Ejercicio 2.3.1.

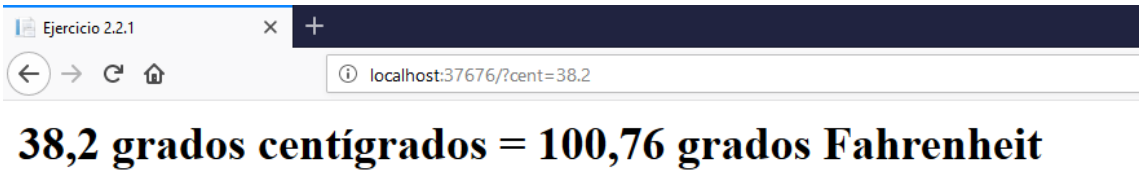
Queremos una aplicación web que convierta una temperatura en grados centígrados a grados Fahrenheit. Para ello debemos pasar la temperatura en grados como un parámetro de nombre *cent* en la url.

Ejemplo: `http://localhost:37676/?cent=38`

Nuestro HomeController debe tener un método que calcule y devuelva el valor en grados Fahrenheit a partir de los grados centígrados que recibirá como parámetro. La fórmula para la conversión es la siguiente:

Grados Fahrenheit = 9/5 * grados centrígrados + 32

El resultado debe ser similar al siguiente:



38,2 grados centígrados = 100,76 grados Fahrenheit

Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(double cent)
        {
            ViewData["degrees"] = cent;
            ViewData["fahrenheit"] = Fahrenheit(cent);

            return View();
        }

        private double Fahrenheit(double degrees)
        {
            return degrees * 9 / 5 + 32;
        }
    }
}
```

Index.cshtml

```
<html>
<head>
    <title>Ejercicio 2.3.1</title>
</head>
<body>
    <h2>@ViewData["degrees"] grados centígrados = @ViewData["fahrenheit"] grados
    Fahrenheit</h2>
</body>
</html>
```

Ejercicio 2.3.2.

Queremos calcular el número de Tarot de una persona, que se obtiene sumando las cifras de la fecha de nacimiento una y otra vez, hasta que se reduzca a un único dígito.

Por ejemplo: 12/12/1900 = 1 + 2 + 1 + 2 + 1 + 9 = 16 = 1 + 6 = **7**

Para esto pasaremos a la aplicación 3 parámetros, day, month y year. Debemos crear una función que sea capaz de calcular el número del Tarot a partir de estos datos, el día, mes y año de nacimiento.

La salida será similar a la que se muestra:

Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(int day, int month, int year)
        {
            ViewData["day"] = day;
            ViewData["month"] = month;
            ViewData["year"] = year;
            ViewData["tarot"] = Tarot(day, month, year);

            return View();
        }

        private int Tarot(int day, int month, int year)
        {
            int tarot = SumOfDigits(day) + SumOfDigits(month) +
                SumOfDigits(year);

            while (tarot > 9)
            {
                tarot = SumOfDigits(tarot);
            }

            return tarot;
        }

        private int SumOfDigits(int number)
        {
            int result = number % 10;
            int n = number / 10;

            while(n > 0)
            {
                result += n % 10;
                n /= 10;
            }

            return result;
        }
    }
}
```

Index.cshtml

```
<html>
<head>
    <title>Ejercicio 2.3.2</title>
</head>
<body>
    <h2>Número de Tarot para la fecha
        @ViewData["day"]/@ViewData["month"]/@ViewData["year"]:
        @ViewData["tarot"]</h2>
</body>
</html>
```

