Autorización y Autenticación

- Autenticación
- Autorización
- Cookies.
- Sesiones.
- Configuración.
- Métodos de sesión

Autenticación y Autorización

La autenticación es el proceso de verificar la identidad de un usuario. Es básicamente confirmar que el usuario es quien dice ser. Se suele hacer a través de credenciales (usuario y contraseña), pero puede incluir otros métodos como OTPs, autenticación de dos factores (2FA), etc.

Flujo:

1. Un usuario envía sus credenciales
2. El sistema las compara con los registros
3. Si coinciden, se confirma la identidad.

Ejemplo común: Iniciar una sesión en una página con tu email y contraseña.

Un usuario envía
sus credenciales

Validación

Confirmación

Autenticación y Autorización

Una vez autenticado, la autorización define lo que el usuario puede hacer o acceder dentro del sistema. En pocas palabras, después de saber quién es el usuario, el sistema verifica si tiene permisos para realizar cierta acción o acceder a recursos específicos.

Flujo: Después de autenticarse, el sistema revisa los permisos del usuario para acciones específicas (por ejemplo, si es "admin" puede acceder a más secciones).

Ejemplo común: Alguien inicia sesión, y dependiendo de su rol (usuario, moderador, administrador), tiene acceso a diferentes funcionalidades en la aplicación.

Autenticación y Autorización

Resumiendo…

Confirma si el usuario es quien dice ser

Determina qué permisos tiene el usuario para accesos a los recursos.

Manejos de sesión

HTTP

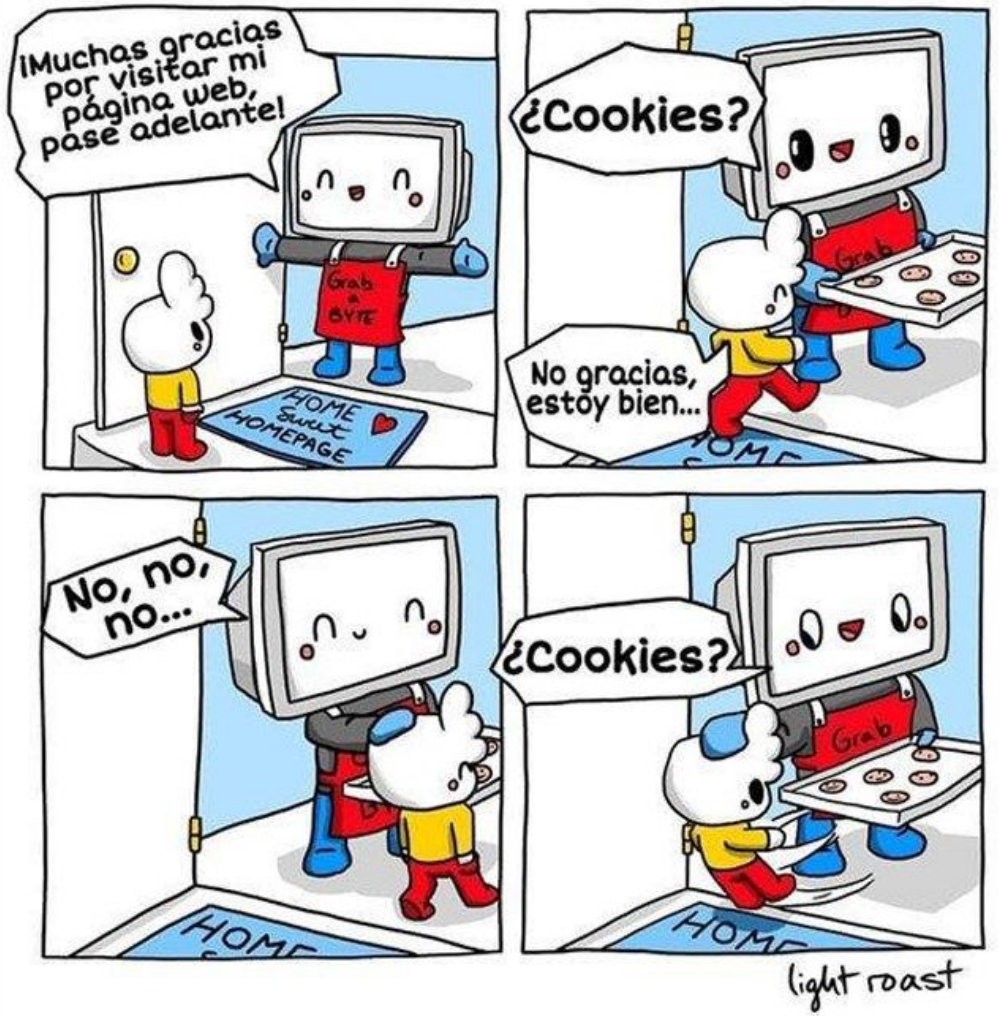
HTTP, el protocolo que se encarga de la comunicación entre un servidor y un cliente en la web, se conoce como protocolo sin estado. En otras palabras, si un usuario solicita dos páginas en un servidor, no se compartirá automáticamente ninguna información entre estas dos solicitudes.

