nuevo más tarde.
        </div>
    </div>
}
```

Perfil

22. Dentro de la carpeta llamada `Views/Perfil` agregue un nuevo archivo llamado `Index.cshtml` con el siguiente código.



```
@model AuthUser
{@
    ViewData["Title"] = "Perfil de usuario";
    ViewData["SubTitle"] = "Perfil";
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-4">@ViewData["Title"]</h2>

<div class="card mt-3">
    <div class="card-body">
        <div class="mb-3 ps-3">
            <label asp-for="Email" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Email</div>
        </div>
        <div class="mb-3 ps-3">
            <label asp-for="Nombre" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Nombre</div>
        </div>
        <div class="mb-3 ps-3">
            <label asp-for="Rol" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Rol</div>
        </div>
        <div class="mb-3 ps-3 small">
            <label asp-for="Jwt" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Jwt</div>
        </div>
        <div class="mb-3 ps-3">
            <label class="form-label form-text">Tiempo restante:</label>
            <div class="form-text text-body">@ViewBag.timeRemaining</div>
        </div>
    </div>
</div>
```

Usuarios

23. Dentro de la carpeta llamada `Views/Usuarios` agregue un nuevo archivo llamado `_DetallePartial.cshtml` con el siguiente código.



```
@model Usuario

<div class="card mt-3">
    <div class="card-body">
        <div class="mb-3 ps-3">
            <label asp-for="Email" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Email</div>
        </div>
        <div class="mb-3 ps-3">
            <label asp-for="Nombre" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Nombre</div>
        </div>
        <div class="mb-3 ps-3">
            <label asp-for="Rol" class="form-label form-text"></label>
            <div class="form-text text-body">@Model.Rol</div>
        </div>
    </div>
</div>
```

24. Dentro de la carpeta llamada **Views/Usuarios** agregue un nuevo archivo llamado **Crear.cshtml** con el siguiente código.

```
● ● ●

@model UsuarioPwd
 @{
    ViewData["Title"] = "Usuarios";
    ViewData["SubTitle"] = "Crear";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

<form asp-action="Crear">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="card mt-3">
        <div class="card-body small">
            <div class="form-floating mb-3">
                <input asp-for="Email" class="form-control" placeholder="Email">
                <label asp-for="Email" class="form-label"></label>
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-floating mb-3">
                <input asp-for="Password" class="form-control" placeholder="Password">
                <label asp-for="Password" class="form-label"></label>
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-floating mb-3">
                <input asp-for="Nombre" placeholder="Nombre" class="form-control" />
                <label asp-for="Nombre" class="form-label"></label>
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-floating mb-3">
                <select class="form-select" asp-for="Rol" class="form-control" asp-items="ViewBag.Rol">
                    <option value="">Seleccione un rol de usuario</option>
                </select>
                <label asp-for="Rol" class="control-label"></label>
                <span asp-validation-for="Rol" class="text-danger" />
            </div>
        </div>
    </div>
</div>

<div class="text-center mt-3">
    <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Guardar</button>
    <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
</div>
</form>
```

25. Dentro de la carpeta llamada `Views/Usuarios` agregue un nuevo archivo llamado `Detalle.cshtml` con el siguiente código.

```
● ● ●  
 @model Usuario  
 {@  
     ViewData["Title"] = "Usuarios";  
     ViewData["SubTitle"] = "Detalle";  
 }  
  
<nav aria-label="breadcrumb">  
    <ol class="breadcrumb small">  
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado"  
            asp-action="Index">Listado</a></li>  
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>  
    </ol>  
</nav>  
  
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>  
  
<partial name="_DetallePartial" />  
<div class="text-center mt-3">  
    <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>  
</div>
```

26. Dentro de la carpeta llamada `Views/Usuarios` agregue un nuevo archivo llamado `Editar.cshtml` con el siguiente código.

