UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN – LEÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

INGENIERÍA EN SISTEMAS DE LA INFORMACIÓN

AÑO LECTIVO: 2025

SEMESTRE: II

Componente Curricular: # PROGRAMACION ORIENTADA A LA WEB II

Grupo: 1

Sub-Grupo:

Profesor(a): JUAN CARLOS LEYTON BRIONES

Autor:

Luis Ignacio Hidalgo Molina.

León, Nicaragua, 2025.

“¡A la Libertad por la Universidad!”

PRACTICA 1

EJERCICIO 1

Controlador EJERCICIO1.cs

El controlador **Ejercicio1** pertenece a la capa de controladores de ASP.NET Core MVC. Su propósito es manejar peticiones relacionadas con manipulación de cadenas y carga de vistas.

Métodos del controlador:

1. Index(String id)

```
public String Index(String id)
{
   id = new string(id.ToUpper().Reverse().ToArray());
   return id;
}
```

Descripción:

- Recibe como parámetro una cadena (id).
- Convierte toda la cadena a mayúsculas.
- Invierte el orden de los caracteres.
- Retorna la nueva cadena transformada.

Ejemplo de uso:

- **Entrada:** "Hola"
- **Proceso:** → "HOLA" → "ALOH"

- **Salida:** "ALOH"

2. Comparar(String id, String cadena)

```
public String Comparar(String id, String cadena)
{
    if (id.Equals(cadena))
    {
        return "Son iguales";
    }
    else
    {
        return "Son diferentes";
    }
}
```

Descripción:

- Recibe dos parámetros de tipo String: id y cadena.
- Compara ambas cadenas usando el método Equals().
- Retorna un mensaje indicando si son exactamente iguales o diferentes.

Ejemplo de uso:

- **Entrada:** "Hola", "Hola" → **Salida:** "Son iguales"
- **Entrada:** "Hola", "Mundo" → **Salida:** "Son diferentes"

3. Perfil()

```
public IActionResult Perfil()
{
    return View("~/Views/Home/Perfil.cshtml");
}
```

Descripción:

- No recibe parámetros.
- Retorna una vista ubicada en la ruta Views/Home/Perfil.cshtml.
- Se utiliza para renderizar y mostrar la página Perfil en la aplicación.

Ejemplo de uso:

- URL asociada → /Ejercicio1/Perfil
- Acción → Muestra la vista Perfil.cshtml

Vista Perfil.cshtml

La vista Perfil.cshtml se utiliza en conjunto con el método Perfil() del controlador Ejercicio1. Su función es mostrar información personal y una imagen dentro de la interfaz web.

Código de la Vista

```
@{
    ViewData["Title"] = "HOLA AMIGUITOS";
}

<div class="text-center">
    <h1>Mi Nombre es Luis Ignacio Hidalgo Molina</h1>
    <h2>Tengo 20 años</h2>
    <h3>Estudio Ingenieria en Sistemas de Informacion</h3>
```

```
    <img src="@Url.Content("~/img/1.jpg")" style="width: 700px;">

    <p>
        Aca muestro un bug algo dificil de ver.
        Un arquitecto/constructor en un pizzaboy en vez del clásico trabajador del Pizza Stack's
    </p>
</div>
```

Descripción de los elementos

- ViewData["Title"]
- Define el título de la página que se mostrará en la pestaña del navegador.
- En este caso: "HOLA AMIGUITOS".
- Etiquetas HTML de presentación (h1, h2, h3)
- Muestran información personal del usuario:
- Nombre completo.
- Edad.
- Carrera universitaria.
- Etiqueta <img>
- Muestra una imagen localizada en la carpeta /img/1.jpg.
- Se renderiza con un ancho de 700 píxeles.
- Etiqueta <p>
- Contiene un texto explicativo sobre un "bug" curioso relacionado con el juego (referencia a un trabajador del Pizza Stack’s reemplazado).

Relación con el controlador Ejercicio1

El método en Ejercicio1 que carga esta vista es:

```
public IActionResult Perfil()
{
   return View("~/Views/Home/Perfil.cshtml");
}
```

Cuando un usuario accede a la ruta /Ejercicio1/Perfil, el controlador devuelve esta vista. El contenido que se muestra será el HTML definido en Perfil.cshtml.

EJERCICIO 2

Controlador EJERCICIO2.cs

El controlador **Ejercicio2** pertenece a la capa de controladores de ASP.NET Core MVC y se encarga de trabajar con un arreglo de cadenas (Cadenas) para obtener valores y compararlos.

Campos del controlador:

```
private String[] Cadenas = new string[] {"Hola", "Que tal", "Como esta", "Hi", "Hola, Amigos"};
```

Cadenas es un arreglo de 5 cadenas predefinidas.

Se usa en los métodos Index y Comparar para buscar o devolver valores.

Métodos del controlador:

1. Index(int id)

```
public String Index(int id)
```

```
{
    try
    {
        if (id <= 4 && id >= 0)
            return Cadenas[id];
        else
            return "Numero invalido";

    }
    catch (Exception ex)
    {
        return "Error: " + ex.Message;
    }
}
```

Descripción:

- Recibe un índice numérico (id).
- Valida que el índice esté entre 0 y 4 (ya que el arreglo tiene 5 elementos).
- Si es válido, retorna la cadena correspondiente en el arreglo.
- Si es inválido, retorna "Número inválido".
- Usa try-catch para capturar posibles errores inesperados (por ejemplo, si el arreglo cambia de tamaño).

Ejemplos:

- **Entrada:** Index(0) → "Hola"
- **Entrada**: Index(4) → "Hola, Amigos"
- **Entrada:** Index(5) → "Numero invalido"

2. Comparar(String id)

```
public String Comparar(String id)
{
   for (int i = 0; i < 5; i++)
   {
      if (id.Equals(Cadenas[i]))
      {
         return "Cadena encontrada: " + Cadenas[i];
      }
   }

   return "No existe";
}
```

Descripción:

- Recibe un string (id).
- Recorre el arreglo Cadenas buscando coincidencias exactas (Equals).
- Si encuentra la cadena, retorna "Cadena encontrada: <valor>".
- Si no encuentra ninguna coincidencia después de recorrer todo el arreglo, retorna "No existe".

Ejemplos:

- **Entrada:** Comparar("Hola") → "Cadena encontrada: Hola"
- **Entrada:** Comparar("Hi") → "Cadena encontrada: Hi"
- **Entrada:** Comparar("Adios") → "No existe"