En su lugar, un desarrollador tendrá que confiar en algo llamado cookies para compartir información entre las solicitudes (requests). Esto es extremadamente útil en muchas situaciones, por ejemplo, para mantener un usuario conectado entre varias solicitudes, etc

https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-7.0

Manejos de sesión

Cookies



Manejos de sesión

Cookies

Una cookie es básicamente un archivo físico de texto sin formato almacenado por el cliente (normalmente un navegador), vinculado a un sitio web específico. El cliente permitirá a este sitio web específico leer la información almacenada en este archivo en solicitudes posteriores, básicamente permitiendo que el servidor (o incluso el propio cliente) almacene información para su uso posterior.

Manejos de sesión

Sesión

El estado de sesión es un recurso de ASP.NET Core para el almacenamiento de datos de usuario mientras el usuario examina una aplicación web. El estado de sesión usa un almacén mantenido por la aplicación para conservar los datos en las solicitudes de un cliente. Los datos de sesión están respaldados por una memoria caché y se consideran datos efímeros

Estado de sesión

Para mantener el estado de sesión, ASP.NET Core proporciona una cookie al cliente que contiene un identificador de sesión. El identificador de sesión de la cookie:

- Se envía a la aplicación con cada solicitud.
- Lo usa la aplicación para capturar los datos de la sesión.

Manejos de sesión

Como usar sesiones en nuestro código

El middleware para gestionar el estado de sesión está incluido en el marco de trabajo. Para habilitar el middleware de sesión, hay que asegurarte de que el archivo Program.cs contenga lo siguiente:

1 – Una implementación de IDistributedCache:

Se utiliza como memoria auxiliar para la sesión. Se puedes utilizar la implementación en memoria para este propósito.

2 - Para habilitar el middleware de sesión, Startup debe contener:

Llamadas a AddSession y UseSession para configurar y habilitar el middleware de sesión.

3- Incluir los espacios de nombres necesarios a nuestros controladores donde queramos usar sesiones

- **Microsoft.AspNetCore.Session:** Contiene el middleware necesario para habilitar y gestionar el estado de sesión en aplicaciones ASP.NET Core.
- **Microsoft.AspNetCore.Http:** Incluye tipos fundamentales para el manejo de solicitudes y respuestas HTTP, como HttpContext, HttpRequest y HttpResponse

Manejos de sesión

Configurando el servicio de sesión

Lo que se agrega al Program.cs:

- **Agregar soporte para sesiones:** Las sesiones permiten almacenar datos específicos del usuario entre solicitudes HTTP. En este caso, se usarán para guardar si el usuario está autenticado y quién es.

- **Configurar cookies:** Las cookies son necesarias porque las sesiones dependen de ellas para identificar a los usuarios entre solicitudes. Una cookie actúa como un identificador único para la sesión en el servidor.

- **Registrar los servicios necesarios:** Se registra el servicio de sesiones y se configura cómo se manejarán las cookies en la aplicación.

- **Middleware para manejar sesiones:** Se agrega el middleware para que ASP.NET Core pueda interceptar solicitudes y leer o escribir en la sesión según sea necesario.

Manejos de sesión

Configurando el servicio de sesión

Agregamos AddSession en **builder.Services**: Este servicio le dice a ASP.NET que queremos usar sesiones en nuestra aplicación. Se configura el tiempo de vida de la sesión (IdleTimeout) y opciones de la cookie asociada.

```
// Habilitar servicios de sesiones
builder.Services.AddSession(options =>
{       options.IdleTimeout = TimeSpan.FromMinutes(30);   // Tiempo de expiración de la sesión
        options.Cookie.HttpOnly = true; // Solo accesible desde HTTP, no JavaScript
        options.Cookie.IsEssential = true; // Necesario incluso si el usuario no acepta cookies
});
```

Parámetros configurados anteriormente

- **options.IdleTimeout:** configuar el tiempo vida de una sesión.
- **HttpOnly = true:** Asegura que las cookies de sesión solo sean accesibles por el servidor, bloqueando intentos maliciosos de leerlas mediante JavaScript
- **JavaScript.IsEssential = true**: Garantiza que las cookies necesarias para la funcionalidad básica de la aplicación (como la sesión) siempre estén habilitadas, incluso si el usuario rechaza cookies no esenciales.

Manejos de sesión

Configurando el servicio de sesión

UseSession en el pipeline de middleware: Esto activa el soporte para sesiones en la aplicación, permitiendo que las solicitudes lean y escriban en la sesión.

```
// Middleware para usar sesiones
app.UseSession();
```

Manejos de sesión

Usos – Establecer y obtener valores de sesión

Se tiene acceso al estado de sesión desde la clase Controller de MVC con *HttpContext.Session*.