```

@model Usuario
{
    ViewData["Title"] = "Usuarios";
    ViewData["SubTitle"] = "Editar";
}

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

@if (ViewBag.PuedeEditar)
{
    <form asp-action="Editar">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <div class="card mt-3">
            <div class="card-body small">
                <div class="form-floating mb-3">
                    <input asp-for="Id" placeholder="Id" readonly class="form-control-plaintext" />
                    <label asp-for="Id" class="form-label"></label>
                </div>
                <div class="form-floating mb-3">
                    <input asp-for="Email" placeholder="Email" readonly class="form-control-plaintext" />
                    <label asp-for="Email" class="form-label"></label>
                </div>
                <div class="form-floating mb-3">
                    <input asp-for="Nombre" placeholder="Nombre" class="form-control" />
                    <label asp-for="Nombre" class="form-label"></label>
                    <span asp-validation-for="Nombre" class="text-danger"></span>
                </div>
                <div class="form-floating mb-3">
                    <select class="form-select" asp-for="Rol" class="form-control" asp-items="ViewBag.Rol">
                        <option value="">Seleccione un rol de usuario</option>
                    </select>
                    <label asp-for="Rol" class="control-label"></label>
                    <span asp-validation-for="Rol" class="text-danger"></span>
                </div>
            </div>
        </div>
        <div class="text-center mt-3">
            <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Guardar</button>
            <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
        </div>
    </form>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning" role="alert">
            El usuario actual <strong>@Model.Email</strong> no puede editarse.
        </div>
    </div>

    <div class="text-center mt-3">
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
    </div>
}

```

27. Dentro de la carpeta llamada `Views/Usuarios` agregue un nuevo archivo llamado `Eliminar.cshtml` con el siguiente código.

```
● ● ●

@model Usuario
 @{
    ViewData["Title"] = "Usuarios";
    ViewData["SubTitle"] = "Eliminar";
}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item"><a class="text-decoration-none" title="Regresar al listado" asp-action="Index">Listado</a></li>
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>

<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>

@if (ViewBag.PuedeEditar)
{
    <p class="text-danger">¿Está seguro que desea eliminar este elemento?</p>

    <partial name="_DetallePartial" />
    <form asp-action="Eliminar">
        <div class="text-center mt-3">
            <button id="btnEnviar" type="submit" class="btn btn-sm btn-danger">Eliminar</button>
            <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
        </div>
    </form>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning" role="alert">
            El usuario actual "<strong>@Model.Email</strong>" no puede editarse.
        </div>
    </div>

    <div class="text-center mt-3">
        <a class="btn btn-sm btn-outline-secondary" asp-action="Index" title="Cancelar">Cancelar</a>
    </div>
}
```

28. Dentro de la carpeta llamada **Views/Usuarios** agregue un nuevo archivo llamado **Index.cshtml** con el siguiente código.

```
● ● ●
@model List<Usuario>
@{
    ViewData["Title"] = "Usuarios";
    ViewData["SubTitle"] = "Listado";
}
<nav aria-label="breadcrumb">
    <ol class="breadcrumb small">
        <li class="breadcrumb-item active" aria-current="page">@ViewData["SubTitle"]</li>
    </ol>
</nav>
<h2 class="text-center mb-3">@ViewData["Title"] <small class="text-muted fs-5">@ViewData["SubTitle"]</small></h2>
<div class="row small mb-3">
    <div class="col">
        <a class="text-decoration-none" asp-action="Crear" title="Crear nuevo">Crear nuevo</a>
    </div>
    <div class="col text-end">
        Mostrando @Model.Count() elementos
    </div>
</div>
@if (Model.Count() > 0)
{
    <div class="table-responsive">
        <table class="table table-striped table-bordered small">
            <thead class="text-center">
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Id)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Email)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Nombre)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.First().Rol)
                    </th>
                    <th></th>
                    <th></th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.Id)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Email)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Nombre)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Rol)
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Detalle"
                                asp-route-id="@item.Email">Detalle</a>
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Editar"
                                asp-route-id="@item.Email">Editar</a>
                        </td>
                        <td width="1">
                            <a class="text-decoration-none small text-uppercase" asp-action="Eliminar"
                                asp-route-id="@item.Email">Eliminar</a>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
}
else
{
    <div class="mt-5">
        <div class="alert alert-warning" role="alert">
            No se han encontrado elementos. Inténtelo de nuevo más tarde.
        </div>
    </div>
}
```

29. Enhorabuena, ha realizado correctamente este apartado.

Instrucciones. Configurar su programa principal

1. Abra su archivo `Program.cs` y agregue el siguiente código en la parte superior.

