

Evidencia del diseño y especificación de la API

Criterios seguidos para asegurar Rest

Nos podemos asegurar que la aplicación desarrollada cumple los criterios REST por lo siguiente:

Aplicación orientada a Cliente-servidor: esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y el servidor se “despreocupa” de cómo son usados los datos que envía al cliente. Por ahora solo se desarrolló el backend, posteriormente se implementará una capa de para el cliente

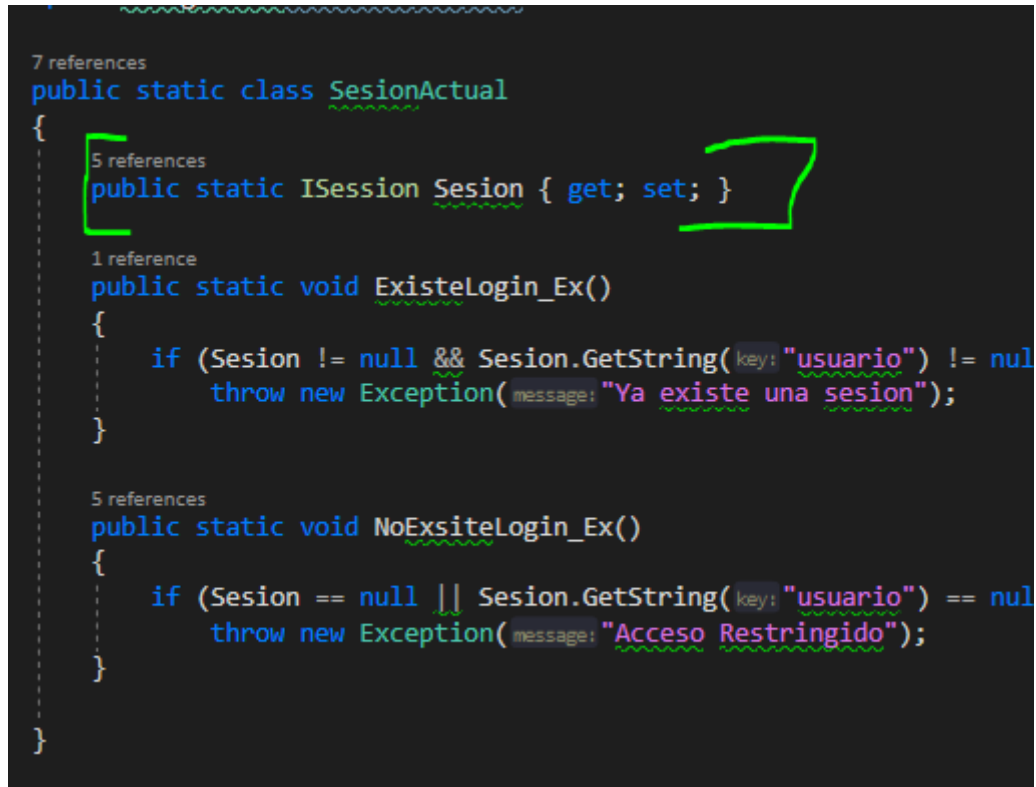
Sin estado: cada petición que recibe el servidor debería es independiente, es decir, no es necesario mantener sesiones. Aunque si existen una sesión mediante cookies, para identificar a los administradores de la página.

Cacheable: se admite un sistema de almacenamiento en caché.

Sistema de capas: el servidor dispone de varias capas para su implementación.

Mecanismo de Autenticación

Se utiliza una clase (estática) para manejar las sesiones de los administradores la cual es la siguiente:



```
7 references
public static class SessionActual
{
    5 references
    public static ISession Session { get; set; }

    1 reference
    public static void ExisteLogin_Ex()
    {
        if (Session != null && Session.GetString(key: "usuario") != null)
            throw new Exception(message: "Ya existe una sesion");
    }

    5 references
    public static void NoExsiteLogin_Ex()
    {
        if (Session == null || Session.GetString(key: "usuario") == null)
            throw new Exception(message: "Acceso Restringido");
    }
}
```

The image shows a code editor with a dark background. The code is in C# and defines a static class named `SessionActual`. Inside the class, there is a static property `Session` of type `ISession` with `get` and `set` accessors. There are two static methods: `ExisteLogin_Ex()` and `NoExsiteLogin_Ex()`. The `ExisteLogin_Ex()` method checks if the `Session` is not null and if the user string is not null, throwing an exception if both are true. The `NoExsiteLogin_Ex()` method checks if the `Session` is null or if the user string is null, throwing an exception if either is true. There are green annotations: a bracket around the `Session` property, a bracket around the `ExisteLogin_Ex()` method, and a bracket around the `NoExsiteLogin_Ex()` method. The code is color-coded with syntax highlighting.

Sostiene una property interfaz `ISession`, para manejar la sesión activa de cada petición http. Esto es una librería que nos aporta Microsoft.

Resources

URL Base/Principal:

<https://localhost:44336/Regiones>

Resto de las endpoints

<https://localhost:44336/PuntosTuristicos/Busqueda?region=1>

<https://localhost:44336/PuntosTuristicos/Busqueda?region=1&categorias=3>

<https://localhost:44336/PuntosTuristicos/Alta> [POST]

<https://localhost:44336/Login/Ingresar?email=emi@gmail.com&contra=1234>

<https://localhost:44336/Reserva/Reservar> [POST]

<https://localhost:44336/Reserva/CambiarEstado?codigo=tzonymijukf&estado=2>

<https://localhost:44336/Reserva/Estado?codigo=tzonymijukf>

<https://localhost:44336/Hospedajes/Busqueda> [POST]

<https://localhost:44336/Hospedajes/Alta> [POST]

<https://localhost:44336/Hospedajes/Baja> [POST]

<https://localhost:44336/Hospedajes/Modificar?nombre=Radison&Disponibilidad=true>