La implementación ISession proporciona varios métodos de extensión para establecer y recuperar valores de cadena y enteros. Los nuevos métodos de extensión se encuentran en el espacio de nombres *Microsoft.AspNetCore.Http y son:*

- Get(ISession, String)
- GetInt32(ISession, String)
- GetString(ISession, String)
- SetInt32(ISession, String, Int32)
- SetString(ISession, String, String)

Ejemplo de uso en el código

```
HttpContext.Session.SetString(“Nombre", "Juan"); // establece una variable de sesión denominada
                                                 “Nombre” con el valor Juan

string Nombre = HttpContext.Session.GetString(“Nombre");   //recupera el contenido de la variable nombre
```

Práctica en .NET

Crear un servicio de autenticación

Práctica en .NET

Configuración para el Uso de Sesiones en Program.cs:

Agregar la configuración para el uso de sesiones en el archivo Program.cs:

services.AddSession(options => {...}); // para configurar las opciones de la sesión (por ejemplo, expiración).

app.UseSession(); // para habilitar las sesiones en la aplicación.

Práctica en .NET

Crear el Modelo de Usuario Creamos un modelo de usuario simple con un nombre de usuario, contraseña y un rol (para fines de autorización).

```
public class User
{
    public string Username { get; set; }
    public string Password { get; set; }
    public string Role { get; set; } // Ej: "Admin", "User"
}
```

Práctica en .NET

Controlador de Login (LoginController):

Crear un controlador LoginController con métodos para mostrar la vista de login y manejar las solicitudes POST.

- **Acción Index:** Muestra la página de login.
- **Acción Login:** Procesa las credenciales y maneja la autenticación.
- **Acción Logout:** Cierra la sesión del usuario y redirige al login.

Práctica en .NET

LoginViewModel:

- Crear un **ViewModel** LoginViewModel para manejar los datos del formulario de login:
  - Propiedades: Username, Password, ErrorMessage, IsAuthenticated.
  - Este modelo permite pasar información desde el controlador a la vista de manera estructurada.

Práctica en .NET

Crear el Repositorio para la gestión de Usuarios

Crear una interfaz IUserRepository y una implementación SqliteUserRepository para interactuar con la base de datos o fuente de datos.

Método GetUser(string username, string password) que valida si las credenciales del usuario son correctas.

```
public interface IUserRepository
{
    User? GetUserByUsername(string username);
}

public class UserRepository : IUserRepository
{
    private readonly List<User> _users = new List<User>
    {
        new User { Username = "admin", Password = "password123", AccessLevel = "Admin" },
        new User { Username = "user", Password = "pass456", AccessLevel = "User" }
    };

    public User? GetUserByUsername(string username)
    {
        return _users.FirstOrDefault(u => u.Username == username);
    }
}
```

Práctica en .NET

Autenticación Vamos a crear un servicio para manejar el login y logout que va a utilizar luego el controlador

```
public interface IAuthenticationService
{
    bool Login(string username, string password);
    void Logout();
    bool IsAuthenticated();
}


public class AuthenticationService : IAuthenticationService
{
    private readonly IUserRepository _userRepository;
    private readonly IHttpContextAccessor _httpContextAccessor;

    public AuthenticationService(IUserRepository userRepository, IHttpContextAccessor
httpContextAccessor)
    {
        _userRepository = userRepository;
        _httpContextAccessor = httpContextAccessor;
    }
}
```

Práctica en .NET

Inyectando dependencias

En el Achivo programa.cs debemos inyectar las dependencias necesarias para usar el repositorio de usuario y el servicio de autenticación

Registrar El repositorio y el servicio de autenticación:

```
builder.Services.AddScoped<IUserRepository, InMemoryUserRepository>();
builder.Services.AddScoped<IAuthenticationService, AuthenticationService>();
```

También debes registrar IHttpContextAccessor en el contenedor de dependencias:

```
builder.Services.AddHttpContextAccessor();
```

Práctica en .NET

Autenticación Si queremos verificar si el usuario esta logueado directo desde una vista, podemos injectar el **HttpContextAccessor**

```
@inject Microsoft.AspNetCore.Http.IHttpContextAccessor HttpContextAccessor
@{
    var isAuthenticated = false;
    var username = string.Empty;

    if (HttpContextAccessor.HttpContext.Session.GetString("IsAuthenticated") == "true")
    {
        isAuthenticated = true;
        username = HttpContextAccessor.HttpContext.Session.GetString("User");
    }
}
```

Práctica en .NET

Autenticación Si queremos verificar si el usuario esta logueado directo desde una vista, podemos injectar el **HttpContextAccessor**

```
...

<li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
                    </li>

                     @if (isAuthenticated)
                    {
                        <li class="nav-item">
                            <span class="nav-link text-dark me-3">Bienvenido, @username</span>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" href="@Url.Action("Logout", "Login")">Cerrar sesión</a>
                        </li>
                    }
                    else
                    {
                        <li class="nav-item">
                            <a class="nav-link text-dark" href="@Url.Action("Index", "Login")">Iniciar sesión</a>
                        </li>
                    }
```