```
● ● ●

using frontendnet.Middlewares;
using frontendnet.Services;
using Microsoft.AspNetCore.Authentication.Cookies;

var builder = WebApplication.CreateBuilder(args);

// Agregamos los servicios
builder.Services.AddControllersWithViews();

// Soporte para consultar el API
var UrlWebAPI = builder.Configuration["UrlWebAPI"];
builder.Services.AddHttpContextAccessor();
builder.Services.AddTransient<EnviaBearerDelegatingHandler>();
builder.Services.AddTransient<RefrescaTokenDelegatingHandler>();
builder.Services.AddHttpClient<AuthClientService>(httpClient => { httpClient.BaseAddress = new Uri(UrlWebAPI!); });
builder.Services.AddHttpClient<CategoriasClientService>(httpClient => { httpClient.BaseAddress = new Uri(UrlWebAPI!); })
    .AddHttpMessageHandler<EnviaBearerDelegatingHandler>()
    .AddHttpMessageHandler<RefrescaTokenDelegatingHandler>();
builder.Services.AddHttpClient<UsuariosClientService>(httpClient => { httpClient.BaseAddress = new Uri(UrlWebAPI!); })
    .AddHttpMessageHandler<EnviaBearerDelegatingHandler>()
    .AddHttpMessageHandler<RefrescaTokenDelegatingHandler>();
builder.Services.AddHttpClient<RolesClientService>(httpClient => { httpClient.BaseAddress = new Uri(UrlWebAPI!); })
    .AddHttpMessageHandler<EnviaBearerDelegatingHandler>()
    .AddHttpMessageHandler<RefrescaTokenDelegatingHandler>();
builder.Services.AddHttpClient<PeliculasClientService>(httpClient => { httpClient.BaseAddress = new Uri(UrlWebAPI!); })
    .AddHttpMessageHandler<EnviaBearerDelegatingHandler>()
    .AddHttpMessageHandler<RefrescaTokenDelegatingHandler>();
builder.Services.AddHttpClient<PerfilClientService>(httpClient => { httpClient.BaseAddress = new Uri(UrlWebAPI!); })
    .AddHttpMessageHandler<EnviaBearerDelegatingHandler>()
    .AddHttpMessageHandler<RefrescaTokenDelegatingHandler>();
```

2. Todo este bloque se encarga de registrar los servicios que la aplicación utilizará y construirla.

```
● ● ●

// Soporte para Cookie Auth
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
{
    options.Cookie.Name = ".frontendnet";
    options.AccessDeniedPath = "/Home/AccessDenied";
    options.LoginPath = "/Auth";
    options SlidingExpiration = true;
    options.ExpireTimeSpan = TimeSpan.FromMinutes(20);
});

var app = builder.Build();
```

3. Observe como estamos configurando la autenticación por cookies de sesión.

4. A continuación, debajo agregue las configuraciones y funcionalidades de los servicios.

```
● ● ●  
// Agregamos un middleware para el manejo de errores  
app.UseExceptionHandler("/Home/Error");  
  
app.UseStaticFiles();  
app.UseRouting();  
  
app.UseAuthentication();  
app.UseAuthorization();  
  
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}");  
  