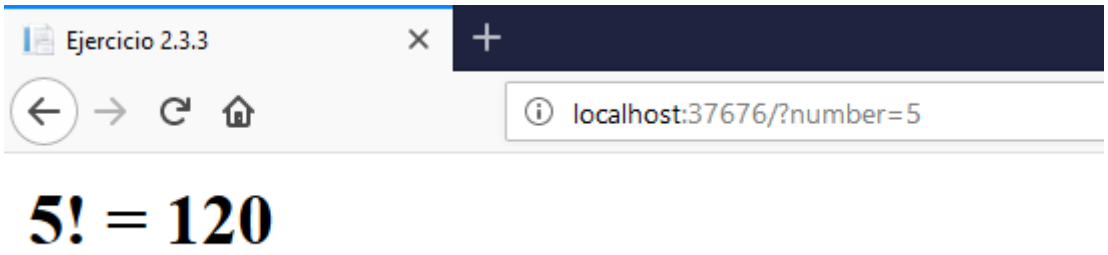
Ejercicio 2.3.3.

Para este ejercicio incluiremos un parámetro *number* en la url, de modo que pasemos un número entero a nuestra aplicación. El resultado será una página con el factorial de dicho número.

Para ello utilizaremos un método en el *HomeController*, que dado un número entero nos devuelva su factorial:

n! = 1 si n<=1
n! = n * (n-1)! si n > 1

El resultado sería similar al siguiente:



5! = 120

Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(int number)
        {
            ViewData["n"] = number;
            ViewData["factorial"] = Fact(number);

            return View();
        }

        private long Fact(long n)
        {
            if(n <= 1)
            {
                return 1;
            }
            else
            {
                return n * Fact(n - 1);
            }

            // Código equivalente al anterior
            // return (n <= 1) ? 1 : n * Fact(n - 1);
        }
    }
}
```

Index.cshtml

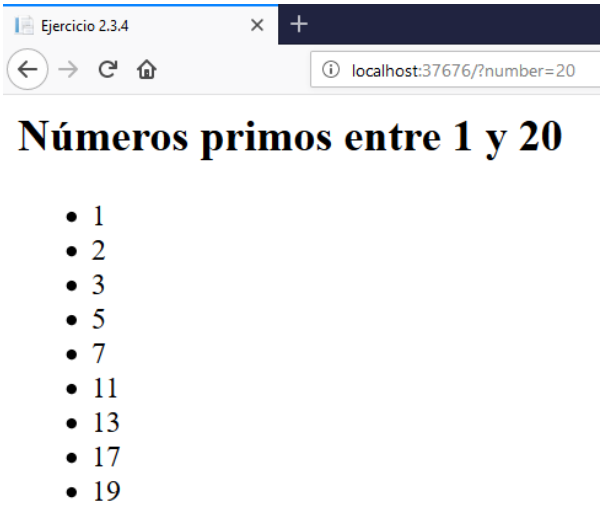
```
<html>
<head>
    <title>Ejercicio 2.3.3</title>
</head>
<body>
    <h2>@ViewData["n"]! = @ViewData["factorial"]</h2>
</body>
</html>
```

Ejercicio 2.3.4.

En este caso lo que queremos es una página donde se muestren todos los números primos entre uno y un número pasado como parámetro.

En este caso adoptaremos un enfoque diferente. Vamos a definir un método estático en el controlador, de modo que pueda ser ejecutado desde la vista, donde recorreremos los números y los mostraremos si son primos. Esto es así porque todavía no hemos visto de que modo pasar colecciones de datos a la vista.

La salida sería como la que se muestra a continuación:



Solución.

HomeController.cs

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

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(int number)
        {
            ViewData["n"] = number;

            return View();
        }

        public static bool IsPrime(int n)
        {
            for (int i = 2; i <= Math.Sqrt(n); i++)
            {
                if (n % i == 0)
                {
                    return false;
                }
            }

            return true;
        }
    }
}
```

Index.cshtml

```
<html>
<head>
    <title>Ejercicio 2.3.4</title>
</head>
<body>
    <h2>Números primos entre 1 y @ViewData["n"]</h2>
    <ul>
        @for (int i = 1; i <= (int)ViewData["n"]; i++)
        {
            if(Ejercicio.Controllers.HomeController.IsPrime(i))
            {
                <li>@i</li>
            }
        }
    </ul>
</body>
</html>
```