app.Run();
```

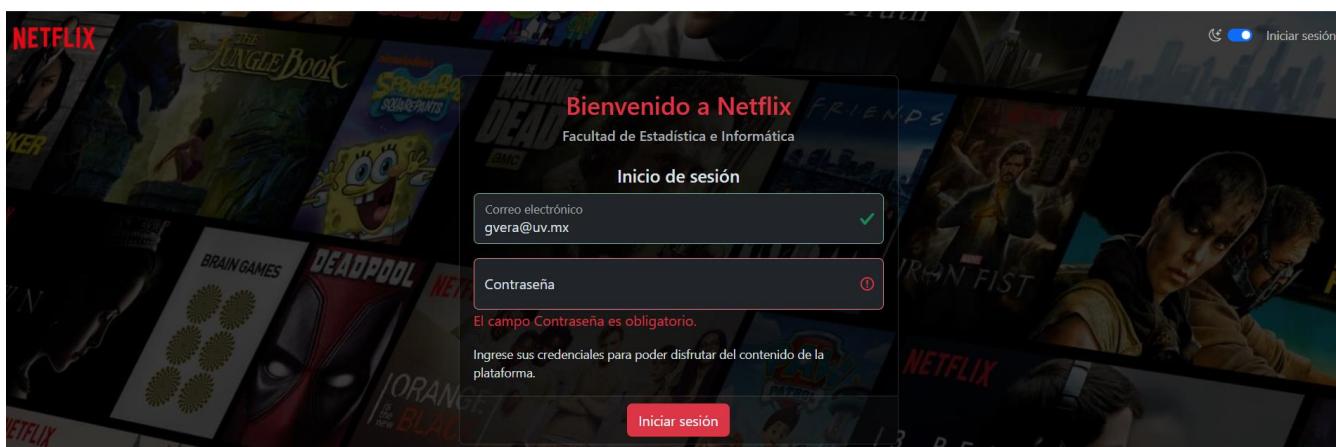
5. La última instrucción, ejecuta su aplicación.

Probar el funcionamiento

- Ejecute uno de sus proyecto de backend y manténgalo corriendo en el puerto **3000**.
- Ejecute su aplicación frontend usando `dotnet run` o `dotnet watch run` y una ventana de su navegador se abrirá.



- Pruebe todos los botones de la parte pública y que todo funcione correctamente.
- Inicie sesión con la cuenta de un usuario **administrador**.

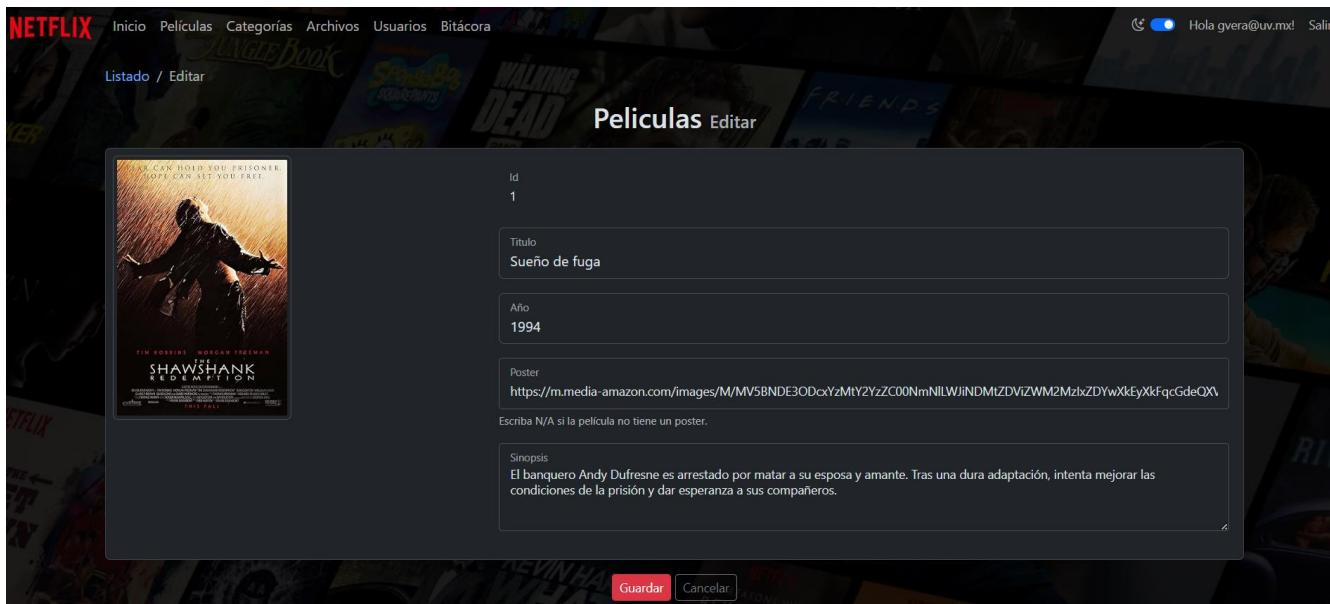


- Verifique que las validaciones funcionen correctamente.
- Ahora pruebe todas las secciones del menú principal.

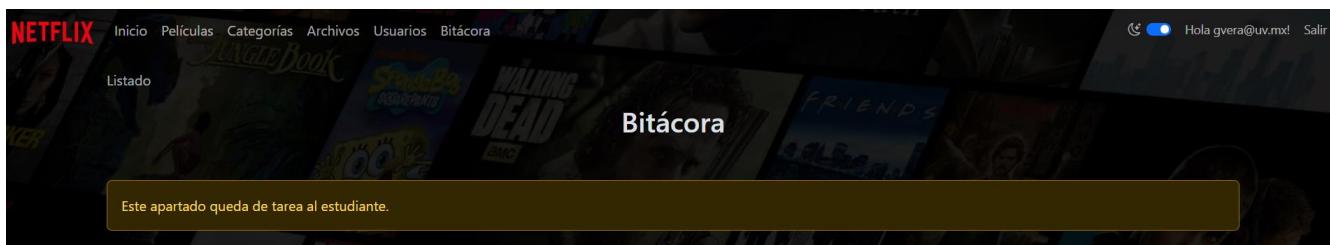
A screenshot of the Netflix application's main menu. The top navigation bar includes links for "Inicio", "Películas", "Categorías", "Archivos", "Usuarios", and "Bitácora". On the right, there is a sign-in status for "Hola gvera@uv.mx!" and a "Salir" button. The main content area is titled "Películas Listado" and shows a table of movies. The table has columns for "Id", "Poster", "Título", "Año", and actions "DETALLE", "CATEGORÍAS", "EDITAR", and "ELIMINAR". Three movies are listed:

ID	Poster	Título	Año	DETALLE	CATEGORÍAS	EDITAR	ELIMINAR
1		Sueño de fuga El banquero Andy Dufresne es arrestado por matar a su esposa y amante. Tras una dura adaptación, intenta mejorar las condiciones de la prisión y dar esperanza a sus compañeros.	1994	DETALLE	CATEGORÍAS	EDITAR	ELIMINAR
2		El padrino El patriarca de una organización criminal transfiere el control de su clandestino imperio a su recién nacido hijo.	1972	DETALLE	CATEGORÍAS	EDITAR	ELIMINAR
3		El caballero oscuro Cuando el Joker emerge causando caos y violencia en Gotham, el caballero de las noches deberá aceptar una de las pruebas psicológicas y físicas más difíciles para poder luchar con las injusticias del enemigo.	2008	DETALLE	CATEGORÍAS	EDITAR	ELIMINAR

7. Agregue, modifique y elimine elementos de las secciones Películas, Categorías, Usuarios.



8. Revise que todas los apartados funcionen de manera adecuada.
9. Observe los apartados de Archivos y Bitácora.



10. Estos apartados se dejan disponibles para que el estudiante construya los artefactos correspondientes.
11. Enhорабуена, su aplicación funciona correctamente.

Instrucciones. Colocar su aplicación a un repositorio de código fuente.

1. Por último, antes de subir su aplicación a GitHub, genere un archivo llamado `.gitignore` que previene el subir información innecesaria al repositorio público.

```
dotnet new gitignore
```

```
PS C:\codigo\tw\mvc> dotnet new gitignore
La plantilla "archivo gitignore de dotnet" se creó correctamente.
```

2. Este nuevo archivo aparece en la raíz de su proyecto.

```
> obj
> Properties
> Views
❖ .gitignore
{ appsettings.Development.json
{ appsettings.json
bd.sql
mvc.csproj
mvc.sln
C# Program.cs

18
19 # Mono auto generated files
20 mono_crash.*
21
22 # Build results
23 [Dd]ebug/
24 [Dd]ebugPublic/
25 [Rr]elease/
26 [Rr]eleases/
27 x64/
28 x86/
29 [Ww][Ii][Nn]32/
30 [Aa][Rr][Mm]/
```

3. Ahora ya puede publicar su archivo en GitHub para terminar la práctica.

4. Felicidades, ha creado su aplicación de manera correcta.